Maulana Azad National Institute of Technology

(An Institute of National Importance)

Bhopal – 462003 (India)

Department of Computer Science and Engineering

Major Project Report

α-Net: Architecture, Models, Applications

Submitted in partial fulfillment for the degree of Bachelor of Technology

Authors:
Adya Sharma (161112102)
Jishan Shaikh (161112013)
Ankit Chouhan (161112051)
Avinash Mahawar (161112042)

Advisors:
Prof. Bholanath Roy
Prof. Sweta Jain

Session 2019–20
CSE–419 and CSE–429

Maulana Azad National Institute of Technology

(An Institute of National Importance)

Bhopal – 462003 (India)

Department of Computer Science and Engineering

Declaration of Authorization

We hereby declare that, all the work (including code) produced in this document entitled α**-Net: Architecture, Model, Applications**, is solely created by us as a part of Major project (CSE–419 and CSE–429) under the supervision of **Prof. Bholanath Roy and Prof. Sweta Jain** with proper citations. We are fully aware that any violation of the institute's work ethics or department's legacy policies will clearly make us bear the charges produced by the institute/department according to their etiquette. We also declare that we have not copied anyone's work to be named as our work. All the sources are referred properly at the end of the section for appropriate authorization.

This declaration is the proof that the work produced as a complete document is first handed, and is not sent anywhere with confidentiality, before submission of it in the department and to the supervisor. The facts mentioned above are true to the best of our knowledge.

Authors:
Adya Sharma (161112102)
Jishan Shaikh (161112013)
Ankit Chouhan (161112051)
Avinash Mahawar (161112042)

Advisors:
Prof. Bholanath Roy
Prof. Sweta Jain

Maulana Azad National Institute of Technology

(An Institute of National Importance)

Bhopal – 462003 (India)

Department of Computer Science and Engineering

Certificate

This is to certify that **Adya Sharma, Jishan Shaikh, Ankit Chouhan, and Avinash Mahawar**, all students of final year B.Tech. (Computer Science and Engineering) batch of 2016–20, have successfully completed their Major project entitled **Alpha-Net: Architecture, Model, Applications** in the partial fulfillment of their Bachelor of Technology degree in Computer Science and Engineering at Maulana Azad National Institute of Technology, Bhopal during session 2019–20.

Reviewer and Coordinator:

Prof. Saritha KHETAWAT

Prof. Sanyam SHUKLA
Department of CSE,
MANIT, Bhopal (462003)

Advisors:

Prof. Bholanath ROY

Prof. Sweta JAIN
Department of CSE,
MANIT, Bhopal (462003)

Contents

Acknowledgement **iii**

Abstract **v**

List of Figures **vi**

List of Tables **viii**

List of Acronyms **xi**

1 Introduction **1**

1.1 Theory 1

1.2 Problem Definition 3

1.3 Research Objectives 3

2 Literature Survey **5**

2.1 Residual Networks (ResNet) 5

2.2 ResNeXt 7

2.3 Densely connected CNN 7

2.4 Deep Network with Stochastic Depth 8

3 Methodology and Work Description **11**

3.1 Research Methodology 11

3.2 Work Description 11

3.2.1 Dataset 11

3.2.2 Face alignment 12

3.2.3 Loss function 13
3.2.4 Alpha-Net Architecture 14

4 Tools and Technologies Used **17**
4.1 Minimum System/Software/Hardware requirements 17
4.2 Resources Usage 18
4.3 Functional Requirements 18

5 Implementation **19**
5.1 Coding details 19
5.2 Testing 20

6 Results Analysis **21**
6.1 Accuracy Metric 21
6.2 Comparison with Layer Structure 21
6.3 Comparison with Loss function 23
6.4 Comparison with Normalization function 23
6.5 Comparison between state-of-the-art architectures 23

7 Conclusion **29**

Bibliography **30**

Acknowledgement

We would like to express special thanks with deep sense of gratitude to our respected guide, mentor, supervisor **Prof. Bholanath Roy and Prof. Sweta Jain**, for their valuable help and guidance. Its been a great honor to work under such instructors. We are thankful for the encouragement and motivation that they have given us in initialization phase; giving us a solid head-start. Their rigorous evaluation and constructive criticism were of great assistance. It is imperative for us to mention the fact that this project would not have been accomplished without the periodic suggestions, advice, and assessment of our supervisors.

We also appreciate the time and effort Project reviewers and Coordinators **Prof. Saritha Khetawat and Prof. Sanyam Shukla** put in the session 2019–20. We undergo flawless project management with proper deadlines and motivations, only because of them. Thanks for their support and assistance.

We are also grateful to our respected director **Dr. Narendra Raghuvanshi** for permitting us to utilize all the necessary facilities of the college for this project. Needless to mention is the additional help and support extended by our respected Head of the department **Dr. Nilay Khare**, in allowing us to use the departmental laboratories and other services on the period of time and also maintaining a healthy discipline in management, tutorials, feedback, and other activities.

We are thankful to all the other faculties, professors, associate professors, assistant professors, staff-members, teaching assistants, laboratory attendants, instructors, seniors, and our fellow branch mates of our work culture for their kind co-operation, periodic evaluation, help, and support. We thank all others who directly/indirectly involved in this project and help us making it successful. We also thank all other teams for a maintaining an active communication aside of healthy competition and sportsman's spirit.

We would also like to express our deep appreciation towards our family members for making us such kind; to do 'more' by providing surplus support and encouragement in all aspects of life. This is a hard time, and we would pass it soon. At last, we recall our gratitude to the one eternal almighty: The God.

Abstract

Deep learning network training is usually computationally expensive and intuitively complex. We present a novel network architecture for custom training and weight evaluations. We reformulate the layers as ResNet-similar blocks with certain inputs and outputs of their own, the blocks (called Alpha blocks) on their connection configuration form their own network, combined with our novel loss function and normalization function form the complete Alpha-Net architecture. We provided empirical mathematical formulation of network loss function for more understanding of accuracy estimation and further optimizations. We implemented Alpha-Net with 4 different layer configurations to express the architecture behavior comprehensively. On a custom dataset based on ImageNet benchmark we evaluate Alpha-Net v1, v2, v3, and v4 for image recognition to give accuracy of 78.2%, 79.1%, 79.5%, and 78.3% respectively. The Alpha-Net v3 gives an improved accuracy of approx. 3% over last state-of-the-art network ResNet 50 on ImageNet benchmark. We also present analysis on our dataset with 256, 512, and 1024 layers and different versions of the loss function. The input representation is also very crutial for training as initial preprocessing will take only a handful of features to make training less complex than it needed to be. We also compared network behaviour with different layer structures, different loss functions, and different normalization functions for better quantitative modeling of Alpha-Net.

Keywords: Alpha-Net, Architecture, Neural Network.

List of Figures

1.1 AI, ML and DL. 1
1.2 Why we used deep learning? 2
1.3 Differences in simple NN and deep learning NN. 2

2.1 Increasing layers decrease accuracy. 6
2.2 Different variants of ResNet. 6
2.3 Block structure of ResNext. 7
2.4 Layer structure of densely connected CN Network. 8
2.5 ResNet architectures over ImageNet benchmark. 9

3.1 Data generation using Generative adversarial network. 12
3.2 68 landmark points for alignment (Ref: Openface) 13
3.3 Face alignment on a sample image. 14
3.4 Alpha-Net architecture base for (i) VGG (ii) 34-layer plain. 16

6.1 Graph showing Layer Structure vs Top 1 Accuracy (%). 22
6.2 Graph showing Loss function vs Top 1 Accuracy (%). 24
6.3 Graph showing Normalization function vs Top 1 Accuracy (%). 25
6.4 Graph showing Architecture vs Top 1 Accuracy (%). 26

List of Tables

6.1 Accuracy comparison of Alpha-Net models vs Layer structure. 22
6.2 Accuracy comparison of Alpha-Net models vs Loss function. 23
6.3 Accuracy comparison of Alpha-Net models vs Normalization function. . . . 24
6.4 Accuracy comparison of various architectures over ImageNet based benchmark. 27

List of Acronyms

HOG *Histogram of Oriented Gradients*

ResNet *Residual Network (architecture)*

DenseNet *Densly connected CNN*

CNN *Convolutional Neural Network*

AM *Additive Margin*

BN *Batch Normalization*

GAN *Generative Adversarial Network*

ReLU *Rectifier Linear Unit*

conv *Convolutional (layer)*

Chapter 1

Introduction

1.1 Theory

Deep learning is a division of machine learning in artificial intelligence (AI) that mimics the structure of the human brain in processing data and generating patterns to use in making important decisions. It is an artificial intelligence operation that has a meshwork of neurons capable of learning associations for unseen from data that is structured or unstructured. It is hence rightly known as deep neural learning or deep neural network.

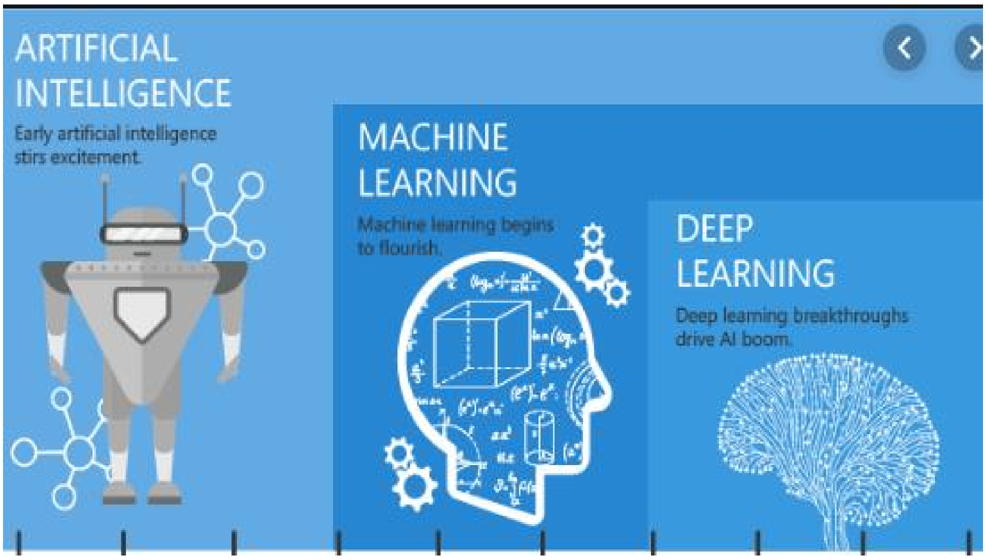

Figure 1.1: AI, ML and DL.

In Deep Convolutional Neural Networks (CNN), the introductory layers axiomatically learn the features laid out for years or even decades, and the subsequent layers further learn advanced level abstraction. Ultimately, the combination of these advanced level abstractions depicts facial identity with unprecedented stability.

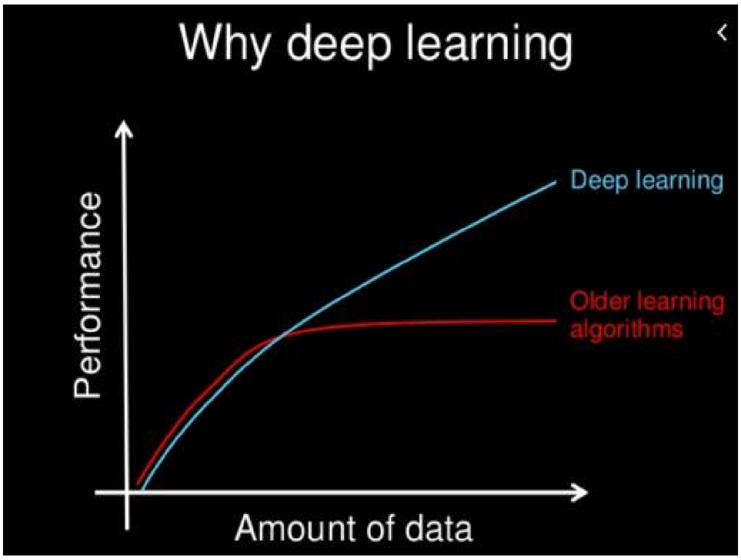

Figure 1.2: Why we used deep learning?

The principal distinguishing element of deep learning compared to more conventional approaches is the ability of the effectiveness of the classifiers to go large scale with an increase in volume of data. Former machine learning methods typically plateau in performance after it reaches a threshold of training data.

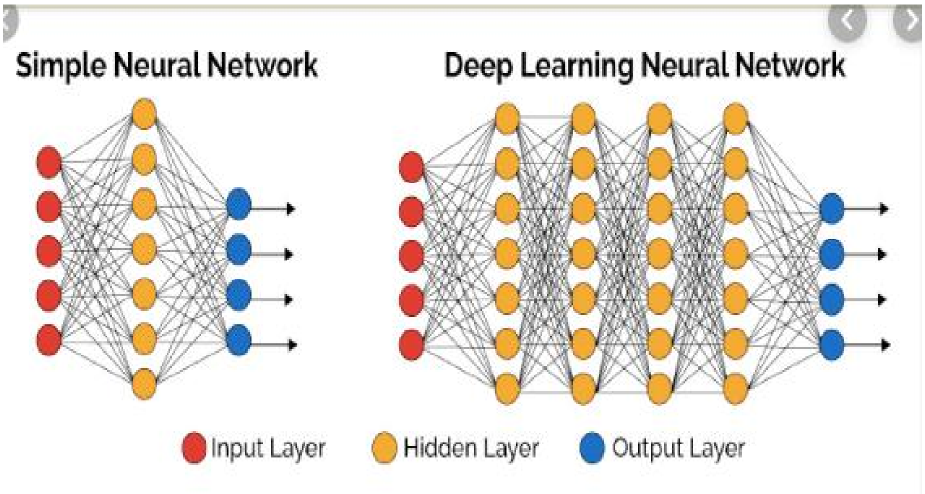

Figure 1.3: Differences in simple NN and deep learning NN.

Deep learning is a distinctive algorithm whose performance enhances with increase in data as more the data fed, the better the classifier is trained on, resulting in surpassing the classical models/algorithms.

1.2 Problem Definition

Deep learning network training is a tedious task because of high computational resources and large number of functions to be tuned. Training purposes mainly for face recognition and object detection methods. As yet, various methods have been designed for face recognition, but it still has remained extremely challenging in real life applications and till date, there is no technique which parallels human ability to distinguish faces despite countless diversity in appearance that the face can take in a scene and implement a powerful solution to all situations accurately irrespective of variations due to light, aging, expressions, similarity in faces and other methodical problems like noise, image acquisition, video camera distortion. Each and every method invented till date suffers from limitations, which hinder the process of achieving cent percent accuracy. **The main task of the project is create a novel DL based architecture, with which we can create separate models for use-cases and use it for our applications.**

Older machine learning algorithms typically plateau in performance after it reaches a threshold of training data. In this scenario, deep learning algorithms come to the rescue as they provide high performance even with an increase in data. In fact, it follows two principles: "The more the data, the better the model" and the "deeper the network, the more powerful it gets".

In these kind of situations, it is imperative to enhance the training methods by introducing a new method using more powerful and efficient functions and architectures, which are robust enough to work for all kinds of variations, yielding better accuracy in less time.

1.3 Research Objectives

In order to overcome the drawbacks of the already existing algorithms, we based our method of creating a new architecture that can train complex networks efficiently and in optimal manner.

The method is based on an improvement over the original ResNet architecture, which has been a benchmark architecture for computer vision tasks such as object detection, face recognition, and segmentation.

The novelties in the project can be summarized as:

1. GAN based dataset for more data; improves training accuracy.

2. Alpha-encodings and transformations of dataset items; for near optimal dataset size and features.

3. New method of interconnection between blocks based on stochastic (random) nature from data items; for better training results in slower time
4. Combined results of convolution filters (e.g. average of 5*5 and 10*10 filters) at each block of the network;
5. Loss function (optimizer); A new variant of AM Softmax function.

The objective of this work is to define a technique robust to the noise to represent, detect and recognize human faces efficiently. It is shown that the described strategy supports not only to acquire and perceive the examples provided during the training phase, but also to generalize them, thus enabling us to detect occurrences of the activities that have not been enclosed in the training set. The expected results show that our system reliably recognizes sequences of human faces with a high recognition rate.

Chapter 2

Literature Survey

2.1 Residual Networks (ResNet)

Residual Networks are one of the most studied networks in neural network architectures. The basic idea of ResNet is to train large number of layers of a neural network efficiently and in a lesser time. ResNet gives an effective analysis of why increasing number of layers at certain point decreases accuracy. The compelling performance of ResNet and its variants can be explained by its residual blocks and identity mappings. The powerful and complex representational ability of ResNet allows computer vision tasks efficiently and easily such as object detection, face recognition, and image segmentation. Figure 1 shows the decrease in accuracy for 20-layer and 56-layer ResNet architecture.

Universal approximation theorem states that any single layer of feed forward network with sufficient capacity can represent any function. However, the layer will get enormous and the network becomes prone to overfitting the data. The trend in community blasts of increasing layers increases training comprehensively, but increasing layers did decrease the training error at higher number of iterations.

Vanishing gradient problem does not let Deep NNs to easily train in most cases – the problem that leads to huge trouble in training neural networks for any fundamental task. Result of which, the performance of it became saturated and even degraded in some of the tasks.

Early attempts to handle the vanishing gradient problem is to add an auxiliary loss layer in the network to adjust the gradient of the network at each iteration of training. The core idea of ResNet is to bring a block of residual network lapsed with different parts such as ReLU, etc. and activation layer, and connection of which leads to the complete network.

The variants of ResNets can be explained by changing its block nature, size, and layers

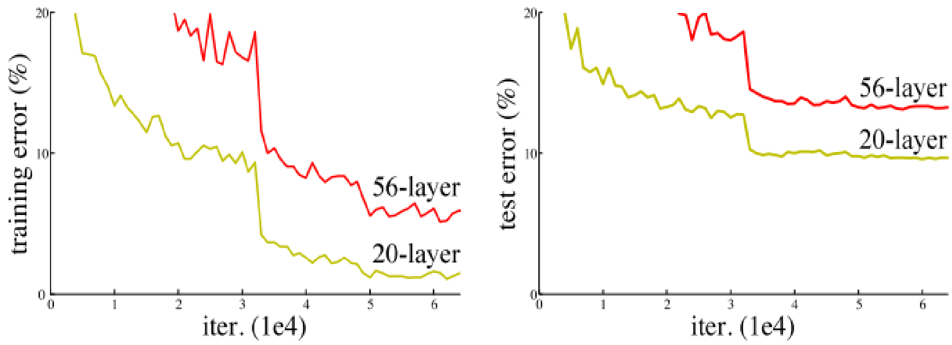

Figure 2.1: Increasing layers decrease accuracy.

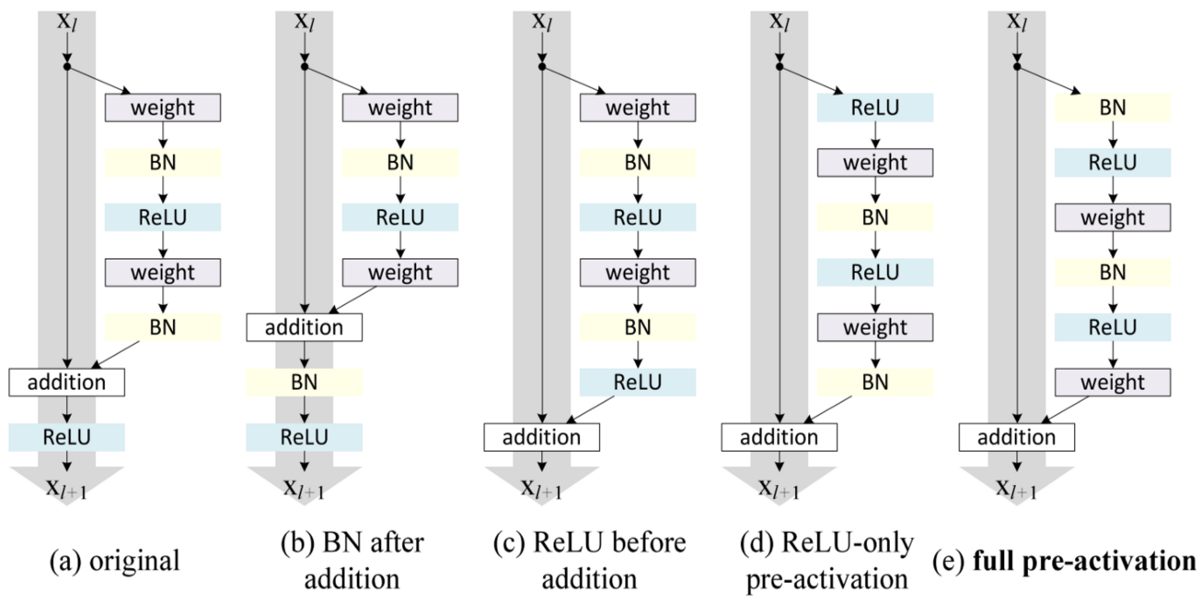

Figure 2.2: Different variants of ResNet.

including the BN after addition, ReLU before addition, ReLU - only pre-activation, and full-activation.

2.2 ResNeXt

The authors introduced a concept called cardinality, to ensure a novel way of regulating the model capacity. They showed that accuracy can be gained more methodically by increasing the cardinality, instead of going deeper or wider network-wise. The authors testified that this network framework is relatively easy to train as compared to Inception Module of GoogleNet since it requires only one hyperparameter, and Inception module requires multiple hyperparameters to tune.

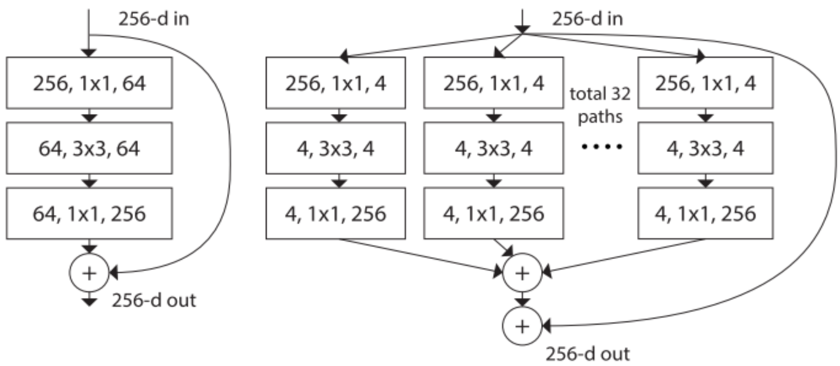

Figure 2.3: Block structure of ResNext.

The “split-transform-merge” is commonly done by pointwise grouped convolutional layer, which partitions its input into groups of feature maps and perform normal convolution respectively, then their outputs are depth-concatenated and then sent as input to a 1x1 convolutional layer.

2.3 Densely connected CNN

Also called as DenseNet, it connects all layers directly with each other. In this contemporary framework, the input of each layer is composed of the feature maps of all former layers, and its output is fed to each subsequent layer. The feature maps are then combined using depth-concatenation.

Apart from dealing with the vanishing gradient problem, the authors state that this architecture also promotes feature reuse, making the network extremely parameter-efficient. One straightforward analysis of this is that, the output of the identity mapping was supplemented to the next blocks, which might deter information flow if the feature maps of two layers have very different distributions.

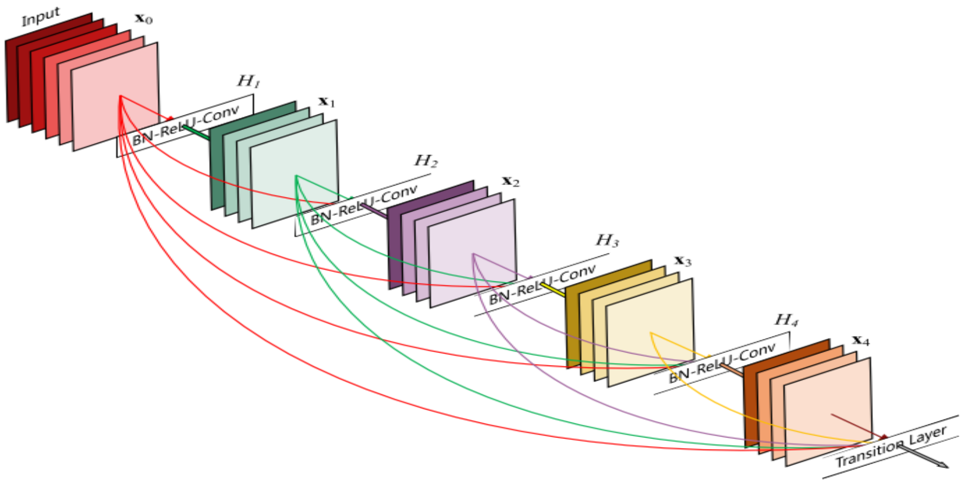

Figure 2.4: Layer structure of densely connected CN Network.

2.4 Deep Network with Stochastic Depth

In spite of ResNet proving its effectiveness in many applications, its one prominent drawback is that deeper networks usually need weeks of training, rendering it practically infeasible in real life applications. To handle this problem, Huang et al. devised a counter-intuitive approach of randomly dropping layers during training, and using the entire framework in testing.

They used only residual blocks for network architecture, the mapping was not particularly identity, the flow can be 2-way in the network. **This inspires us to search for stochastic mapping with convenient network size.** The survival probability is randomly dropped during training time.

Layers	Output Size	DenseNet-121($k = 32$)	DenseNet-169($k = 32$)	DenseNet-201($k = 32$)	DenseNet-161($k = 48$)
Convolution	112×112	7×7 conv, stride 2			
Pooling	56×56	3×3 max pool, stride 2			
Dense Block (1)	56×56	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$
Transition Layer (1)	56×56	1×1 conv			
	28×28	2×2 average pool, stride 2			
Dense Block (2)	28×28	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$
Transition Layer (2)	28×28	1×1 conv			
	14×14	2×2 average pool, stride 2			
Dense Block (3)	14×14	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 24$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 48$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 36$
Transition Layer (3)	14×14	1×1 conv			
	7×7	2×2 average pool, stride 2			
Dense Block (4)	7×7	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 16$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 24$
Classification Layer	1×1	7×7 global average pool			
		1000D fully-connected, softmax			

Figure 2.5: ResNet architectures over ImageNet benchmark.

Chapter 3

Methodology and Work Description

3.1 Research Methodology

The most important steps of the proposed method can be summarized in the following points:

- Dataset organization: ImageNet + CASIA-webface + GAN based dataset + some random images.
- Face alignment using landmark algorithms.
- Input normalization using Alpha-transformations.
- Custom encoding of normalized Numpy matrices for blocks.
- Assigning random weights to interconnect layers initially.
- Add linear weights + Additive margin softmax as loss function layer
- Alpha-Net architecture for training data
- Results combination and analysis

3.2 Work Description

3.2.1 Dataset

We are augmenting a dataset of human faces using GAN. The input to this step is the CASIA-WebFace dataset which is the largest dataset publicly available.

Discriminative algorithms tend to try to classify input data; that is, given the features of an occurrence of data, they try to conclude a category or label to which that data corresponds. For instance, given the set of words in an email (the data instance), a discriminative algorithm can conclude as to whether the message is spam or not. Therefore, discriminative algorithms map features to categories or labels. They are interested entirely with that correlation.

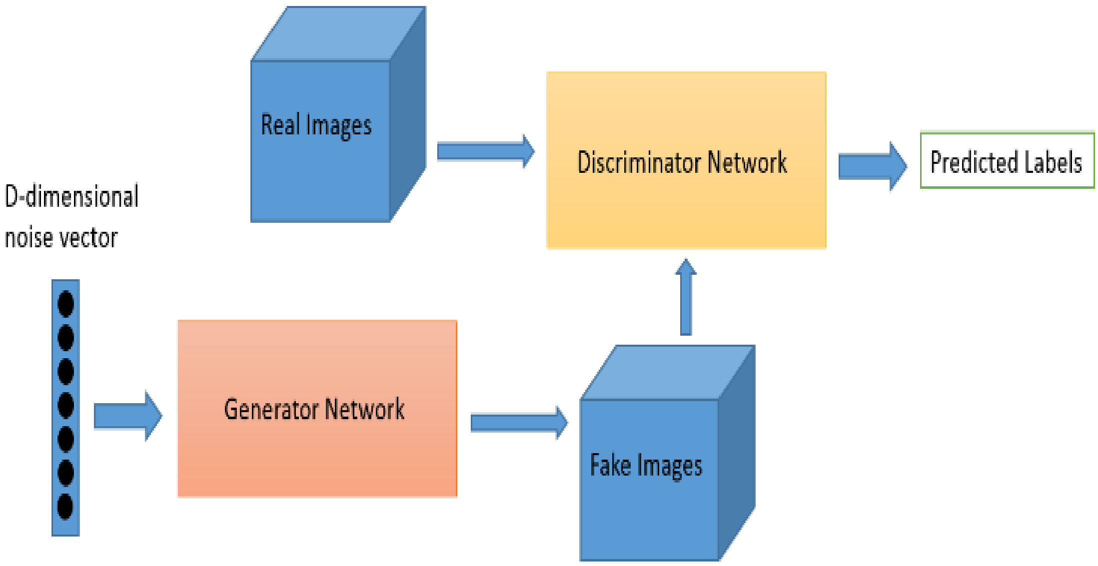

Figure 3.1: Data generation using Generative adversarial network.

Generative adversarial networks (GANs) are deep neural net framework composed of two networks, pitting one against the other (thus called "adversarial"). Rather than assigning a label given certain features, they try to conclude features given a certain label. Generative models design the distribution of separate classes. One neural net, named the generator, generates novel data occurrences, while the other, known as the discriminator, gauges them for legitimacy; i.e. the discriminator determines whether each occurrence of data that it assesses belongs to the actual training dataset or not.

The GAN dataset + CASIA webface dataset + ImageNet dataset is then combined with some random images for more diverseness, and combined dataset is then used for training.

3.2.2 Face alignment

When we clip an image of a person or object, the direction of the person/object facing as well as the position of camera matters a lot. Sometimes other factors such as luminosity and image quality also affect the qualitative features of the image/face. The face orientation is a complex problem, but we can use a simple template of 68 landmark points to assign the face with each point for mapping purposes. After this we can wrap the picture with respect to the template only, and later on we can only use template image features for training and classification purposes.

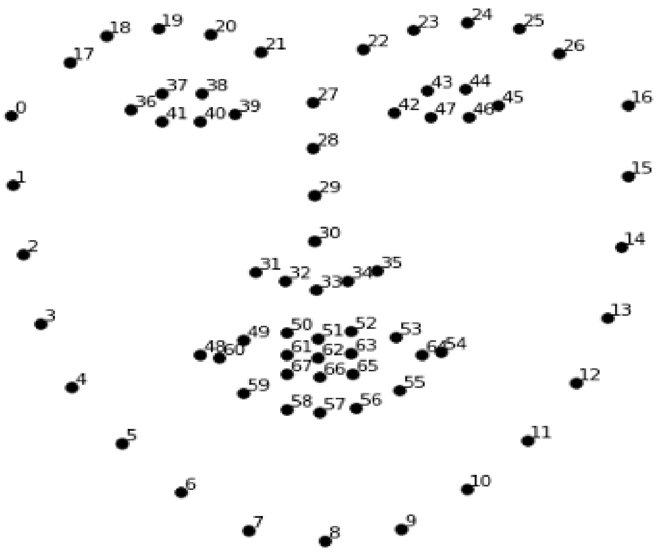

Figure 3.2: 68 landmark points for alignment (Ref: Openface)

When we map a face with a template we know where all the biometrics are. We can then use affine transformations (rotation, translation, and shear) for verification of the face alignment and mapping. Often the case, the image is usually 30 to 60 degree tilted either to right or left, which can be easily aligned by rotation or transformation used properly with size.

The orientation of the face in the data image does not matter, as we can map it to the given particular template. The same landmark points can be used for verification and authentication purposes at later stage, if used manually.

3.2.3 Loss function

Softmax function is widely used in computer vision and deep learing community for classification and learning tasks. This is because softmax function provides less inter-class correlation, and high intra-class correlation. We modified the Additive margin version of Softmax function, to add linear weights trained specially for ImageNet benchmark. The formulation of AM Softmax via $cos(\theta - m)$, can be mathematically noted as:

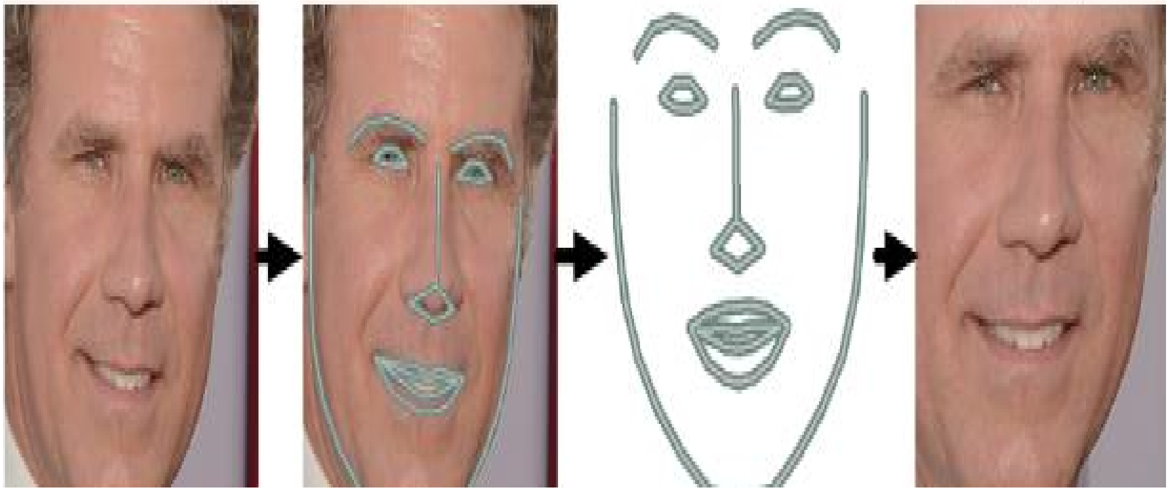

Figure 3.3: Face alignment on a sample image.

$$\Lambda_{AMS} = -\frac{1}{n}\sum_{i=1}^{n}\frac{e^{s(cos\theta_{y_i}-m)}}{e^{s(\cos\theta_{y_i}-m)}+\sum_{j=1,j\neq y}^{c}e^{s\cos\theta_j}}$$

For linear weights ($y = ax_1 + c$), we can add a piece-wise division of it to make AM Softmax function with linear weights trained on ImagNet benchmark, as follows:

$$\Lambda_{AMS} = \begin{cases} -\frac{1}{n}\sum_{i=1}^{n}\frac{e^{s(cos\theta_{y_i}-m)}}{e^{s(\cos\theta_{y_i}-m)}+\sum_{j=1,j\neq y}^{c}e^{s\cos\theta_j}} & \theta - m > 0 \\ ax_i + c & \theta - m <= 0 \end{cases}$$

Where:

Λ_{AMS} = Calculated Loss

n = Training instances

θ = Angle with the origin

m = Gradient for the instance

s = Sample value of the current instance

a and c = Linear weight coefficients

3.2.4 Alpha-Net Architecture

We have designed 4 type alpha nets, and have observed comprehensive results for our dataset. The two basic types of model are described as:

- **Plain Networks:** The designed plain networks are based on VGG Network for image recognition. The layers having convolutions mostly have 3x3 size and for same number of size same output feature map size, the layers have the same number of filters. Number of filters are doubled, when the map size is halved. This is because to keep the lower complexity per layer. By stride of 4, we perform down sampling so as to reduce layer data size and complexity. No. of weighted layers in the image is 34, and is expandable as in v1 for 128, v2 for 256, v3 for 512, and v4 for 1024.

- **Block Networks:** Based upon plain networks, but each and every layer is blocked by a bunch of predetermined set of layers; called as blocks. Each block has an input size 256×256 of data image and outputs a similar size image with convoluted data weights and function approximations. Each block has modified batch normalization (BN) layer, a loss function layer (AM softmax variant with linear weights), 3x3 convolutional layers, and stochastic pooling layer.

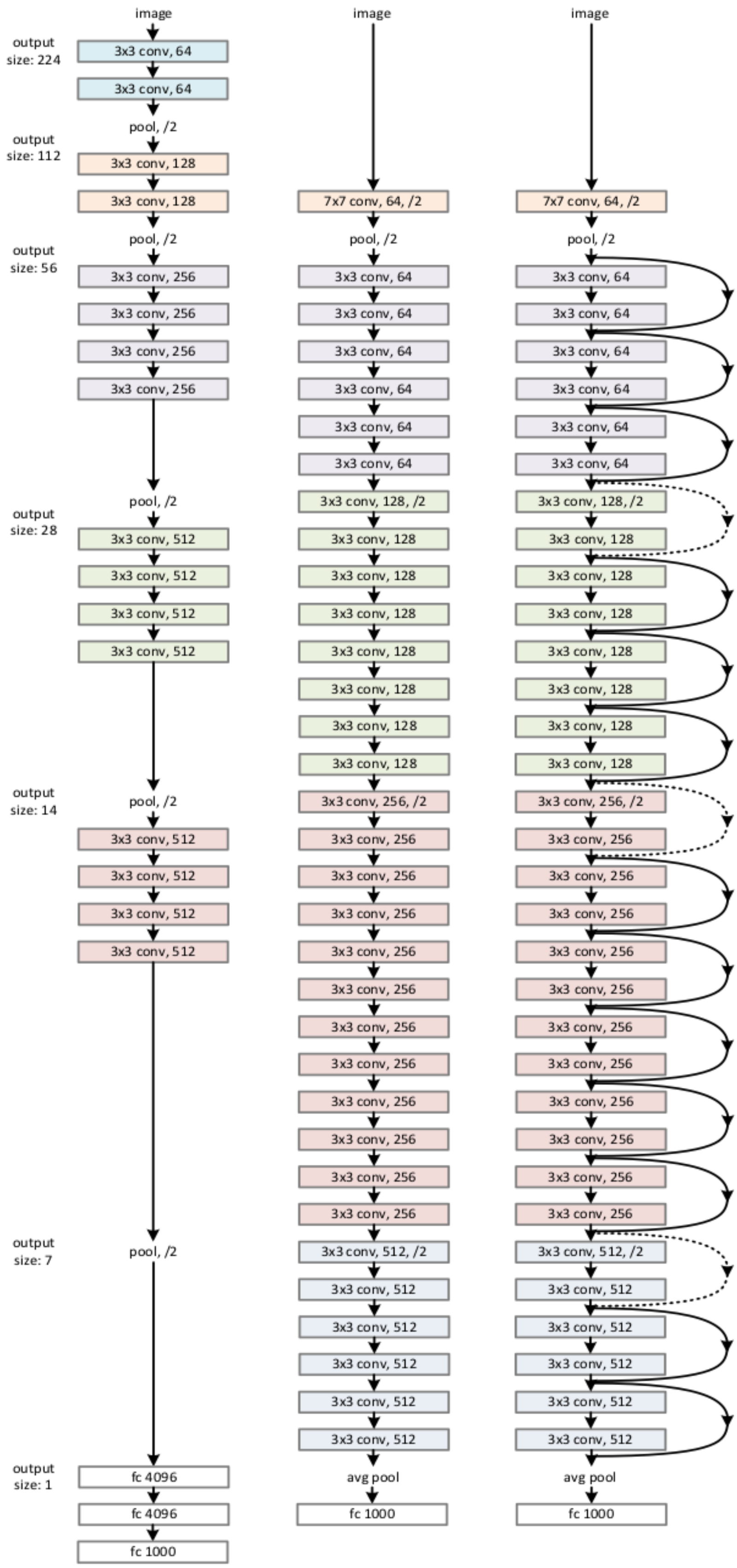

Figure 3.4: Alpha-Net architecture base for (i) VGG (ii) 34-layer plain.

Chapter 4

Tools and Technologies Used

4.1 Minimum System/Software/Hardware requirements

- Windows 10 / Linux / Mac OS / Chromium OS.
- Google Cloud Platform (GCP) for faster trainings and processings (If local system not compatible)
- Memory requirements: 16 GB (RAM) with a GPU or TPU.
  - Recommended NVIDIA GEFORCE 940+MX
- Secondary memory requirements (Hard disk): 200 GB (ROM).
  - Recommended 512 GB / 1 TB for dataset storage and processing
  - HDFS if distributed is necessary
- Latest versions of web browsers - Chrome, Chromium, Firefox, or Opera.
- Python libraries: Scikit-learn, Numpy, Pandas, Python 3.8
- Writer and diagram designing software such as LibreOffice Draw.
- Clock Speed: 1.5 Ghz
- Virtual Memory: 32 bits (minimum).
- Cache Memory: 512 MB, etc.

4.2 Resources Usage

- Operating System: Windows 10 / Ubuntu 20.04 / **Kali Linux 20.0.4**
- Memory: **8 GB** (RAM)
- Secondary memory: **1 TB** (ROM)
- Firefox, Chrome, **Chromium**, and Opera web browsers.
- Microsoft Word 16 / **LibreOffice Writer 5.4** (Linux).

4.3 Functional Requirements

- Cached data storage and retrieval.
- Proper User Interface as in accordance with common use-cases; i.e. the terminal.
- Customized testing functionalities in accordance with the proposed model (Python3.8).

Chapter 5

Implementation

5.1 Coding details

Our implementation for the proposed dataset follows the standard pratice of training and testing methodology based on Keras v2. The image is resized with its shorter side randomly sampled in [256, 480] for scale augmentation. All the images with faces are aligned so as to minimize the complexity of the angle and orientation of the image. Augmentation of color is used as described in original ResNet paper. We take convolutional layer, activation layer and batch normalization to form a block, in our alpha blocks as similar to residual blocks. We initialize the weights as in the ImageNet dataset and train the network. The learning rate (β) starts from 0.01 and is divided by 10 when the error converges, and the models are trained for up to 60x10, 8 for each v1, v2, v3, and v4. For consistency reasons, dropout was not used.

Parameters used for training are as follows:

- Momentum: 0.9
- Weight decay: 0.0001
- Initial learning rate: 0.01
- Batch size: 512
- Batch size division: 10

5.2 Testing

For testing purposes, we used 10 crop technique for qualitative analysis. The data is first normalized as per the image size and is used as per our transformations, and the scores are averaged over certain image sizes such as 32, 64, 128, 256, and 512. It is also important to note that stochastic pooling for custom objects is not available in Keras, so we implemented it separately for convenience.

Chapter 6

Results Analysis

6.1 Accuracy Metric

We use Top 1 accuracy as a comparison metric between performance of different models and architectures. Top 1 accuracy is the accuracy when a model gives a single accuracy probability and is match with the trained dataset accuracy. There are other variants of accuracy metric such as Top 5 accuracy and Top 10 accuracy but they give 5 and 10 values of probabilities respectively and out of which the closest one is to be compared; the disadvantage of Top 5 and Top 10 accuracy is that they might produce false results since the accuracy range is pretty narrow in ImageNet based benchmark we created, and as from ImageNet competition.

Since we created 4 models with Alpha-Net architecture: v1, v2, v3, and v4 with different layers, we analyze all the possible combinations arrive over different parameter and technique.

6.2 Comparison with Layer Structure

The comparison of all the Alpha-Net models with respect to layer structure is shown in Table 6.1 with layer structures as plain layer structure, residual block structure, and alpha block structure. We used Alpha blocks for our benchmark.

Best results are shown by alpha-blocks because of careful formulation of layers inside the alpha-block. We can also see that the accuracy of v4 model is less than its expected accuracy because of the possible reason of overfitting with large number of layers in them.

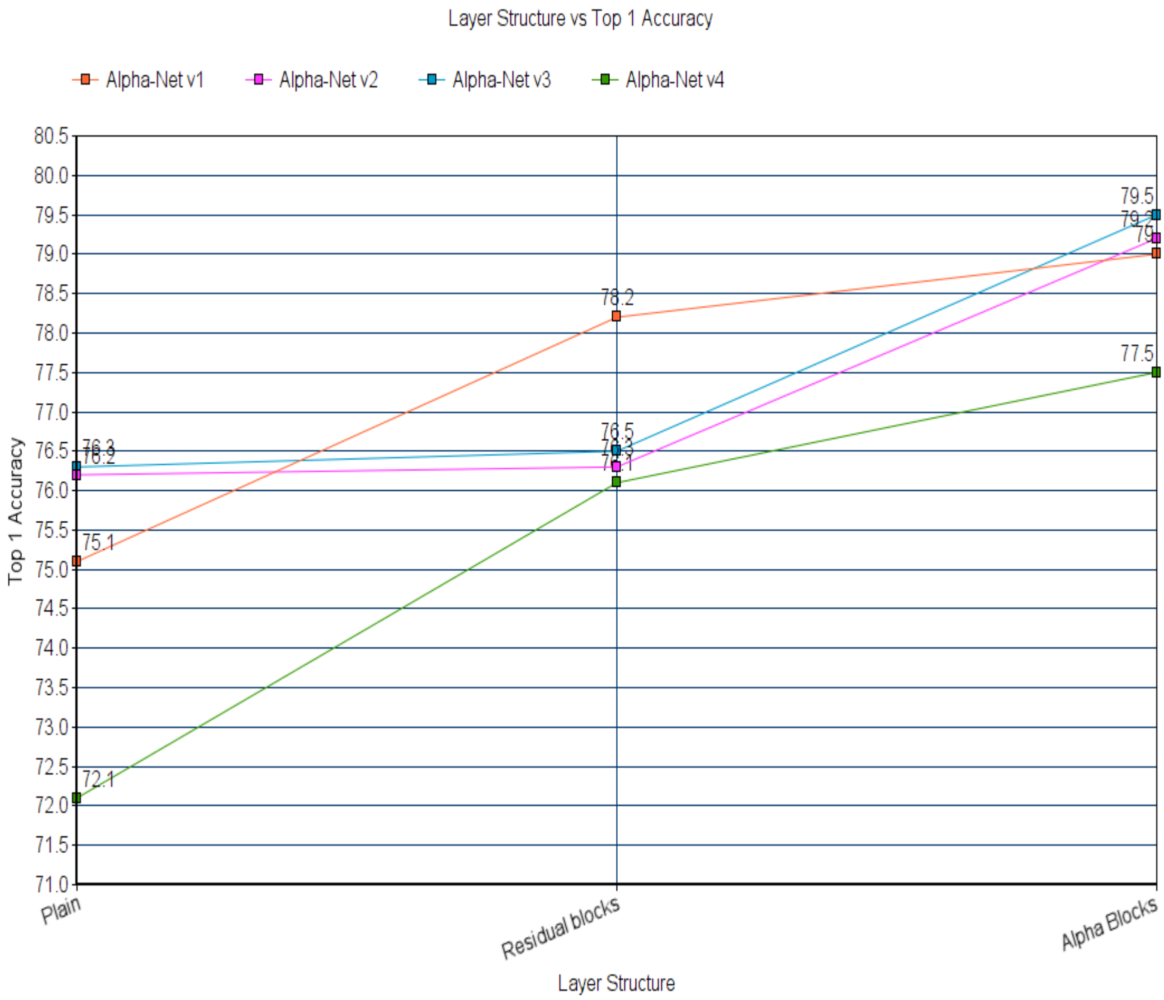

Figure 6.1: Graph showing Layer Structure vs Top 1 Accuracy (%).

Architecture	No. of Layers	Layer structure (Top 1 Accuracy)		
		Plain	Residual Blocks	Alpha Blocks
Alpha-Net v1	128	75.1%	78.2%	79.0%
Alpha-Net v2	256	76.2%	76.3%	79.2%
Alpha-Net v3	512	76.3%	76.5%	79.5%
Alpha-Net v4	1024	72.1%	76.1%	77.5%

Table 6.1: Accuracy comparison of Alpha-Net models vs Layer structure.

Architecture	No. of Layers	Loss Function (Top 1 Accuracy)		
		Softmax	AM Softmax	AM Softmax + Linear Weights
Alpha-Net v1	128	72.1%	74.3%	76.2%
Alpha-Net v2	256	71.3%	74.3%	77.1%
Alpha-Net v3	512	72.1%	74.3%	77.2%
Alpha-Net v4	1024	71.2%	73.1%	75.1%

Table 6.2: Accuracy comparison of Alpha-Net models vs Loss function.

6.3 Comparison with Loss function

The comparison of all the Alpha-Net models with respect to loss functions is shown in Table 6.2 with loss functions as Softmax function, AM Softmax function, AM Softmax function with linear weights. We used AM Softmax function with linear weights for our benchmark.

Best results are shown by AM Softmax function with linear weights because of linear weights improvements over normal Additive margins. We can also see that the accuracy of v4 model is less than its expected accuracy because of the possible reason of overfitting with large number of layers in them.

6.4 Comparison with Normalization function

The comparison of all the Alpha-Net models with respect to the normalization function is shown in Table 6.3 with normalization functions as log scaling, z-score, and Alpha-encoding. We used Alpha encoding for our benchmark.

Best results are shown by Alpha encoding because of simple and consistent feature extraction by alpha-transformations. We can also see that the accuracy of v4 model is less than its expected accuracy because of the possible reason of overfitting with large number of layers in them.

6.5 Comparison between state-of-the-art architectures

The comparison of all the Alpha-Net models (v1, v2, v3, and v4) with different architectures is shown in Table 6.4. The global benchmark is pop out to be InceptionResNet v2. InceptionResNet v2 is the second improvement with combinational features of residual

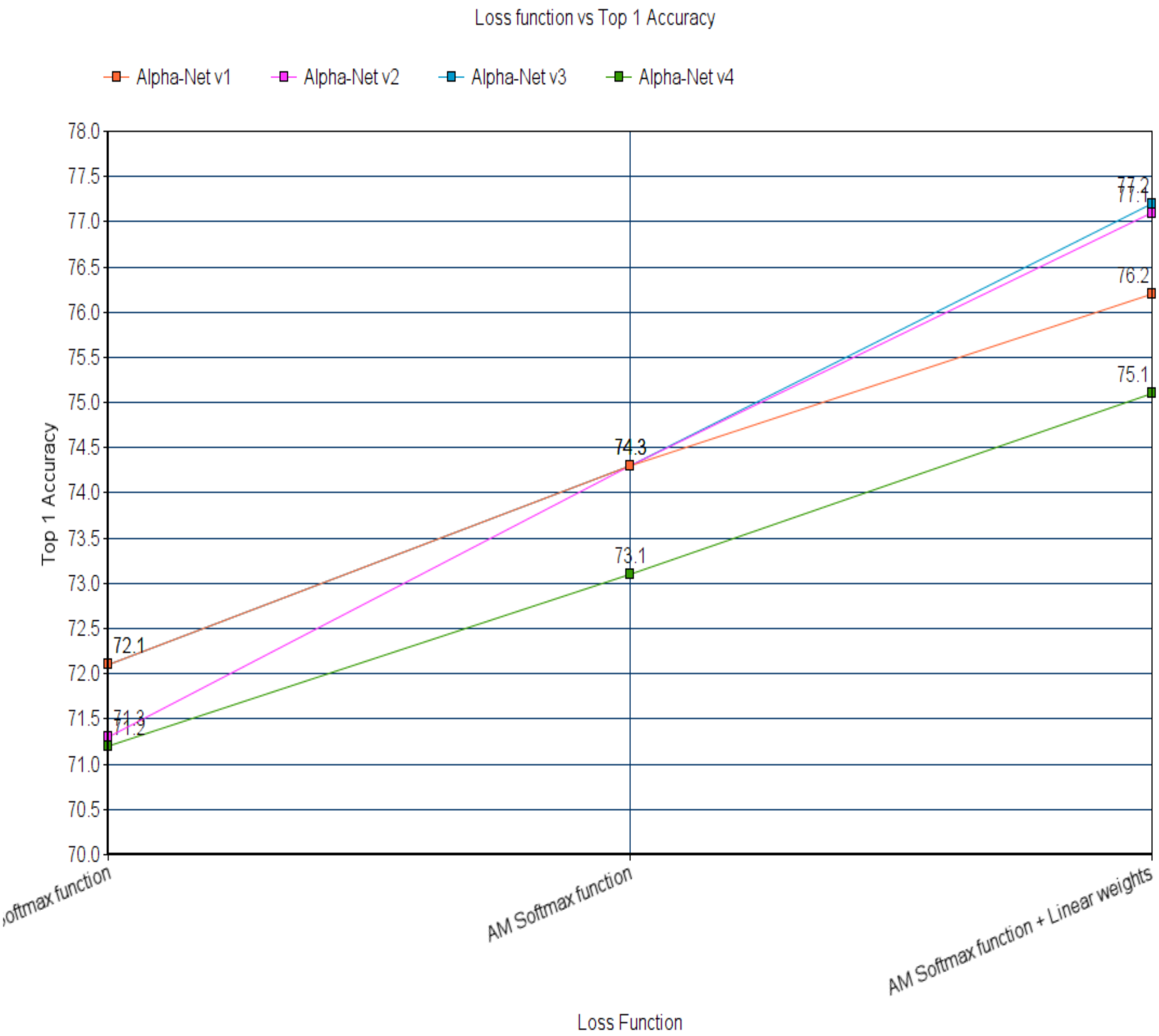

Figure 6.2: Graph showing Loss function vs Top 1 Accuracy (%).

Architecture	No. of Layers	Normalization (Top 1 Accuracy)		
		log-scaling	z-score	Alpha-encoding
Alpha-Net v1	128	69.2%	71.2%	71.0%
Alpha-Net v2	256	69.5%	70.1%	71.2%
Alpha-Net v3	512	70.1%	70.1%	71.5%
Alpha-Net v4	1024	71.2%	69.5%	70.5%

Table 6.3: Accuracy comparison of Alpha-Net models vs Normalization function.

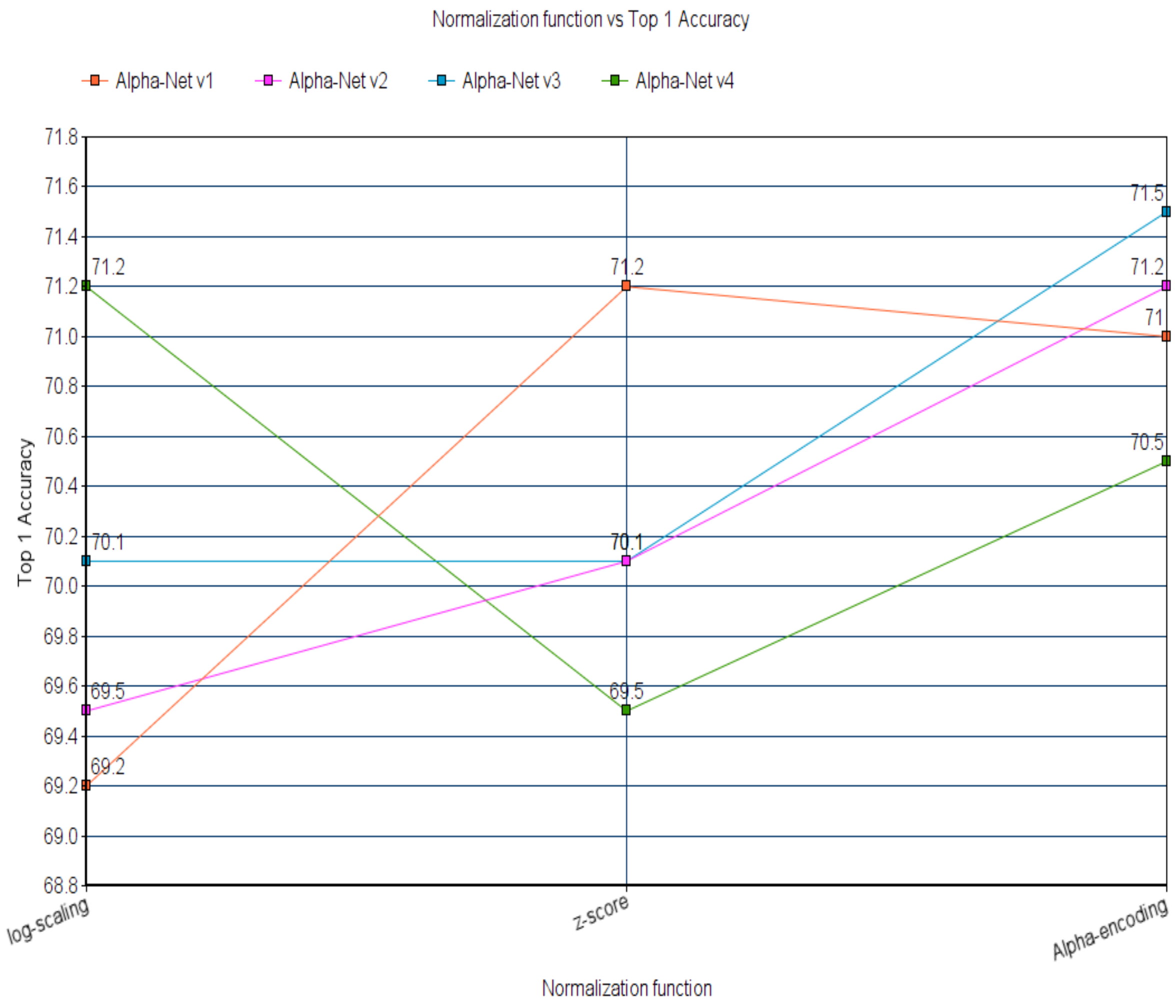

Figure 6.3: Graph showing Normalization function vs Top 1 Accuracy (%).

blocks and representation with Inception module functionalities.

Our benchmark produces **second best results for v3**, and are accompanied by AM Softmax function with linear weights, Alpha-encoding, and Alpha-blocks in combination.

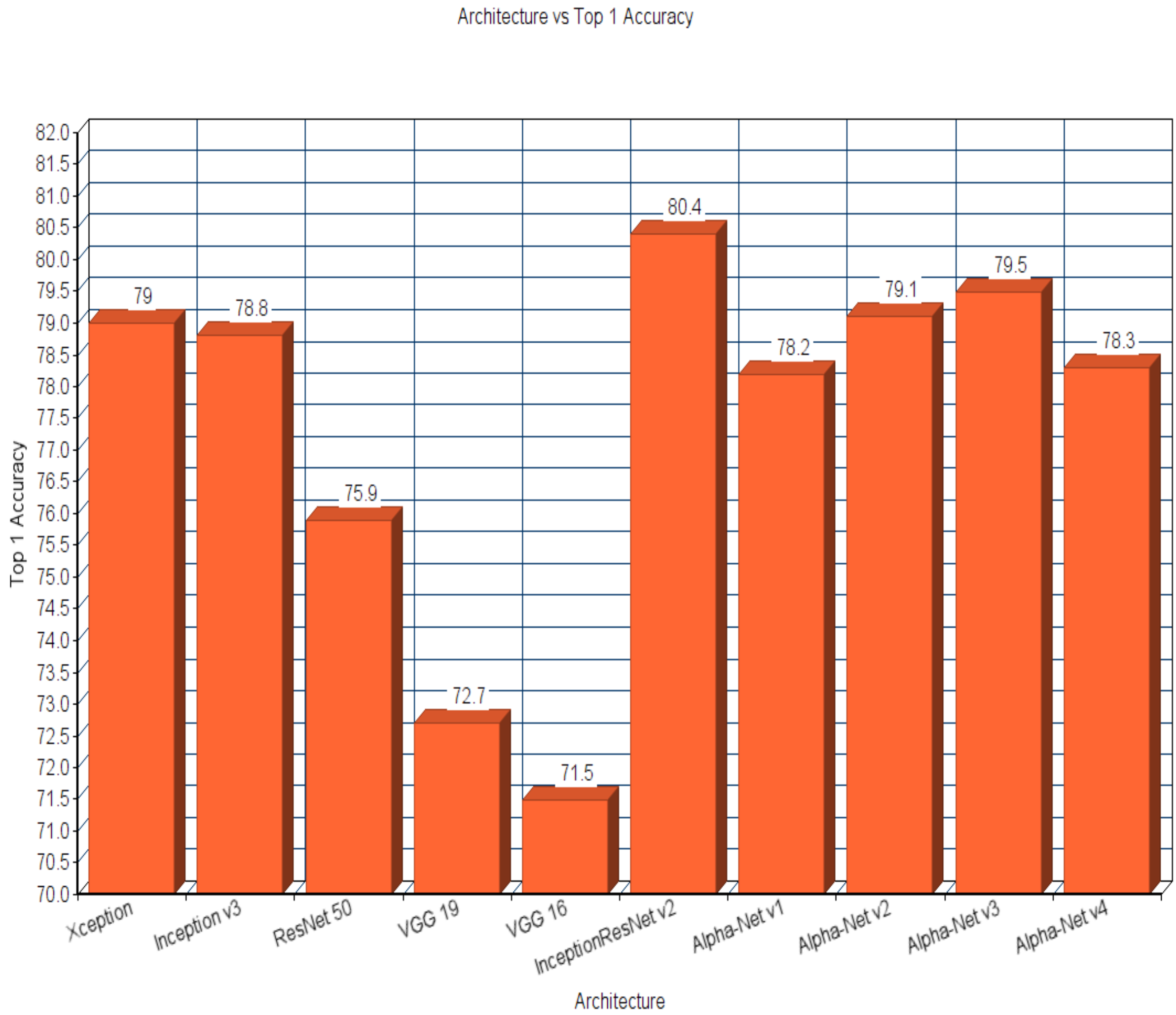

Figure 6.4: Graph showing Architecture vs Top 1 Accuracy (%).

Architecture	Top 1 Accuracy
Xception	79.0%
Inception v3	78.8%
ResNet 50	75.9%
VGG 19	72.7
VGG 16	71.5
InceptionResNet v2	80.4%
Alpha-Net v1	78.2%
Alpha-Net v2	79.1%
Alpha-Net v3	79.5%
Alpha-Net v4	78.3%

Table 6.4: Accuracy comparison of various architectures over ImageNet based benchmark.

Chapter 7

Conclusion

Deep learning systems are complex and non-comprehensive by their nature of deep layers and data representations. Combining powerful features to a single layer and processing is a computationally expensive task. Much of which require a cloud platform (GCP or AWS) for processing and storage. A new architecture is proposed named Alpha-Net based on the alpha-transformations of the input data; which is important because to keep data size low for faster processing. Four custom models were implemented based upon Alpha-Net named Alpha-Net v1, v2, v3, and v4 with layers 128, 256, 512, and 1024. The common misconception that increasing layer size increases the training accuracy is also successfully busted when v4 model shows less accuracy than v1, v2, and v3; that means the threshold number of layers is in between v3 and v4; i.e. 512 and 1024. Since, some authors have proved that ResNet has some redundant layers; we used this fact to improve the mapping scheme of the layers based on stochastic approach. The data flow is not linear, and certainly not many to many (increases the training complexity), so we randomly assigned mapping from one layer to all other layers, which adjusts itself in many iterations.

The project has many **novel contributions** including but not limited to the dataset used, input representation, normalization of input data, layer mappings, block structure of layers, and the loss function – linear weights + Additive Margin Softmax function. The result of each and every novelty affects the training accuracy in an indifferent manner; the summary of which is described in results analysis (chapter 6).

From our results analysis, Alpha-Net performs better than ResNet (original and its variants) on ImageNet based benchmark. Multiple trainings of alpha-net models suggest that the optimal number of layers for any computational training task is constant, and is often less than 10,000.

Bibliography

. [1] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In IEEE Conference on Computer Vision and Pattern Recognition, pages 770–778, 2016.

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

[3] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller.Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical report, Technical Report 07-49, University of Massachusetts,Amherst, 2007.

[4] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In Proceedings of the 22nd ACM international conference on Multimedia, pages 675–678. ACM, 2014.

[5] I. Kemelmacher-Shlizerman, S. M. Seitz, D. Miller, and E. Brossard. The megaface benchmark: 1 million faces for recognition at scale. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 4873–4882, 2016.

[6] X. Liang, X. Wang, Z. Lei, S. Liao, and S. Z. Li. Soft-margin softmax for deep classification. 24th International Conference on Neural Information Processing, pages 413–421, 2017.

[7] S. Liao, Z. Lei, D. Yi, and S. Z. Li. A benchmark study of large-scale unconstrained face recognition. In IEEE Inter-national Joint Conference on Biometrics, pages 1–8. IEEE,2014.

[8] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar. Focal loss for dense object detection. arXiv preprint arXiv:1708.02002, 2017.

[9] W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj, and L. Song. Sphereface: Deep hypersphere embedding for face recog- nition. In Proceedings of the IEEE conference on computer vision and pattern recognition, 2017.

[10] W. Liu, Y. Wen, Z. Yu, and M. Yang. Large-margin soft-max loss for convolutional neural networks. In International Conference on Machine Learning, pages 507–516, 2016.

[11] W. Liu, Y.-M. Zhang, X. Li, Z. Yu, B. Dai, T. Zhao, and L. Song. Deep hy-

perspherical learning. In Advances in Neural Information Processing Systems, pages 3953–3963,2017.

[12] Y. Liu, H. Li, and X. Wang. Rethinking feature discrimination and polymerization for large-scale ecognition. arXiv preprint arXiv:1710.00870, 2017.

[13] O. M. Parkhi, A. Vedaldi, and A. Zisserman. Deep facerecognition. In BMVC, volume 1, page 6, 2015.

[14] G. Pereyra, G. Tucker, J. Chorowski, Ł. Kaiser, and G. Hinton. Regularizing neural networks by penalizing confident output distributions. arXiv preprint arXiv:1701.06548, 2017.

[15] R. Ranjan, C. D. Castillo, and R. Chellappa. L2-constrained softmax loss for discriminative face verification. arXiv preprint arXiv:1703.09507, 2017.

[16] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In IEEE Conference on Computer Vision and Pattern Recognition, pages 815–823, 2015

[17] Y. Sun, Y. Chen, X. Wang, and X. Tang. Deep learning face representation by joint identification- verification. In Advances in neural information processing systems, pages 1988–1996, 2014

[18] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. Deepface: Closing the gap to human-level performance in face verification. In IEEE Conference on Computer Vision and Pattern Recognition, pages 1701–1708, 2014

[19] F. Wang, X. Xiang, J. Cheng, and A. L. Yuille. Normface: L2 hypersphere embedding for face verification. In Proceedings of the 25th ACM international conference on Multimedia. ACM, 2017.

[20] Y. Wen, K. Zhang, Z. Li, and Y. Qiao. A discriminative feature learning approach for deep face recognition. In European Conference on Computer Vision, pages 499–515.Springer, 2016.

Abstract — Deep learning network training is usually computationally expensive and intuitively complex. We present a novel network architecture for custom training and weight evaluations. We reformulate the layers as ResNet-similar blocks with certain inputs and outputs of their own, the blocks (called Alpha blocks) on their connection configuration form their own network, combined with our novel loss function and normalization function form the complete Alpha-Net architecture. We provided empirical mathematical formulation of network loss function for more understanding of accuracy estimation and further optimizations. We implemented Alpha-Net with 4 different layer configurations to express the architecture behavior comprehensively. On a custom dataset based on ImageNet benchmark we evaluate Alpha-Net v1, v2, v3, and v4 for image recognition to give accuracy of 78.2%, 79.1%, 79.5%, and 78.3% respectively. The Alpha-Net v3 gives an improved accuracy of approx. 3% over last state-of-the-art network ResNet 50 on ImageNet benchmark. We also present analysis on our dataset with 256, 512, and 1024 layers and different versions of the loss function. The input representation is also very crucial for training as initial preprocessing will take only a handful of features to make training less complex than it needed to be. We also compared network behavior with different layer structures, different loss functions, and different normalization functions for better quantitative modeling of Alpha-Net.

Keywords: Alpha-Net, Architecture, Neural Network.

MANIT Campus
Link Road – III, Near Mata Mandir,
Bhopal – 462003
Madhya Pradesh, India (+91).