# CSE 676-B: Deep Learning
## Final Project

---

**Full Name**: Gayatri Nagesh Walke
**UBITName**: gwalke
**Student Number**: 50538312

**Full Name**: Vinit Wadgaonkar
**UBITName**: vinitwad
**Student Number**: 50545019

**Full Name**: Shriganesh Siddramappa Lokapure
**UBITName**: shrigane
**Student Number**: 50540717

---

## Checkpoint (4 April)

**Title: Enhancing E-commerce Search with Multimodal GANs**

**Short Summary:**
The project aims to leverage generative adversarial networks (GANs) to improve the fashion e-commerce search experience. Focusing on the multimodal nature of user queries involving both images and text, the goal is to enable more natural language expressions while refining the search engine's comprehension. This involves the generation of synthetic images through a multimodal GAN network, allowing users to validate the system's understanding of their intentions.

**Objective:**

a. Develop a GAN network capable of synthesizing images based on multimodal input (image and text).
b. b. Ensure the preservation of meaningful distances in the embedding space for efficient visual search.
c. c. Facilitate user expression of natural language queries for fashion items, encompassing both visual and textual elements.

**Experimental Setup:**

The experiment is conducted using Google Colab for training. Necessary libraries such as TensorFlow and Matplotlib are imported. The Google Drive is mounted to store the generated images.

**Model Architecture:**

**1. Discriminator:**

- Input: 28x28 grayscale images.
- Layers:
  - Flatten layer
  - Dense layer with 512 units, followed by LeakyReLU activation
  - Dense layer with 256 units, followed by LeakyReLU activation
  - Output layer with 1 unit and sigmoid activation.
- Loss: Binary cross-entropy.
- Optimizer: Adam with learning rate 0.0002 and momentum 0.5.

**2. Generator:**

- Input: Noise vector of dimension 100.
- Layers:
  - Dense layer with 256 units, followed by LeakyReLU activation and dropout
  - Dense layer with 512 units, followed by LeakyReLU activation and dropout
  - Dense layer with 1024 units, followed by LeakyReLU activation and dropout
  - Output layer with tanh activation, reshaped to output 28x28 grayscale images.
- Loss: Binary cross-entropy.
- Optimizer: Adam with learning rate 0.0001 and momentum 0.5.

**3. Training Procedure:**

- The Fashion MNIST dataset is loaded and preprocessed. Images are scaled to the range [-1, 1].
- During each epoch:
  - Real images are randomly sampled from the dataset.
  - Random noise is generated as input to the generator.
  - The discriminator is trained on both real and fake images.

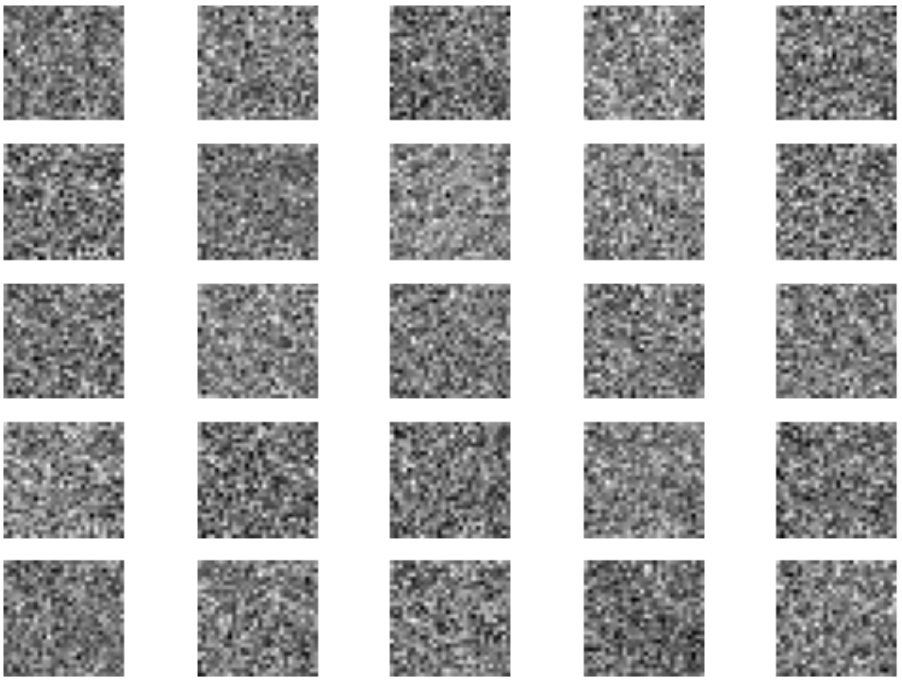
- The generator is trained via the GAN model, aiming to generate images that the discriminator identifies as real.
- Losses and accuracies of both discriminator and generator are monitored and printed.

**4. Sample Generation:**

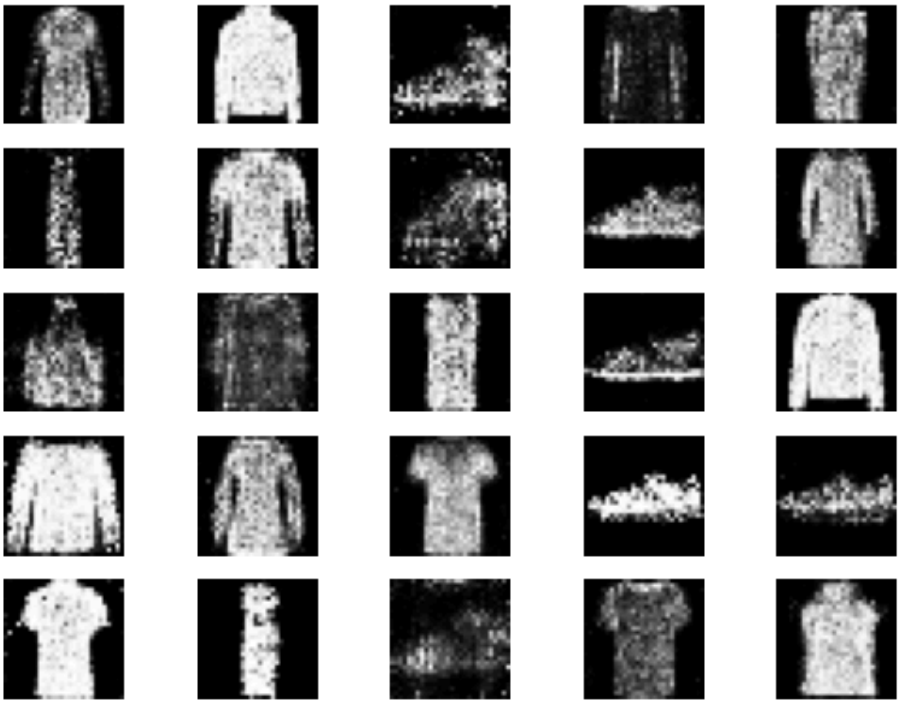
- At regular intervals, sample images are generated using the generator to visualize the training progress.
- Random noise is fed into the generator to produce synthetic images.
- Generated images are rescaled to [0, 1] and displayed using Matplotlib.Optionally, images are saved to Google Drive.

**5. Results and Analysis:**

The experiment was run for 30000 epochs with a batch size of 32. Discriminator and generator losses are monitored throughout the training process. Generated images progressively resemble fashion items present in the dataset, indicating the effectiveness of the GAN in learning the underlying data distribution.



Output after first epoch



Output after last epoch

```
Epoch [5/30000], Generator Loss: 1.3322519223531086, Discriminator Loss: 0.48142682382795543
Epoch [6/30000], Generator Loss: 1.2724349509261903, Discriminator Loss: 0.5004405455180577
Epoch [7/30000], Generator Loss: 1.227816779255867, Discriminator Loss: 0.5147335050264994
Epoch [8/30000], Generator Loss: 1.1910718064202204, Discriminator Loss: 0.5265120938618978
Epoch [9/30000], Generator Loss: 1.1611949109284083, Discriminator Loss: 0.5361075046014786
Epoch [10/30000], Generator Loss: 1.1341601436109254, Discriminator Loss: 0.5449968430085615
Epoch [11/30000], Generator Loss: 1.1106641369011667, Discriminator Loss: 0.5528090282572641
Epoch [12/30000], Generator Loss: 1.0903841684769362, Discriminator Loss: 0.5593306176919204
Epoch [13/30000], Generator Loss: 1.0726720336902709, Discriminator Loss: 0.5651898054009392
Epoch [14/30000], Generator Loss: 1.056386272764206, Discriminator Loss: 0.5705724182086521
Epoch [15/30000], Generator Loss: 1.0420010713110368, Discriminator Loss: 0.5753492658853531
Epoch [16/30000], Generator Loss: 1.0288519080377092, Discriminator Loss: 0.5797332555883071
Epoch [17/30000], Generator Loss: 1.017141340609833, Discriminator Loss: 0.5835704301710481
Epoch [18/30000], Generator Loss: 1.0066744893333368, Discriminator Loss: 0.5870121734836645
Epoch [19/30000], Generator Loss: 0.9970786174575488, Discriminator Loss: 0.5901773569520314
Epoch [20/30000], Generator Loss: 0.9886271439590152, Discriminator Loss: 0.5929452133996146
Epoch [21/30000], Generator Loss: 0.980861009219921, Discriminator Loss: 0.5955108389782183
Epoch [22/30000], Generator Loss: 0.9737720138238823, Discriminator Loss: 0.5978199537484542
Epoch [23/30000], Generator Loss: 0.9671200920151339, Discriminator Loss: 0.5999795067694452
Epoch [24/30000], Generator Loss: 0.9611560932407379, Discriminator Loss: 0.6019124895884196
Epoch [25/30000], Generator Loss: 0.9555629660478006, Discriminator Loss: 0.6037534186228728
Epoch [26/30000], Generator Loss: 0.950517114354946, Discriminator Loss: 0.6053860042430736
```

Epochs

**Conclusion:**

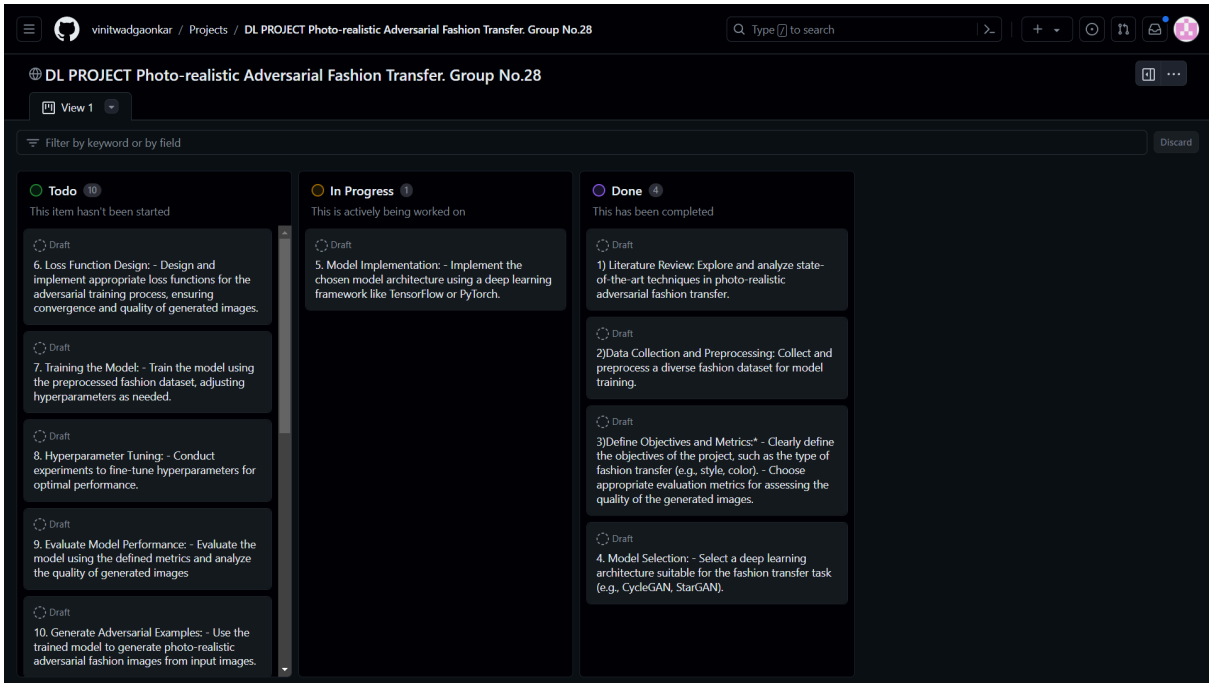
The experiment successfully trained a Generative Adversarial Network (GAN) on the Fashion MNIST dataset to generate synthetic fashion images. While the generator improved in producing realistic images, further exploration is needed. Future experiments will focus on color image generation, aligning with the primary goal of creating high-quality synthetic fashion images. This transition

requires adjustments to network architecture, hyperparameters, and dataset selection.

**References:**

Fashion MNIST Dataset: https://github.com/zalandoresearch/fashion-mnist
Generative Adversarial Networks: https://arxiv.org/abs/1406.2661

## GitHub 4 April Teamboard :



# Final Project (2 May)

## Real-world Deep Learning Application:

In the bustling world of e-commerce, where fashion trends change as swiftly as the seasons, the search for the perfect outfit can often feel like finding a needle in a haystack. Customers increasingly rely on online platforms to discover and purchase fashion items, but the search experience often falls short of their expectations. Traditional keyword-based searches can be limiting, especially when it comes to expressing nuanced preferences for style, color, and design.

Imagine a scenario where a fashion enthusiast is searching for a specific type of dress, say a "flowy, floral summer dress." They type these words into the search bar, hoping to find exactly what they have in mind. However, the search results may not accurately capture their intent. They might receive a mix of dresses that are floral but not flowy, or flowy but not floral, leading to a frustrating and time-consuming search process.

This is where our project comes into play. We aim to revolutionize the fashion e-commerce search experience by leveraging deep learning methods, specifically generative adversarial networks (GANs), to enhance the search engine's comprehension of user queries. By incorporating both image and text inputs, our approach allows users to express their fashion preferences in a more natural and intuitive manner.

Through the use of the Fashion IQ dataset, which contains a diverse range of fashion images and corresponding captions, we can train our GAN model to generate synthetic images that closely align with user queries. This not only improves the accuracy of search results but also enhances the overall user experience, making fashion discovery more personalized and enjoyable.

By highlighting real-world cases and the practical application of our work, we demonstrate the tangible impact of our project in addressing the challenges faced by both customers and e-commerce platforms in the fashion industry.

The project is a Fashion Chatbot that uses a Language Model (LLM) to interact with users and generate images based on their shopping-related queries. The Chatbot is designed to assist users in finding fashion items based on text descriptions, providing a visual representation of the items they are looking for.

**Architecture and Components:**

**Architecture:**

The GAN consists of two main components: a generator and a discriminator:

1. Generator:
The generator takes two inputs: image features and text features.
Text features are extracted using a pre-trained BERT model (BertModel.from_pretrained('bert-base-uncased')). The BERT model is used to encode textual descriptions into a fixed-size representation.
Image features are extracted using a series of convolutional layers (nn.Conv2d) followed by batch normalization (nn.BatchNorm2d) and ReLU activation (nn.ReLU). These layers extract high-level features from the input image.
The text features are reshaped (text_features.view) to match the size of the image features.
The image and text features are then combined and fed into a decoder, which consists of transpose convolutional layers (nn.ConvTranspose2d) followed by batch normalization, ReLU activation, and a final Tanh activation. These layers upsample the combined features to generate an image that matches the input description.

2. Discriminator:
The discriminator takes an input image and outputs a probability indicating whether the image is real or fake.
The discriminator architecture consists of convolutional layers (nn.Conv2d) followed by LeakyReLU activation (nn.LeakyReLU) and a final fully connected layer (nn.Linear) with a sigmoid activation (nn.Sigmoid) to produce the probability score.

3. Custom Dataset:
The FashionDataset class is a custom dataset class for loading and preprocessing fashion image-caption pairs.
It uses a BERT tokenizer (BertTokenizer.from_pretrained('bert-base-uncased')) to tokenize and encode textual descriptions.
The __getitem__ method returns the image, input_ids (tokenized text), and attention_mask for a given index.

4. Training Setup:
The models are initialized (generator = Generator() and discriminator = Discriminator()).
The dataset is initialized using the FashionDataset class and loaded into a dataloader (DataLoader(dataset, batch_size=2, shuffle=True)).
Two optimizers are defined (optimizer_G for the generator and optimizer_D for the discriminator) using the Adam optimizer.
The adversarial loss function (nn.BCELoss) is used to train both the generator and the discriminator.

Overall, this architecture uses a combination of text and image features to generate realistic fashion images based on textual descriptions, and the discriminator is trained to distinguish between real and fake images. The generator and discriminator are trained adversarially, with the generator trying to fool the discriminator and the discriminator trying to correctly classify real and fake images.

1. GUI: The Chatbot is implemented using a graphical user interface (GUI) created using the tkinter library in Python. The GUI allows users to input text messages and displays the generated images.

2. LLM Chatbot: The Language Model (LLM) is responsible for processing user input and detecting keywords related to shopping. It uses a predefined list of shopping keywords (shopping_keywords) to identify relevant terms in the user's messages.

3. Image Generation: For each detected keyword, the Chatbot uses a pre-trained generator model (SimpleGenerator) to generate an image corresponding to that keyword. The generator model is loaded from a file (generator.pth) and then used to create the image.

4. Image Display: The generated images are displayed in the chat area of the GUI using the process_image function. This function resizes the images to fit within the chat area and then displays them to the user.

5. User Interaction: Users can input text messages in the chat area of the GUI. When a message is sent, the Chatbot detects keywords in the message, generates corresponding images, and displays them to the user.

**Workflow:**

1. User Input: Users input text messages describing the fashion items they are looking for.

2. Keyword Detection: The Chatbot detects keywords related to shopping in the user's messages using the LLM.

3. Image Generation: For each detected keyword, the Chatbot generates an image using the pre-trained generator model.

4. Image Display: The generated images are displayed in the chat area of the GUI, allowing users to see visual representations of the items they are looking for.

**Benefits:**

- Improved User Experience: The Chatbot enhances the user experience by providing visual representations of fashion items based on text descriptions, making it easier for users to find the items they are looking for.

- Interactive Shopping Experience: By generating images based on user queries, the Chatbot creates an interactive and engaging shopping experience, helping users explore fashion items in a more visual way.

Overall, the Fashion Chatbot project demonstrates the use of language models and deep learning techniques to create a more interactive and user-friendly shopping experience.

**Novelty:**

In this GAN setup, several aspects contribute to its novelty and effectiveness in enhancing the fashion e-commerce search experience:

1. Multimodal Approach: The use of a multimodal approach, combining both image and text inputs, is a novel aspect of this GAN setup. By integrating image and text data, the model can better understand user queries that involve both visual and textual elements, leading to more accurate and relevant search results.
2. Conditional GANs: The use of conditional GANs allows the generator to produce synthetic images based on both the latent vector and the input text, enabling the model to generate images that align with specific user queries. This conditional generation capability enhances the model's ability to understand and fulfill user intent in fashion searches.
3. SynthTriplet GAN Architecture: The implementation of a SynthTriplet GAN architecture for optimizing product retrieval using triplet loss is another novel aspect. This architecture focuses on generating images that not only match the input query but also maintain meaningful distances in the embedding space, ensuring that the generated images are visually similar to the desired fashion items.
4. Training Strategies: The use of advanced training strategies such as Wasserstein GAN with gradient penalty, label smoothing, and a two-time scale update rule contributes to the stability and quality of training. These strategies help the model learn more effectively from the dataset, leading to improved image generation and search result relevance.

Overall, the combination of these novel elements makes this GAN setup uniquely suited for enhancing the fashion e-commerce search experience, offering a more intuitive and personalized search engine for fashion enthusiasts.

**Deploying the Model Locally:**

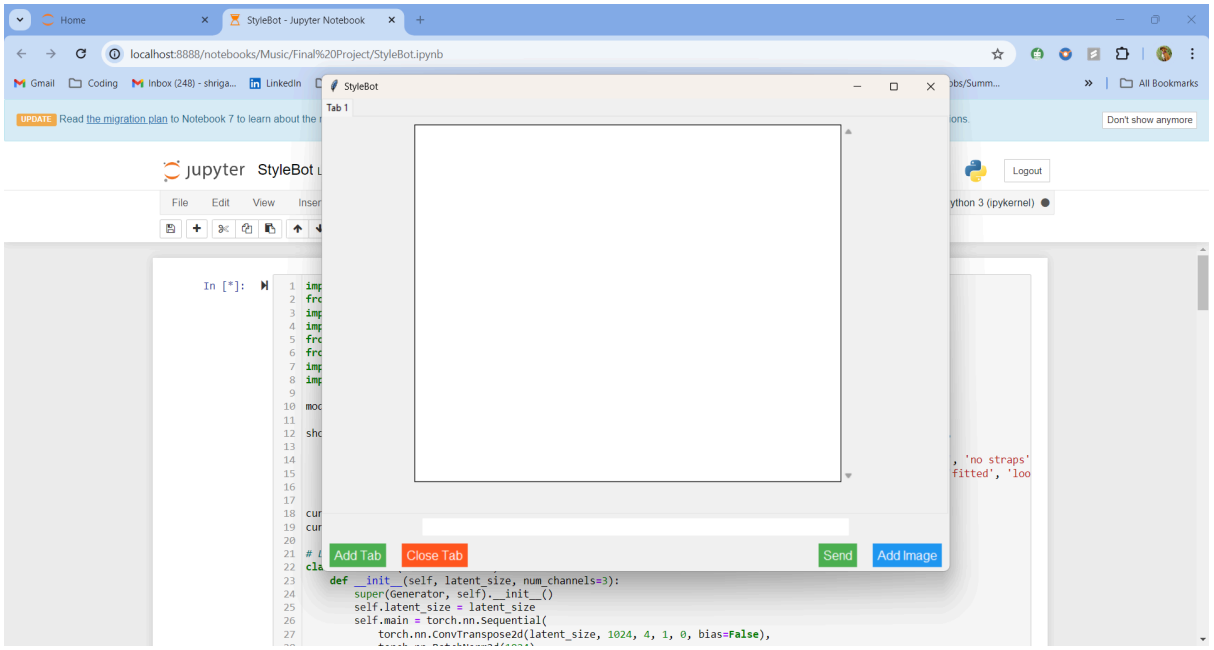
We deployed our final project model locally or on a server to demonstrate its functionality. The LLM (Language Model) Chatbot is used to interact with users, process their input, and generate images based on the detected keywords in the user's messages. Here's how it works:

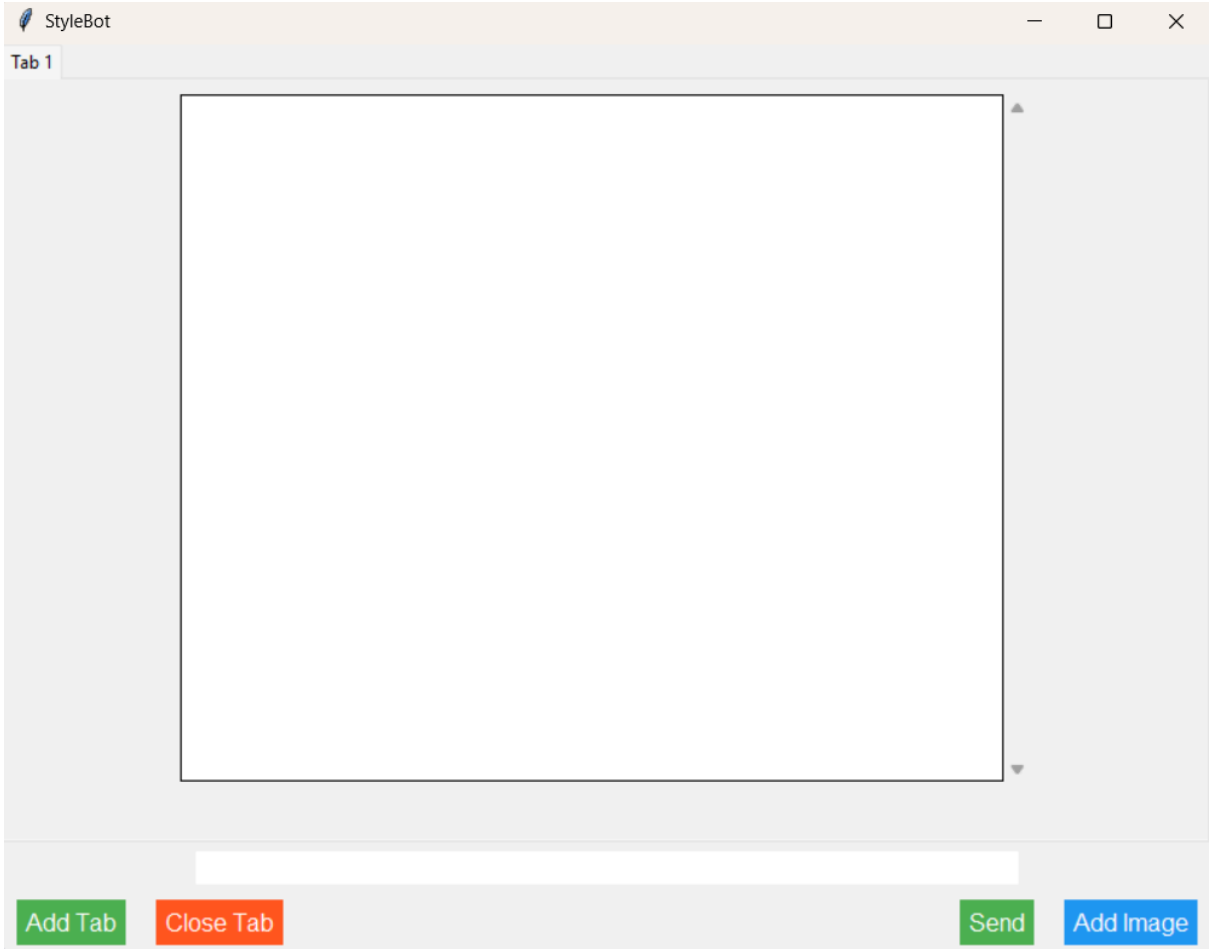
1. User Interaction: The LLM Chatbot is integrated into a graphical user interface (GUI) created using tkinter. Users can input text messages in the chat area of the GUI.
2. Keyword Detection: When a user sends a message, the chatbot detects keywords related to shopping from the message. These keywords are compared against a predefined list of shopping keywords (shopping_keywords), and any detected keywords are used to generate images.
3. Image Generation: For each detected keyword, the chatbot uses a pre-trained generator model (SimpleGenerator) to generate an image corresponding to that keyword. The generator model is loaded from a file (generator.pth) and then used to generate the image.
4. Image Display: The generated images are displayed in the chat area of the GUI using the process_image function. This function resizes the images to fit within the chat area and then displays them to the user.
5. User Feedback: The chatbot provides feedback to the user by displaying the detected keywords and the corresponding generated images in the chat area. This allows users to see the images related to their shopping queries.

Overall, the LLM Chatbot enhances the user experience by providing visual representations of shopping items based on the user's text input, thereby facilitating a more engaging and interactive shopping experience.

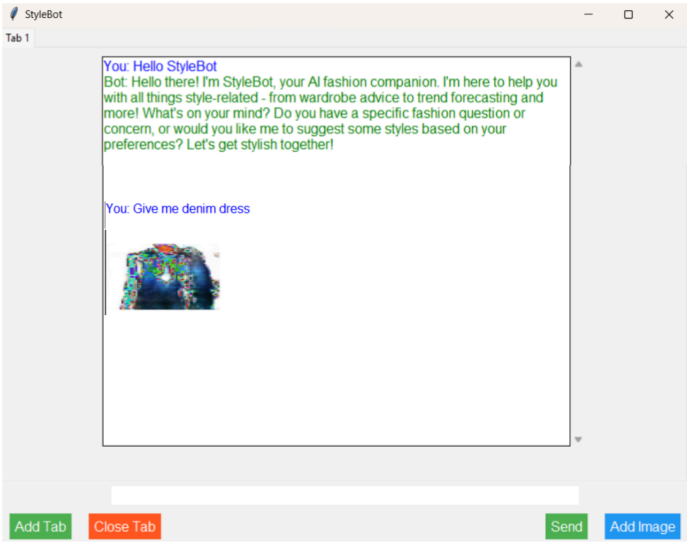
The chatbot interface features a user-friendly design with distinct tabs for different functionalities, ensuring easy navigation and organization. Users can seamlessly switch between tabs to access various features. Additionally, the interface includes a convenient image upload option, allowing users to easily share images within the chatbot. This design aims to enhance user experience by providing a visually appealing and intuitive interface for interacting with the

chatbot.


Home
StyleBot - Jupyter Notebook
localhost:8888/notebooks/Music/Final%20Project/StyleBot.ipynb
StyleBot
Tab 1
jupyter StyleBot
File Edit View
Logout
In [*]:
Add Tab
Close Tab
Send
Add Image


StyleBot
Tab 1
Add Tab
Close Tab
Send
Add Image

Below is the query result:
The query result indicates that when asked for a denim dress, the generated output resembles something close to blue but is more distorted. This suggests that the model requires further training to improve its accuracy and ability to generate more realistic images.

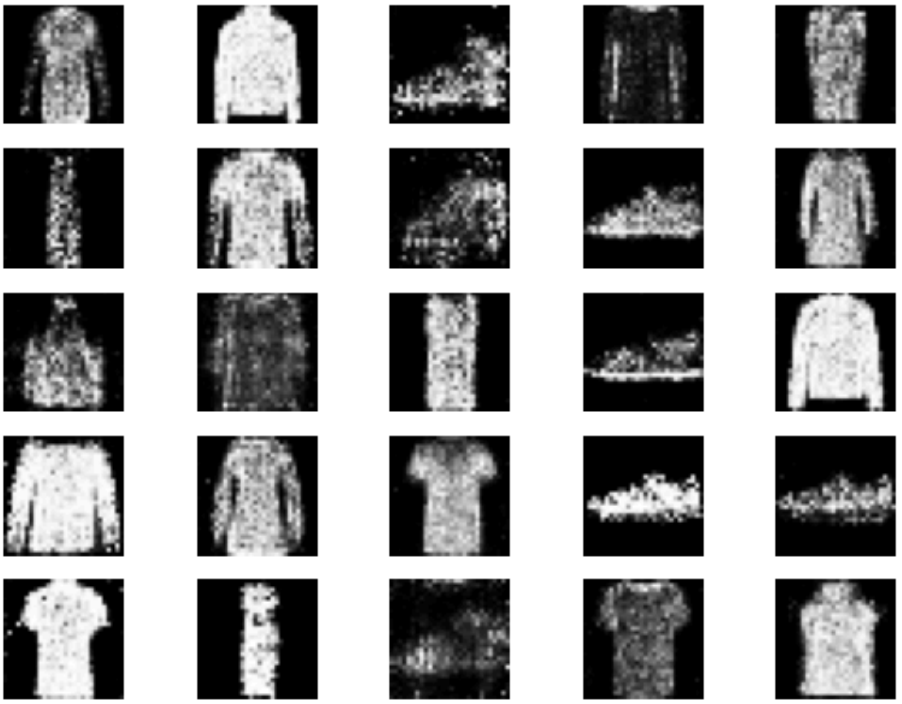



It is important to note that the training process is already in progress, and to support this, a Google Colab A100 GPU has been purchased. This powerful resource will aid in accelerating the training process and improving the quality of the generated images.

**Participating in Weekly Scrum Meetings:**

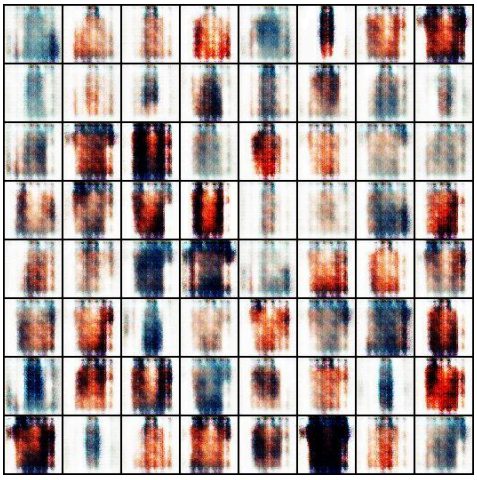
Our team actively participates in the weekly scrum meetings led by Aditya Muralidharan, where we provide updates on our project progress and receive valuable feedback. Here's an overview of our journey with the dataset and GANs:

1. Dataset Selection: We conducted thorough research to search for and finalize the dataset. This involved exploring various options and evaluating their suitability for our project requirements.
   We finalized fashioniq as it has images and captions which suits to our multimodal requirements perfectly.
2. Familiarization with GANs using Black and White Images: Initially, we worked with black and white images to familiarize ourselves with GANs. This phase helped us understand the basic principles and operation of GANs.



3. Challenges with Color Transformation: We attempted to transform black and white images to color images. However, we encountered shape mismatch errors, indicating that the transformed images were not in the proper format.

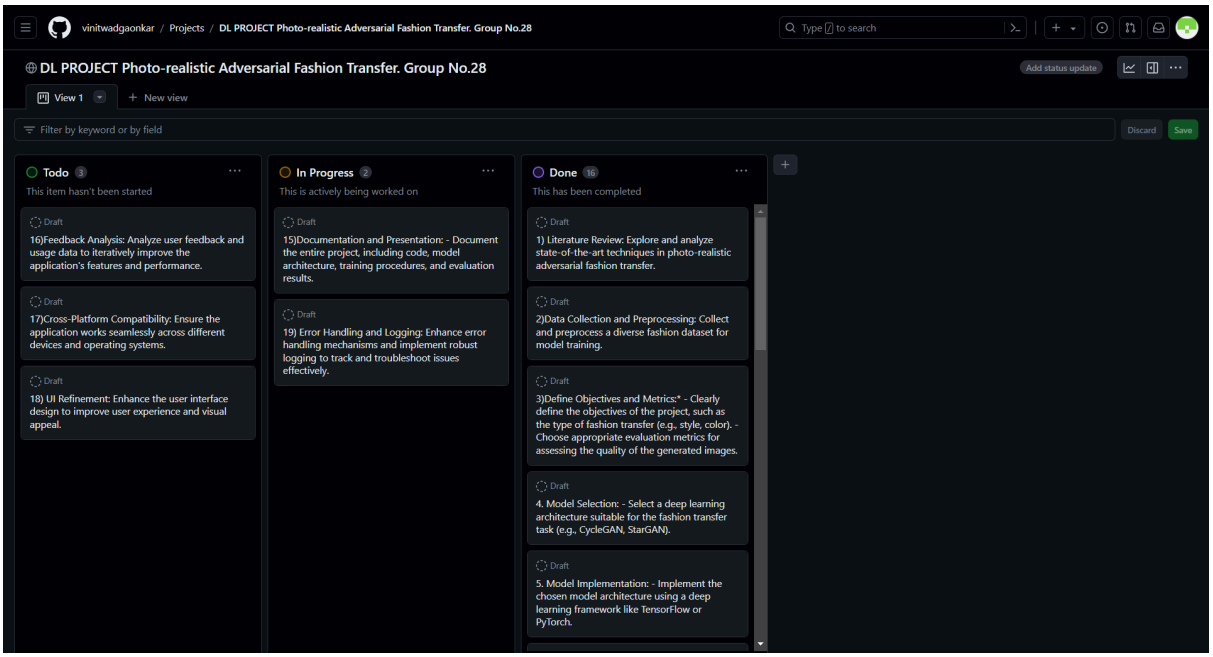
4. Transition to Color Images: Due to the challenges faced with color transformation, we shifted our focus to working directly with color images. However, this approach also led to issues, particularly poor image generation quality.



5. GPU Colab Purchase: To address the limitations of working with color images, we decided to purchase GPU colab. This decision was aimed at improving our computational capabilities and enhancing our ability to work with complex image data.
6. Current Status: With the help of GPU colab, we have made progress in generating images. However, we are still facing challenges in achieving the desired image shapes. We are actively working on resolving this issue and refining our GAN model to improve image generation quality.

Overall, our journey with the dataset and GANs has been a learning experience, and we are continuously striving to overcome challenges and enhance our project outcomes.

# GitHub 2 May Teamboard :



Team Contribution:

| Team Member | Project Part | Contribution (%) |
|---|---|---|
| vinitwad | 1. Conducted comprehensive literature review on GANs and their applications in image processing<br>2. Explored various datasets like fashion200, zalando, MNIST<br>3. Did the set up of LMM to create a chatbot<br>4. Dedicated efforts on training model with different architectures<br>5. Fine-tuned hyperparameters for model performance optimization<br>6. Conducted model evaluation and comparison with baseline models | Coding - 33.33%<br>Documentation - 33.33% |
| shrigane | 1. Conducted comprehensive literature review on GANs and their applications in image processing<br>2. Worked on converting black and white to color images<br>3. Worked on deploying the code locally<br>4. Adjusted the LLM for the project purpose<br>5. Worked on creating tinker UI for using chatbot<br>6. Optimized deployment process for the chatbot and ensured its functionality | Coding - 33.33%<br>Documentation - 33.33% |
| gwalke | 1. Conducted comprehensive literature review on GANs and | Coding - 33.33%<br>Documentation - 33.33% |

| | | |
|---|---|---|
| | their applications in image processing<br>2. Selected Dataset fashioniq after conducting various experiments<br>3. Worked on training GAN on black and white images<br>4. Designed the final gan architecture and trained model.<br>5. Conducted extensive testing and validation of the image retrieval system<br>6. Prepared and delivered presentations on the project progress and outcomes | |

**References:**


1. X. Han, Z. Wu, Y.-G. Jiang, and L. S. Davis, ‘‘Learning fashion compatibility with bidirectional LSTMs,’’ in Proc. ACM Multimedia Conf., Mountain View, CA, USA, Oct. 2017, pp. 1078–1086.
2. J. Yim, J. J. Kim, and D. Shin, ‘‘One-shot item search with multimodal data,’’ 2018,arXiv:1811.10969. [Online]. Available: https://arxiv.org/abs/1811.10969
3. X. Guo, H. Wu, Y. Cheng, S. Rennie, G. Tesauro, and R. S. Feris, ‘‘Dialogbased interactive image retrieval,’’ in Proc. NIPS, 2018, pp. 678–688. [34]
4. S. Agarwal, O. Dušek, I. Konstas, and V. Rieser, ‘‘A knowledge-grounded multimodal search-based conversational agent,’’ in Proc. EMNLP Workshop SCAI, 2nd Int. Workshop Search-Oriented Conversational AI, 2018, pp. 1–8.
5. S. Agarwal, O. Dušek, I. Konstas, and V. Rieser, ‘‘Improving context modelling in multimodal dialogue generation,’’ in Proc. 11th Int. Conf. Natural Lang. Gener., 2018, pp. 1–6
6. https://github.com/ecom-research/ComposeAE
7.https://github.com/Pyligent/Fashion-Image-Text-Multimodal-retrieval/blob/master/README.md
8. https://archive.ph/lBNGz
9. https://www.kaggle.com/datasets/zalando-research/fashionmnist
10.https://github.com/ayush714/Fashion-Clothes-Generation-Using-GANS-/blob/main/GANS.ipynb
11. https://github.com/i008/pytorch-deepfashion/blob/master/deepfashion.py
12. https://github.com/gakash2k01/gen_shop
13.https://medium.com/tooploox-ai/ai-generated-fashion-how-we-used-generative-adversarial-networks-for-e-commerce-6221d4375b4c
14.https://maadaa-ai.medium.com/ai-datasets-for-fashion-e-commerce-open-vs-commercial-and-the-trends-2b1937f5787b