# Fuel or Fool: Decoding Your Food Choices

Pratham Mehta
Georgia Tech
pratham.mehta@gatech.edu

Xiangsheng Gu
Georgia Tech
xgu304@gatech.edu

Yu-Ling (Renee) Lu
Georgia Tech
ylu603@gatech.edu

Zeezoo Ryu
Georgia Tech
zeezooryu@gatech.edu

## Abstract

*Unhealthy dietary habits are a leading cause of chronic diseases such as obesity, diabetes, and heart conditions. To promote healthier eating, we present a novel application of Generative Adversarial Networks (GANs) that generates personalized visual representations of healthy and unhealthy meals. Our approach utilizes a conditional GAN (cGAN) framework trained on the Food-11 dataset, which we manually labeled into "healthy" and "unhealthy" categories. We incorporate advanced techniques like Wasserstein loss with gradient penalty and minibatch discrimination to enhance training stability and mitigate issues such as mode collapse. Experimental results demonstrate the model's ability to generate diverse and realistic food images conditioned on health labels at a resolution of 128×128 pixels. Despite challenges with computational resources and training instability, our work highlights the potential of GAN-based systems to encourage healthier dietary choices and provides suggestions for future research in high-resolution food image synthesis.*

## 1. Introduction

Health issues related to unhealthy diets consisting of high sugar, sodium, fat, and cholesterol, combined with a lack of exercise and frequent dining out, are the common causes of various chronic diseases today. For example, obesity, diabetes, heart diseases, and even mental health issues like depression, etc., are all possible health diseases caused by poor food choices. Maintaining a healthy diet can be challenging due to limited awareness and knowledge about nutrition. To address this, our team aims to develop a deep learning model using a Generative Adversarial Network (GAN) to generate healthy and unhealthy food images. The goal is to provide personalized visual meal options based on user responses and encourage a healthy diet. Our work prompts a user to select their preference for a healthy or unhealthy dish. Once the selected healthy food is selected, users will see the generated healthy food. If selected unhealthy, users will be exposed to the generated unhealthy food, learn about how distasteful they are, and will be geared towards making healthier dietary choices.

## 2. Related Work

The idea of implementing GAN to generate healthy and unhealthy food images is inspired by CookGAN [1], which is a GANs-based computational framework catered to the synthesis of photo-realistic food meal images from textual lists of ingredients. CookGAN employs an Attention-based Association model first to match an ingredient list and its corresponding image in a joint latent space. This mapping allows CookGAN to establish meaningful associations between ingredient characteristics and visual features. Once this association is established, the generative component, a network based on StackGAN-v2, synthesizes meal images from the latent representations of the ingredients, generating images that reflect the appearance and spatial arrangements of multiple ingredients.

In our project, we extend this approach by using a conditional GAN (cGAN) [2], where both the generator and discriminator are conditioned on an additional label specifying whether the food is healthy or unhealthy. Furthermore, to improve the diversity and quality of the generated images, we incorporate the minibatch discrimination technique proposed by Salimans et al. [3]. This allows the discriminator to consider relationships between samples in a minibatch, helping to prevent mode collapse and encourage the generator to produce a wider variety of images. Additionally, to quantitatively assess the quality of the generated images, we employ the Fréchet Inception Distance (FID) metric as introduced by Heusel et al. [4]. The FID score measures the similarity between the distributions of the generated images and real images, providing a robust evaluation of how

closely our synthetic images resemble actual food images. By conditioning image generation on labeled data, cGAN is well-suited for this application, allowing us to generate good images based on the healthiness of the ingredients.

The classification of foods as "healthy" or "unhealthy" is central to public health initiatives and has evolved significantly over the past few decades. Early nutrition guidelines primarily focused on broad dietary advice, encouraging consumers to eat more or eat less of specific food categories, like fruits and vegetables versus sugary or fatty foods. However, with the rise of processed foods and consumer confusion over dietary guidelines, attention has shifted to more food-specific labeling to assist consumers in making healthier choices. One of the primary tools developed by the UK Food Standards Agency to address this need is the Single-Score model, combining "positive" scored nutrients (e.g., oats, brown rice, chicken breast, olive oil, carrots) with "negative" ones (e.g., processed meats, glucose, butter) to yield a single score that can be directly applied as classification thresholds [5]. This approach inspired us in defining "healthy" and "unhealthy" categories for our training dataset.

## 3. Technical Approach

### 3.1. cGAN

We propose using a Conditional Generative Adversarial Network (cGAN) to address the problem of generating food images based on user preference for healthy or unhealthy food. A cGAN is an extension of a standard GAN, where both the generator and the discriminator are conditioned on additional information — in this case, the label indicating whether the food should be healthy or unhealthy. The generator will take random noise and a label (healthy/unhealthy) as input and generate corresponding images, while the discriminator will classify whether the input image is real or generated, taking into account the condition (label), which will be created from the dataset using machine learning to implement dimensionality reduction by condensing the nutritional features into a single label.

The design of the Conditional GAN would require us to train two neural networks, as is standard for GANs, the Generator network, whose job is to generate realistic images that are as indistinguishable from real images as possible, and the Discriminator network, whose job is to try to identify the generated images from the real ones, both of which will be trained with an additional condition of the healthy/unhealthy labels. The Generator takes the feedback from the Discriminator network and refines itself. This process goes on iteratively until we reach the desired results (i.e., the generated output is actually food). Due to this, we believe that the proposed algorithm/model is appropriate for this problem statement. We plan to use the loss function described in the paper for conditional GANs, which states: The loss function for the Conditional GAN (cGAN) [2] is given by:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{\text{data}}(x|y)}[\log D(x|y)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z|y)|y))]$$

Where:

- $D(x|y)$ is the probability that the discriminator assigns to real data $x$ conditioned on $y$,
- $G(z|y)$ is the generator's output, which generates data conditioned on $y$ from a noise vector $z$,
- $p_{\text{data}}(x|y)$ is the distribution of real data conditioned on $y$,
- $p_z(z)$ is the distribution of noise used as input to the generator.

The discriminator's loss function is:

$$\mathcal{L}_D = -\mathbb{E}_{x\sim p_{\text{data}}(x|y)}[\log D(x|y)] - \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z|y)|y))]$$

The generator's loss function is:

$$\mathcal{L}_G = -\mathbb{E}_{z\sim p_z(z)}[\log D(G(z|y)|y)]$$

In these equations, the generator aims to minimize the loss, while the discriminator tries to maximize it, following the adversarial setup.

Considering our later switch to Wasserstein loss [6], the Wasserstein loss for Conditional GANs (cGAN) is given by:

$$\min_G \max_{D\in\mathcal{D}} V(D,G) = \mathbb{E}_{x\sim p_{\text{data}}(x|y)}[D(x|y)] - \mathbb{E}_{z\sim p_z(z)}[D(G(z|y)|y)]$$

Where:

- $D$ is the critic (or discriminator) function, outputting a scalar score instead of probabilities,
- $\mathcal{D}$ is the set of 1-Lipschitz functions, ensuring the Lipschitz constraint via weight clipping or gradient penalty.

The critic loss function is:

$$\mathcal{L}_D = -\mathbb{E}_{x\sim p_{\text{data}}(x|y)}[D(x|y)] + \mathbb{E}_{z\sim p_z(z)}[D(G(z|y)|y)]$$

The generator loss function is:

$$\mathcal{L}_G = -\mathbb{E}_{z\sim p_z(z)}[D(G(z|y)|y)]$$

To enforce the Lipschitz constraint, a gradient penalty is applied, defined as:

$$\mathcal{L}_{\text{GP}} = \lambda \cdot \mathbb{E}_{\hat{x} \sim p_{\hat{x}}} \left[ (\|\nabla_{\hat{x}} D(\hat{x}|y)\|_2 - 1)^2 \right]$$

Here:

- $\lambda$ is a hyperparameter that controls the strength of the gradient penalty,
- $p_{\hat{x}}$ is the distribution of interpolated samples between real and generated data.

The final discriminator (critic) loss with the gradient penalty is:

$$\mathcal{L}_D = -\mathbb{E}_{x \sim p_{\text{data}}(x|y)}[D(x|y)] + \mathbb{E}_{z \sim p_z(z)}[D(G(z|y)|y)] + \mathcal{L}_{\text{GP}}$$

### 3.2. Model Implementation

For the initial setup and testing, we developed a streamlined conditional GAN (cGAN) architecture tailored for 64x64 images labeled as "healthy" and "unhealthy." This preliminary design served as a proof of concept, aimed at verifying the core components of the GAN and generating initial results.

Our generator network was intentionally shallow, beginning with a fully connected layer to process input noise and class labels, followed by three convolutional layers. Each convolutional layer was equipped with Batch Normalization and ReLU activation functions to enhance feature extraction and maintain stability during training.

The discriminator network, likewise, adopted a minimalistic structure with two convolutional layers, each incorporating Batch Normalization and LeakyReLU activations, followed by a fully connected layer to output the final classification. This design aims to keep the architecture lightweight and efficient for testing purposes.

For the adversarial training loss, we utilized binary cross-entropy with a sigmoid activation, allowing the model to differentiate real and synthetic samples accurately. This simple but effective loss function helped establish a stable training foundation for both the generator and discriminator.

Once we utilized this initial setup to verify the working of all the modules of the GAN as well as test and qualitatively evaluate some initial generated results, we set out to resolve some of the problems we identified through this test, namely mode collapse and occasionally exploding gradients, and to increase the generated image size as well as make the network architecture deeper to increase the quality of the generations. The architecture is visualized in Figure 1.

To ensure that we could quantitatively judge the generations of our models, we implemented the Fréchet Inception Distance (FID) [7] as a measure. This helped us to test out different approaches to see whether our model was improving, especially in cases where it was difficult to judge the differences qualitatively.

Once we had the measures to test the model in place, we started with our attempts to improve the GAN. In the first refinement phase, we adopted principles from Wasserstein GAN with Gradient Penalty [6]. We replaced the traditional adversarial loss utilized in our GAN to a Wasserstein Loss which gave real number values as a loss instead of a probabilistic value as is the case with traditional GANs and also implemented a gradient penalty system. This was done to improve the stability of our model and also because it is known to help reduce mode collapse issues. To further increase the stability we also added Xavier initialization to all the trainable which refers to the convolutional layers and the fully connected layers. With these changes in place, we began the next testing phase for this attempt. Our previous attempt, though successful to some extent, suffered severely from mode collapse with only minor changes in subsequent generations. To identify whether this was still an issue mid-way through training, we also added code to generate a grid of 16 images so that we can compare several generations side by side and stop the training early in case of issues. Through the testing for this phase, we were able to successfully overcome mode collapse for 64x64 images and achieved a stable generation that had diverse generations every time we entered our choice of label ('healthy' or 'unhealthy'). We also achieved an increase in the image quality.

Once the testing for this phase concluded, we set out to work on our next attempt which was to accomodate the principles of Deep Convolutional GAN (DCGAN) [8]. For this phase we set out our plan to initially implement these principles, make our network deeper, and then test it out on the 64x64 images to see the performance before switching over to higher resolutions. For doing this we followed the principles specified in the paper such as using only ReLU in the generator architecture and only LeakyReLU in the discriminator. We changed our generator and discriminator networks to have 5 convolutional layers as well. Through the testing of this phase we realized that a DCGAN needed a larger number of epochs compared to our initial network to produce decent generations, and that the generations this time were more detailed compared to the previous attempts and took longer time for both training as well as generating. This concluded our testing for this phase.

Moving onto our next phase, we decided to make the objective of this phase to generate 256x256 images instead of 64x64 which would bring us closer to our goal of achieving realistic generations. To also accomodate this large change in resolution, we changed the filter sizes in both our generator and discriminator networks. With this

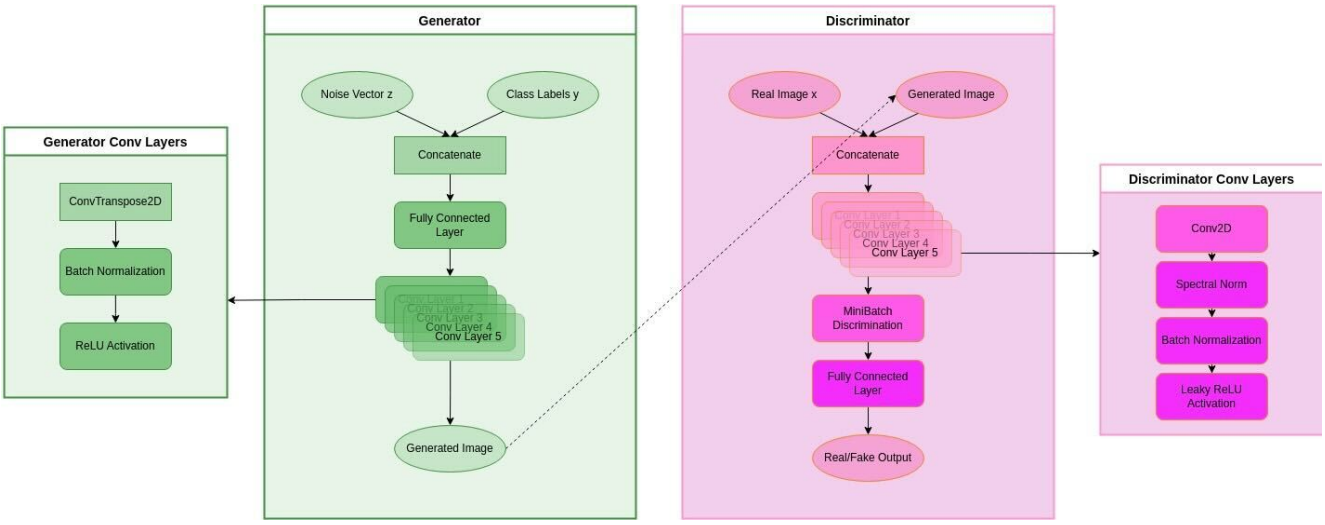



Figure 1. Architecture of a Generative Adversarial Network (GAN), highlighting the interplay between the Generator and the Discriminator. The Generator transforms a noise vector and class labels into synthetic images, while the Discriminator evaluates both real and generated images to guide the Generator's improvement through iterative training.

minimal change, we started the testing phase immediately since we anticipated it would take several days to train, considering that 256x256 images contain 65536 pixels each compared to only 4096 pixels each in 64x64 images. The testing phase was proceeding according to our expectations with some epochs taking over thirty minutes. We realized early on that the training may not finish in time at this rate. Despite that we decided to leave the training on for a while to at the very least collect intermediary results before we made a decision on whether to pull the plug on this phase. Before we were able to reach a point where it generated food-like images, we were forced to stop due to memory limitations on the test machines. This concluded this phase.

To address these limitations, we reduced the resolution to 128x128 images and tried training the network in hopes that we would be able to sustain this training for longer and be able to train faster. This, however, made us realize that at higher resolutions mode collapse issues resurfaced, which had previously been mitigated by increasing stability through the use of Wasserstein Loss and gradient penalties. To resolve this issue once and for all, we implemented MiniBatch Discrimination [3]. In this approach, the discriminator evaluates images in small batches instead of individually. If the discriminator encounters several similar looking generations within the same batch it automatically rejects those as fake samples which helps prevent any form of mode collapse. With these enhancements, we began our final phase of testing and training the model.

## 4. Data

As outlined in our proposal, we aim to preprocess the dataset by applying a binary classifier to label each ingredient as "healthy" or "unhealthy" based on its nutritional values. Then, these labels will be used as input for the GAN model in the deep learning stage. To achieve this, we used three datasets to implement our approach: The "Food Ingredients" dataset [9], The "Food Ingredients and Recipe Dataset with Image Name Mapping" dataset [10], and the "Food-11" dataset [11].

The "Food Ingredients" dataset provides comprehensive nutritional information for food items, detailing calories, protein, fat content, and other key metrics essential for nutritional analysis. This data allows us to build a healthiness dictionary, categorizing ingredients as healthy or unhealthy based on nutritional profiles. The "Food Ingredients and Recipe Dataset with Image Name Mapping" dataset [10] is then used to incorporate real-world recipes with ingredient lists, preparation steps, and corresponding images. These enable the development and evaluation of a robust recipe classification model while supporting our GAN model in generating realistic visual meal options, effectively bridging nutritional profiling and practical application.

However, after exploring several preprocessing techniques, including unsupervised learning, Single-Score model, and nutrient-profiling schemes developed by health professionals, we were not able to generate unbiased labeling due to the complexity of the nutrition information and the need for a diverse dataset. Thus, we manually labeled the "Food-11" dataset, containing 16643 food images grouped in 11 major food categories such as bread, dessert,

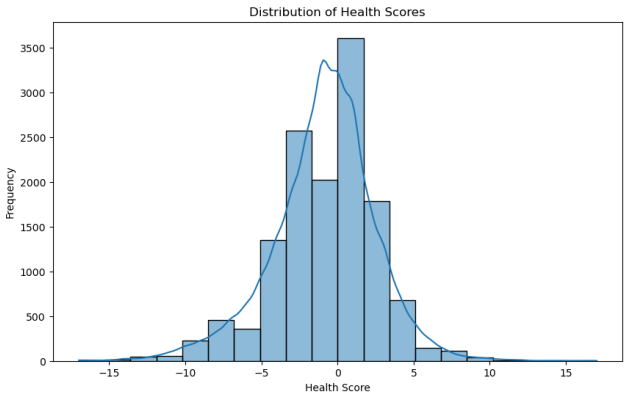



Figure 2. The distribution of calculated health scores is roughly symmetrical. Food recipes tend to be moderately balanced but may include a mix of both healthy and unhealthy ingredients. Most recipes don't lean strongly towards either extreme, with only a few highly healthy or unhealthy recipes.

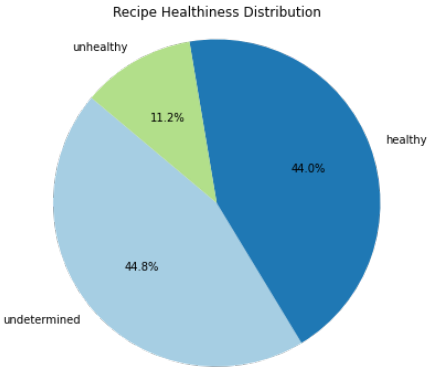



Figure 3. Distribution of recipe classifications into healthy, unhealthy, and undetermined categories using the nutrition profiling scheme.

diary products, etc., into healthy and unhealthy categories, and use it as our input for cGAN.

Below shows our attempts to generate accurate healthiness labeling for each ingredient.

### 4.1. Unsupervised Learning

The first approach we explored is using an unsupervised learning model, specifically KMeans clustering, to classify food ingredients into two categories. We chose KMeans clustering because it is a fast, simple model that works well with large data, and it allowed us to specify the desired number of clusters (in this case, 2 clusters: healthy and unhealthy). However, we discovered that, rather than separating ingredients by healthiness, the generated KMeans clusters separate light, snack-type ingredients from meal components like meat and carbohydrates. Additionally, the Silhouette score of the KMeans model is 0.21, showing that it does not capture meaningful patterns in the dataset efficiently. As a result, we decided to explore alternative methods to preprocess the dataset.

### 4.2. Single-Score Model

The second approach we explored is the Single-Score model. We defined functions to classify a subset of recipes from the Recipe1M dataset as healthy or unhealthy. First, we compiled two lists of ingredients associated with positive and negative nutritional characteristics. Next, for each recipe, each ingredient was assigned a single health score based on predefined indicators. Then, it produced a single numerical health score combined for each recipe, representing the overall nutritional profile of the food: A higher score indicates a healthier recipe, while a lower score suggests an unhealthy recipe. The resulted dataset has a balanced variety of ingredients, with some recipes containing mostly unhealthy ingredients and others favoring healthier components (Figure 2). However, the exact nutritional values for each ingredient are not taken into account in this approach, making the healthy/unhealthy label not completely accurate or reliable.

### 4.3. Nutrition Profiling Scheme

To accurately classify recipes as "healthy," "unhealthy," or "undetermined," we implemented a nutritional profiling model inspired by Lobstein et al. [5]. Our classification system uses the Food Ingredients and Recipe Dataset combined with an updated cleaned ingredients list that includes labels for healthiness. This dataset provides a comprehensive collection of recipes, each with a list of ingredients, preparation instructions, and metadata. The diverse set of real-world recipes allows us to better understand how combinations of ingredients contribute to the overall healthiness of a recipe.

We first preprocess each ingredient to remove punctuation, convert text to lowercase, and split the string into individual words. This ensures consistency in how ingredients are compared and minimizes discrepancies caused by variations in formatting. Next, we employ set intersections to quickly check if ingredients from the recipes match entries in our healthiness dictionary. If at least 60% of the ingredients in a recipe are classified as "healthy," we label the recipe as "healthy." If there are no matches at all, the recipe defaults to "undetermined." This method allows us to classify recipes efficiently while still considering the balance of healthy and unhealthy ingredients. Lastly, we selected a 60% threshold after testing different values to account for the mixed nature of most recipes, which often contain both healthy and unhealthy components. Adjusting this ratio was crucial for improving classification accuracy and ensuring our model made more balanced decisions.

Figure 3 shows the results of using this method for labeling the healthiness of recipes. Our initial tests revealed a strong bias toward labeling recipes as "unhealthy." To address this, we refined our approach by adding partial matching capabilities and adjusting the healthiness ratio to be more inclusive. While these changes made a significant difference, we also found that our healthiness dictionary needed to be more comprehensive to cover a broader range of ingredients and their variations. Additionally, we realized that incorporating semantic similarity measures could help our model better capture the nuances of different ingredient names, reducing misclassifications. These ongoing refinements are critical for integrating our nutritional profiling model seamlessly into our GAN framework.

However, due to the project scope and the limited time we had, we were unable to build a comprehensive dictionary that can accurately capture the complexity of nutrition information, as well as continuing with the refinements. We also faced some challenges in linking the ingredient information to their corresponding images. Therefore, our team decided to utilize the Food-11 image dataset [11] and manually label the food categories into healthy/unhealthy categories. By limiting the scope to just 11 categories (e.g., bread: "healthy", desserts: "unhealthy", etc.), we were able to define a more focused scope for our GAN model, which allowed us to concentrate on the development of the model and fine-tuning it for a better performance.

## 5. Results

We conducted several training experiments on different subsets of the dataset and across varied epochs. For the initial testing phase, images in the dataset were first downsampled and normalized to 64x64 using ffmpeg, allowing for faster training and efficient model testing. The first cGAN model was trained on an image resolution of 64 x 64 pixels, with a subset of 960 images (480 images for each label) for 1,000 epochs using the shallow test network. This training was conducted on a laptop GPU with a batch size of 32. The conditional generation results from this setup are illustrated in Figure 4a and Figure 4d

In contrast, the second model was trained on a larger subset of 3,840 images (1920 images for each label) for a total of 10,000 epochs, using a desktop GPU with a batch size of 128. Results from this model's conditional generation are displayed in Figure 4b, Figure 4c, Figure 4e, and Figure 4f.

This initial setup effectively verified the functionality of the GAN model, confirming the operational integrity of its core components. However, notable challenges arose, particularly with mode collapse in the smaller model, where subsequent generations appeared highly similar. While the larger model showed less resemblance among generated images, the overall diversity remained limited. This test helped validate the setup and identify areas for enhancement moving forward.

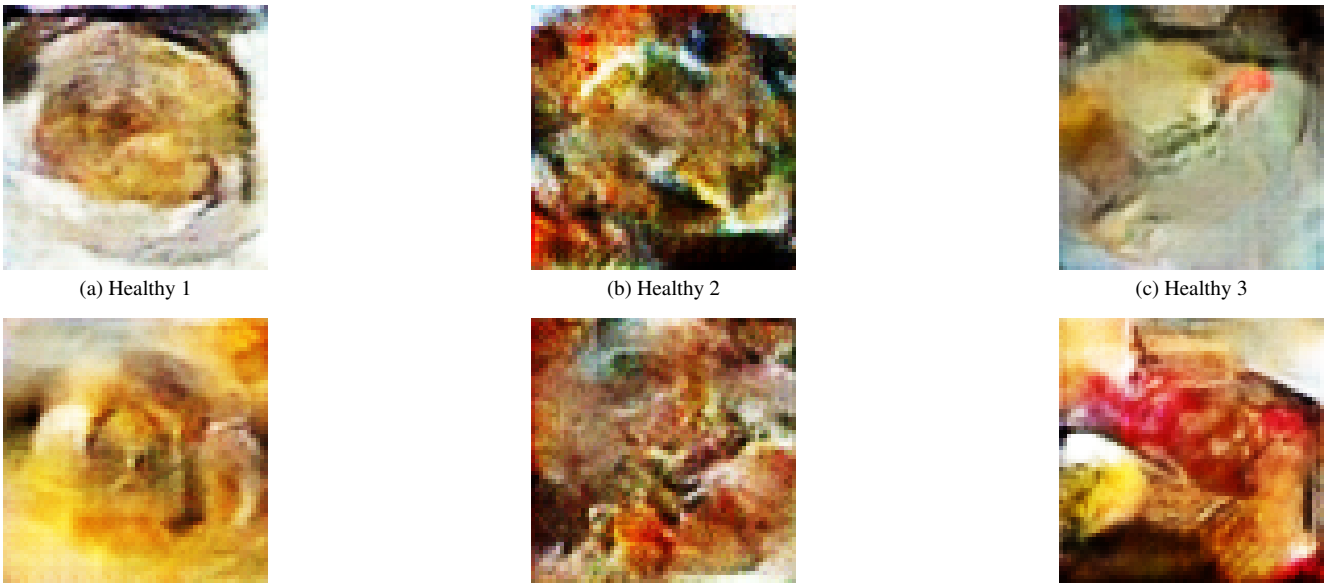
In our experimental results, as illustrated by Figure 5, we observed the improvements and challenges in training our conditional GAN model. Figure 5a shows sample images from our training dataset at a resolution of 64x64 pixels, serving as a baseline for evaluating the quality of generated images. In Figure 5b, we present images generated by our initial cGAN model without the implementation of minibatch discrimination or advanced stabilization techniques. While the images exhibit some food-like structures, they lack clarity and diversity, indicating issues with mode collapse and insufficient training stability.

To improve our model, we incorporated Wasserstein loss with gradient penalty and increased the depth of our network, and the results can be seen from Figure 5c. With the improvements, we scaled up the image resolution to 128x128 pixels and further refined our model by adding minibatch discrimination to mitigate mode collapse. Figure 5d showcases the best images generated at this resolution, displaying more detailed and more realistic food representations.

Despite these improvements, as the training proceeded, we encountered challenges with training stability. In Figure 5e, the images begin to show signs of degradation, with reduced clarity and emerging artifacts, which occurred in the 2030th epoch. At epoch 2050 as seen by Figure 5f, the generated images seemed like white noise and it seemed like the model was re-initialized. We hypothesize that this decline is attributed to the generator experiencing instability after a large number of epochs as this attempt was run for 3100 epochs. Despite continuous training without re-initialization of model weights, the generator started to produce less coherent outputs. This phenomenon can also be due to issues like overfitting, where the generator becomes too specialized on the training data, or the imbalance between the generator and discriminator learning rates, leading to the discriminator overpowering the generator. Furthermore, we can also speculate factors such as vanishing/exploding gradients or improper handling of gradient penalty and minibatch discrimination.

By analyzing the results from Figure 5, we highlight the importance of maintaining training stability throughout the entire training process. Addressing the issue of careful hyperparameter tuning, regular monitoring of the training process, implementing early stopping mechanisms or learning rate schedules to prevent overfitting, and maintaining a balance between the generator and discriminator will be crucial for generating high-quality images in GANS, especially when training over many epochs.

Figure 6 illustrates that both generator and discriminator test losses are relatively low in the early epochs, representing the initialization phase where neither network is particularly effective. As training progresses, losses fluctu-



(a) Healthy 1 (b) Healthy 2 (c) Healthy 3

(d) Unhealthy 1 (e) Unhealthy 2 (f) Unhealthy 3

Figure 4. Examples of healthy and unhealthy food images (64x64) generated by the preliminary shallow model.

(a) Sample Images from Training Set (b) 64x64 without Minibatch Discrimination (c) 64x64 with Minibatch Discrimination

(d) 128x128 Best Generated Image (e) 128x128 Right before Gradient Issues (f) 128x128 Image after Gradient Issues

Figure 5. Generated images by the different attempts of the cGAN models.

| Model Config. | Resolution | Epochs | G Loss | D Loss | Batch Size | FID |
|---|---|---|---|---|---|---|
| Baseline cGAN + Adversarial Loss | 64×64 | 1000 | 1.3470 | 0.8183 | 32 | 311.53 |
| cGAN + W-Loss | 64×64 | 3000 | -32.6031 | 488.3872 | 64 | 204.18 |
| DCGAN + W-Loss | 64×64 | 10000 | -6.4623 | 153.4116 | 32 | 203.82 |
| DCGAN + MD + W-Loss | 64×64 | 10000 | 38.7270 | 82.4689 | 32 | 202.57 |
| DCGAN + MD (Best Image) + W-Loss | 128×128 | 3000 | 1745.0259 | 850.5171 | 32 | 418.12 |

Table 1. FID scores for different GAN models. W-Loss: Wasserstein Loss, MD: Minibatch Discrimination.

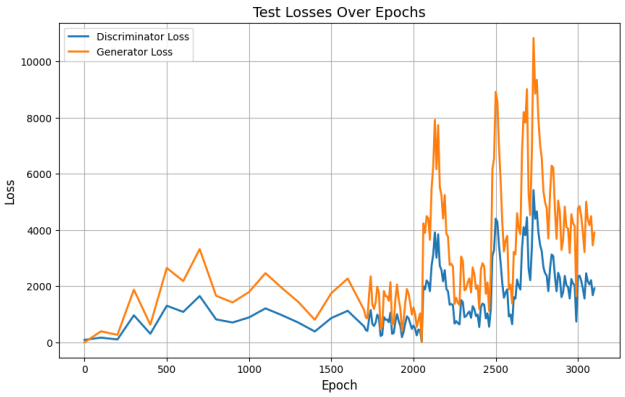



Figure 6. Discriminator and generator test losses change over epochs 0 to 3100 for images of resolution 128x128.

ate due to the adversarial nature of GANs. In later epochs, the losses tend to stabilize, suggesting that the generator and discriminator are approaching equilibrium. However, persistent oscillations in the graph indicate the inherent instability of GAN, which is a common challenge that may require improved architectures or regularization.

From the results shown in Table 1, we observe that improvements to our conditional GAN (cGAN) architecture led to a slight reduction in FID scores when using 64×64 images. However, the FID scores increased significantly when using 128×128 images. This can be attributed to the calculation of FID scores on higher-resolution images, as the increase in pixel count makes it more challenging for the generator to replicate the fine-grained details of real images, resulting in a higher score (indicating greater distance from real images).

Additionally, the lack of a drastic improvement in FID scores could be due to the high variability in the food dataset, as well as similarities between the two classes. These factors likely require the cGAN to be trained for a much larger number of epochs than we were able to train it on to effectively learn the underlying patterns and generate images with lower FID scores that are closer to real images.

## 6. Conclusion

In this project, our objective is to develop a Generative Adversarial Network (GAN) that can effectively generate healthy and unhealthy food images, aiming to encourage healthier food choices. We leveraged the Food-11 image dataset, manually labeling each food category into healthy or unhealthy, and utilized the labels as the condition for training our cGAN model. After a series of iterations, including training and fine-tuning, we successfully obtained the best food images by incorporating DCGAN and minibatch discrimination with an image resolution of 128x128 pixels into our model. However, despite achieving promising results, our model still exhibits some limitations, including mode collapse and unrealistic food images. During the training process, our team faced several challenges, including insufficient computing power, which hindered our ability to further refine the model through additional training on higher-resolution images (256x256) or building a deeper architecture. We also observe some noises in generated images, showing that the model is still somewhat unstable despite implementing noise regulations and minibatch discrimination.

Despite these challenges, our project demonstrates the potential of GANs for generating realistic food images. Future works can build upon this foundation by exploring alternative architectures and techniques, such as PacGAN or bigGAN frameworks, to investigate model stability and performance. Additionally, utilizing a high-performance computing system will be essential for training more complex models and achieving higher image resolutions. To further reduce mode collapse and build a more stable model, utilizing a diffusion model could be a solution to address some of the challenges we faced during model training, such as mode collapse and the unstable training process. The architecture of a diffusion model can reduce mode collapse and generate a robust result through a probabilistic representation of the data distribution.

In conclusion, while our project faced some challenges in generating realistic images using the GAN architectures, we believe that the insights gained from this work will contribute to future research in deep learning-based food image synthesis that can promote healthier dietary choices.

## 7. Team Contributions

| Team member | Contributions |
|---|---|
| Zeezoo Ryu | Data prerocessing, experimenting with different model architectures for fine-tuning, poster making, result analysis, report writing |
| Yu-Ling Lu | Data preprocessing, experimenting with different model architectures for fine-tuning, poster making, result analysis, report writing |
| Pratham Mehta | Designing technical strategies, data preprocessing, model creation, reviewing literature to refine model, result analysis, report writing |
| Xiangsheng Gu | Data preprocessing, experimenting with different model architectures for fine-tuning, result analysis, report writing |

## References


[1] Fangda Han, Ricardo Guerrero, and Vladimir Pavlovic. Cookgan: Meal image synthesis from ingredients. In *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1439–1447, 2020. 1

[2] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets, 2014. 1, 2

[3] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans, 2016. 1, 4

[4] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017. 1

[5] T Lobstein and S Davies. Defining and labelling 'healthy' and 'unhealthy' food. *Public Health Nutrition*, 12(3):331–340, 2009. 2, 5

[6] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein gans, 2017. 2, 3

[7] Ayush Thakur. How to evaluate gans using frechet inception distance (fid). https://wandb.ai/ayush-thakur/gan-evaluation/reports/How-to-Evaluate-GANs-using-Frechet-Inception-Distance-FID---Vmlldzo0MTAxOTI, 2023. Accessed: 2024-11-04. 3

[8] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks, 2016. 3

[9] The Devastator. Now with more nutrients. https://www.kaggle.com/datasets/thedevastator/now-with-more-nutrients, 2021. Accessed: 2024-11-04. 4

[10] PES12017000148. Food ingredients and recipe dataset with images. https://www.kaggle.com/datasets/pes12017000148/food-ingredients-and-recipe-dataset-with-images, 2020. Accessed: 2024-11-04. 4

[11] Aleksandr Antonov. Food-11 image dataset. https://www.kaggle.com/datasets/trolukovich/food11-image-dataset/data, 2019. Accessed: 2024-11-04. 4, 6