Data Augmentation with Part Deformations

A Project Report

submitted by

DATTA SAI KRISHNA VARDHAN REDDY KARNA

in partial fulfilment of the requirements

for the award of the degree of

BACHELOR OF TECHNOLOGY &

MASTER OF TECHNOLOGY

DEPARTMENT OF ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY, MADRAS.

May 2020

THESIS CERTIFICATE

This is to certify that the thesis titled **Data Augmentation with Part Deformations**, submitted by **Datta Sai Krishna Vardhan Reddy Karna**, to the Indian Institute of Technology, Madras, for the award of the degree of **Bachelor of Technology & Master of Technology**, is a bonafide record of the research work done by him under our supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Prof. Anurag Mittal
Research Guide
Dept. of CSE
IIT Madras, 600036

Place: Chennai
Date: 6th May 2020

ACKNOWLEDGEMENTS

I am very grateful to my research guide Professor Anurag Mittal for his constant invaluable guidance and support throughout the project. Also, special thanks to Arul S Kumar and Vismay Patel for constantly resolving my doubts and helping me in my thought process. Learnt a great lesson from this project that I should always cross check my work and results before proceeding further as it may lead to undesirable situations.

ABSTRACT

Data augmentation is one of the prominent techniques that is often used to improve the performance of a network, thereby making the network more robust in learning various features and variations. Though traditional data augmentation techniques such as mirroring, random cropping and color-shifting improve performance to some extent, they are at the global level and are not at the object level. (1) introduces a novel method to augment 2D shape data of deformable images by identifying parts from the shape and deforming them. Though this technique shows improvements on the shape datasets, the maximum amount of a part deformation is fixed irrespective of the object and part.

In this work, we will extend the same method to augment real-world images and show the effectiveness of the technique on the CUB-200-2011 dataset (7). Also, we overcome the drawback of fixed maximum amount of deformation by proposing an algorithm based on reinforcement learning and adversarial training to learn part-specific values for deformations. We will illustrate that our method performs better and improves accuracy.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

CNN - Convolutional Neural Network

GAN - Generative Adversarial Network

NOTATION

r	Amount to rotate a part
s	Amount to scale a part
s_h	Amount to scale a part along its major axis
s_v	Amount to scale a part along its minor axis
σ_r	standard deviation for Normal distribution from which we sample r
σ_s	standard deviation for Normal distribution from which we sample s
σ_{hs}	standard deviation for Normal distribution from which we sample s_h
σ_{vs}	standard deviation for Normal distribution from which we sample s_v
Framework 1	Framework based on Previous Work
Framework 2	Our proposed Framework
E	Expectation
J	Objective function

CHAPTER 1

INTRODUCTION

Over the past couple of years, Deep Convolutional Neural Networks has revolutionized many aspects of research and industry related to computer vision tasks and improved the performance in these tasks. In order to train such networks and learn complex functions, we need large amounts of data. We have enough data for some tasks, but not for all. Most often we might not have enough data to train complex networks especially in the case of computer vision tasks which try to learn information present in the pixels. Data augmentation is often useful in such cases which adds more data to the training set and also avoids the network from overfitting, thus improving the accuracy.

Traditional data augmentation techniques such as mirroring, random cropping, random rotations and color-shifting have proved to improve performance for most of the data hungry algorithms. Though these types of augmentation methods improve accuracy to some extent, they are at the global level and consider image as a whole and don't consider any object-specific changes and hence lack to serve class specific object variations. Fig 1.1 shows conventional data augmentation methods performed on a sample image.

In this report, we would like to explore other plausible data augmentation techniques that involve object specific variations. One way is to generate a new image of the object in a different pose compared to the original pose i.e., image when viewed from a different position. This can be done by first mapping the 2D image of the object onto a 3D model as shown in Fig 1.2. Now view the model from another angle

Figure 1.1: Conventional data augmentation methods: random rotations, random cropping, mirroring. Source: `https://kharshit.github.io/blog/2019/04/12/data-augmentation`

and transform it into a 2D image as done in (3). But it is in general difficult to get the 3D models of all types of classes and map the 2D images onto these 3D models with detailing for all objects involved in vision tasks.

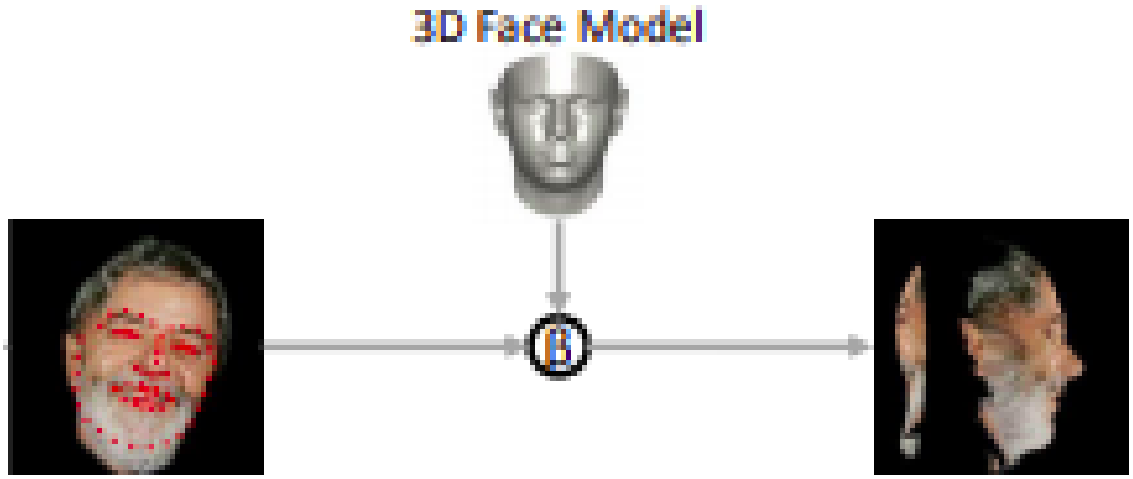

Figure 1.2: Mapping of 2D image to 3D model and vice versa. Image taken from (3)

Another way is to generate new images of the object by deforming the 2D image. This is only feasible for those objects that are not rigid and can be deformed such as humans, animals, birds, etc. We want to utilize the fact that these objects have parts that can be rotated or scaled. In this work we will exploit such part base object variations to augment new data. In the following chap 2, we will study a

novel data augmentation technique introduced by (1) that uses such part based object deformations to augment data for 2D shape dataset for the task of shape classification.

In chap 3, we will extend the same technique to real world images and show the effectiveness of this method on CUB-200 dataset (7). In section 3.2 we will overcome the limitation of (1) i.e., having the same value for maximum amount of deformation for all parts and objects without any dependence. We propose an algorithm based on reinforcement learning and adversarial training to learn this maximum amount of deformation given an object and its part to deform.

CHAPTER 2

BACKGROUND AND RELATED WORK

In order to deform a given image for augmentation, as mentioned in the previous section, we first need to identify parts. (1) proposes a new method to identify and localize parts using a novel cut detection technique. It basically involves discovering potential cuts. Cuts are line segments that join parts which are obtained by identifying concave points on the contour of the shape. These concave points serve as endpoints for the cuts. It then removes all invalid cuts with some valid heuristics such as

- length of the cut should be less than some threshold
- the entire cut should be within the shape
- the cut should intersect the skeleton structure only once
- the cut should not be parallel to the skeleton segment
- the cut should not be parallel to the contour

Fig 2.1 clearly shows above mentioned heuristics. After obtaining the potential cuts, it constructs a graph with endpoints of cuts as vertices. Any two vertices in the graph are connected if either they are neighbors on the contour or they are the endpoints of a cut. Now it identifies possible parts by detecting cycles in the graph and prune them with some additional valid heuristics. Fig 2.2 demonstrates the graph construction from cuts.

After identifying the parts, it randomly selects one of them to apply a local transformation such as rotation and/or scaling. This local transformation will be

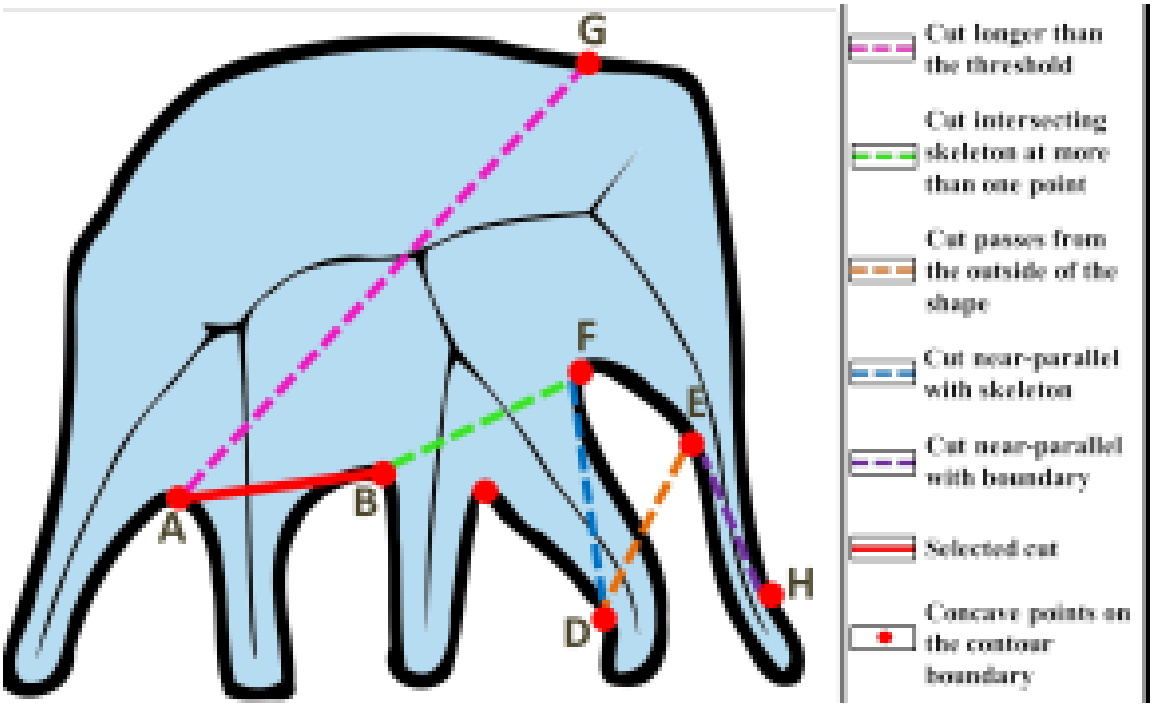

Figure 2.1: Heuristics for potential cuts. Image taken from (1)

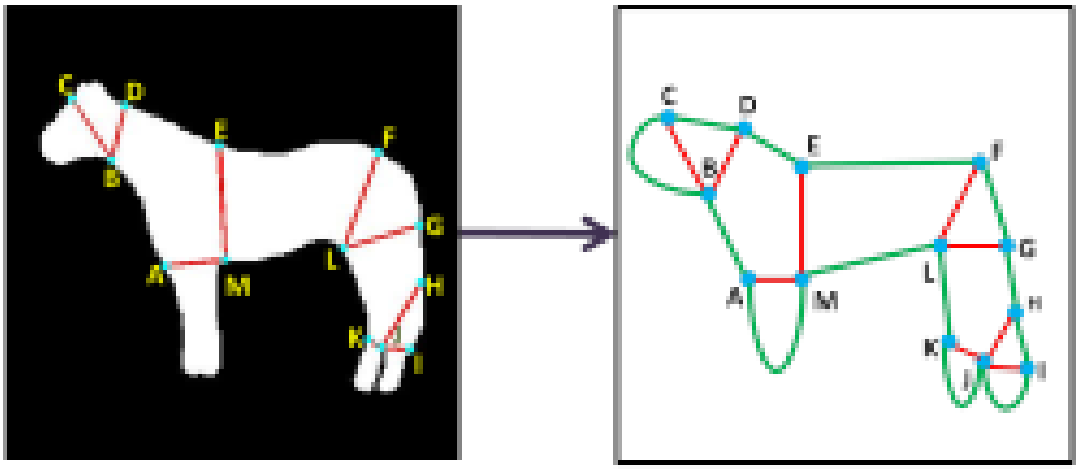

Figure 2.2: Graph construction from cuts. Image taken from (1)

spread across the rest of the shape with the shape manipulation technique proposed by (2) and hence keeps the image consistent. (2) treats the shape as a mesh of triangles and applies the deformation first on this mesh which will be mapped to the original shape later.

The vertices of the triangles which reside in the part are considered as handle

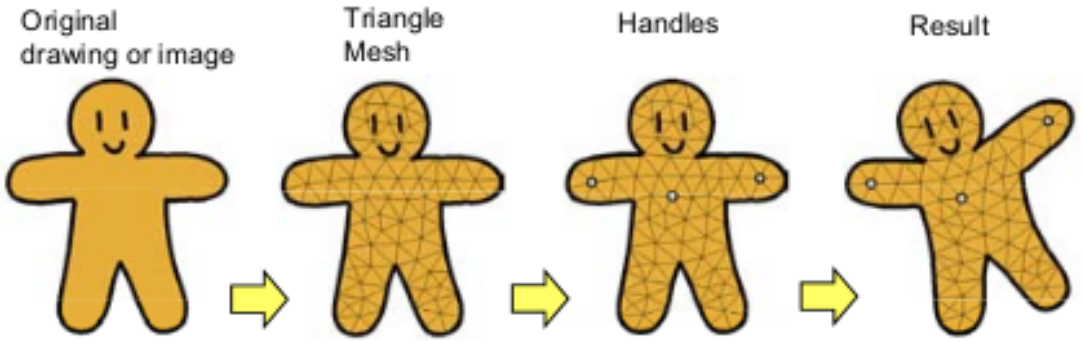

Figure 2.3: Image deformation using mesh optimization. Image taken from (2)

points to which we apply the local transformation. The positions of the remaining vertices will then be computed to minimize the associated distortions of all triangles. Thus, it carries deformation from one part of the shape to the whole mesh and makes the deformation physically constrained as shown in Fig 2.3.

The paper (1) performed experiments and demonstrated the improvements in results on the Animal-Shape dataset (4) and the MPEG-7 dataset (5). Some of the deformed images are shown in Fig 2.4. These datasets are just shapes of animals and birds in the form of binary images. In the following chap 3, we will first extend this method to real-world images and show its potential in data augmentation with improvements in accuracy on Caltech-UCSD Birds-200 dataset (7).

In the method proposed by (1), while deforming a selected part, the amount of deformation i.e. rotation and scaling are sampled from a normal distribution $N(0, \sigma_r^2)$ and $N(0, \sigma_s^2)$ respectively with σ_r and σ_s as hyper parameters. But these values depend on the object and the part selected in the object. In order to overcome this issue, we will explore ways to compute the maximum amount of deformation σ.

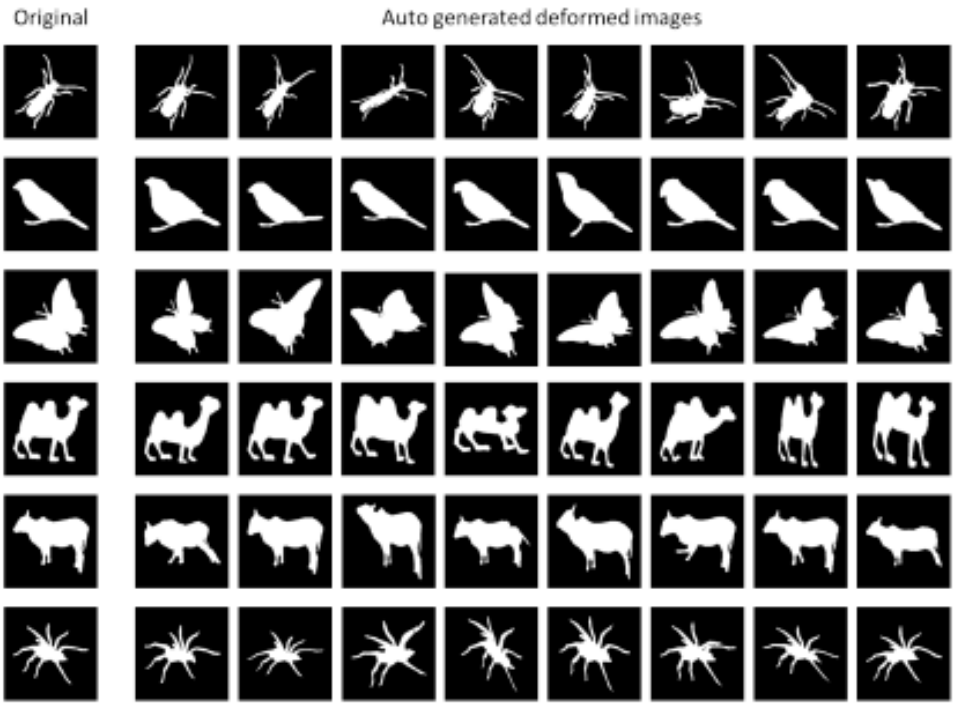

Figure 2.4: Deformed shapes from original images with the technique proposed by (1). Image taken from (1)

CHAPTER 3

PART BASED DATA AUGMENTATION

In this chapter we will look at two frameworks. In the first framework we simply extend (1) to the real-world images i.e. treating σ_r and σ_s as hyper parameters. In the second framework we will learn these parameters given the object and its part.

3.1 Framework 1

In order to extend the method proposed by (1) to real-world images, we need bounding boxes and segmentation masks for the objects in the images. To show the efficacy of this method on real-world images, we use CUB-200 dataset(7) as it contains both segmentation masks and bounding boxes.

With the segmentation mask we can extract the object from the original image leaving the image with background alone. We will fill this void, created by extracting the object, with image inpainting technique as shown in Fig 3.1. With the help of bounding boxes we can keep track of the location of the object so that we can place back the deformed object on top of the image inpainted background. We do this inpainting once for the entire dataset and store it as metadata.

Segmentation masks can be treated as shapes of the object. We first perform mesh analysis on the segmentation mask and store the information of vertices of triangles as metadata. We perform the shape analysis on the segmentation mask, as done in (1), to identify and localize parts and store all the information about parts such as

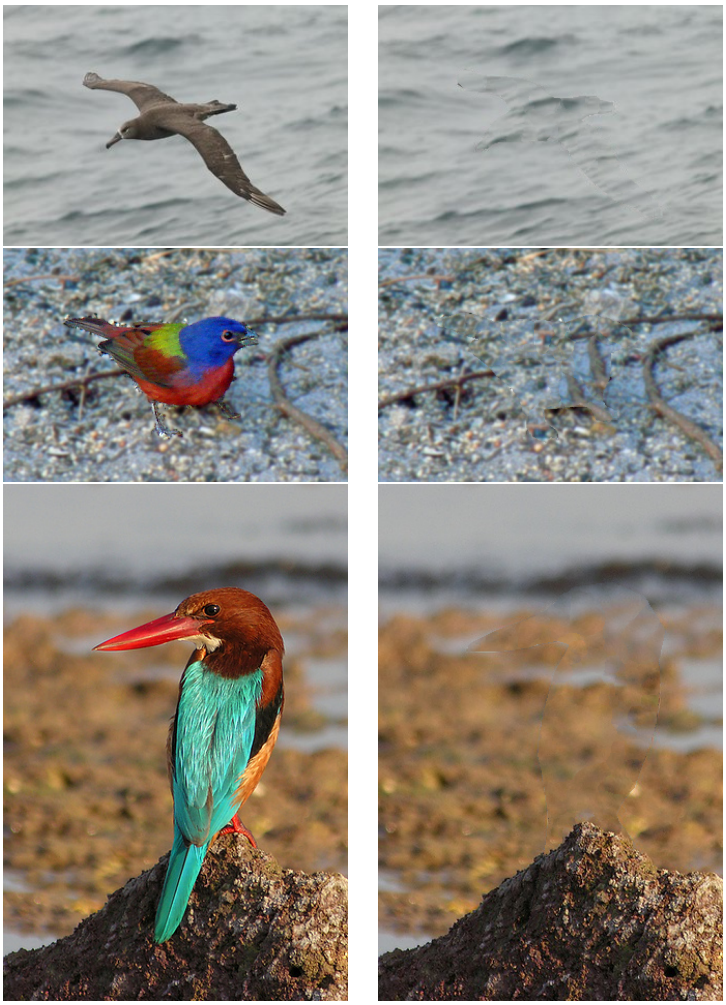

Figure 3.1: Filling void after extracting the object with image inpainting techinque

their location, handle points(vertices of mesh triangles that lie inside the part), can be rotated or not, etc as metadata. This information will later be used to augment data.

Now given an image, we will select a part randomly from its metadata. We will sample r and s i.e., the amount to rotate and amount to scale respectively, from a normal distribution as described in (1).

$$r \sim N(0, \sigma_r^2) \tag{3.1}$$

$$s_h \sim N(0, \sigma_{hs}^2) \tag{3.2}$$

$$s_v \sim N(0, \sigma_{vs}^2) \tag{3.3}$$

We will now scale the part along its major axis by $(1 + s_h)$ times and $(1 + s_v)$ times along its minor axis. Here major axis is the along the skeletal segment that passes through the part. If there are many skeletal segments, we take the middle among them as explained in (1). The minor axis will be perpendicular to the major axis. For the rotation, we will first check if we can rotate the part as all parts cannot be rotated freely. Only parts with a single cut i.e., only exterior parts can be rotated freely as explained in (1). If this part can be rotated we will rotate this part by r degrees.

We apply all these transformations first to the handle points present in the part. After which positions of other vertices of mesh triangles will be computed minimizing the distortion as explained in previous chapter. These transformations will then be mapped to the shape. giving us a new deformed shape. We will do the same set of transformations to the RGB image of the object and place it back on the background with the help of bounding box coordinates, thus giving us a new augmented image. In section 4.2, we will show the effectiveness of this method and compare it with the conventional data augmentation techniques.

In this section, we have treated σ as a hyper parameter i.e., it will be the same for all parts in all objects. This assumption may not be reasonable as the maximum amount of deformations depend on the object and the part selected in it. In the following section we will try to learn this σ given an object and a part in it.

3.2 Framework 2

Let us first formulate the required task i.e, we need a function F to tell how much maximum a particular part can be rotated or scaled for the given object. Let the input image of the object be I and the extracted object be i (extracted with the help of segmentation mask) and p be the part selected to deform. Now the problem statement is to build or model this function F that inputs i , p and outputs $\sigma^2(\sigma_r^2\sigma_{hs}^2\sigma_{vs}^2)$, from which we sample the amount this part p can be rotated, scaled along its major axis and scaled along its minor axis i.e.

$$\sigma^2 = F(i, p) \tag{3.4}$$

This σ will then be used to deform the image as explained in the previous section. In order to get the required σ from the input object i and for the selected part p, we need to build the function F which is going to be complex. The best way to model or build such a complex function with the image as input is to use deep convolutional neural networks(CNN). We, therefore, model F with a CNN namely a suggestion network that suggests how much this particular part can be rotated or scaled, as shown in Fig 3.2.

In order to train this network F, we have to define a loss function. But we don't have any targets to define a loss. We, therefore, use a Discriminator Network(D) motivated from Generative Adversarial Networks(6) to evaluate how good the deformed object is. We also use a Feature Extractor (H) that extracts features from both the original object and the final deformed object to compare them and tell us how close the deformed image is with the original object as we don't desire too much variation.

Though the problem seems to be solved, it is difficult to propagate gradients

through the deformation module i.e., through the method proposed in (2) to deform the object. We, therefore, incorporate the idea of policy gradient from Reinforcement Learning to train the Suggestion Network F.

We will treat the Discriminator(D) and feature extractor(H) as the environment. Policy is the functional behaviour of an agent. Our goal is to learn a policy from which the we sample our actions to maximize the total reward. In other words, by looking at the rewards we make our policy to move in the direction that makes it better by adjusting it with gradients. In our case, actions are sampling of amount of deformations from the policy. We want our action a(amount to deform) to be continuous and the best policy in such scenarios is to use Gaussian Policy. We, therefore, define our policy π for the current state $S(i, p)$ and action a to be as follows.

$$\pi_\theta(S, a) = P(a/S, \theta) \rightarrow N(0, \sigma^2) \tag{3.5}$$

where θ parameterizes the suggestion network and P denotes the probability of an action given a state and θ.

$$\pi_\theta(S, a) \rightarrow N(0, \sigma^2) \tag{3.6}$$

where σ^2 is a function of input object i and the selected part p and the function is parameterized by θ as in

$$\sigma^2 = F_\theta(i, p) \tag{3.7}$$

$$a \sim N(0, \sigma^2) \tag{3.8}$$

$$a = \begin{pmatrix} r \\ s_h \\ s_v \end{pmatrix} \sim \begin{pmatrix} N(0, \sigma_r^2) \\ N(0, \sigma_{hs}^2) \\ N(0, \sigma_{vs}^2) \end{pmatrix} \tag{3.9}$$

where r, s_h, s_v denotes the amount to rotate, scale along major axis and minor axis of the part respectively.

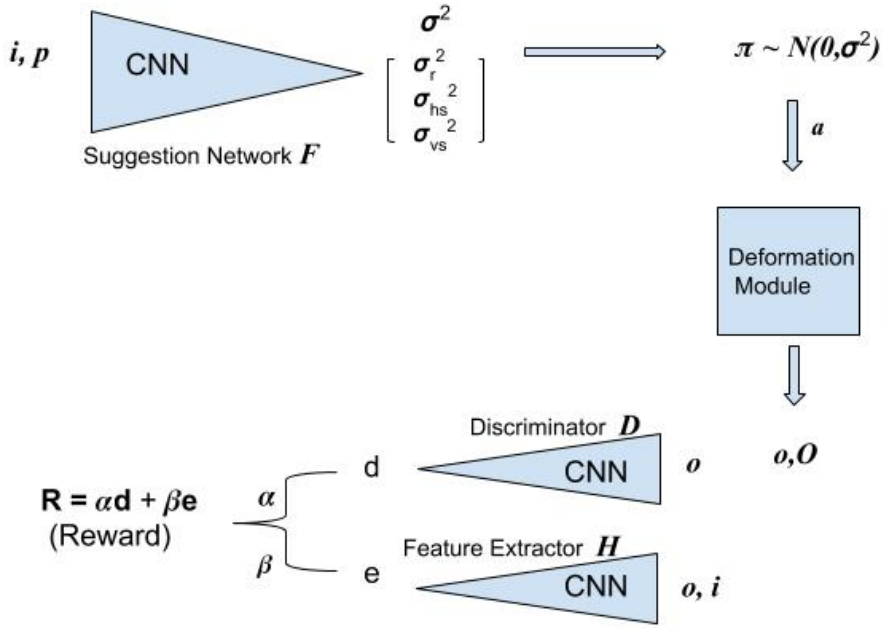

Figure 3.2: Model architecture for Framework 2. Given an object i with a part p, Suggestion Network(F) outputs σ^2 that helps in building a policy π from which we sample an action a, amount to deform. The deformation module outputs the deformed object o and the deformed image O. D tells if o is valid or not. H compares the features of the o and i. Reward R helps in propagating gradients to F.

So given an object and part, we get the σ from the Suggestion Network(F). We now sample an action a from the policy defined by σ as shown in Eq 3.9 and deform the object by passing it through the deformation module. The Discriminator Network

D, then judges whether the deformed image is valid or not. We train this D like we train the discriminator in the Generative Adversarial Networks(GAN)(6). We mark all images from the training data as positive examples and the deformed images as negative. The objective function for the Discriminator Network will be as follows.

$$J_D = E_i(L_2(1, D(i))) + E_{i,p}(L_2(0, D(G(i, p)))) \tag{3.10}$$

where G is the combined model of suggestion network and deformation module similar to a generator in GAN.

We also use a Feature Extractor Network H to compare the generated object with the original object, so that both have similar features. H will be a VGG-16 network pretrained on ImageNet.

In order to train the Suggestion network, we need to calculate the reward for the sampled action a. We calculate the reward R as a linear combination of losses d and e with hyperparameters α and β as shown in Eq 3.11

$$R = \alpha d + \beta e \tag{3.11}$$

Here d is the loss from the Discriminator for the deformed object which it should have classified as positive example and e is the loss from the Feature Extractor between the features of the deformed and the original image. α and β values will be negative so that high losses in d and e would imply less reward and vice versa.

Our objective is to maximize the expectation of the reward over the entire dataset. The Objective function and hence the gradient calculation for the Suggestion Network

will be as follows.

$$\begin{aligned} J &= E_{\pi_\theta}(R) \\ &= \sum_{x \in X} q(x) \sum_{a \in A} \pi_\theta(x, a) R_{x,a} \end{aligned}$$

where x denotes a data point constituting (i, p) and q denotes the distribution of sample space X. A denotes the action space.

$$\begin{aligned} \nabla_\theta J &= \sum_{x \in X} q(x) \sum_{a \in A} \nabla_\theta \pi_\theta(x, a) R_{x,a} \\ &= \sum_{x \in X} q(x) \sum_{a \in A} \pi_\theta(x, a) \nabla_\theta log(\pi_\theta(x, a)) R_{x,a} \end{aligned}$$

as $\nabla z = z \nabla log(z)$

$$\nabla_\theta J = E_{\pi_\theta}(\nabla_\theta log(\pi_\theta(x, a)) R_{x,a}) \tag{3.12}$$

Substituting the Gaussian distribution for policy π will result in Eq 3.13

$$\nabla_\theta J = E_{\pi_\theta}(\nabla_\theta log(\frac{1}{\sqrt{2\pi}\sigma} e^{\frac{a^2 - 0}{2\sigma^2}}) R_{x,a}) \tag{3.13}$$

$$\nabla_\theta J = E_{\pi_\theta}(\nabla_\theta (\frac{1}{2} log(\frac{1}{\sigma^2}) - \frac{a^2 - 0}{2\sigma^2}) R_{x,a}) \tag{3.14}$$

$$\nabla_\theta J = R_a(-\frac{1}{2\sigma^2} + \frac{a^2}{2\sigma^4}) \frac{\partial \sigma^2}{\partial \theta} \tag{3.15}$$

$$\nabla_\theta J = R_a(-\frac{1}{2(\sigma^2 + \varepsilon)} + \frac{a^2}{2(\sigma^2 + \varepsilon)^2}) \frac{\partial \sigma^2}{\partial \theta} \tag{3.16}$$

$$\theta \longleftarrow \theta + \alpha \nabla_{\theta} J \tag{3.17}$$

Here, instead of updating the parameters once for the entire distribution, we do stochastic gradient ascent. Thus eq 3.15 gives the gradient required to update θ. Since there is a possibility of gradients exploding because of σ in the denominator, we add a small value $\varepsilon > 0$ to σ^2 as in eq 3.16. We update θ with a suitable learning rate α as in eq 3.17. Thus we can train the Suggestion network, parametrized by θ, using the above objective function without passing gradients through the deformation module.

Once we have trained the Suggestion network F, we can now get σ given an object i and a part p. This σ will then be used to deform the object by the deformation module and thus generates a new augmented image. In the following chapter we will show the effectiveness of both frameworks on CUB-200 dataset.

3.2.1 Architecture Details

In this subsection we will look at the architecture details of the Suggestion Network and Discriminator.

As shown in Fig 3.3, the Suggestion Network inputs 5 channel tensor consisting of RGB object, mask for the object and mask for the part. Across the whole network we use 3×3 filters and Leaky Relu as activation function. We use MaxPool layers to reduce the size of the tensors except for the first two convolutional layers. Finally at the end we use sigmoid to output 3 values representing our σ^2.

As shown in Fig 3.4, Discriminator inputs 3 channel tensor i.e., either the RGB object from the dataset or the deformed RGB object. Across the whole network we

use 3×3 filters and Leaky Relu as activation function. Instead of MaxPool Layers we use strides during convolution to reduce the size of the tensor.

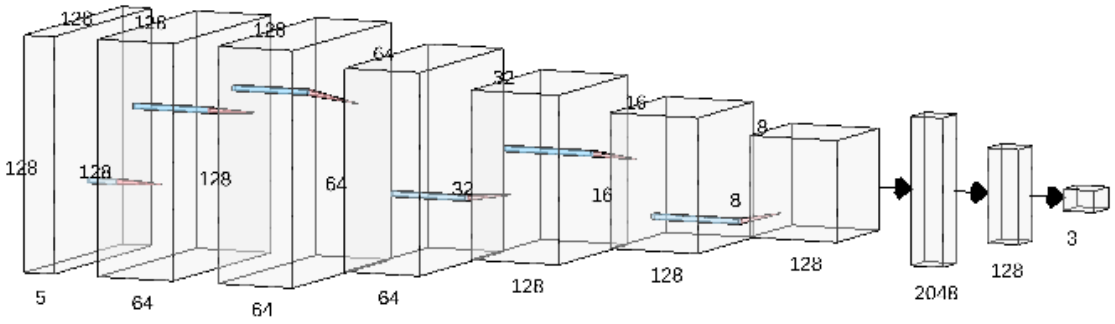

Figure 3.3: Network architecture for Suggestion Network.

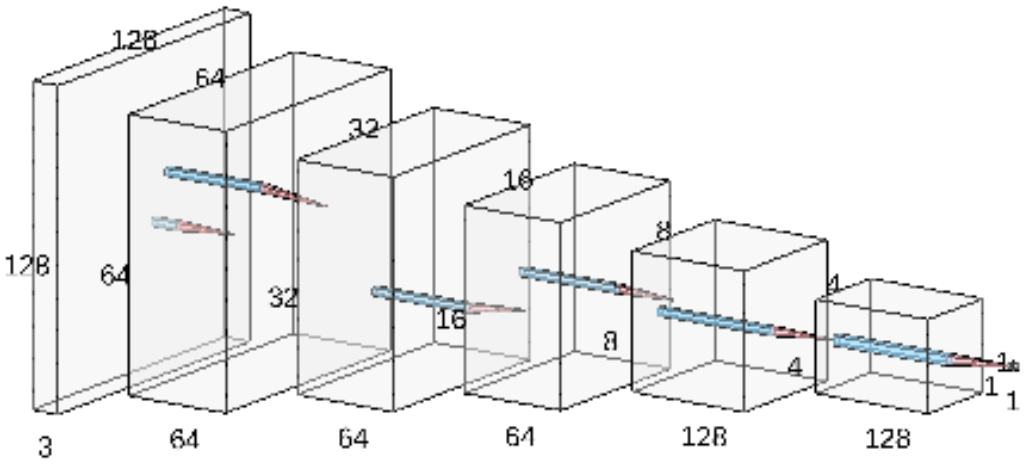

Figure 3.4: Network architecture for Discriminator Network

CHAPTER 4

EXPERIMENTS AND RESULTS

To demonstrate the effectiveness of both frameworks on real-world images, we will use and perform the experiments on CUB-200 dataset(7). In order to show the performce of the data augmentation techniques, we will train ResNet-18 model to classify this CUB-200 data with and without data augmentation techniques.

4.1 CUB-200 dataset

Caltech-UCSD Birds 200 (CUB-200) dataset(7) is a collection of approximately 12000 RGB images of birds of 200 species. Each class or species contains 30 images for training and 30 for testing. It also contains annotations such as segmentation masks, bounding box and attributes. In the following section we will simply extend the method proposed by (1) on this dataset.

4.2 Experiments with framework 1

In this section we will demonstrate the effectiveness of the data augmentation technique proposed in Framework 1 and compare it with conventional data augmentation techniques such as mirroring, random cropping, color variations and random rotations. As discussed earlier, in this framework, we treat σ^2 as hyper parameter. The maximum value we can sample from $N(0, 1)$ will be greater than 3 which will be too high for scaling and doesn't look realistic. So we will choose $N(0, 0.3)$ normal

Dataset	σ_{hs}^2	σ_{vs}^2	σ_r^2	Test Accurcay
CUB-200	-	-	-	64.79
CUB-200+ Framework 1*	0.2	0.2	20	67.40
CUB-200+ Framework 1*	0.3	0.3	30	67.52
CUB-200+ Framework 1*	0.4	0.4	40	67.22

Table 4.1: Test accuracies on ResNet-18 when trained using different datasets(with and without augmentation)
*:Augmented each image in the training data twice with Framework 1 i.e., with σ as hyper parameters.

distribution as the maximum value we can sample will be closer to 1. We have varied $\sigma^2(\sigma_r^2\sigma_{hs}^2\sigma_{vs}^2)$ values as shown in table 4.1 and (0.3,0.3,30) values for σ^2 have shown better accuracy. We use these values for all other experiments related to Framework 1.

We will first take entire training data i.e., 30 images per class out of 30 training images per class and train the ResNet-18 model first without any data augmentation. We will evaluate the trained ResNet-18 model on testset consisting of 30 images per class. Now we will use our technique to see the improvement. Augment each image in the training set twice so that we will have 90 images per class after augmentation. Now train the model with the original (30 per class training data) and the augmented data(60 per class). Form the Fig 4.1, we can clearly see the improvement in performance due to this data augmentation technique.

We will now perform the same experiment as above but now we will take only 20 images per class for training i.e., with 2/3 of the training data. We do the experiment with and with out the data augmentation. From Fig 4.2 we can clearly notice the effectiveness of this technique.

Now we will repeat the experiment but with only 10 images per class i.e., with only 1/3 of the training data. Again from Fig 4.3 we can show that this data augmentation technique gives better accuracy.

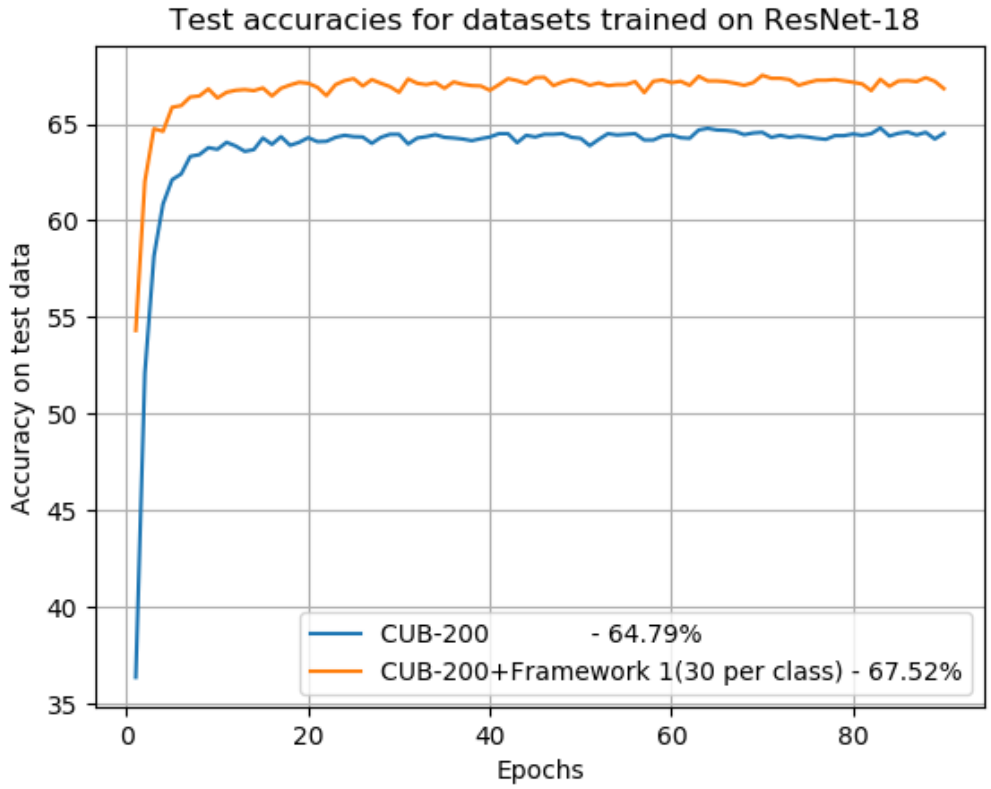

Figure 4.1: Comparision of test accuracies when trained on ResNet-18 between original CUB-200 dataset and original dataset with each training image(30 per class) augmented twice using Framework 1

We will now compare this data augmentation technique with conventional data augmentation techniques. We will first train the ResNet-18 model with conventional data augmentation techniques namely, random rotations, random cropping, mirroring. Clearly from the Fig 4.4, conventional data augmentation methods outperforms our data augmentation technique. Also, when we combine our method with conventional methods we see a slight dip in accuracy. One possible reason could be that conventional data augmentation methods are online i.e., generated during training where as our method is offline i.e., we generate the augmented data say two times the dataset before any training. Also, there are high chances of not noticing significant deformation as we are sampling our action from a $N(0, \sigma^2)$ and values closer to 0 are more probable.

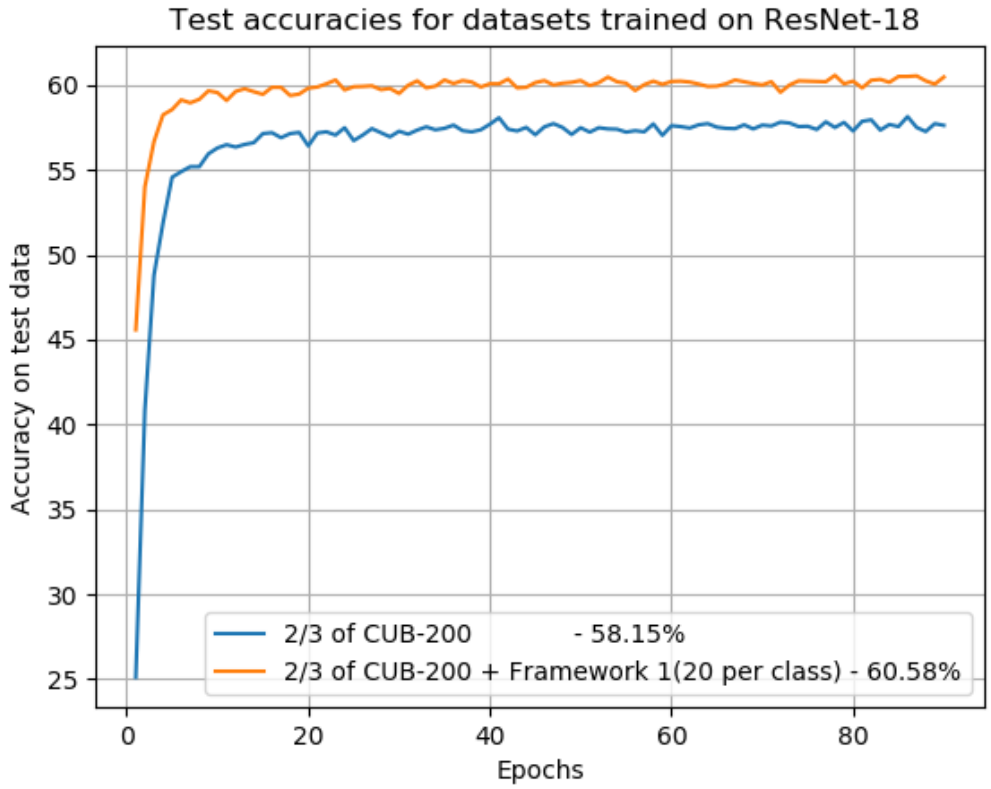

Figure 4.2: Comparision of test accuracies when trained on ResNet-18 between 2/3rd of original CUB-200 dataset and 2/3rd original dataset with each training image(20 per class) augmented twice using Framework 1

Some of the augmented images with framework 1 are shown in Fig 4.5.

4.3 Experiments with framework 2

In this section we will compare the effectiveness of both frameworks. From the previous section it is evident that Framework 1 improved accuracy by a significant amount.

We will train the Suggestion network F as described in section 3.2. Now for every image from the original CUB-200 training data we randomly select a part. We pass this object and part through the suggestion network and get σ. We will then sample

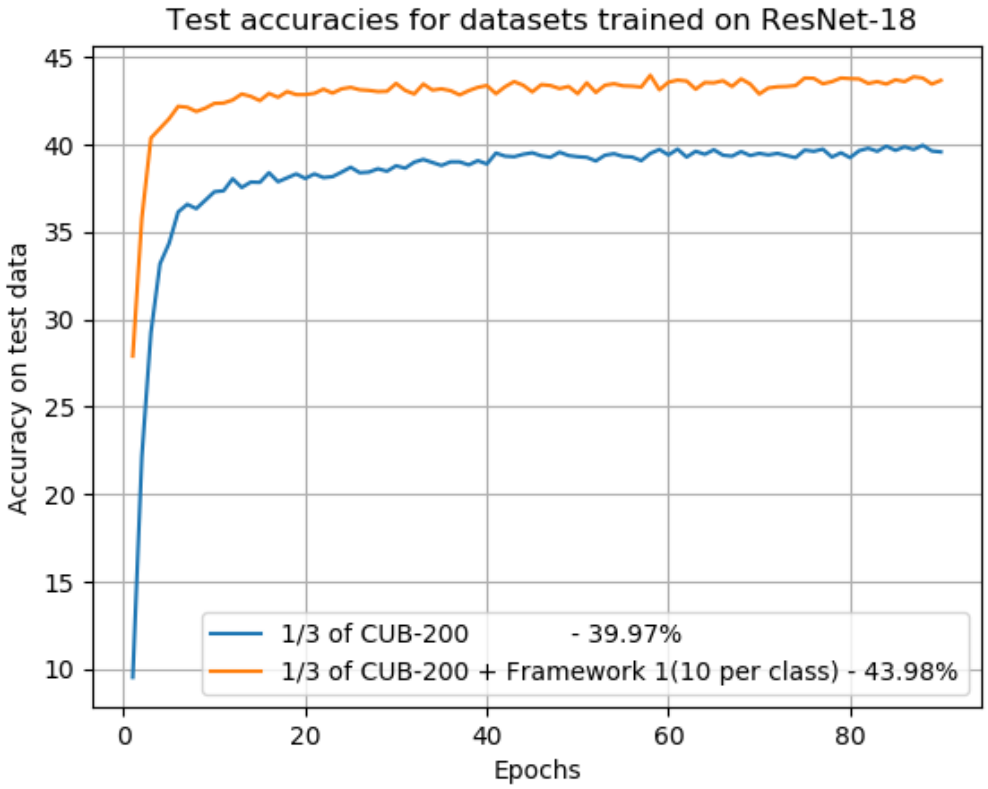

Figure 4.3: Comparision of test accuracies when trained on ResNet-18 between 1/3rd of original CUB-200 dataset and 1/3rd original dataset with each training image(10 per class) augmented twice using Framework 1

the amount of deformation from this σ and deform it with the deformation module. We do this for entire training dataset twice. We see from Fig 4.6 that conventional data augmentation methods outperforms this method. But still it is better than accuracy when trained with out any data augmentation. Less accuracy compared to conventional methods can be attributed to its offline nature and having sampling the deformations from a Gaussian which has a more probability for close to 0 values as explained in previous section.

Also, it is slightly better than framework 1 as can be seen in Fig 4.7. Comparing these two frameworks is a bit uncertain as the accuracy depends on the augmented data which in turn depends on sampling the amount of deformations. For example framework 1, when experimented again without any changes gave slightly better

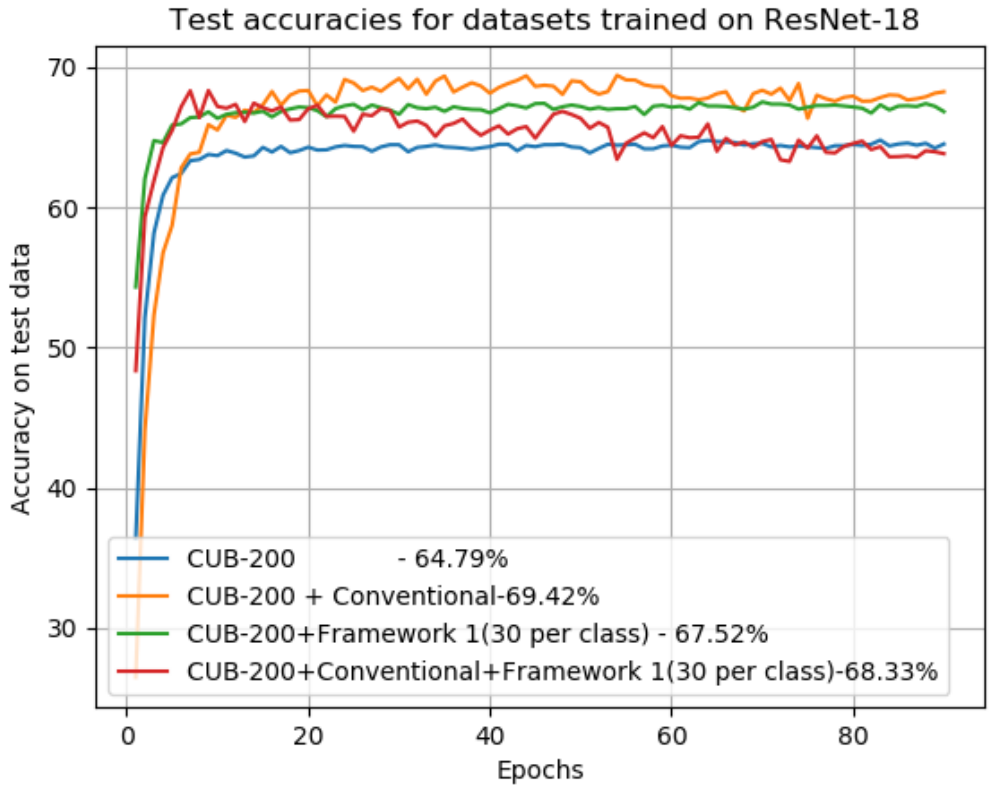

Figure 4.4: Comparision of test accuracies when trained on ResNet-18 between conventional data augmentation methods and Framework 1
Blue: Original CUB-200 without any data augmentation
Orange: Original CUB-200 + data augmented with rotations, cropping, mirroring
Green: Original CUB-200 + data augmented twice using Framework 1
Red: Original CUB-200 + Framework 1 + data augmented with rotations, cropping, mirroring

result(0.1% more) than Framework 2.

Some of the augmented images with framework 2 are shown in Fig 4.8.

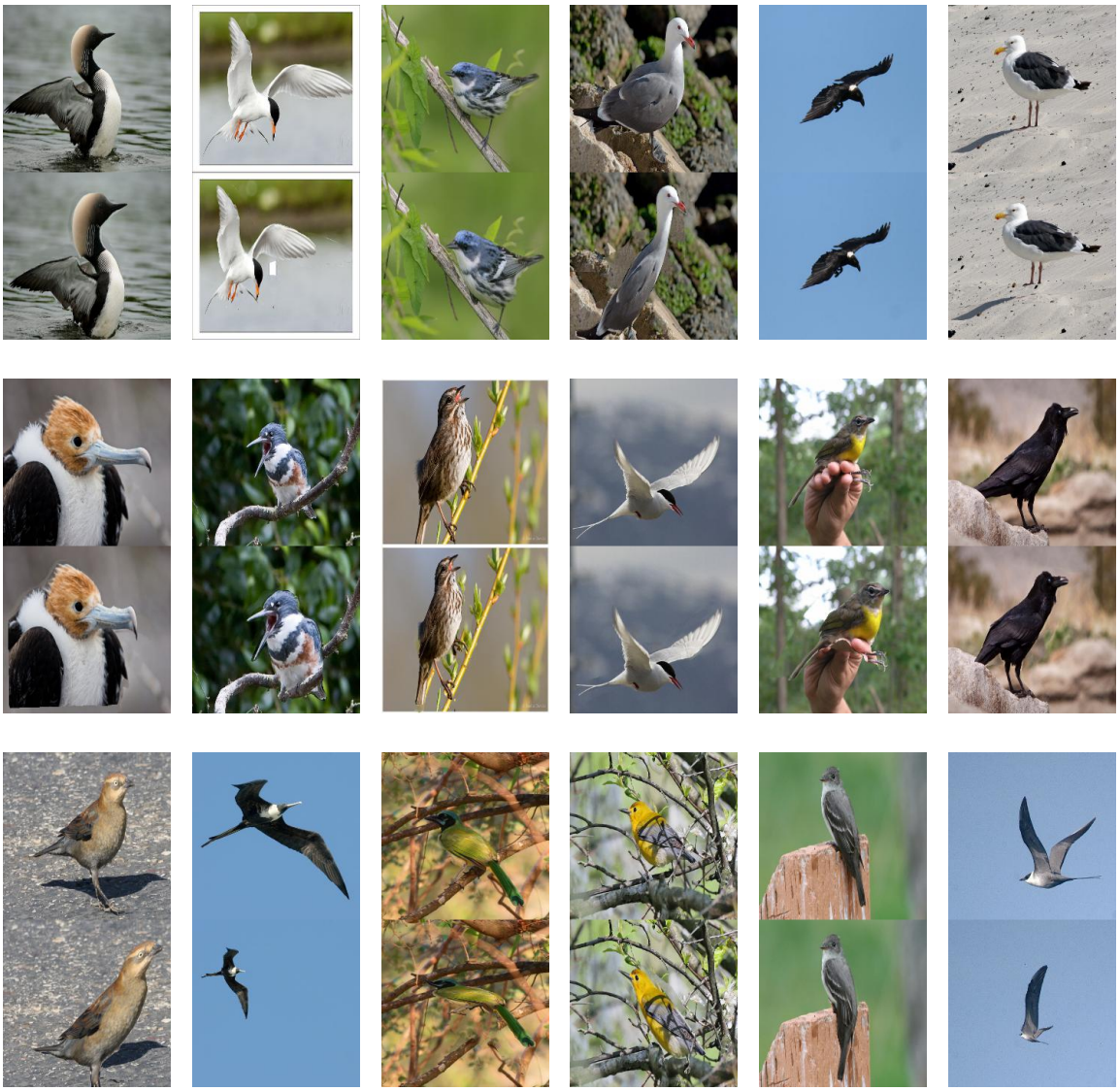

Figure 4.5: Augmented images using framework 1 with original images at top and the deformed images at bottom

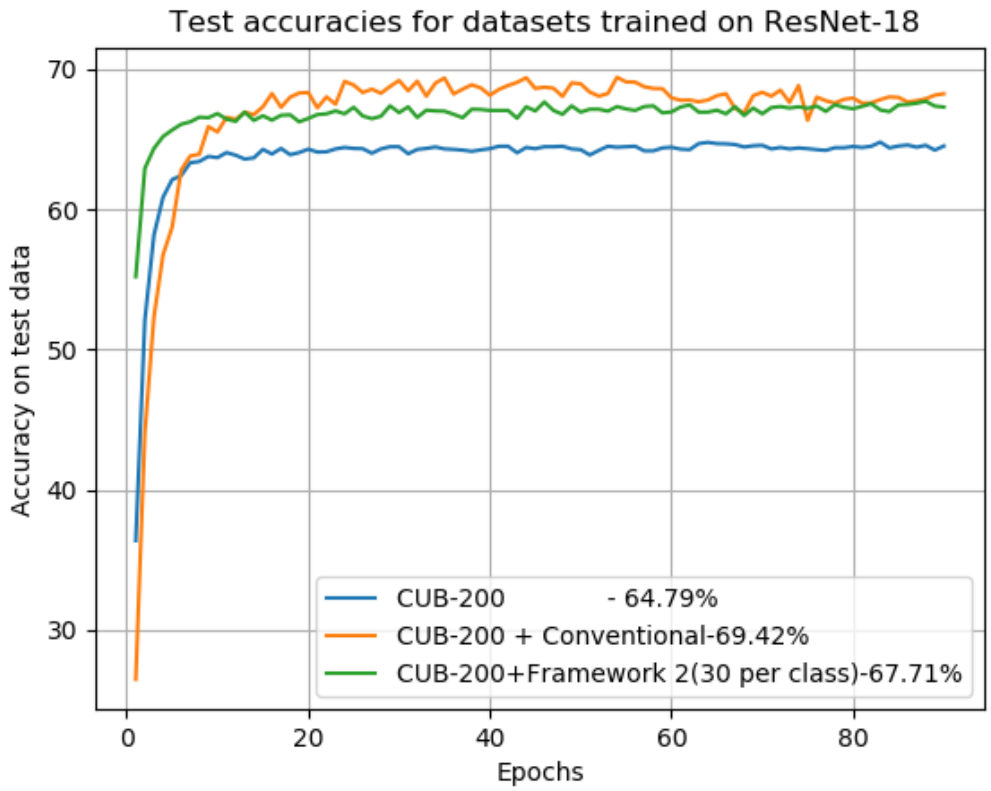

Figure 4.6: Comparision of test accuracies when trained on ResNet-18 between conventional data augmentation methods and Framework 2

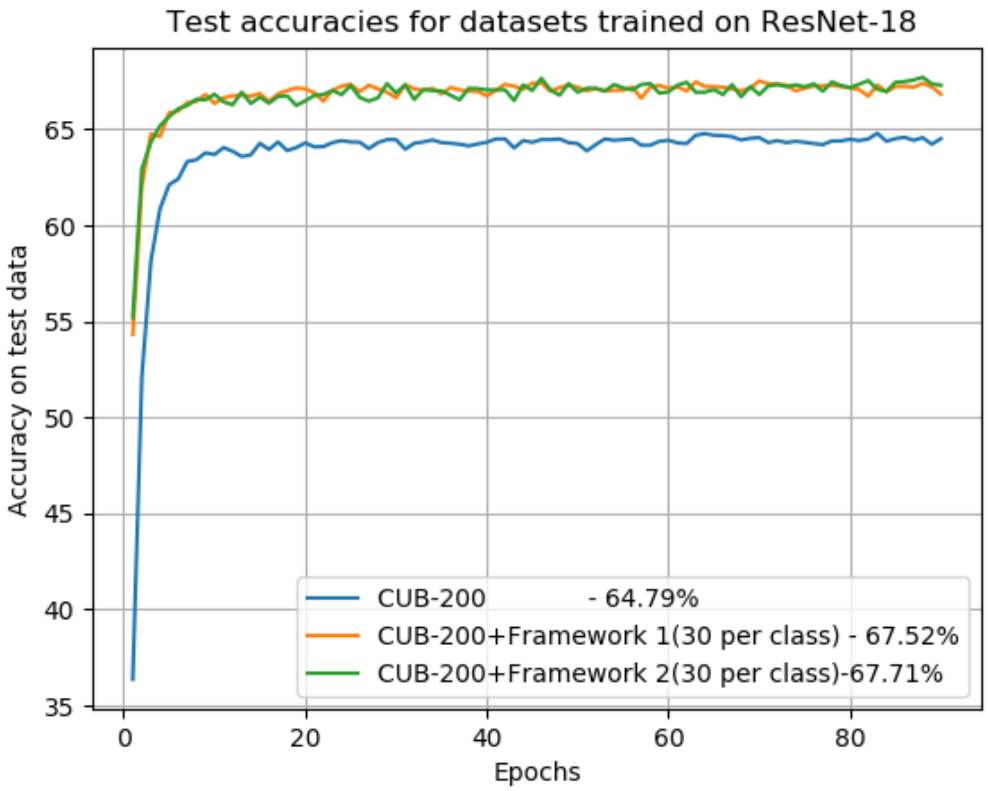

Figure 4.7: Comparision of test accuracies when trained on ResNet-18 between Framework 1 and Framework 2

Figure 4.8: Augmented images using framework 2 with original images at top and the deformed images at bottom

CHAPTER 5

CONCLUSION AND FUTURE WORKS

In this work, we have extended (1) to real-world images which gives us a new data augmentation technique. This technique actually improved the accuracy though it didn't outperform other conventional data augmentation techniques due to various reasons. Also, in order to deform a part we have learnt the maximum amount of deformation we can perform with the reinforcement learning and adversarial based algorithm. Thus we have a potential automatic data augmentation method and have shown its effectiveness on CUB-200 dataset.

We will extend this technique to other datasets which have segmentation masks and bounding boxes and show its effectiveness on them. Also, we would like to further improve framework 2 as it hasn't shown much improvement compared to framework 1 and conventional methods. Finally, this technique has a potential to extend to other computer vison tasks such as human pose estimation.

REFERENCES

[1] Vismay Patel, Niranjan Mujumdar, Prashanth Balasubramanian, Smit Marvaniya, Anurag Mittal *Data Augmentation Using Part Analysis for Shape Classification.* 2019 IEEE Winter Conference on Applications of Computer Vision (WACV) 1223-1232 (2019).

[2] T. Igarashi, T. Moscovich, J. F. Hughes *As-rigid-as-possible shape manipulation.* ACM transactions on Graphics (TOG), ACM, vol. 24, pp. 1134-1141 (2005).

[3] J. Zhao, L. Xiong, J. Li, J. Xing, S. Yan, and J. Feng *3D-Aided Dual-Agent GANs for Unconstrained Face Recognition.* IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume: 41, Issue: 10 (Oct. 1, 2019).

[4] X. Bai, W. Liu, and Z. Tu . *Integrating contour and skeleton for shape classification.* In Computer Vision Workshop (ICCV Workshops), 2009 IEEE 12th International Conference on, pages 360–367 (2009).

[5] L. J. Latecki, R. Lakamper, and T. Eckhardt *Shape descriptors for non-rigid shapes with a single closed contour.* In Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on volume 1, pages 424–429 (2000).

[6] Ian J. Goodfellow and Jean Pouget-Abadie and Mehdi Mirza and Bing Xu and David Warde-Farley and Sherjil Ozair and Aaron Courville and Yoshua Bengio *Generative Adversarial Networks.* arXiv 1406.2661 (2014).

[7] Wah C., Branson S., Welinder P., Perona P., Belongie S *The Caltech-UCSD Birds-200-2011 Dataset.* Computation and Neural Systems Technical Report CNS-TR-2011-001