Using GAN for Fashion

A Project Report

Submitted by:

Sneh Singh (191B262)

Vanama Yaswanth (191B278)

Vidhi Mathur(191B282)

Under the Guidance of: Dr. Ajay Kumar

In partial fulfilment for the award of the degree

of

BACHELOR OF TECHONOLOGY IN

COMPUTER SCIENCE AND ENGINEERING

At

Jaypee University of Engineering and Technology,

Guna, Madhya Pradesh ,473226

DECLARATION

We hereby declare that the work reported in 7th semester Major project entitled "Using GAN for Fashion", in partial fulfilment for the award of the degree of B.Tech. submitted at Jaypee University of Engineering and Technology, Guna, as per the best of our knowledge and belief there is no infringement of intellectual property rights and copyright. In case of any violation, we will solely be responsible.

Signature of Student

Sneh Singh (191B262)

Vanama Yaswanth (191B278)

Vidhi Mathur(191B282)

Department of Computer Science and Engineering,

Jaypee University of Engineering and Technology,

Guna ,473226

Date:

CERTIFICATE

This is to certify that the project titled "**Using GAN for Fashion**" is the bonafide work carried out by **Sneh Singh**, **Vanama Yaswanth** and **Vidhi Mathur**. We are students of B.Tech (CSE) at Jaypee University of Engineering and Technology Guna (MP) during the academic year 2019-2023 in partial fulfilment of the requirements for the award of the degree Of Bachelor of Technology (Computer Science and Engineering) and that the project has not formed the basis for the award previously of any other degree, diploma, fellowship or any other similar title.

Signature of Guide

Department of Computer Science and Engineering,
Jaypee University of Engineering and Technology,
Guna, 473226
Date:

ACKNOWLEDGEMENT

We take this opportunity to express our deep gratitude and most sincere thanks to our project supervisor Dr. Ajay Kumar and project coordinator Dr. Dinesh Kumar Verma for giving the most valuable suggestions, helpful guidance and encouragement in the execution of this project work. We would like to thank our supervisor and coordinator for guiding us. Last but not the least we are grateful to all the team members of “Using GAN for Fashion”.

Sneh Singh (191B262)

Vanama Yaswanth (191B278)

Vidhi Mathur (191B282)

Department of Computer Science and Engineering,
Jaypee University of Engineering and Technology,
Guna, 473226

Date:

Executive Summary

We propose a new framework for estimating generative models via an adversarial process, in which we simultaneously train two models: a generative model G that captures the data distribution, and a discriminative model D that estimates the probability that a sample came from the training data rather than G. The training procedure for G is to maximize the probability of D making a mistake. This framework corresponds to a minimax two-player game. In the space of arbitrary functions G and D, a unique solution exists, with G recovering the training data distribution and D equal to 1 2 everywhere. In the case where G and D are defined by multilayer perceptrons, the entire system can be trained with backpropagation. There is no need for any Markov chains or unrolled approximate inference networks during either training or generation of samples. Experiments demonstrate the potential of the framework through qualitative and quantitative evaluation of the generated samples.

In this project we introduce new methods for the improved training of generative adversarial networks (GANs) for image synthesis. We construct a variant of GANs employing label conditioning that results in 128 × 128 resolution image samples exhibiting global coherence. We expand on previous work for image quality assessment to provide two new analyses for assessing the discriminability and diversity of samples from class-conditional image synthesis models. These analyses demonstrate that high resolution samples provide class information not present in low resolution samples. Across 1000 ImageNet classes, 128 × 128 samples are more than twice as discriminable as artificially resized 32 × 32 samples. In addition, 84.7% of the classes have samples exhibiting diversity comparable to real ImageNet data.

Practical implications – we believe such personalized experience can increase the sales of fashion stores and here provide the feasibility of such a clothes generation system.

Originality/value – Applying GANs from the deep learning models for generating fashionable clothes

Keywords: Neural network, Cross-domain relations, Fashion clothes, Generative adversarial network.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Fashion is a constant presence in a person’s life but it is not consistent. People follow fashion to express themselves. One combines his clothes according to his own opinion and creates a unique style. As fashion trends change so frequently, some of them are very extreme and opposite to what a person likes. Recent advances in the artificial intelligence domain and especially in machine learning has led to tremendous collaborative works between the fashion and computer-based designs. In the textile and fashion world cloth generation systems that take images of latest fashion trend and the clothes purchased by the customers as input to produce new line of trends are highly desirable.

In this work, we study a deep learning based system, namely, generative adversarial networks (GANs) that utilizes the input information to generate new clothes, which are trendy and at the same time are derived from the user style. These can be also be used as a blueprint design by the manufacturers to make new clothes. Such a system will give a personalized shopping experience to the person. Generation of new and trendy clothes based on based on the current fashion trend by understanding the styling of the customer using GANs a new topic and to the best of our knowledge it has not been studied in this context. To generate such type of data the system must be able to relate the two different image domains. For this purpose, we need to create function, which maps images from one domain to the other, thus identifying the cross-domain relationships. This can be achieved using a GAN couple. This system will be supported by a relevance feedback in which the output is customized based on previous selections made by the user. If the user selects a particular image, then the weight all the images that have the same style as the selected image will be increased. This weight will be used to sort the generated images in order to their relevance to the user. We believe our system can have many applications. The same system can be employed to create new fashion clothes from old fashion clothes. This will need two types of images – one will be from latest fashion and the other will be from old fashion. This system can also be modified to work for clothes for different occasions as well. For this purpose, the data set needs to be updated accordingly. Such a system can also change the process of designing and manufacturing clothes. The designers can understand what people like and can use their creativity efficiently.

In the proposed adversarial nets framework, the generative model is pitted against an adversary: a discriminative model that learns to determine whether a sample is from the model distribution or the data distribution. The generative model can be thought of as analogous to a team of counterfeiters, trying to produce fake currency and use it without detection, while the discriminative model is analogous to the police, trying to detect the counterfeit currency. Competition in this game drives both teams to improve their methods until the counterfeits are indistiguishable from the genuine articles.

This framework can yield specific training algorithms for many kinds of model and optimization algorithm. In this article, we explore the special case when the generative model generates samples by passing random noise through a multilayer perceptron, and the discriminative model is also a multilayer perceptron. We refer to this special case as adversarial nets. In this case, we can train both models using only the highly successful backpropagation and dropout algorithms and sample from the generative model using only forward propagation. No approximate inference or Markov chains are necessary.

There are several metrics proposed for computing and comparing the results of our experiments. Some of the most popular metrics include: Precision,Recall,Accuracy,F1 score, True Rate and False alarm rate. A typical confusion table for our problem is given below along with illustration of how to compute our required metric.

Table 1- Confusion matrix content

	Machine says Yes	Machine says No
Human says Yes	T_p	F_n
Human says No	F_p	T_n

T_p =true positive T_n =true negative

F_n =false negative F_p =false positive

Precision $P = \mathrm{Tp} / (\mathrm{Tp} + \mathrm{Fp})$

Recall R= $Tp/(Tp + Fn)$

Accuracy A= $(Tp + Tn)/(Tp + Tn + Fp + Fn)$

2. Related Work

A GAN includes two interconnected systems. One system tries to learn patterns from the original data so that it can discriminate between what is original and what is fake, when a data item is presented to it after training. Therefore, this system can be thought of as a discriminator. The second network will try to generate the data which it similar to the original data, starting from some noise. This system is called a generator. The generated data will then be passed to the discriminator for the evaluation. The discrimination error from this system will be used by the generator to improve. Therefore, such a model forms the base of our system. However, the data generated in such a manner will be of the type on which the discriminator was trained. GANs have been used for previously (Zhu et al., 2017) to generate new clothes based on the description provided by the user and at the same visualize himself in those clothes. This research develops a system capable of learning the posture of the person and then dresses the person with this newly developed dress by GAN. This shows that GANs are capable enough to learn the complex clothing patterns. Nevertheless, the newly generated clothes will similar to what is in the market which means the latest trend or the data set used for the learning purpose. This means the system is capable of capturing a one domain or type of clothes. What we aim is to generate new clothes, capturing the ideas of two very different domains. Some researchers (Matzen et al., 2017) have tried to understand the most popular styles in different cities of the various countries around the globe. Therefore, people in different cities follow different fashion trends. This can form the third type of problem where the first domain includes fashionable and latest clothes and the second domain includes clothes popular in that region. The data set used by the researchers (Matzen et al., 2017) is based on the pictures taken from the social networking websites like Instagram. Such images are very noisy and are not consistent in nature. By noise and consistency, we mean that the background of the images is not same, lighting and the capturing angle is different in every picture and the picture captures different poses. This makes the learning process extremely difficult. There are many existing fashion based recommendation system that will recommend users new dresses that the user might like based on the previous shopping history of the user. The recommended item is similar to or complements the items that have already been purchased by the user or the items that the user is interested in (Gu, Liu, Cai and Shen, 2017). But these recommendation systems recommend already existing fashion styles or designs that are there in the database. This is where the novelty of our system comes in to generate new designs and patterns based on the trendiest items that are in fashion on a single piece of cloth. There are some existing systems that will generate new images for people in different poses based on the image of a person wearing clothes (Ma et al., 2017). These types of systems employ the technique of pose integration and image refinement based on the existing image to synthesize the image of the same cloth when looked from a different angle. So, no new designs are generated in these systems. Few existing systems such as Liu et al. (2016) can analyze the styles of the clothing that the user is interested in based on the feature extractions of the clothes such as visual features, distribution of styles, aesthetic word spaces, etc. They also combine the features of tops and bottoms. Some systems like Chen et al. (2015) extract the visual features such as the colors and the designs of the clothes to find out the similarities in the styles of different clothes. Chen et al. (2015) constructs a database of the clothes from the New York Fashion Shows and compare the trends. They elaborate on the current trends on fashion and changes in styles. But no new patterns or clothes are generated in these systems either. These modern styles and trends from different fashion shows have been incorporated in our system too. There are few systems like Gu, Wong, Peng, Shou, Chen and Kankanhalli (2017) which take into account the street fashion data and identify some of the long term and short term fashion styles. We used the styles which are long-term and have a perennial presence in the fashion industry for the generation of our data set so that the new images generated by our system will also include the patterns that are always in trend. Some proposed systems like He et al. (2016) check the similarity of trends and the pattern of the items that the user is interested in. These systems will not only compare clothes, they can also compare other items like shoes, jewelry, handbags, etc. These systems help us in gaining valuable insights into fashion and the current trends. Few models like Lassner et al. (2017) a semantic segmentation of the body and clothing is generated and then a conditional model is learned on the resulting segments that creates realistic images. But they are taking into account, the entire body structure, due to that the design generations and the details of the patterns are not clear, and hence they cannot be used by the manufacturers to make new clothes. Our system on other hand takes the cloth piece alone and generates new and clear design patterns. Some other systems like Xian et al. (2018) use the textures given by the user to synthesize objects consistent with the texture suggestions. They can also be applied on a wide variety of objects such as clothes, shoes, purses, etc. The details of the textures can be controlled using texture patches. Though this system will give user the ability to customize, but from the millions of designs, it is not possible to for the user to generate the best design or to search for the trendiest texture patch. This also poses a problem for cloth manufacturers as there is no parameter to decide which texture they should use for the generation of designs that will be in fashion for the longest time so that those designs can be used for mass production.

3. Proposed Work

The main objective of this project is to perform Classification of dress and generation of new dress. Our project consists of 2 modules, the first being the classification of clothes based on pattern on the dress and the second is the generation of garments.

4. Design and Implementation

To achieve this objective discussed above, following methodology is used:

- A thorough study of existing approaches and techniques in field of GAN.
- Collection of related data from Kaggle and Amazon web store.
- Pre-processing of data collected from Aws and Kaggle to fit for CCN Model.
- To build a classifier based on different supervised Deep learning techniques.
- Training and testing of built classifier using dataset collected from AWS.
- Checking the accuracy of classification and GAN and comparing results of each classifier and visualising the results
- Using GAN generate new design dresses.

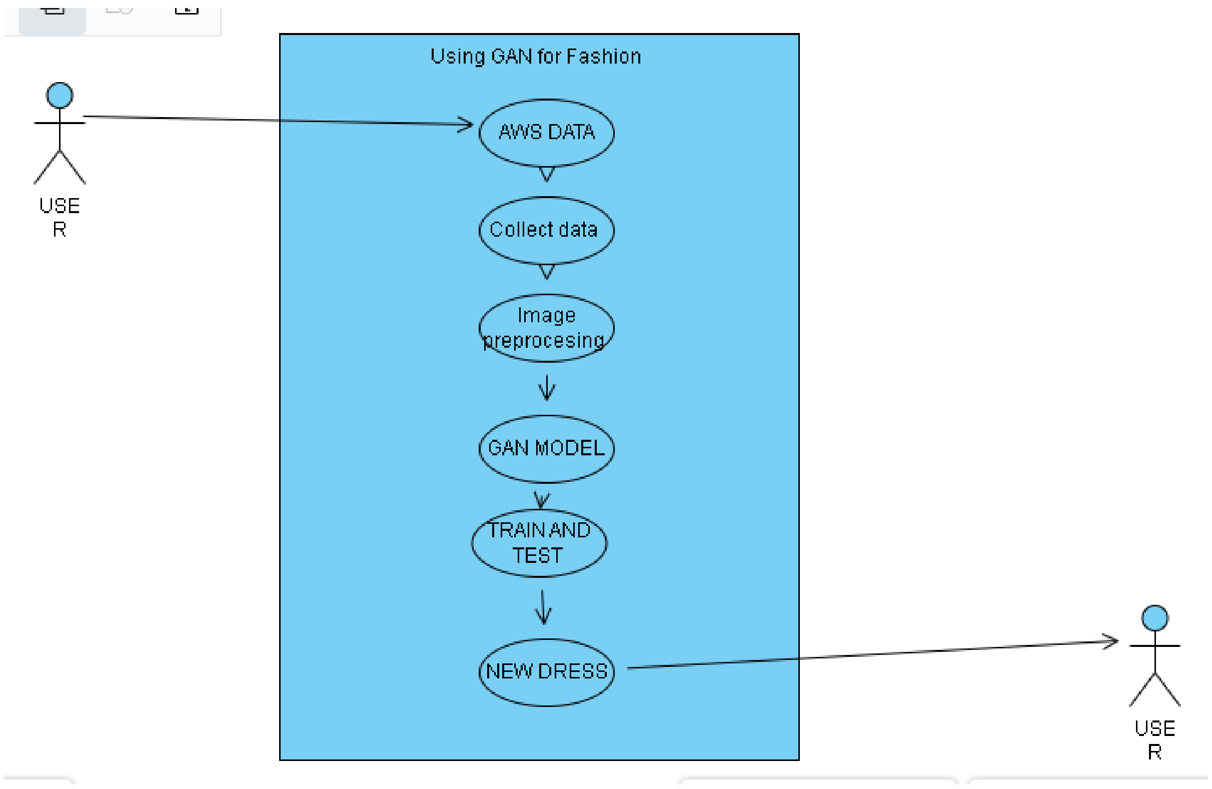


Figure 1: Use Case Diagram for the model

Tools/Technique Used

Tools used

- Jupyter Notebook/Colab
- Pycharm
- VS Code

Packages Used

- Tensorflow/Keras
- CV2
- Pytorch
- Scikit

Techniques Used

- Convolutional Neural Network
- VGG16
- ResNet
- Inception Resnet V2
- Stacked Ensemble model
- GAN
- DCGAN
- Color Detection using OpenCV

Generative Adversarial Nets

The adversarial modeling framework is most straightforward to apply when the models are both multilayer perceptrons. To learn the generator's distribution pg over data x, we define a prior on input noise variables pz(z), then represent a mapping to data space as G(z; θg), where G is a differentiable function represented by a multilayer perceptron with parameters θg. We also define a second multilayer perceptron D(x; θd) that outputs a single scalar. D(x) represents the probability that x came from the data rather than pg. We train D to maximize the probability of assigning the correct label to both training examples and samples from G. We simultaneously train G to minimize log(1 − D(G(z))): 2

In other words, D and G play the following two-player minimax game with value function

V (G, D): min G max D V (D, G) = Ex∼pdata(x) [log D(x)] + Ez∼pz(z) [log(1 − D(G(z)))].

In the next section, we present a theoretical analysis of adversarial nets, essentially showing that the training criterion allows one to recover the data generating distribution as G and D are given enough capacity, i.e., in the non-parametric limit. See Figure 1 for a less formal, more pedagogical explanation of the approach. In practice, we must implement the game using an iterative, numerical approach. Optimizing D to completion in the inner loop of training is computationally prohibitive, and on finite datasets would result in overfitting. Instead, we alternate between k steps of optimizing D and one step of optimizing G. This results in D being maintained near its optimal solution, so long as G changes slowly enough. This strategy is analogous to the way that SML/PCD [31, 29] training maintains samples from a Markov chain from one learning step to the next in order to avoid burning in a Markov chain as part of the inner loop of learning. The procedure is formally presented in Algorithm 1. In practice, equation 1 may not provide sufficient gradient for G to learn well. Early in learning, when G is poor, D can reject samples with high confidence because they are clearly different from the training data. In this case, log(1 − D(G(z))) saturates. Rather than training G to minimize log(1 − D(G(z))) we can train

G to maximize log D(G(z)). This objective function results in the same fixed point of the dynamics of G and D but provides much stronger gradients early in learning.

Theoretical Results

The generator G implicitly defines a probability distribution pg as the distribution of the samples G(z) obtained when z ∼ pz. Therefore, we would like Algorithm 1 to converge to a good estimator of pdata, if given enough capacity and training time. The results of this section are done in a nonparametric setting, e.g. we represent a model with infinite capacity by studying convergence in the space of probability density functions. We will show in section 4.1 that this minimax game has a global optimum for pg = pdata. We will then show in section 4.2 that Algorithm 1 optimizes Eq 1, thus obtaining the desired result.

```
for number of training iterations do for $k$
  steps do
    • Sample minibatch of $m$ noise samples $\{z^{(1)},...,z^{(m)}\}$ from noise prior $p_g(z)$.
    • Sample minibatch of $m$ examples $\{x^{(1)},...,x^{(m)}\}$ from data generating
      distribution $p_{\text{data}}(x)$.
    • Update the discriminator by ascending its stochastic gradient:
          $\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)\right]$.
  • Sample minibatch of $m$ noise samples $\{z^{(1)},...,z^{(m)}\}$ from noise prior $p_g(z)$.
  • Update the generator by descending its stochastic gradient:
  $\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)$.
```

Global Optimality of $p_g = p_{\text{data}}$

We first consider the optimal discriminator D for any given generator G. Proposition 1. *For G fixed, the optimal discriminator D is*

data(*x*) $D^*_G(\boldsymbol{x}) =$ p

(2) *pdata*(*x*) + *pg*(*x*)

Proof. The training criterion for the discriminator D, given any generator *G*, is to maximize the quantity $V(G,D)$

$$V(G,D) = \int_{\boldsymbol{x}} p_{\text{data}}(\boldsymbol{x}) \log(D(\boldsymbol{x}))dx + \int_z p_{\boldsymbol{z}}(\boldsymbol{z}) \log(1 - D(g(\boldsymbol{z})))dz$$
$$= \int_{\boldsymbol{x}} p_{\text{data}}(\boldsymbol{x}) \log(D(\boldsymbol{x})) + p_g(\boldsymbol{x}) \log(1 - D(\boldsymbol{x}))dx \quad (3)$$

For any $(a,b) \in \mathbb{R}^2 \setminus \{0,0\}$, the function $y \to a\log(y) + b\log(1-y)$ achieves its maximum in $[0,1]$ at $\frac{a}{a+b}$. The discriminator does not need to be defined outside of $Supp(p_{\text{data}}) \cup Supp(p_g)$, concluding the proof. □

Note that the training objective for D can be interpreted as maximizing the log-likelihood for estimating the conditional probability $P(Y = y|x)$, where Y indicates whether x comes from p_{data} (with $y = 1$) or from p_g (with $y = 0$). The minimax game in Eq. 1 can now be reformulated as:

$$C(G) = \max_D V(G,D)$$

$$= \mathbb{E}_{x\sim p_{\text{data}}}[\log D^*_G(x)] + \mathbb{E}_{z\sim p_z}[\log(1 - D^*_G(G(z)))] \quad (4)$$

$$= \mathbb{E}_{x\sim p_{\text{data}}}[\log D^*_G(x)] + \mathbb{E}_{x\sim p_g}[\log(1 - D^*_G(x))]$$

$$= \mathbb{E}_{x\sim p_{\text{data}}}\left[\log \frac{p_{\text{data}}(\boldsymbol{x})}{P_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})}\right] + \mathbb{E}_{\boldsymbol{x}\sim p_g}\left[\log \frac{p_g(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})}\right]$$

Theorem 1. *The global minimum of the virtual training criterion $C(G)$ is achieved if and only if $p_g = p_{data}$. At that point, $C(G)$ achieves the value* $-\log 4$.

Proof. For $p_g = p_{\text{data}}$, $D^*_G(\boldsymbol{x}) = \frac{1}{2}$, (consider Eq. 2). Hence, by inspecting Eq. 4 at $D^*_G(\boldsymbol{x}) = \frac{1}{2}$, we find $C(G) = \log\frac{1}{2} + \log\frac{1}{2} = -\log 4$. To see that this is the best possible value of $C(G)$, reached only for $p_g = p_{\text{data}}$, observe that

$$\mathbb{E}_{x\sim p_{\text{data}}}[-\log 2] + \mathbb{E}_{x\sim p_g}[-\log 2] = -\log 4$$

and that by subtracting this expression from $C(G) = V(D_G^*, G)$, we obtain:

$$C(G) = -\log(4) + KL\left(p_{\text{data}} \left\| \frac{p_{\text{data}} + p_g}{2}\right.\right) + KL\left(p_g \left\| \frac{p_{\text{data}} + p_g}{2}\right.\right) \tag{5}$$

where KL is the Kullback–Leibler divergence. We recognize in the previous expression the Jensen– Shannon divergence between the model's distribution and the data generating process:

$$C(G) = -\log(4) + 2 \cdot JSD(p_{\text{data}} \,\mathrm{k} p_g) \tag{6}$$

Since the Jensen–Shannon divergence between two distributions is always non-negative and zero only when they are equal, we have shown that $C^* = -\log(4)$ is the global minimum of $C(G)$ and that the only solution is $p_g = p_{\text{data}}$, i.e., the generative model perfectly replicating the data generating process. □

Convolutional Neural Networks

CNNs are widely used for image classification. Each image is passed through a series of convolutional layers with filters, MaxPooling, flattening and dense layer. The extraction of features and learning is done in the initial few layers

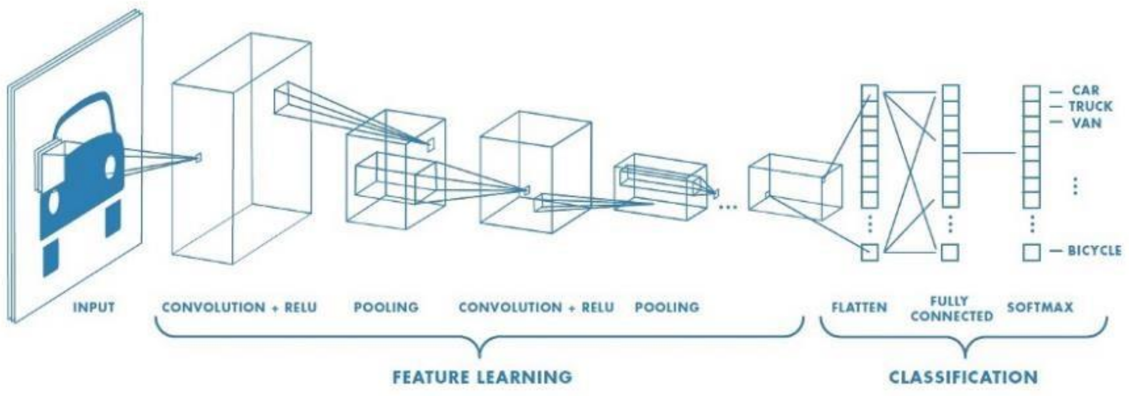


Figure 2 - Convolutional Neural Network

Resnet

A deep neural network is difficult to train because of the vanishing gradient problem and performance gets saturated or even starts degrading. A deeper network is ideally expected to work at least as good as a shallow network if we assume many layers have not learnt features, we could simply stack identity mappings. Researchers hypothesize that letting the stacked layers fit a residual mapping is easier than letting them directly fit the desired underlaying mapping. The residual block below explicitly allows it to do precisely that by introducing skip connections.

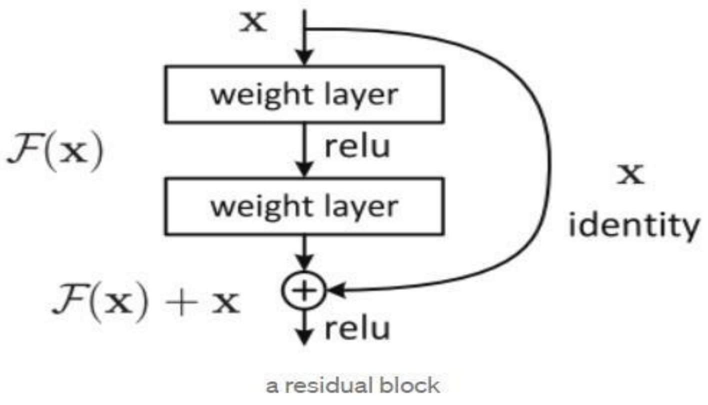


Figure 3 resnet model

Inception Resnet V2

Choosing right kernel size for convolution becomes tough, deep networks is prone to overfitting and randomly stacking convolution layers comes with a heavy cost. Hence the idea of inception net was born to utilize filters of multiple size operating on the same level, which drastically reduces the computation cost. Inception network combined with the concept of residual connections was introduced which helps in boosting the performance

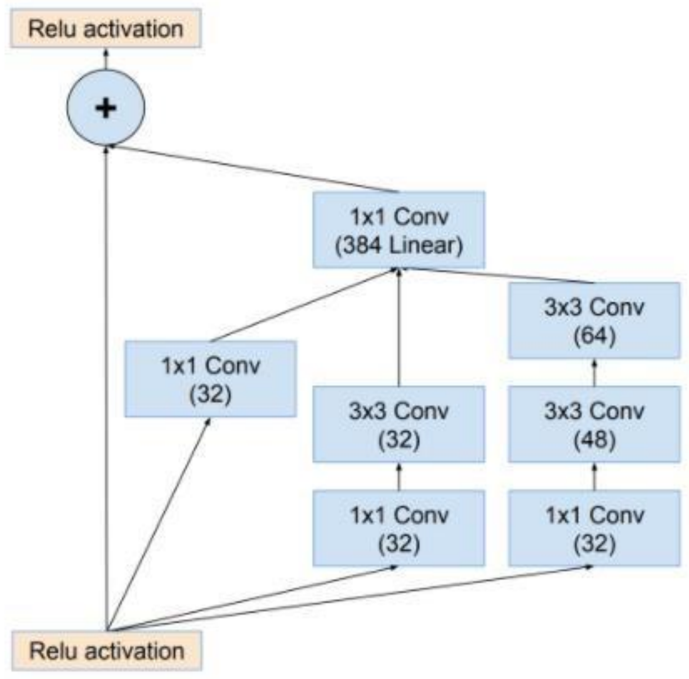


Figure 4 Inception ResNet – A

VGG net

The used VGG 16 is much deeper which consists of 16 weight layers including thirteen convolutional layers with filter size of 3 X 3, and fully connected layers with filter size of 3 X 3, and fully connected layers. The stride and padding of all convolutional layers are fixed to 1 pixel. All convolutional layers are divided into 5 groups and each group is followed by a max-pooling layer.

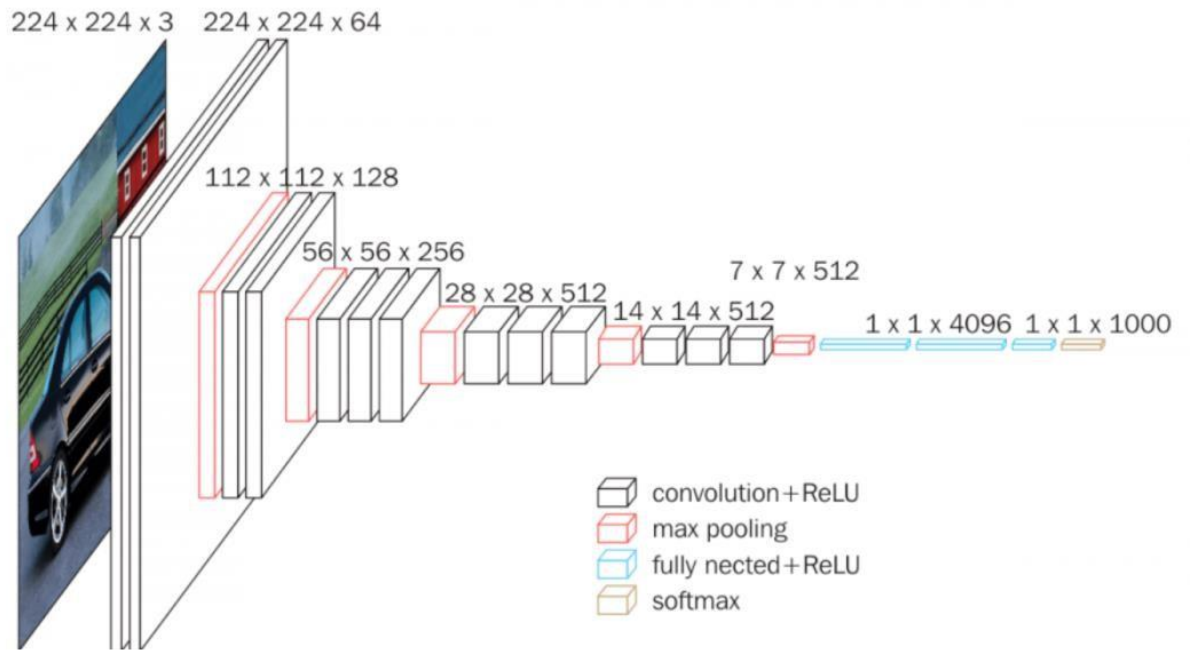


Figure 5 VGG NET

GAN / DCGAN

GAN comprises of two neural networks, the discriminator and the generator. The purpose of the discriminator is to identify if the given image is from the original dataset or if it is newly generated image, and the generator generates new images in order to cheat the discriminator while the discriminator tries to increase the performance of guessing it right. The two components work hand in hand to give out more realistic images as output.

DCGAN is a simple variation of GAN where there is use of convolutional and transpose convolutional layers in discriminator and generator.

The difference between GAN and DCGAN is as follows –

• Use of convolutional stride instead of MaxPooling

- Use of transposed convolution for upsampling
- Eliminating fully connected layers
- Use of BatchNorm except the output layer for the generator and the input layer of the discriminator
- Use of ReLU in the generator except for the output which uses tanh
- Use LeakyReLU in the discriminator

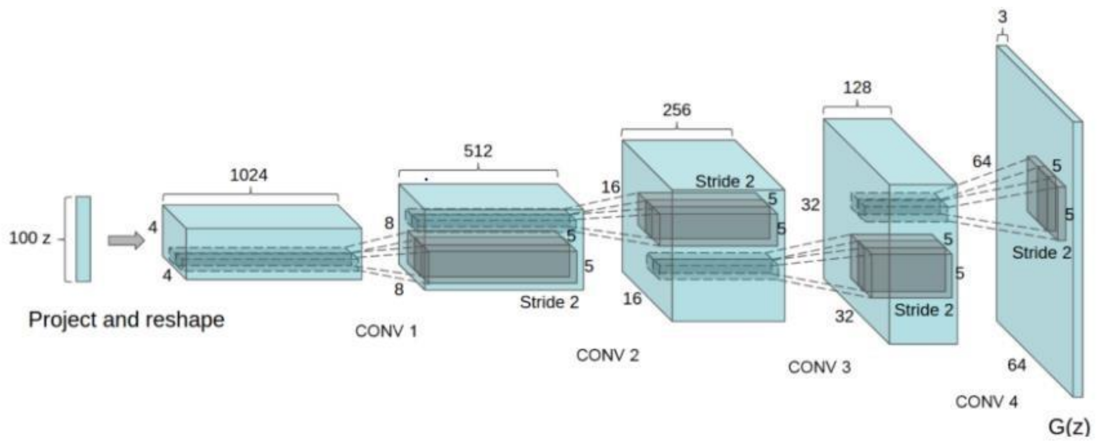


Figure 6 GAN

System Design

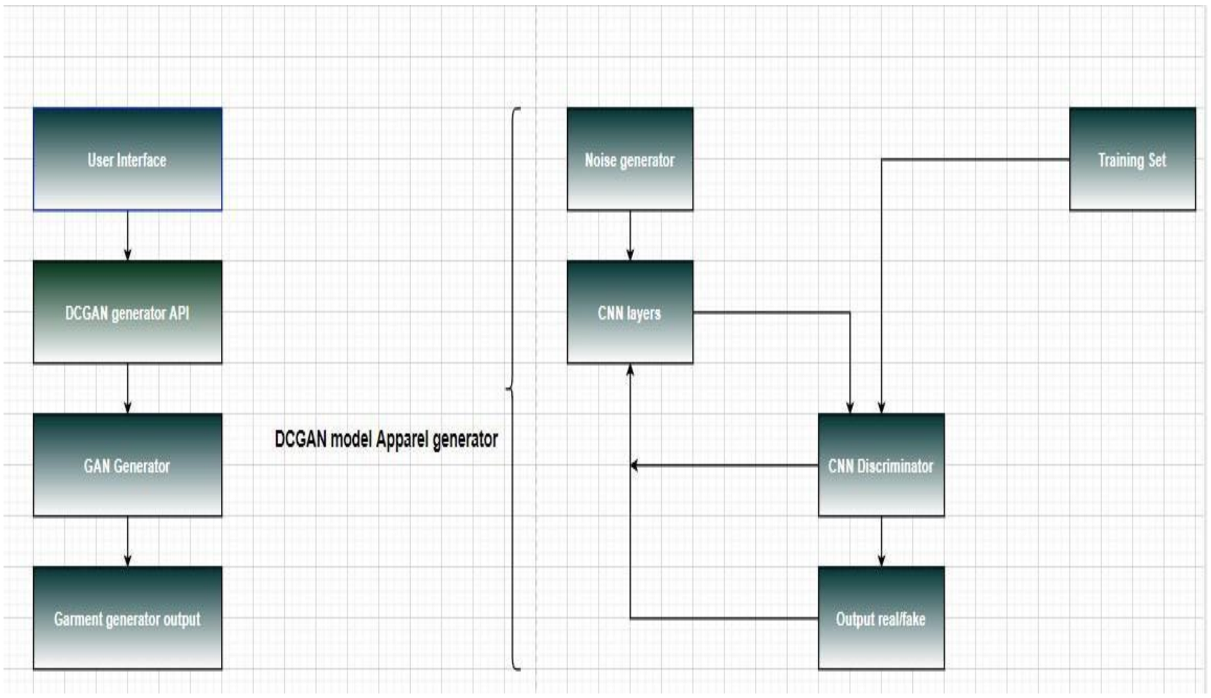


Fig 7 – System Design for Apparel Generation

The system architecture is mainly divided into two parts:

- **DCGAN** – Apparel generation
- **Stacked Ensemble Model** – Apparel classification

We have used the **DCGAN** for apparel generation during the training phase.
The API will in turn call the DCGAN model. This model uses a combination of GAN generator along with a series of convolutional layers. The CNN model is used as a discriminator that will classify the apparel as fake or real image.

In the testing and validation phase with the help of the user interface, we can call the apparel generator API. The generated garments are then sent as output to the fashion designer who can then verify the designs that have been auto generated.
The second part of the system architecture is the **stacked ensemble model**.

The fashion designer can select any apparel of his/her choice and find out the features he/she would like to know about the garment. This will help him/her analyze the data and help in making quick decisions.

The garment is fed to different models for classification.

- 2 CNN models
- Resnet
- Resnet-Inception
- VGGNET

Some of the features extracted are:

- Sleeve length
- Cloth pattern
- Dress length
- Neckline
- Color

Each model will perform a separate classification on the pre-trained model weights. The pre-trained model weights are obtained from training a separate CNN model for pattern recognition, sleeve length recognition etc.

For the cloth pattern attribute classification each weak learner model has distinct features and different accuracy and loss values which when stacked together will contribute to provide an enhanced accuracy.
The output of all the models is added together and provided as an input to the stacked ensemble model which is tasked with the classification of the garment.

Dataset

The dataset contains around 15k images with details of dress patters. The dataset was taken from the following https://s3.eu-central1.amazonaws.com/fashion-gan/images.zip.

Image Classification

In the first module we have developed classifier model to detect various attributes from a given image(256x256). The following are the few attributes we obtain from the image –

1. Pattern
2. Sleeve Length
3. Length
4. Color
5. Fit
6. Neckline

System Performance

Pattern Classification

There are 6 output classes for the design patterns of the clothes. We developed multiple CNN models as base models over which a stacked ensemble model is built for classification. The distribution of data in the class is ununiform.

TABLE 2 NO OF DRESSES

Class	No. of image data
Floral	2591
Lace	1018
Polkadots	428
Print	1220
Stripes	1010
Unicolors	6098

Base Model-1

We have trained the dataset on a sequential Convolutional neural network with the following architecture and observed a validation accuracy of 78.01%.

The hyper parameters set were RMSprop optimizer with a learning rate of 0.0001, there are many dropout layers with 25% rate and 50% in one of the last layers. Two convolution layers before the dense layers were set to L2 kernel regularization of penalty 0.001.

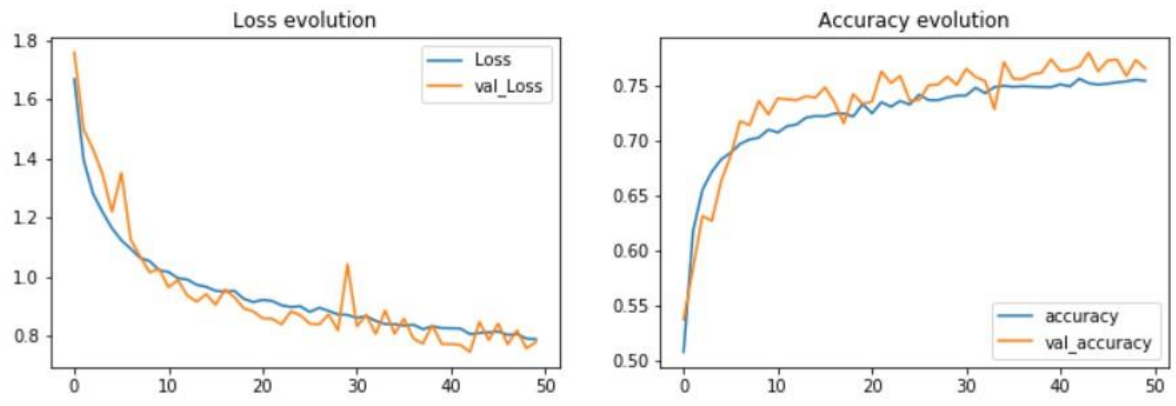
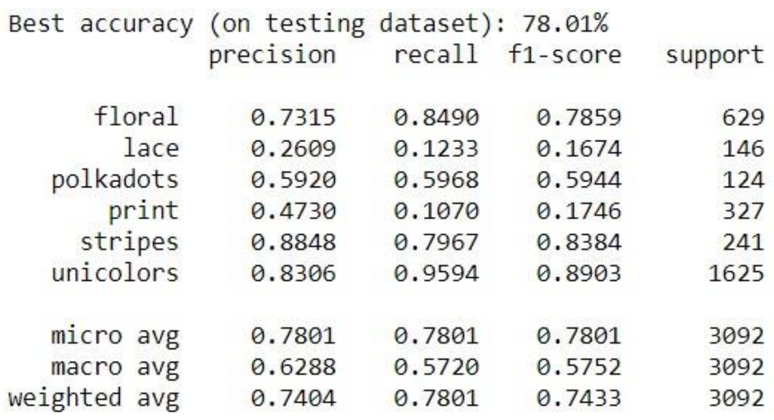


Fig 8– Pattern Classification Base Model 1 – loss and accuracy evolution over 50 epochs.

The following is the confusion matrix for the above trained model.



```
Best accuracy (on testing dataset): 78.01%
              precision    recall  f1-score   support

      floral     0.7315    0.8490    0.7859       629
        lace     0.2609    0.1233    0.1674       146
   polkadots     0.5920    0.5968    0.5944       124
       print     0.4730    0.1070    0.1746       327
     stripes     0.8848    0.7967    0.8384       241
   unicolors     0.8306    0.9594    0.8903      1625

   micro avg     0.7801    0.7801    0.7801      3092
   macro avg     0.6288    0.5720    0.5752      3092
weighted avg     0.7404    0.7801    0.7433      3092
```

Fig 9 – Pattern Classification Base Model 1 – Classification Report

Base Model-2

We have trained the dataset on a VGG Net (VGG 16) and observed a validation accuracy of 79.66%.

The hyper parameters set were Adam optimizer with a learning rate of 0.00005. The following is the plot for training and test set loss :

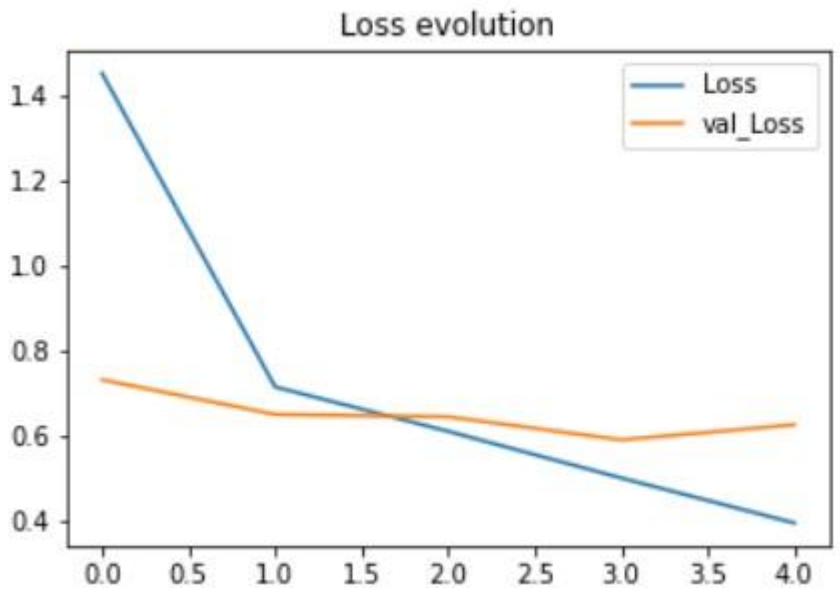


Fig 10– Pattern Classification Base Model 2 – loss evolution over 5 epochs.

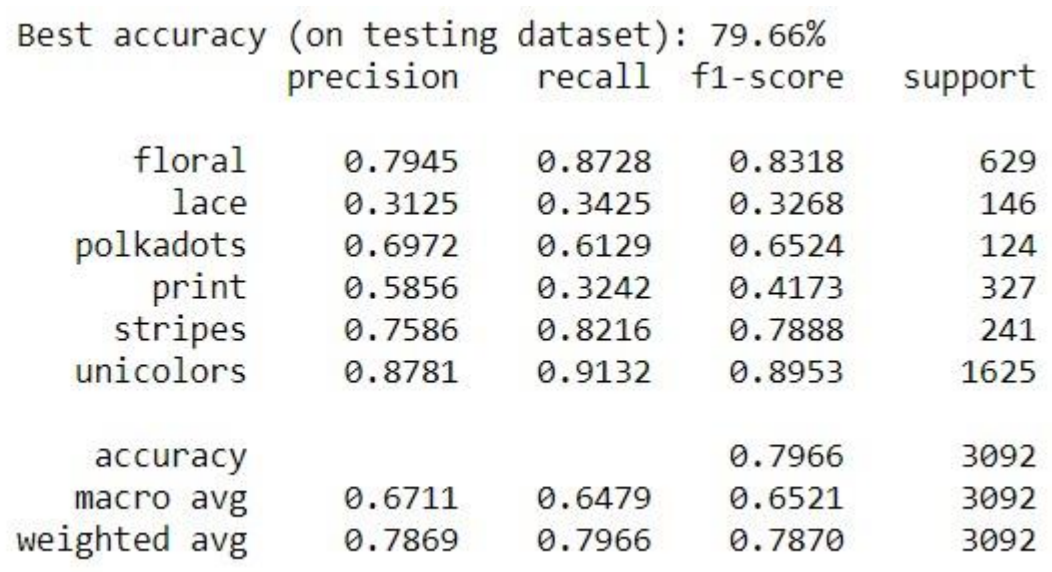

```
Best accuracy (on testing dataset): 79.66%
              precision    recall  f1-score   support

      floral     0.7945    0.8728    0.8318       629
        lace     0.3125    0.3425    0.3268       146
   polkadots     0.6972    0.6129    0.6524       124
       print     0.5856    0.3242    0.4173       327
     stripes     0.7586    0.8216    0.7888       241
   unicolors     0.8781    0.9132    0.8953      1625

    accuracy                         0.7966      3092
   macro avg     0.6711    0.6479    0.6521      3092
weighted avg     0.7869    0.7966    0.7870      3092
```

Fig11– Pattern Classification Base Model 2 – Classification Report

Dress Length Classification

There are 5 classes used for the classification of dress length in the dataset. We preprocessed the data to have a balanced dataset. Each class must have equal data. For classification, we added a CNN model. The distribution of data for each class is as follows.

Table 3 dress length

Class	**No. of image data**
3-4	1200
knee	1200
long	1200
normal	1200
short	1200

We have also plotted the training accuracy and loss.

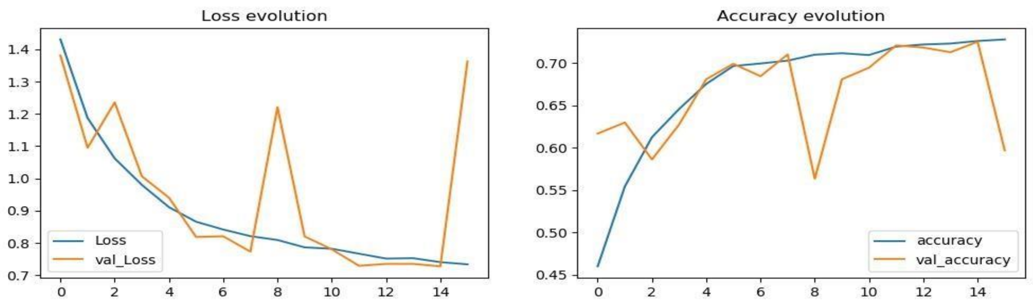
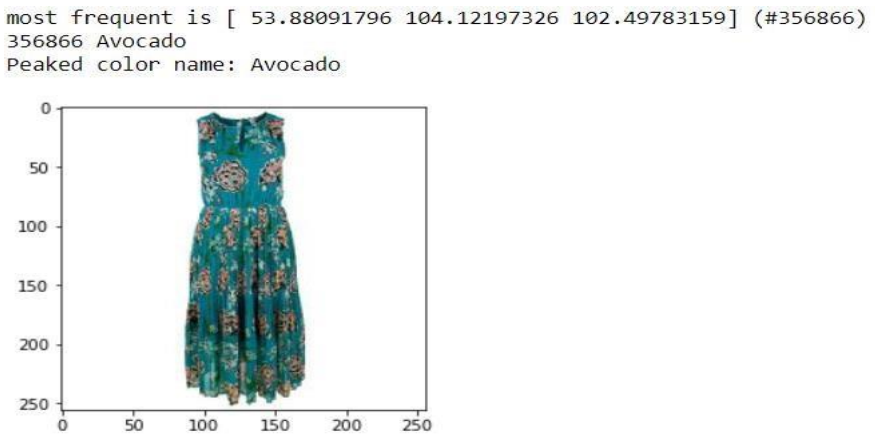


Fig 12– Sleeve length Classification CNN Model – loss and accuracy evolution over 20 epochs

Color Prediction

Extraction of color is obtained based on the k means clustering of the pixel intensities of the given RGB Image, the number of clusters is set as 2 and the most frequent is taken the color of the dress. The nearest color name is also calculated using Euclidean distance from the smaller list of hex-codes and name.

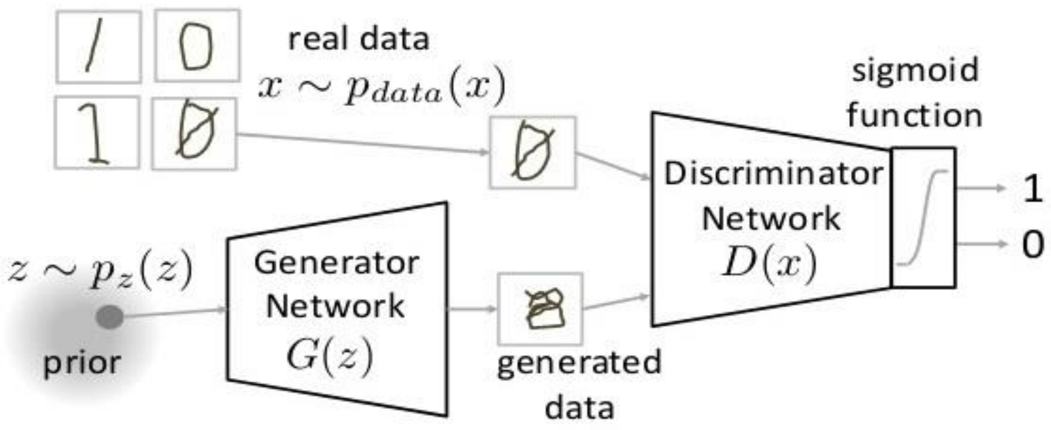


```
most frequent is [ 53.88091796 104.12197326 102.49783159] (#356866)
356866 Avocado
Peaked color name: Avocado
```

Fig 13 Color detection

GAN

Generation of new dress was achieved by training the Generative Adversarial Network where the generator has multiple dense layers with tanh activation function in the final dense layer and discriminator also has multiple dense layers with sigmoid activation function in the final dense layer. The training was done for 200 epochs and received the following output:



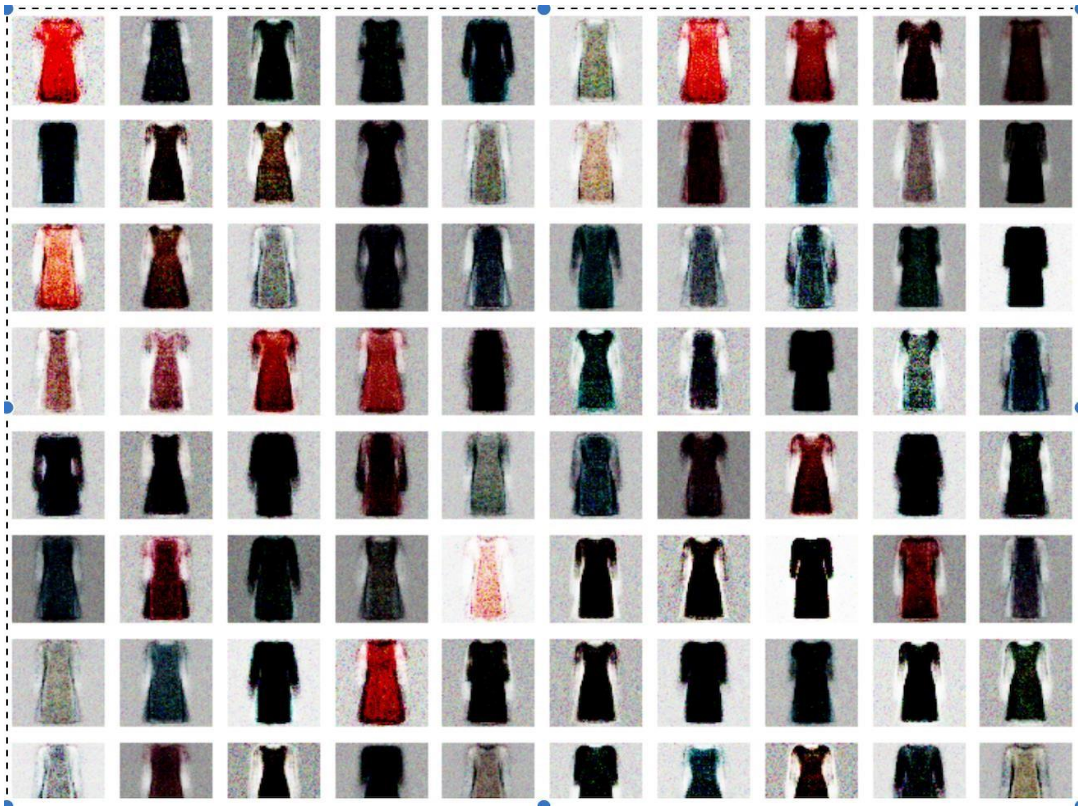
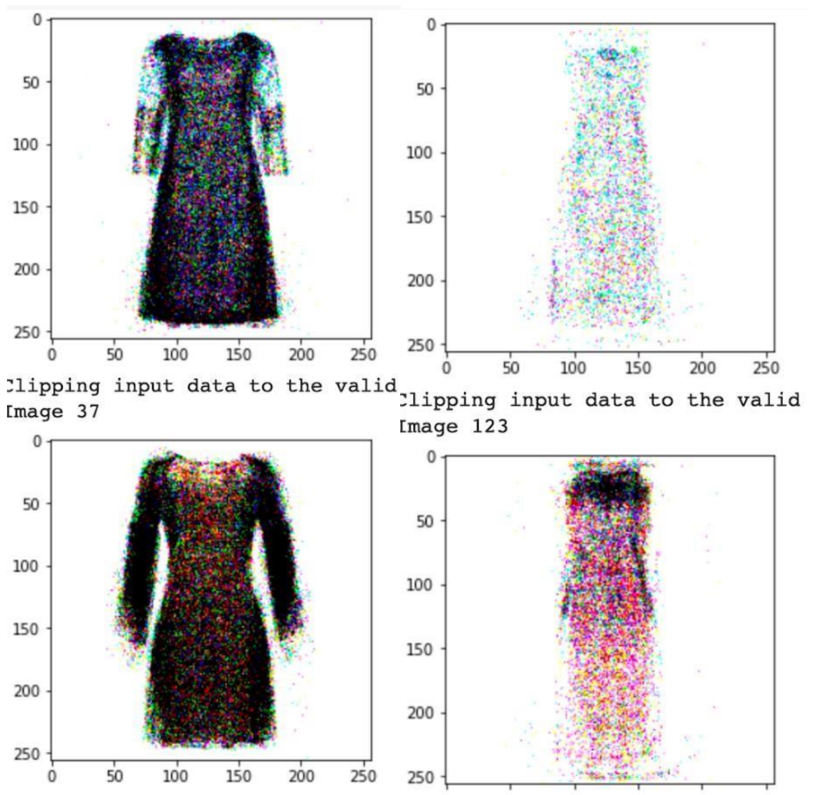


Fig 14– GAN 256x256 Images generated



```
Clipping input data to the valid
Image 37
Clipping input data to the valid
Image 123
```

Fig15 – GAN 256x256 Images generated

DCGAN

Generation of new dress was achieved by training the deep convolutional Generative Adversarial Network where there are 5 convolution layers in discriminator and 5 transpose convolution layers in the generator block. The training was done for 100 epochs and the generator and discriminator loss are as follows.

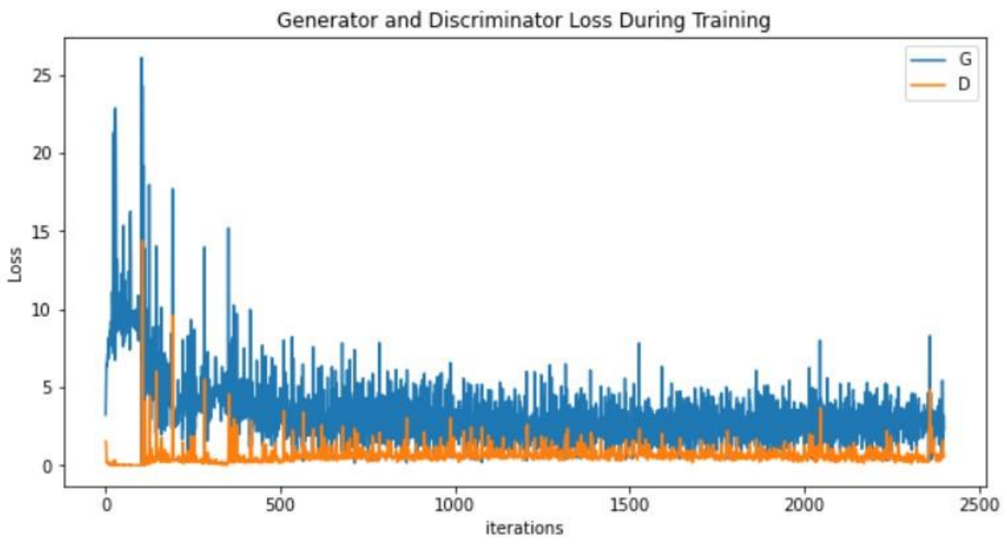


Fig 15– DCGAN Generator and Discriminator Loss over 100 epochs, the graphs show for every batch over 100 epochs

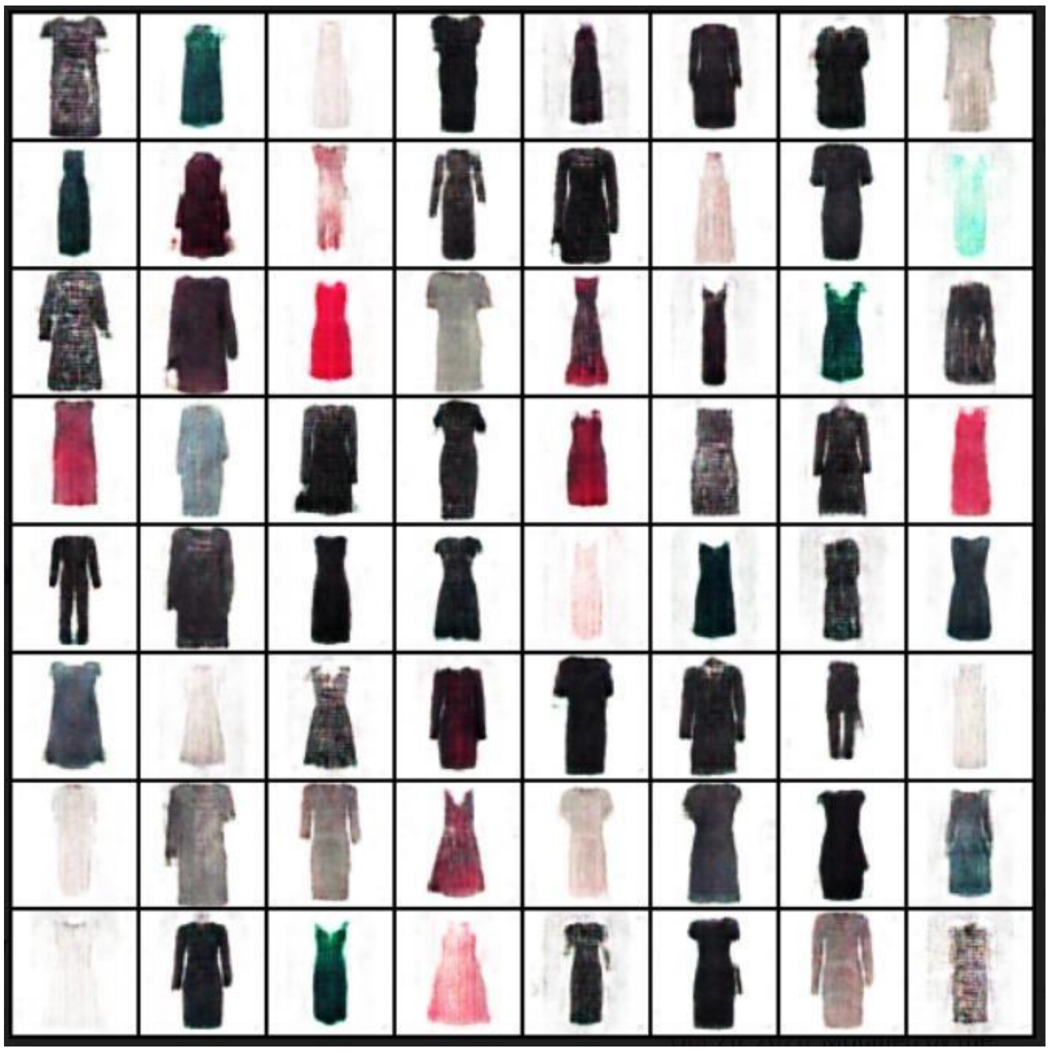


Fig 16– DCGAN 64x64 Images generated

5. Conclusion and future work

The shape and dynamics of a learning curve can be used to diagnose the behaviour of a machine learning model. The training and validation accuracy and loss indicate fitting of the model in the training and testing phase. Close curves of the training and accuracy loss show that the training was fairing good like in the case of CNN model used in pattern prediction. But for models like Resnet and Inception Resnet the curve seems to diverge after few epochs, indicating poorly fit model, where the model training should be stopped after reaching point of inflation. Also models like Resnet require lesser training because ResNet architecture is for reducing the complexity and solving the degradation while keeping good performance. By reducing complexity, a smaller number of parameters need to be trained and spending less time on training as well.

Table 4 Result

Models used for Pattern Classification	**Model Accuracy**
CNN-1	78.01%
CNN-2	79.81%
ResNet	76.58%
Inception Resnet	77.87%
VGG Net	79.66%

As it can be seen in the table above, we were able to achieve highest accuracy for CNN-2 model. The hyperparameters were optimized to accommodate the changes. Instead of using the RMSprop optimizer which was used in CNN-1 model we used the Adam optimizer as it is computationally efficient even with large datasets and works well on problems with normal or sparse gradients where the dataset could be sparse. We also maintained a learning rate of 0.0001 % which helped us in increasing the accuracy even further. With the help of the learning rate the gradients did not explode.

We also maintained a constant dropout rate of 25% in all the convolutional layers which as compared to the dropout rate of 50% maintained in the last layers of CNN-1 model. This helped us in removing only those weights which did not actually contribute to increasing the overall accuracy of the model.

We had additional convolutional layers which included having a batch normalization layer, max pooling layer and a dropout layer which refined our accuracy by a factor of 0.12%.

We also used an L1 regularization kernel instead of the L2 regularization kernel used in the CNN-1 model. This also helped us as L1 is computationally

The overall accuracy of the pattern classification model has improved with the use of stacked ensemble model. We used a concatenate layer and two dense layers as a secondary meta learner giving weighted contribution of each sub models. The Precision and recall values for all the model for the class Lace seems to be low and therefore it is observed that the models are unable to decipher these patterns to the best compared to the other classes. This inability to identify this class has also taken a toll on the ensemble classification this is probably because of merge amount of data available in the dataset for these classes especially for lace.

Data augmentation and pre-processing of images was not required for classification of Pattern, Sleeve-Length, Length and Neckline as we have considered on the front view of the dress images and without and human model posing with the apparel. Cropping the input image for finding the colour of the dress was adopted as the background of the dress was also considered as one of the major colour contributors in the dress, therefore the centre of the image was considered for obtaining the colour. Here clustering was performed with cluster size as 2, although any number of clusters can be implemented, to get the major colour in the dress the most frequent is considered. To get the names of the colour from hex-code, we have taken a list of colour names and the corresponding hex-code and performed Euclidean distance between the two hex-codes. The accuracy of getting the names correct is not high because the code, colour name considered is very less compared to the number of hex-codes present in each channel (16*16).

The results for GAN trained on 256x256 are found to be a little spotty compared to that of the results produced by DCGAN. The use of convolutional layers and transpose convolutional layers instead of fully connected layers has helped in improving the image clarity. Use of convnet tries to find the areas of correlation within the images looking for spatial correlations.

Transfer Learning (TL) is a concept related to machine learning (ML) which focuses on saving or storing the knowledge gained while solving for one problem and then applying it to a new but related problem. For example, the knowledge that is gained while learning to identify or recognize cars can be applied when trying to recognize buses and trucks. Using transfer learning, instead of starting the learning or training process from scratch, we can start from patterns that have already been learned when solving a different problem. In this way we can leverage earlier learnings and avoid beginning from scratch again. From the practical perspective, reusing information from previously learned tasks in order to the learn new tasks has the potential to highly improve the efficiency of a reinforcement learning agent.

6. REFERENCES

1. Liu, Z., Yan, S., Luo, P., Wang, X., Tang, X.: Fashion landmark detection in the wild. Lecture Notes in Computer Science, pp. 229–245 (2016)
2. Lao, B., Jagadeesh, K.: Convolutional neural networks for fashion classification and object detection. Semantic Scholar (2015)
3. Yu, W., Liang, X., Gong, K., Jiang, C., Xiao, N., Lin, L., Yat-Sen, S.: University, DarkMatter AI research. Layout-Graph Reasoning for Fashion Landmark Detection (2019)
4. Cetinic, E., Lipic, T., Grgic, S.: A deep learning perspective on beauty, sentiment and remembrance of art. In: Institute of Electrical and Electronics Engineers (IEEE), pp. 1–1 (2019)
5. . Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. Visual Geometry Group, Department of Engineering Science, University of Oxford (2015)
6. Yangjie Cao, Li-Li Jia, Yong-Xia Chen, Nan Lin, Cong Yang, Bo Zhang, Zhi Liu, XueXiang Li, Honghua Dai: Recent advances of generative adversarial networks in computer vision. In: Institute of Electrical and Electronics Engineers (IEEE), 7, pp. 14985–15006 (2019)
7. Atenas, F., Sanhueza, F., Valenzuela, C.: Redes neuronales adversarias convolucionales para generación de imágenes (2017)
8. Jun-Yan, Z., Park, T., Isola, P.A.: Unpaired image-to-image translation using cycleconsistent adversarial networks efros. Berkeley AI Research (BAIR) laboratory, UC Berkeley (2015)
9. Li, F.F., Johnson, J., Yeung, S.: CS231n: Convolutional neural networks for visual recognition (2017)

APPENDICES

A.

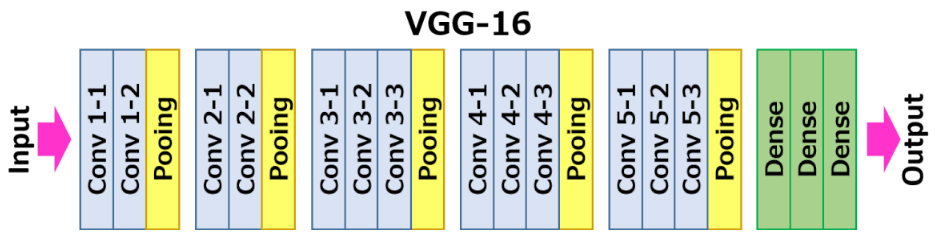
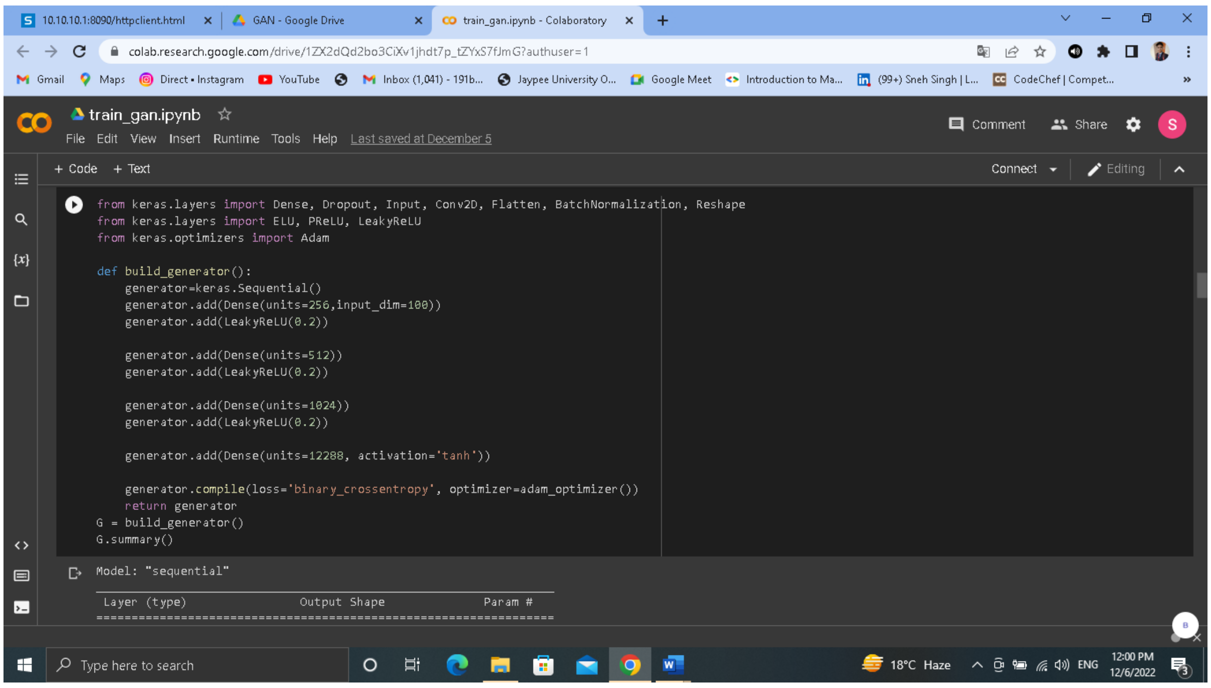


Fig A– Base Model 2 – VGG Pattern Prediction Architecture

B.

.



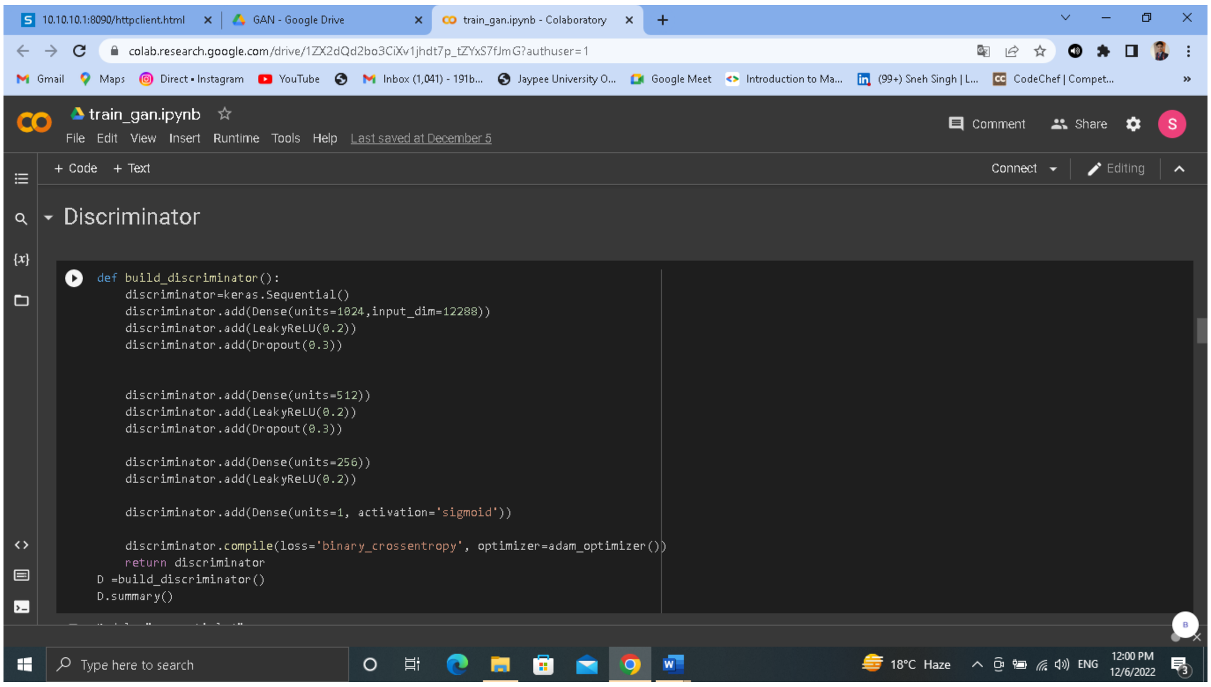
train_gan.ipynb
File Edit View Insert Runtime Tools Help Last saved at December 5
+ Code + Text
Comment
Share
Connect
Editing

```
from keras.layers import Dense, Dropout, Input, Conv2D, Flatten, BatchNormalization, Reshape
from keras.layers import ELU, PReLU, LeakyReLU
from keras.optimizers import Adam

def build_generator():
    generator=keras.Sequential()
    generator.add(Dense(units=256,input_dim=100))
    generator.add(LeakyReLU(0.2))

    generator.add(Dense(units=512))
    generator.add(LeakyReLU(0.2))

    generator.add(Dense(units=1024))
    generator.add(LeakyReLU(0.2))

    generator.add(Dense(units=12288, activation='tanh'))

    generator.compile(loss='binary_crossentropy', optimizer=adam_optimizer())
    return generator
G = build_generator()
G.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
```

Fig B– DCGAN Architecture for Generator

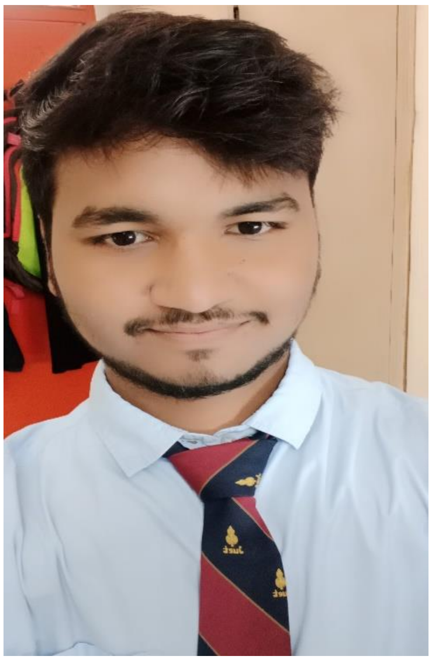
C.


train_gan.ipynb
File Edit View Insert Runtime Tools Help Last saved at December 5
+ Code + Text
Connect
Editing
Discriminator

```
def build_discriminator():
    discriminator=keras.Sequential()
    discriminator.add(Dense(units=1024,input_dim=12288))
    discriminator.add(LeakyReLU(0.2))
    discriminator.add(Dropout(0.3))


    discriminator.add(Dense(units=512))
    discriminator.add(LeakyReLU(0.2))
    discriminator.add(Dropout(0.3))

    discriminator.add(Dense(units=256))
    discriminator.add(LeakyReLU(0.2))

    discriminator.add(Dense(units=1, activation='sigmoid'))

    discriminator.compile(loss='binary_crossentropy', optimizer=adam_optimizer())
    return discriminator
D =build_discriminator()
D.summary()
```

Fig C– DCGAN Architecture for Discriminator

Students Profile



NAME: SNEH SINGH
ER. NO: 191B262
COURSE: Bachelor in Technology
BRANCH: Computer Science and Engineering
MOTHER’S NAME: Mrs. SULOCHANA SINGH
FATHER’S NAME: Mr. JAIMANGAL SINGH
EMAIL: singhsneh58@gmail.com
MOBILE: +91-9340575521

NAME: VANAMA YASWANTH
ER. NO: 191B278
COURSE: Bachelor in Technology
BRANCH: Computer Science and Engineering
MOTHER’S NAME: Mrs. Vanama Sunitha Rani
FATHER’S NAME: Mr. Vanama Suresh
EMAIL: vanamayaswanth1212@gmail.com
MOBILE: +91-9705888236

NAME: VIDHI MATHUR
ER. NO: 191B282
COURSE: Bachelor in Technology
BRANCH: Computer Science and Engineering
MOTHER’S NAME: Mrs. SARITA MATHUR
FATHER’S NAME: Mr. AJAY KUMAR MATHUR
EMAIL: vidhi8255@gmail.com
MOBILE: +91-7869727730