IMAGE AUGMENTATION

Nikita Arora
Mca Scholar
DeptOf SCOPE
nikita.arora2019@vit.ac.in

Ramesh Ragala
Professor
Dept of SCOPE
ramesh.ragala@vit.ac.in

ABSTRACT

Data Augmentation defined as supplementing a data set with a data set that is created from information data set. It is very vast technique to improve the quality of modern image classifiers. The main example of data augmentation is image augmentation, whereas image augmentation specifies the parameters used for increasing the data sample count and perform various image transformations like zoom, shear, rotation, flipping etc. Many tools are used for image augmentation, such as imgaug, keras, smart augment, autoaugment etc. Our aim is to find the fastest and most flexible method for image augmentation and to add more transformation to the images.

1INTRODUCTION

Machine learning (ML) is very wide technologies now a days and requires a huge amount of data but unfortunately the amount of data is not sufficient to fulfill the need as required.In most settings, data is not available in sufficient quantities to avoid the counter fitting of the data set. The process of expanding the training sets by applying transformations as well as preserving the integrity of data is known as data augmentation. It has become a very powerful solution to the problem of data scarcity.

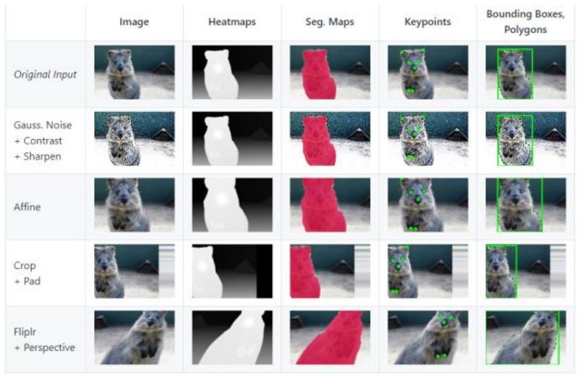
Data augmentation focuses on increasing the data by performing many operations on it and then trains the resultant data which is known as training data set then the training data set is used by many machine learning, deep learning and neural networks for their models. The goal of data augmentation is to increase the data and reduce the cost.

Similarly, when there is need of increasing the multimedia data and mainly the images, then it is known as image augmentation. Artificially Expanding the data set containing image as an input is basic working of image augmentation.By artificially expanding the data set we are trying to say that we are taking only one or two images as our input and then applying transformations on it which eventually results in increased data.

This is helpful when data set is given with very few data samples.Augmentation passes the generated information to the network and hence reduces the network loss. When dealing with images, it often includes the image transformations such as rotation, shearing, flipping, zoom, translation, blurring and many more modifications.

Image samples generated using image transformations genrally results in increase of existing data sample sets by nearly 3 to 4 times.

Figure 1.1 and Figure 1.2 are examples of image augmentation where only a single image is taken as input and then many operations are applied on the same image which results in the increased data set.



1.1 Converting a set of input images into a new much larger set of slightly altered images.

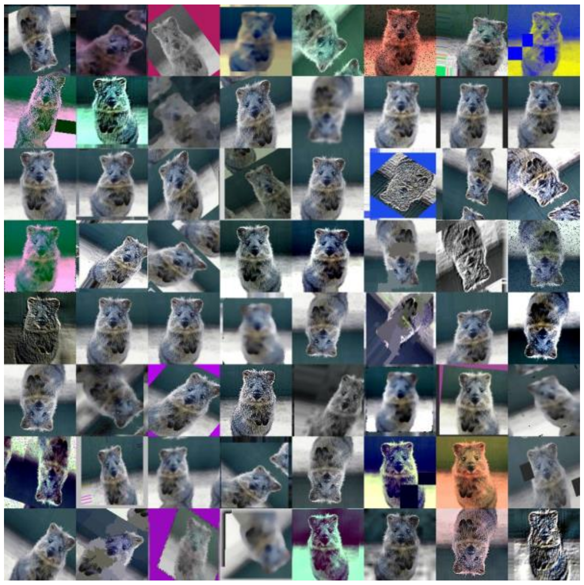


fig 1.2 Strong example augmentation of one input image

As the world is moving towards more technical world image augmentation has become need of the hour. But many times image augmentation leads to performance variability. For example,when the rotation is applied on 6 it becomes 9 but can't be distinguish by MNIST dataset.

2.PREVIOUS WORK

2.1 ALBUMENTATION

A very fast solution for image augmentation is Albumentation. Albumentation is python library contain 'n' no of transformation techniques. It is also known as easy to wrap library.Color , saturation , lightning and their combinations are used in wide range of computer vision task. The difference between original image and resultant image can be seen very easily according to the parameters taken. A powerful and very simple interface is provided by the albumentation library which is applicable for different tasks like classifications of images, segementing the images , detecting the images and many more.

Figure 1.3 demonstrates a task specific augmentation, where two transformations horizontal flip and cropping simultaneously.

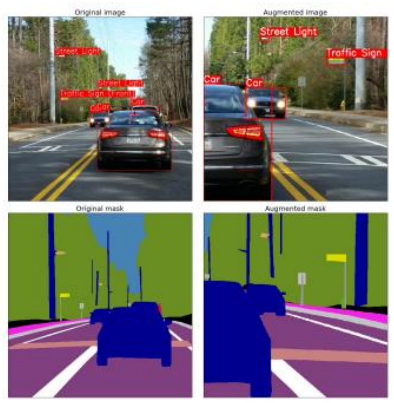


Fig 1.3 applying transformation simultaneously

Time taken by albumentation to perform various transformation is shown in table 1.1

Tranformations	Time taken
Randomcrop	0.0017
Verticalflip	0.178
Rotate	3.8538

Table 1.1 Time in seconds per image transformation

2.2 SMART AUGMENTATION

While training the deep neural networks the best strategy for image augmentation is chosen by Smart Augmentation. It does so by merging two or more data sets together and then perform some operations on it select the best one. Smart augmentation works for reducing the network loss by generating the training data set in a network. By using this the system is allowed to minimize the error for that network. All data sets are tested which results in increased accuracy by illustrating the specific measures.

Following are some steps followed by smart augmentation.

Step1: Data sets are combined.

Step 2: Combined data set gets trained.

Step3: Network loss is calculated.

Step4: 1-3 repeated for n no. of combinations.

Step 5: combined data set with minimum network loss is selected.

The above steps can also be understood by Figure 1.5

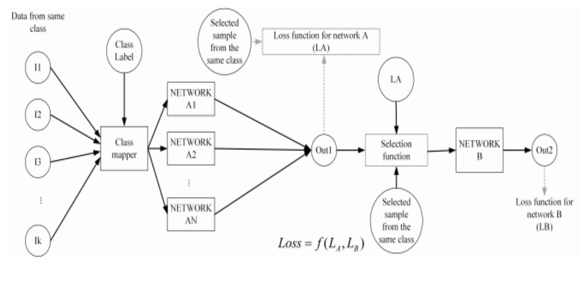


Fig1.5 showing the step by step working of smart augmentation

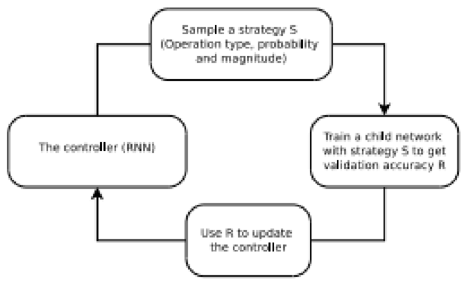
Smart Augmentation is best known for reducing the network loss.

2.3 AUTO AUGMENT

As the name suggests Auto augments works on the principle of finding the best strategy for augmentation automatically. It overcomes the traditional manual augmentation. A method auto augment is used to select the the strategy. It works on simple theory which divides the data set into smaller dataset and then searching for the desired image to augment it. The data set used for autoaugment is CIFAR-10.

The methodology on which auto augment works is that it divides the data set into mini policies and with each mini policies two operations are assigned which contains 1. Transformation to be performed, 2. Magnitude.Then search space algorithm is applied which searches for the best mini policy depending upon the accuracy it yields.

Figure 1.6 shows the working of autoaugment.


Sample a strategy S (Operation type, probability and magnitude)
Train a child network with strategy S to get validation accuracy R
Use R to update the controller
The controller (RNN)

2.4 IMGAUG

Imgaug is strongest library built in python for augmenting the images. It converts a set of input images into a new, much larger set of slightly altered images. This python library not only augment the images but also support bounding boxes ,heat maps and segmentation maps. Imgaug supports a huge number of augmentation technique moreover combining these techniques in any order is also possible.

Following code demonstrates the usage of imgaug, the output of code is also attached in fig 1.7-:

```
importimageio
importimgaug as ia
fromimgaug import augmenters as iaa
matplotlib inline
image=imageio.imread("
C:\Users\nikita\Desktop\rbl\image1.png")
print("Original:")
ia.imshow(image)
rotate = iaa.Affine(rotate=(-25, 25))
image_aug = rotate.augment_image(image)
print("Augmented:")
ia.imshow(image_aug)
```

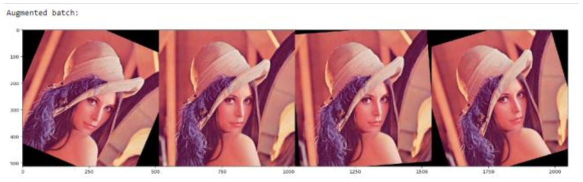


Fig 1.7 showing output of above code

The table 1.2 below shows the time taken by transformations it performs.

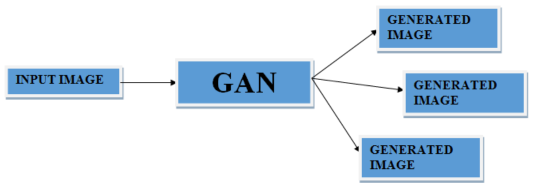
TRANSFORMATION	IMGAUG
HORIZONTAL FLIP	2.2299
VERTICAL FLIP	0.3899
ROTATE	16.16

Table 1.2 showing time in seconds to perform transformations.

3 METHODOLOGY

The GENERATIVE ADVERSIAL NETWORKS, which creates artificial instances of an image. This generative model is unsupervised which learns an underlying distribution.

The network takes an image as an input and converts into multiple number of images.



3.1MODULES

The model further consist of two networks, generator and discriminator, the generator model generator model takes one image or set of images as an input and then converts it into 'n' no. of images .

Now, the real images and generated images are fed into the discriminator model, which then discriminates the real and fake images.

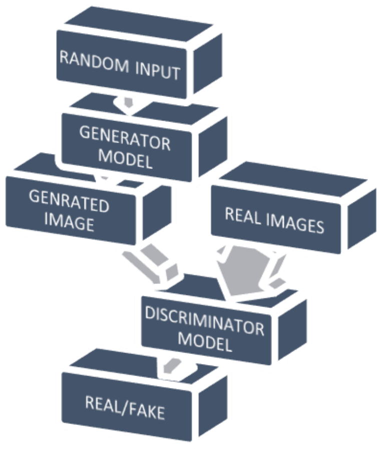


Fig1.8 Showing the architecture of GAN

The above diagram is representing the working of GAN model.

4 **ALGORITHM**

- Load the data from google drive.
- Reshape and resize the dataset.
- Import all the layers like dense, batch normalization , flatten.
- Define the constructor and intialize all the values like rows and columns in which resultant images must be displayed.
- Define the generator model and in the generator model add all the desired layers of the model and also add the noise to that model ,so that it can create artificial images .
- Now, define the discriminator model and in that model add all the desired layers so that it can tell the difference between artificial and fake images.
- Now the two models are combined,by defining the training function where we can define the training steps and batch size.
- Set a for loop from 0 to no. of images in the dataset.
- Feed the data onto generator model so that it can get trained.
- Feed the real images onto the discriminator model and also the fake images.
- Save the images and then display them.

5 **RESULTS**

The MNIST dataset was taken which contains images of handwritten numbers(0-9),the dataset contains total 70,000 images and the dataset is divided as follows-:

- Train dataset-: 55,000
- Validation dataset -: 5000
- Test dataset -:10,000

Now, the dataset is grouped into the epochs, every epoch is trained 10,000 times with the batch size of 256 image in each batch, means at a time 256 images are getting trained.

The total time taken for training is 5days.The images we get after 1332 iterations is

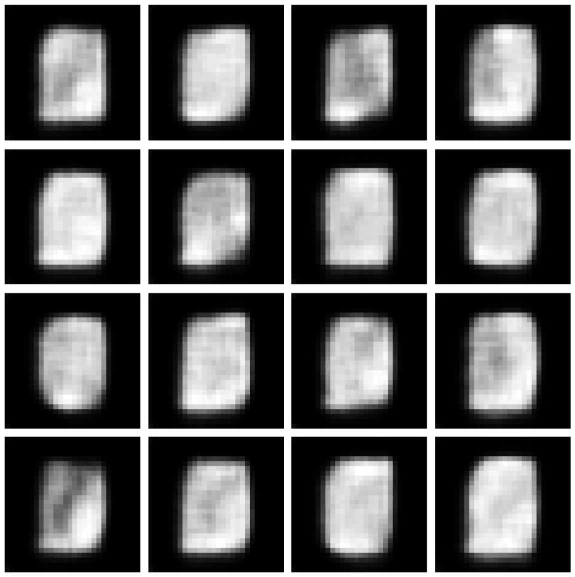


Fig 1.10 augmented after 1332 iterations

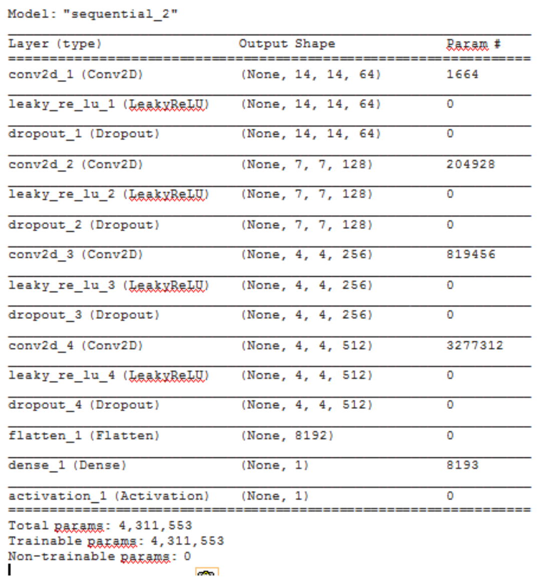
The images are not clear after 1332 iterations as there is noise in the resultant images which will be removed as the number of iteration increases. The more the no of iterations, the more clear the results are.

Figure 1.11 is the summary of the layers attached with the models we implemented.

The summary depicts the name of the model, the name of the layer, the output shape and parameters added for the better results.

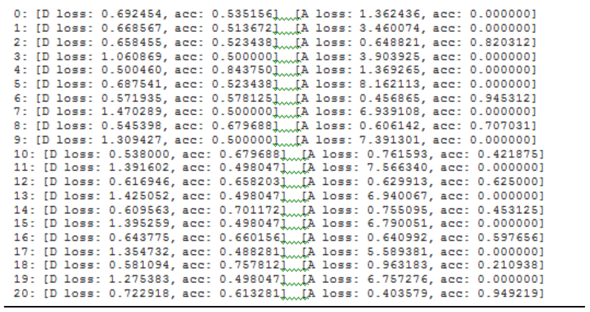
It also tells the total parameters, the trainable parameters and non trainable parameters.

Figure 1.12 shows the various stages when the data is getting trained.



```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 14, 14, 64)        1664
_________________________________________________________________
leaky_re_lu_1 (LeakyReLU)    (None, 14, 14, 64)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 14, 14, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 7, 7, 128)         204928
_________________________________________________________________
leaky_re_lu_2 (LeakyReLU)    (None, 7, 7, 128)         0
_________________________________________________________________
dropout_2 (Dropout)          (None, 7, 7, 128)         0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 4, 4, 256)         819456
_________________________________________________________________
leaky_re_lu_3 (LeakyReLU)    (None, 4, 4, 256)         0
_________________________________________________________________
dropout_3 (Dropout)          (None, 4, 4, 256)         0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 4, 4, 512)         3277312
_________________________________________________________________
leaky_re_lu_4 (LeakyReLU)    (None, 4, 4, 512)         0
_________________________________________________________________
dropout_4 (Dropout)          (None, 4, 4, 512)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 8192)              0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 8193
_________________________________________________________________
activation_1 (Activation)    (None, 1)                 0
=================================================================
Total params: 4,311,553
Trainable params: 4,311,553
Non-trainable params: 0
```

Fig 1.11 Summary of the layers



```
0: [D loss: 0.692454, acc: 0.535156]  [A loss: 1.362436, acc: 0.000000]
1: [D loss: 0.668567, acc: 0.513672]  [A loss: 3.460074, acc: 0.000000]
2: [D loss: 0.658455, acc: 0.523438]  [A loss: 0.648821, acc: 0.820312]
3: [D loss: 1.060869, acc: 0.500000]  [A loss: 3.903925, acc: 0.000000]
4: [D loss: 0.500460, acc: 0.843750]  [A loss: 1.369265, acc: 0.000000]
5: [D loss: 0.687541, acc: 0.523438]  [A loss: 8.162113, acc: 0.000000]
6: [D loss: 0.571935, acc: 0.578125]  [A loss: 0.456865, acc: 0.945312]
7: [D loss: 1.470289, acc: 0.500000]  [A loss: 6.939108, acc: 0.000000]
8: [D loss: 0.545398, acc: 0.679688]  [A loss: 0.606142, acc: 0.707031]
9: [D loss: 1.309427, acc: 0.500000]  [A loss: 7.391301, acc: 0.000000]
10: [D loss: 0.538000, acc: 0.679688]  [A loss: 0.761593, acc: 0.421875]
11: [D loss: 1.391602, acc: 0.498047]  [A loss: 7.566340, acc: 0.000000]
12: [D loss: 0.616946, acc: 0.658203]  [A loss: 0.629913, acc: 0.625000]
13: [D loss: 1.425052, acc: 0.498047]  [A loss: 6.940067, acc: 0.000000]
14: [D loss: 0.609563, acc: 0.701172]  [A loss: 0.755095, acc: 0.453125]
15: [D loss: 1.395259, acc: 0.498047]  [A loss: 6.790051, acc: 0.000000]
16: [D loss: 0.643775, acc: 0.660156]  [A loss: 0.640992, acc: 0.597656]
17: [D loss: 1.354732, acc: 0.488281]  [A loss: 5.589381, acc: 0.000000]
18: [D loss: 0.581094, acc: 0.757812]  [A loss: 0.963183, acc: 0.210938]
19: [D loss: 1.275383, acc: 0.498047]  [A loss: 6.757276, acc: 0.000000]
20: [D loss: 0.722918, acc: 0.613281]  [A loss: 0.403579, acc: 0.949219]
```

Fig1.12 Data is getting trained

6 CONCLUSION

So, far we have seen different methods for image augmentation like imgaug, auto augment and the GAN(which we have implemented).The generative adversial network is the most accurate method for image augmentation which results into the highest accurate generated images as the number of iterations are higher in this network and the time complexity is also lower. The generator model creates the batch of the images from the dataset taken and every batch contains 256 images and then every batch is augmented. Hence, we can conclude that the more number of iterations we apply the more clear images we can get.

As we have already seen fig 1.10 the augmented images are blur and not so good but as the number of iterations get increased the results become crystal clear and network loss decreases.

7 REFERENCES

1) Alexander Buslaev, Alex Prinov, Eugene Khvedchenya, Vladimir I. Iglovikov, Alexandar A Kalinin "Albumentation:fast and flexible image augmentations",arXivpreprint arXiv: 1809.06839v1,2018.

2) F. Chollet et al., "Keras," https://keras.io, 2015.

3) J. Lemley, S. Bazrafkan, and P. Corcoran, "Smart augmentation learning an optimal data augmentation strategy," IEEE Access, vol. 5, pp. 5858– 5869, 2017.

4) E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le, "Autoaugment: Learning augmentation policies from data," arXiv preprint arXiv:1805.09501, 2018.

5)A.Jung,"imgaug,"https://github.com/aleju/imgaug, 2017".

6)Maayan Frid-adar, Idit Diamant, Eyal Klang, Michal Amitai, Jacob Goldberger, and Hayit Greenspan "GAN-based Synthetic Medical Image Augmentation for increased CNN Performance in Liver Lesion Classification"

7)Hoo-Chang Shin, Neil A Tenenholtz , Jameson K Rogers, Christopher G Schwarz , Matthew L Senjem , Jeffrey L Gunter , Katherine Andriole , and Mark Michalski "Medical Image Synthesis for Data Augmentation and Anonymization using Generative Adversarial Networks" .

8)Hub.packtpub.com/generative-adversarial-networks-using-keras/.