Defective fruit classification using variations of GAN for augmentation

Prateek Durgapal
IIIT Allahabad, Prayagraj, Uttar Pradesh, India

***Abstract*—Identification and segregation of defective fruits from healthy ones is an important task in fruit processing industry. In this paper, we present a method for defective lemon fruit classification using different flavours of Generative Adversarial Networks (GANs) [1] and then using Transfer Learning. The algorithm begins with preprocessing the lemon images followed by data augmentation using GANs. This is followed by using pre-trained Convolutional Networks (CNNs) for the classification task and then fine tuning the model even further. Lemons quality control Dataset [2] was used as the base dataset for conducting all the experiments throughout this work.**

***Index Terms*—Fruit Quality Detection, GAN, DCGAN, cGAN, WGAN, WGAN-GP, Transfer Learning, CNN, ResNet, Data Augmentation**

I. Introduction

Fruit Quality Detection and Classification of defective fruits is an important task in the fruit packaging and fruit processing industry.The quality of a fruit from inside can be attributed from the fruit's surface. In almost all cases a fruit having surface defects/scratches is of poor quality. To be able to identify fruits having defective surfaces precisely and in lowest possible time is the biggest challenge.

In the Fruit Packaging/Processing industries this task of defective fruit classification was being done by manual labour which itself signifies how slow, inefficient, cost ineffective and prone to manual errors this process is. Also if even a single defective fruit is left unsegregated from a crate of good quality fruits, that single defective fruit is enough to spoil the whole crate in very less time. Hence, an automated system that could determine fruit quality by classifying it on basis of surface defects is need of the hour for enhancing fruit production, for reducing the dependency on manual labour, for reducing losses due to fruit quality deterioration.

Different fruits suffer from different types of defects. The size, color, positioning of defect on fruit cannot be ascertained easily and therefore traditional image processing methods will not provide satisfactory results. It is clear that a method which is found suitable to classify one fruit may not be good enough for some other fruit.

With the evolution of Deep Learning techniques, they are increasingly being applied to the tasks like Object Detection, Image Classification and Segmentation. Deep Convolutional Networks (DCNNs) allows the model to capture high quality features to improve accuracy and performance. DCNNs are being applied to surface defect detection in the past few years. Wang, Chenglong, and Zhifeng Xiao.l [3], for surface defect detection in lychee fruit, used three DCNN based pretrained models and then performed a comparitive analysis of the obtained results.

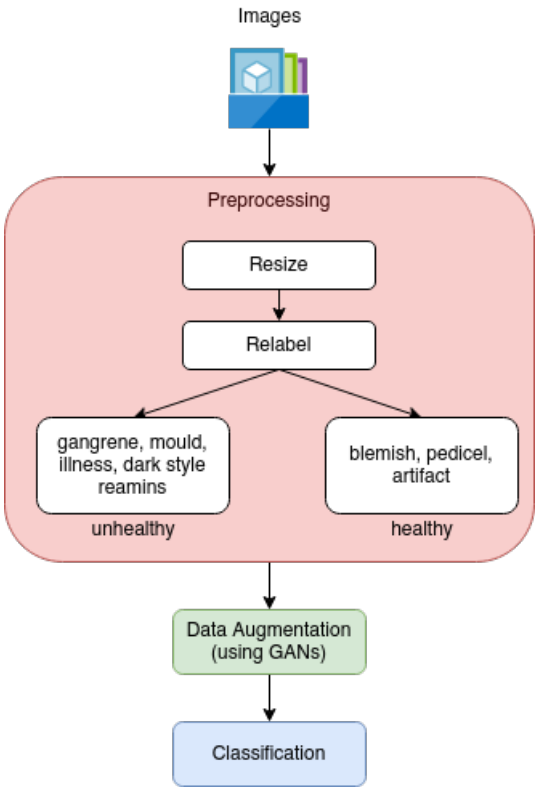

Fig. 1. Overview of the ML pipeline used with the Preprocessing step expanded; steps 2 and 3 have been illustrated further.

To counter fruit image data scarcity, augmentation to improve the dataset for better generalisation is required. Generative Adversarial Networks(GANs) can automatically discover and learn the regularities or patterns in input data in such a way that the model can be used to generate or output new examples/synthetic images. GANs work by framing the task as a supervised learning problem which consists of two sub models, one being the generator which is trained to generate fake samples while the other being the discriminator that classifies the generated sample to be real(already present in the training set) or fake(generated by the generator model). These two sub models are trained together in the form of a zero sum game, adversarial until the generator fools the discriminator

for majority of the samples.

In Conditional GANs, the output of generator can be controlled by specifying the class for which the synthetic image is to be generated. In Bird, Jordan J., et al. [4] the use of Conditional GAN improved the accuracy of their classification model from 83.77% to 88.75% signifying that the use of C-GAN improved the issue of data scarcity and distribution imbalance among different classes in the training dataset.

In this paper we propose to use GANs for adding images to our dataset and then feeding the combined images to a pretrained CNN model which would then be fine tuned for our specific task achieving an overall testing accuracy of 92.11%. The main contribution of the paper can be summarized as follows:

- Usefulness of GANs (and its variations) in data augmentation is shown.
- Transfer learning is used in the form of using ResNet as a pretrained CNN model.
- The state-of-the-art performances on the dataset is reported.

The remaining of the paper is organized as follows: Section II reviews the related work in the field of Defective Fruit Classification. Section III gives the details of the proposed method. Section IV presents the experiments and discussions. Finally, Section V gives the conclusions and future work.

II. Related Work

This section presents detailed review of previous methods used for defective fruit classification for different kind of fruits.

In [6], authors developed VGG16 - Convolutional neural network (CNN) [7] model for cultivar discrimination and fruit recognition for litchi. Authors in [4] worked on fruit quality and defect classification for lemons. Inspired from [6], they proposed VGG16 CNN model for classification. Preprocessing techniques were applied on the dataset which contains 2690 images of size 1056×1056 pixels. They achieved 83.77% accuracy on validation set with the VGG16 - CNN model. Using data augmentation with a Conditional Generative Adversarial Network for creating new synthetic fruit images, they were able achieve the classification accuracy of 88.75%. Convolutional Neural Networks have proved their significance in a lot of challenging problems.

Recently [3], authors worked on surface defect detection of lychee fruit. Lychee fruit is prone to surface damage during cultivation/ gathering times. The dataset which

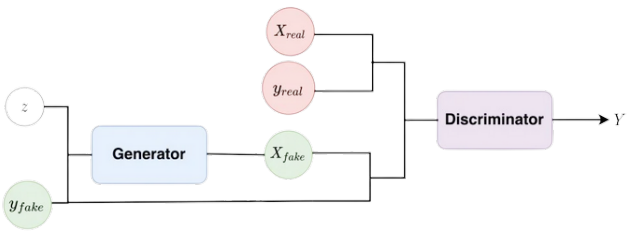

Fig. 2. The working of a conditional GAN [5].

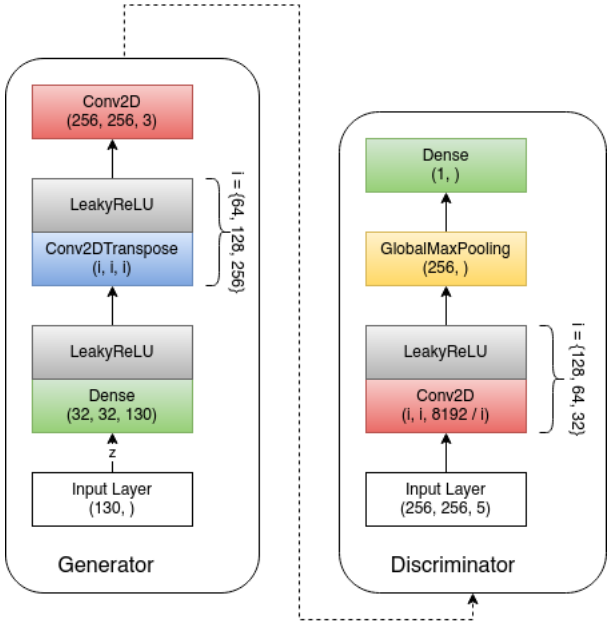

Fig. 3. Complete architecture of the GAN (and its variations) used

they used consisted of 3743 samples classified into 3 classes namely mature, defective and non-defective. Using transformer based Generative Adversarial Network (GAN), they removed distribution imbalance in the training set by introducing miscellaneous samples to rebalance the training examples corresponding to each of the three classes. For the classification task they used three pretrained Deep Convolutional Neural Network Architectures namely SSD - MobileNet V2, Faster RCNN - ResNet50 and Faster RCNN - Inception - ResNet V2 each of which was trained separately under specific settings and then their results were compared.

In [8], Authors proposed a CNN based model for defective apple classification. They achieved 96 % accuracy on the test set. 87% of classification accuracy was observed with the traditional Support Vector Machine (SVM) classifier based model. They used the offline trained CNN model to develop a software for online classification, with testing on 200 new apples images, they achieved 92% of classification accuracy.

In this work [9], The implementation of end to end defective fruit classification software on conveyor belt is described, The main part of the software comprises CNN model which is used for fruit recognition and defect classification, Tiny - YOLO is used to detect the fruit on the conveyor belt. The model was trained on 1915 images of six kind of images (apple, orange, lemon, defective apple, defective orange, defective lemon). The proposed model achieved 88 % accuracy on testset of 6000 fruit images.

In [10], [11], [12], Authors have proposed CNN based model with different architecture for fruit defect classification. Various image processing techniques like Gabor filter, GLCM etc were used in [10]. Artificial neural network (ANN) based model with self organising map (neuron structure can be

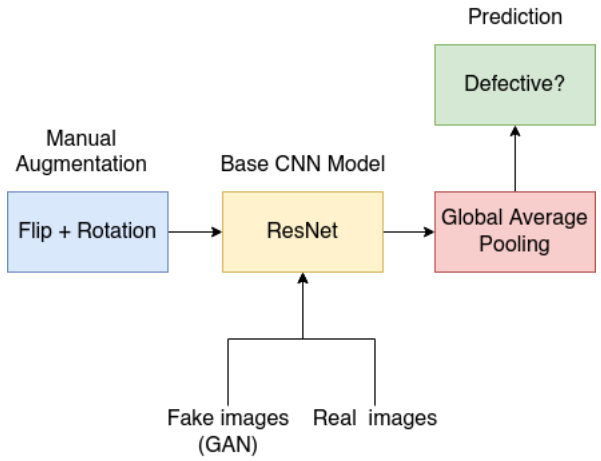

Fig. 4. Overall pipeline for classifying the images.

adjusted) is discussed in [13].

Now a days, deep learning have become very popular and gained a lot of attentions in different fields. Deep learning based techniques have been widely applied on fruit recognition and classification.

III. Proposed Method

As illustrated in the overview in Fig. 1, the proposed deep architecture consists of three components: input preprocessing, data augmentation and classification. We go over each of these one by one.

A. Input Preprocessing

The input images have a resolution of 1056 x 1056 px and have a count of 2690. Also, each picture contains some labels associated with it in COCO format. This means an image might have more than one label associated with it. For example, a lemon image might contain labels *gangrene, mould* and *pedicel*. The dataset also contains areas / segments where these labels occur. We decided to ignore this since it was unneeded for our task. Now we needed to segregate these images into *healthy* and *unhealthy* samples. For this, we chose the following labels for a lemon to be considered unhealthy – illness, gangrene, mould, dark style remains. If a lemon contains at least one of the preceding labels it was considered as unhealthy. Finally, after resizing each image to 256 x 256 px, we stored the images and corresponding labels locally for future use. Now, since the number of images was very less for neural networks which are data hungry, we decided to increase it by augmenting these.

B. Data Augmentation

For this specific subtask, we decided to use Generative Adversarial Networks (GANs). GANs consists of a generator and the discriminator. The job of the generator is to generate fake images that look like the input images / real images. The job of the discriminator, after being provided with both fake and real images, is to identify the fake images from the real ones. At equilibrium, the generator successfully manages to fool the discriminator which now has 50% chance to guess the label of the image.

TABLE I
Different conditions and their counts in the dataset

Conditon	Count
Blemish	2,048
Illness	1,743
Pedicel	1,245
Dark Style Remains	467
Artifact	451
Gangrene	449
Mould	264

GANs have been shown to work extremely well for generation of high quality images which are comparable to real images. Here we use, three variations of GAN – Conditional GAN (cGAN) [14], Wasserstein GAN (WGAN) [15], WGAN with Gradient Penalty (WGAN-GP) [16]. Each model has improvements over the previous models. But first a brief discussion of Deep Convolutional GAN (DCGAN) [17] follows.

1) DCGAN: It is a direct extension of GAN in which the the upsampling in the generator is carried out by transposed convolutional layers and the downsampling in the discriminator is carried out by convolutional layers. Since it is made up of many convolutional layers, it is known as Deep Convolutional GAN.

Fig. 3. shows a brief overview of the layers used in our implementation of DCGAN containing five layers in the Generator and four layers in the discriminator. The generator's input is a latent vector that is drawn from a standard normal distribution and outputs a fake image of size (256, 256, 3). The discriminator, on the other hand, outputs a scalar probability that the input is from the real data distribution. One of the inputs to the discriminator is the generator's output itself, hence the dotted line in the figure.

Note: We have used the same layers of DCGAN in all the subsequent GAN variations. Hence, technically, in the case of cGAN, it should be cDCGAN but we omit the extra 'DC' for the sake of readability.

2) cGAN: The first GAN that we used and implemented was a vanilla cGAN. GANs generate images unconditionally

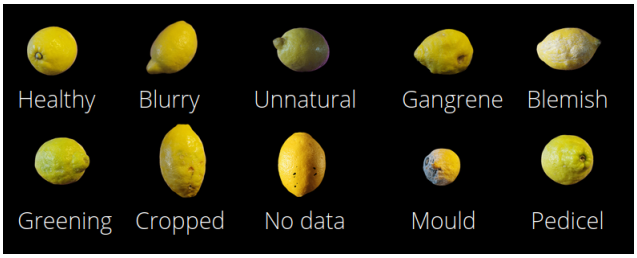

Fig. 5. An overview of various types of lemons present in the dataset along with their labels.

TABLE II
GANs AND THEIR LOSSES AFTER 1000 EPOCHS

GAN	G Loss	D Loss
cGAN	10.21	0.26
WGAN	6.32	0.10
WGAN-GP	3.11	0.02

i.e. we do not have control over the class of the image we want to generate. In our case since we need to augment in a fixed proportion, we need some way of conditionally generating images taking the help of some auxiliary data (class labels in our case). Hence cGAN is used. Also, since the random distribution that the fake images follow will have some pattern (due to the labels being added), convergence will be faster.

Fig. 2 shows the working and architecture of a general cGAN. Here, the input to the generator is not only a latent vector but also the class label which is concatenated with the vector to achieve a new latent vector. Now using this vector, we generate fake images. Now, for the discriminator, while passing the real input images, the class label is concatenated with these images and then passed through the discriminator. As a result, the ideal model can learn multi-modal mapping from inputs to outputs by being fed with different contextual information.

Note: In all the subsequent GANs, the same cGAN is used internally. The only difference from here on is the change in loss function and not the architecture. Hence, technically, in the case of WGAN, it should have been cWGAN but we omit the 'c' for the sake of readability

3) WGAN: The second GAN that we used was WGAN which has some major improvements over vanilla cGANs. WGAN uses a metric known as the Wasserstein metric in the loss function which has a smoother gradient everywhere. WGAN learns no matter the generator is performing or not. Even if there is vanishing gradient problem in the case of GAN, the gradients are linear and smoother everywhere in the case of WGAN. In comparison to GAN, the only difference is in the discriminator (called critic here) which omits the sigmoid function. Also, instead of the probability, a scalar score is the output. This score can be interpreted as how real the input images are. The following is the weight update policy for WGAN:

$$w \leftarrow w + \alpha \cdot RMSProp(w, g_w)$$

$$w \leftarrow clip(w, -c, c)$$

We observe that after updating the weights using RMSProp optimizer in the first step, weight clipping is performed in the second step i.e. the weights must lie in the range $[-c, c]$. In practical implementation, this is achieved by changing the loss function from the usual binary cross-entropy loss to

$$mean(y_{pred} \cdot y_{true})$$

Here y_{true} is the real image and y_{pred} is its output through the discriminator. This means that this loss maybe negative. We need this loss to be as close to 0 as possible. A shift above or below 0 means increasing loss in this case.

4) WGAN-GP: The third and final GAN that we used was WGAN-GP. The difference between this and the preceding GAN is that instead of weight clipping, gradient penalty is used. The issues with clipping lies within the clipping parameter. If the clipping parameter is high, it would be harder to train the critic till optimality as it can take a long time for any of the weights to reach their limit. If the clipping parameter is low, this can lead to the problem of vanishing gradients when batch normalization is not used or the number of layers is large.

Gradient penalty enforces a constraint / penalizes the model such that the gradients of the critic's output with respect to the inputs have a norm of 1. The penalty coefficient, λ, is used to weigh the GP term. In the original paper, for all their experiments, the authors have set λ = 10.

In practical implementation, this can be achieved by the following

$$g \leftarrow \Delta(y_{inter}, y_{pred})$$

$$gp \leftarrow mean((norm(g) - 1)^2)$$

$$d_{loss} \leftarrow d_{loss} + \lambda \cdot gp$$

Here y_{inter} is the interpolated image and y_{pred} is its output through the discriminator. gp is the gradient penalty to be added. d_{loss} is the discriminator loss.

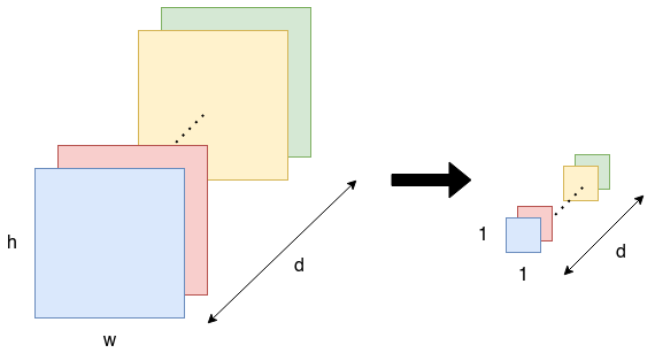

Fig. 6. The process of Global Average Pooling (GAP) for d channels with feature maps of size $h \times w$. After this operation, the size becomes $1 \times 1 \times d$.

C. Classification

Here we simply need to classify whether the given lemon image is defective or not. As illustrated in Fig. 4, the classification pipeline has 3 components. Each of the components is discussed one by one.

1) Manual Augmentation: Along with the GAN augmented images, we decided to combine manual augmentation as well. We are using Horizontal flips and rotations in the range $[\frac{-2\pi}{5}, \frac{2\pi}{5}]$

2) Base CNN Model: We experimented with different pre-trained CNN models – VGG16, Xception, DenseNet, EfficientNet and ResNet. The best results were being provided by ResNet. We removed the top layer and froze the layers of the remaining network for transfer learning. After training for 500 epochs, we unfroze the layers and trained again for 250 epochs to fine-tune the model and get an increase of 2-3% in accuracy of the model.

One thing to note here is that the input to the CNN would be the following – real input images, fake images generated from GAN, manually augmented images.

3) Global Average Pooling: Global Average Pooling is a pooling operation designed to replace fully connected layers in classical CNNs. As can be seen in Fig. 6, the output has the same depth with the feature maps replaced by their averages. This reduces the total number of dimensions and reduces overfitting even further.

Finally, for making predictions, the output of this layer is passed on to a single node to simply output 0 for unhealthy and 1 for healthy lemons.

IV. Experiments

In this section, the proposed method will be evaluated systematically on one public dataset which would be described briefly. And then, the training processes will be described in detail. Finally, the evaluation results will be reported respectively.

A. Dataset Description

Lemons quality control dataset [2] is an open source dataset consisting of 2690 annotated images of lemons, each of size 1056 x 1056 px. It has been prepared to investigate the possibilities to tackle the issue of fruit quality control. Table 1 shows the count of all the labels contained in the dataset.

B. GAN training

After shuffling the dataset and using a batch size of 64, we first trained the cGAN model with Adam optimizer and binary crossentropy loss. After 1000 epochs, Table 2 shows the losses for each of the GAN. We can see the general trend here. Both the generator and discriminator losses seem to go down with WGAN-GP performing the best with the lowest loss. This is the expected behaviour if we analyze the different GANs and their pros over the other as forementioned.

Fig. 7. shows some images generated by our trained WGAN-GP generator. Fig. 10 and Fig. 11 show the generator and discriminator losses for WGAN-GP during its training. If we take a look at Fig. 10 then indeed, the generator loss of 3.11 for the WGAN-GP is by far not the lowest observed but it is important to consider the nature of GANs; the losses of the two networks are relative to one another, i.e. it is an adversarial score. So, it does not infer that a lower generator loss implies higher quality images or more refined images. Also, considering Fig. 11, the loss is quite erratic but reaches

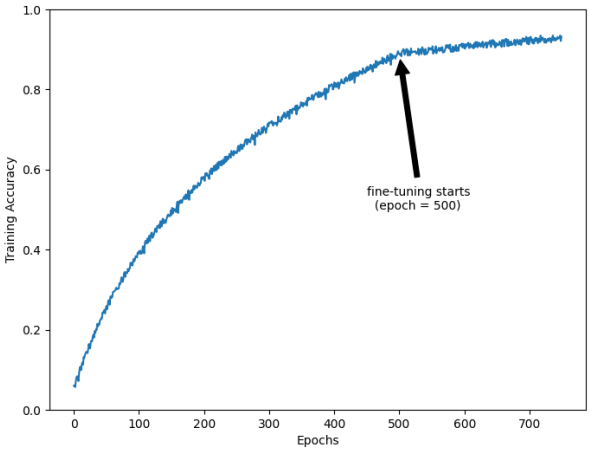

Fig. 8. Training Accuracy vs. Epochs for 2D CNN.

Fig. 7. GAN generated lemon images for both the classes.

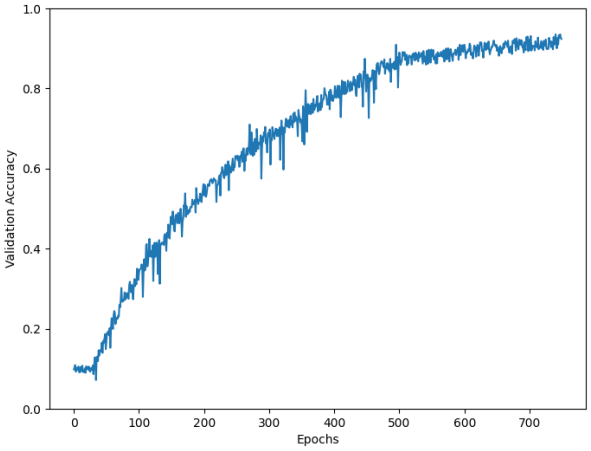

Fig. 9. Validation Accuracy vs Epochs for 2D CNN.

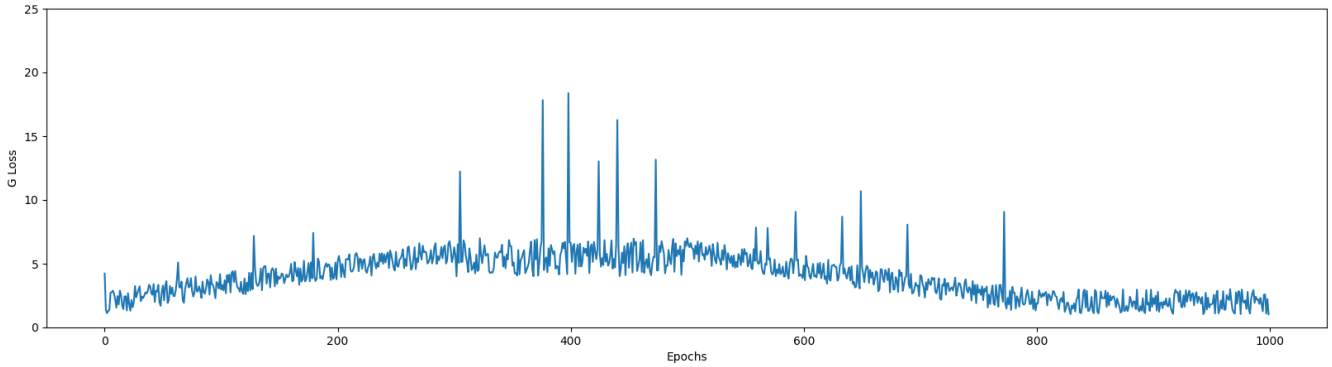

Fig. 10. Generator loss for 1000 epochs for WGAN-GP.

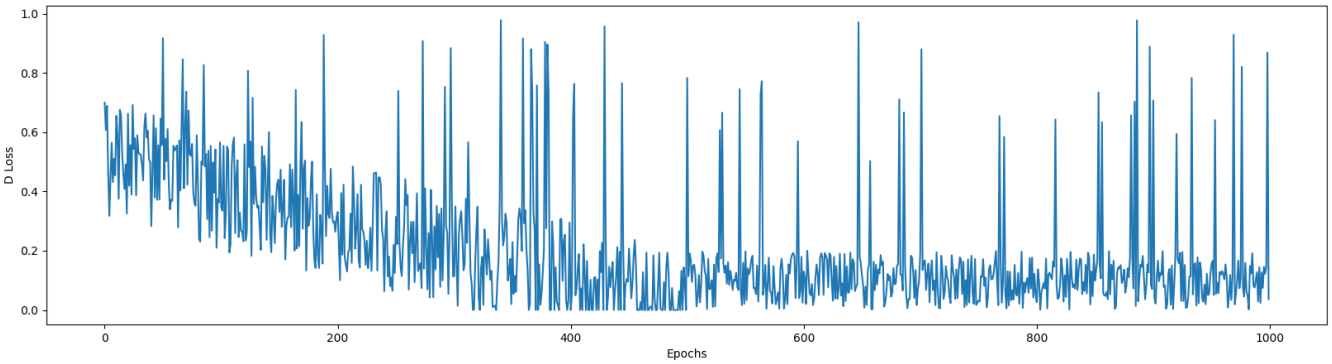

Fig. 11. Discriminator loss for 1000 epochs for WGAN-GP.

a low in the end.

GANs are often very hard to train and suffer from various problems such as mode collapse or inability to reach convergence. So, in order to make the GAN reach convergence faster the following techniques were used:

- Two Time-scale Update Rule: We choose different learning rates for the discriminator and generator. The crux is that if the generator changes slowly enough, then the discriminator still converges, since the generator perturbations are small. We choose a learning rate 0.0004 for the discriminator and 0.0001 for the generator.
- One sided label smoothing: Since, in most cases of mode collapse the discriminator loss goes to zero, we sometimes randomly flip some labels when feeding it into the discriminator. This acts like a kind of regularization.
- Discriminator steps: We always want the discriminator to be ahead of the generator. Otherwise if the discriminator is already calling generator's images as real, generator does not have any incentive to do any better. Hence we can train the discriminator more times per step. We train the discriminator 5 times in our case.
- Noise addition: Adding noise with decay over time to images for the discriminator makes training hard for it and so mode collapse could be avoided.

C. CNN Training

First the images (real, fake and manually augmented) are combined and then shuffled. Then the ResNet CNN was trained with the Adam optimizer with learning rate = 0.0001. Binary Cross-entropy was chosen as the loss function. As mentioned before, the model was trained for a total of 750 epochs – 500 for training and 250 for fine tuning. The training accuracy achieved was 94.23% while the testing accuracy was 92.11%. The results could be visualized in Fig. 8 and Fig. 9.

V. Conclusion

The use of GANs in augmenting data together with transfer learning and fine tuning allowed us to obtain good results on this classification task. We obtained testing classification accuracy of 92.11% in detecting defective lemons and training accuracy of 94.23%. This shows that our model and didn't overfit and has a good ability to generalize. We used a total of 1000 generated images for both the classes. Also, even though the generator loss is not that low, we still managed to achieve generate high quality lemon images.

REFERENCES

[1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[2] M. Adamiak, "Lemons quality control dataset," Jul. 2020. [Online]. Available: https://github.com/softwaremill/lemon-dataset

[3] C. Wang and Z. Xiao, "Lychee surface defect detection based on deep convolutional neural networks with gan-based data augmentation," *Agronomy*, vol. 11, no. 8, p. 1500, 2021.

[4] J. J. Bird, C. M. Barnes, L. J. Manso, A. Ekárt, and D. R. Faria, "Fruit quality and defect image classification with conditional gan data augmentation," *Scientia Horticulturae*, vol. 293, p. 110684, 2022.

[5] [Online]. Available: https://learnopencv.com/conditional-gan-cgan-in-pytorch-and-tensorflow/

[6] Y. Osako, H. Yamane, S.-Y. Lin, P.-A. Chen, and R. Tao, "Cultivar discrimination of litchi fruit images using deep learning," *Scientia Horticulturae*, vol. 269, p. 109360, 2020.

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

[8] S. Fan, J. Li, Y. Zhang, X. Tian, Q. Wang, X. He, C. Zhang, and W. Huang, "On line detection of defective apples using computer vision system combined with deep learning methods," *Journal of Food Engineering*, vol. 286, p. 110102, 2020.

[9] M.-C. Chen, Y.-T. Cheng, and C.-Y. Liu, "Implementation of a fruit quality classification application using an artificial intelligence algorithm," *Sensors and Materials*, vol. 34, no. 1, pp. 151–162, 2022.

[10] H. T. Aguiar and R. C. S. Vasconcelos, "Identification of external defects on fruits using deep learning," in *Pattern Recognition and Image Analysis*, A. J. Pinho, P. Georgieva, L. F. Teixeira, and J. A. Sánchez, Eds. Cham: Springer International Publishing, 2022, pp. 565–575.

[11] M. Nur Alam, S. Saugat, D. Santosh, M. I. Sarkar, and A. A. Al-Absi, "Apple defect detection based on deep convolutional neural network," in *Proceedings of International Conference on Smart Computing and Cyber Security*, P. K. Pattnaik, M. Sain, A. A. Al-Absi, and P. Kumar, Eds. Singapore: Springer Singapore, 2021, pp. 215–223.

[12] L. M. Azizah, S. F. Umayah, S. Riyadi, C. Damarjati, and N. A. Utama, "Deep learning implementation using convolutional neural network in mangosteen surface defect detection," in *2017 7th IEEE International Conference on Control System, Computing and Engineering (ICCSCE)*, 2017, pp. 242–246.

[13] T. Makkar, S. Verma, Yogesh, and A. K. Dubey, "Analysis and detection of fruit defect using neural network," in *Data Science and Analytics*, B. Panda, S. Sharma, and N. R. Roy, Eds. Singapore: Springer Singapore, 2018, pp. 554–567.

[14] M. Mirza and S. Osindero, "Conditional generative adversarial nets," 2014, cite arxiv:1411.1784. [Online]. Available: http://arxiv.org/abs/1411.1784

[15] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," 2017, cite arxiv:1701.07875. [Online]. Available: http://arxiv.org/abs/1701.07875

[16] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans," in *NIPS*, 2017.

[17] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *CoRR*, vol. abs/1511.06434, 2016.