Plate2Recipe: Converting Food Images to Recipes

Nellie K. Cordova ncordova7@gatech.edu
Dennise Vieyra dennise@gatech.edu
Eric Connelly hieu7@gatech.edu
Denis Koshelev koshelev@gatech.edu

Abstract

The "Plate2Recipe" project addresses the compelling challenge of transforming food images into detailed cooking recipes, a tool of great interest to culinary enthusiasts and professionals alike. The main objective is to explore various architectures to learn and determine whether one can build a complex system using them. Utilizing a Vision Transformer (ViT), our approach first classifies the ingredients depicted in food photographs. Subsequently, a specialized NLP model synthesizes these ingredients into coherent, actionable cooking recipes. We fine-tune various models, experiment with hyperparameters, and evaluate our results by identifying key mistakes made during training.

1. Background

In today's digital world, many people encounter visually appealing food online and wish to recreate these dishes but lack the necessary details on ingredients and cooking methods [9]. The goal of this project is to allow users to upload an image of a dish and receive a complete recipe for it.

Traditional methods, such as online searches or simple recipe apps, often rely heavily on the user's ability to describe the dish and may fail to provide comprehensive information, especially for complex dishes or when the image quality is low. As deep learning algorithms grow more widespread, more people are increasingly acknowledging the benefits of interacting with these systems to enhance our creativity and performance [7]. This project simplifies the process of learning to cook new dishes, helps people manage their diets by showing them exactly what goes into their food, and encourages creativity in the kitchen.

To build this system, we used large databases such as Food-101 and Recipe1M+, which contain over one million recipes linked to 13 million images, and the RecipeNLG dataset. These resources were chosen to train the system to recognize and accurately generate diverse recipes.

2. Approach

The project involves two main components:

1. ***Ingredient Classification*** from a food image using Vision Transformer (ViT) model as introduced in Dosovitskiy et al (2021) [5]

2. ***Recipe Generation*** using NLP Models (GPT-2 and LSTM) based on the list of ingredients obtained from the ViT model.

We integrate Vision Transformer (ViT) and Natural Language Processing (NLP) models to transform images into comprehensive cooking recipes:

1. **Food Label Identification with ViT**. The model analyzes food images, classifying the food title label directly from the image. We segment the image into fixed-size patches, embedding these linearly, and then adding position embeddings. Then they are processed through a Transformer encoder trained with an extra learnable "classification token" to identify food types. This step is crucial as it sets the foundation for the accurate generation of the recipe by providing a precise list of ingredients.

2. **Ingredient Feature Extraction with ViT and Ingredient Encoder and Decoder**. Extracting ingredients from food image presents a huge challenge due to the complexity and variability of food image. The task of determining the complete list of ingredients that go into the food requires the model to have deep understanding of the characteristic and textures of the food. We aim to accomplish this following the same methodology outlined in Chhikara et al 2023 [4]. First we employ ViT's attention mechanism in order to extract the image's features. The feature extractor produces the image embeddings which is then fed into three normalization layers and the ingredient decoder which is responsible for extracting ingredients. The decoder consists of 4 consecutive blocks each consists of multiple sequential layers:self-attention, conditional attention, two fully connected layers, and three normalization layers.

3. **Recipe Generation:**

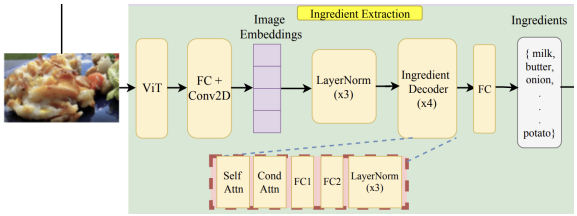

Figure 1: Ingredient Extraction Process [4]

Figure 2: Training image

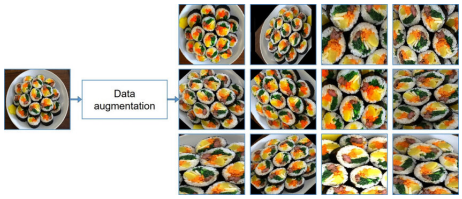

Figure 3: Image Augmentation

- **GPT-2 with Language Modeling Head** generates creative draft recipes based on identified ingredients, using a fine-tuned GPT-2 model on our dataset.
- **LSTM for Structured Text Generation** We also train LSTM on the RecipeNLG dataset as an alternative version of the recipe. The LSTM ensures a logical sequence and consistency in recipe steps, all ingredients are properly incorporated and the instructions flow logically from one step to the next.

The final step in our workflow is the delivery of the complete recipe. This includes the recipe title, required ingredients and quantities, and detailed cooking instructions, providing a comprehensive guide to recreate the dish.

3. Datasets

We use Huggingface Datasets which provides a one-liner command to download and efficiently pre-process any datasets in any major public databases. It is also designed to let the community easily add and share new datasets. It has native support for image file and allow streaming mode to save disk space. Instead of downloading the large dataset all at once, Huggingface Dataset loader allows user to iterate over the dataset one by one.

3.1. Food101 Image

The Food101 contains images of food, organized by type of food. It was used in Bossard et al 2014 [3]. It consists of 101 food categories with 750 training and 250 test images per category, which means the dataset has a total of 101k images. The labels for the test images have been manually cleaned. On purpose, the training images were not cleaned, and thus still contain some amount of noise. This comes mostly in the form of intense colors and sometimes wrong labels.

We used it to train ViT Food Classification model. We first split the data set into training and test set with the split ratio of 80/20. We then resize each image to have maximum width and height to be 224. This will later allow us to divide up the image into patch of 16 x 16.

We process each image by resizing them to a particular size and then normalizing the color channels (R,G,B) using a mean and standard deviation. These often are referred to as image transformations. In addition to this, we also perform data augmentation during training such as random cropping and flipping or rotating. Data augmentation make the model more robust to noise and achieve higher accuracy during validation and inference.

3.2. Recipe1M

The Recipe1M+ is a large-scale, structured dataset that contains over one million structured cooking recipes, each linked to images, totaling approximately 13 million images.

We used it to train Ingredients Extractor model. For preprocessing, we actively remove images that cannot be opened using PIL library and folders that are empty. We then follow the same approach we used for processing Food101 Image.

3.3. RecipeNLG

The RecipeNLG dataset, with over 2 million cooking recipes, was introduced in (Bień et al., INLG 2020) [2]. This dataset builds upon the prior Recipe1M+ dataset by incorporating additional and corrected entries gathered from a variety of cooking websites.

We selected a random subset of recipes to manage computational resources and speed up training. Each recipe was formatted into a single text block containing a title, ingredients list, and cooking instructions, providing a structured context for text generation by the GPT-2 and LSTM models.

The data was then divided into training and validation sets with an 80/20 split. This division ensures a robust train-

ing environment for the model while maintaining a separate set for validation and fine-tuning, ensuring the model's effectiveness in generating novel recipe texts based on learned patterns.

Utilizing both visual and textual data provides a holistic approach to understanding and generating content related to food, potentially enhancing the model's ability to produce accurate and contextually relevant outputs. The approach to managing large datasets through selective sampling and structured preprocessing ensures that the training process is feasible and efficient without compromising the quality of model outputs.

4. Food Name Classification + Ingredients Extraction

4.1. Vision Transformer (ViT) model

Transformer architecture was first introduced in Vaswani et al (2017) [8]. In the context of Natural Language Processing, The Transformer model extract features for each word using a self-attention mechanism to figure out how important all the other words in the sentence are with respect to the aforementioned word.

For this part of the project, we will experiment with using the Vision Transformer (ViT) model to classify the list of ingredients that go into the image of the food.

First, we will split the food image into fixed-size patches. We then linearly embed each of them, and add position embeddings. Then we feed the resulting sequence of vectors to a standard Transformer encoder. During the classification task, we will adopt the standard approach of adding an extra learnable "classification token" to the sequence

For this part, we followed the setup instruction in the vision_transformer. We also borrowed the code from Vision-Transformer-based-Food-Classification and Indian Food Image Classification to pre-process the images and finetune the model. We also use the supplementary code from Chhikara et al 2023 [4] to train and finetune the model to extract the list of ingredients from the food image.

4.2. Finetuning pretrained ViT model

We obtained the Vision Transformer (ViT) model pretrained on ImageNet-21k (14 million images, 21,843 classes) at resolution 224x224 from google/vit-base-patch16-224-in21k. This model was introduced in Dosovitskiy et. al 2020 [5]

We finetuned the pretrained model on the Food101 dataset using T4 GPU from Google Colab. Below is the configuration of the finetuning:

The finetuned ViT model is then used to classify the food name label for a random food image obtained on the internet. The random image is fed into the feature extractor to obtain the image encoding.

Parameter	**Value**
Training Batch Size	16
Evaluation Batch Size	16
LR Scheduler Type	linear
LR Scheduler Warmup Ratio	0.1
Gradient Accumulation Steps	4
Total Training Batch Size	64
Total Number of Training Epochs	4
Learning Rate	2e-4

Table 1: ViT Training Parameters

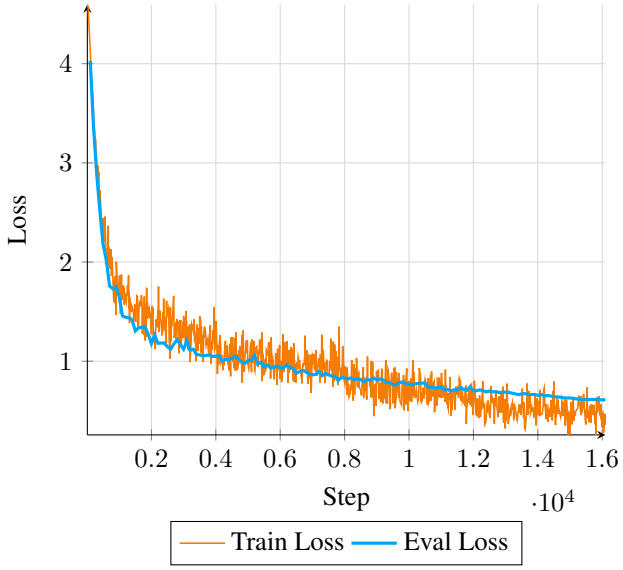

Figure 4: Training and Evaluation Loss per Step

As shown in Figure 4, both the training and evaluation loss consistently decrease as the number of steps increases. This indicates that the model is effectively learning and improving its predictions over time. Concurrently, as depicted in Figure 5, the evaluation accuracy increases with the number of steps. This improvement in accuracy alongside the reduction in loss underscores the model's increasing proficiency in accurately classifying and predicting outcomes as it processes more data.

We also trained the ingredient extraction model for 10 epochs with a batch size of 200 with Adam optimizer and a learning rate of 10e4. At each epoch, we decrease the learning rate by 0.02%. Each input image was also resized to a dimension of 224x224x3. The output is the label and ingredients which matches what the image is showing.

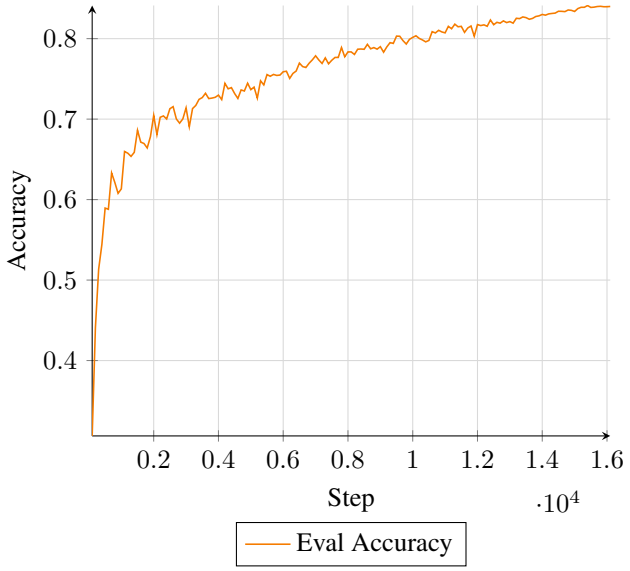

Figure 5: Evaluation Accuracy per Step

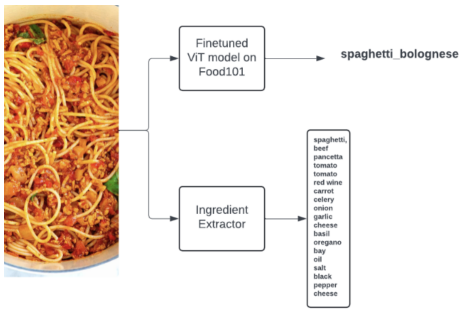

Figure 6: ViT Result

5. Recipe Generation

5.1. GPT-2 Model

The second component of our project involves a language model designed to generate recipes based on ingredients. The GPT-2 model, developed by OpenAI, excels in generating coherent and contextually relevant text. We fine-tune a pre-trained GPT-2 model using the extensive RecipeNLG dataset.

5.2. Fine-Tuning pre-trained GPT-2

We selected the GPT-2 medium model [6] for this purpose. Despite being considered relatively modest in size and capability by 2024 standards, this model suits our needs for several reasons. Firstly, its manageable size allows us to fine-tune it on available hardware without the extensive computational resources required by larger models, although the training process still demands several hours. Secondly, the pre-trained version of GPT-2 does not generate recipes from ingredients unlike newer models as seen on Figure 4. Our goal was to gain practical experience in fine-tuning such models. As a baseline for the training process, we used this notebook [1] that fine-tuned GPT-2 on biographies.

> Input: *beef, salt, pepper*
>
> Output: *beef, salt, pepper, paprika, dried bay leaf, dried thyme, black pepper, cayenne, paprika, oregano, and thyme. Makes about 3 cups (2 sticks or ½ pound). For the veggie burger bun: For the veggie burger bun: Directions: Bake the veggies in a pre-heated 350 degree*

Figure 7: Example of a pretrained GPT model output.

5.2.1 Data Preprocessing

To preprocess our dataset, we first ensured that all target recipes fit within the 768 embedding size limit for the GPT-2 model. Afterwards we format the recipes. Each recipe entry in the dataset consists of a title, a list of ingredients, and cooking directions. We structured the input to the model as follows:

- **Input**: A prompt string that starts with a special token `<|startoftext|>` followed by the word "ingredients:" and a comma-separated list of ingredients.
- **Target Output**: The recipe title, ingredients list, and directions, each separated by special tokens and formatted to provide clear, structured information.

We divided our dataset into training and validation sets, allocating 90% for training and 10% for validation. The complete dataset contains 2,231,141 samples; however, due to limited resources, we train our model on a subset of only 100,000 samples.

5.2.2 Training

We trained the model with the following hyperparameters:

Parameter	100k Samples	10k Samples
Epochs	3	3
Learning Rate	5×10^{-4}	2×10^{-4}
Warm-up Steps	100	500
Epsilon	1×10^{-8}	5×10^{-8}

Table 2: Training Parameters for Different Sample Sizes

Our initial experiments with a smaller sample size did not produce meaningful outputs. We initially assumed that

[1] GPT-2 Fine-Tuning w/ Hugging Face PyTorch

the problem was exclusively due to the amount of data on which we trained, so we conducted extensive training with 100,000 samples of recipes on V100 GPUs with a single epoch taking about 2 hours. However, after we completed the training and examined the results, they were not satisfactory; the model was still producing nonsensical outputs.

> Input: *rice, avocado, beef*
>
> Output: *Forest: Car Nut- Car els 2 Ingredients Car Car c Car14 Car p- Carted.) Car/els car Carrotsels Car.) Car Car Car Car...*

Figure 8: Example of poorly finetuned GPT model output on 100k samples.

At first glance, there appeared to be no issues in our code that could explain the poor performance of the model. We made several adjustments, including changing hyperparameters, reducing the dataset size, and modifying how we defined the entire dataset in our PyTorch setup. After these initial experiments, the results began to make more sense. Consequently, we trained the model on a dataset comprising 10,000 samples with the hyperparameters from Table 2. These adjustments led to significant improvements in the model's performance, indicating that the modifications were effective in addressing the initial shortcomings. The model was able to generate a meaningful recipe and use exclusively ingredients provided in the input. One can see how model generalized better this time with the sample output from each epoch showing more and more details for the recipes:

> Input: *eggs, flour, butter, sugar, vanilla*
>
> Epoch 1: *Instructions: Combine all ingredients. Pour in a 9 x 13-inch pan. Bake at 350° for 1 hour.*
>
> ...
>
> Epoch 3: *Instructions: Beat eggs until thick. Add sugar and beat until fluffy. Add flour, butter and vanilla. Blend in stiffly beaten egg whites. Spread in 9 x 9-inch pan. Bake at 325° for 30 minutes. Cool completely before cutting.*

Figure 9: Example of finetuned GPT model output on 10k samples.

In our evaluation of fine-tuning GPT-2 with 100k and 10k samples, we measured success by looking at training loss and actual performance. Although the model trained on 100k samples had lower training losses, as shown in Figure 10, the model trained on 10k samples performed consistently better, while 100k version was supposedly overfit and couldn't produce meaningful recipes.

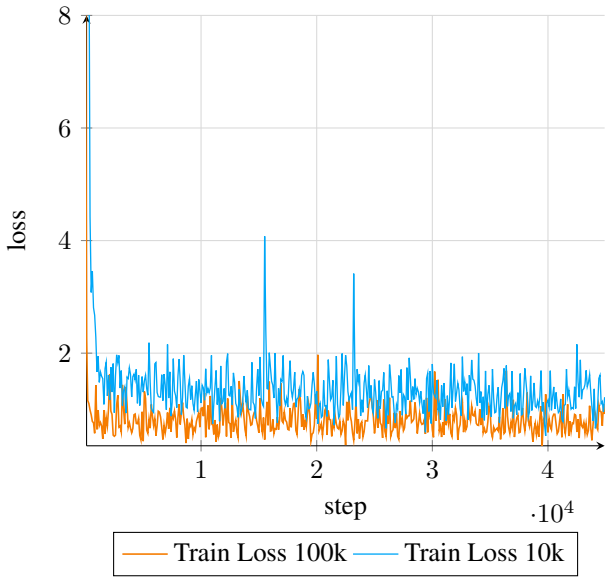

Figure 10: Comparison of training losses for fine-tuning GPT-2 on datasets of 100k and 10k samples.

5.3. LSTM Model

Alongside GPT-2, we employ Long Short-Term Memory (LSTM) networks, which are a type of recurrent neural network (RNN) designed to handle sequence prediction problems. LSTMs are adept at processing and generating text that involves longer sequences, capturing dependencies that occur over extended texts. This capability is particularly useful for generating coherent and contextually relevant sections of the recipe, such as detailed cooking instructions that require maintaining thematic consistency over multiple steps. Unlike traditional RNNs, LSTMs can remember information for a longer duration within the sequence, thanks to their internal gate mechanisms. This feature allows the LSTM model to maintain the context over the entirety of a recipe's length, improving the flow and logic of the generated text. This is crucial in ensuring that the generated recipes make sense from start to finish, adhering closely to culinary logic. For this part, we followed the steps from Text Generation with an RNN[1].
We used a multi-layer LSTM for character-level language modeling. Character-level models generate text one character at a time, learning to predict the next character in a sequence given the previous characters.

5.3.1 Data Preprocessing

The data is batched into groups of 16. Special markers (`<t>`, `<i>`, `<d>`) are used to segment each recipe section. Characters, including markers, are tokenized with unique integer indices, essential for LSTM's character-level pro-

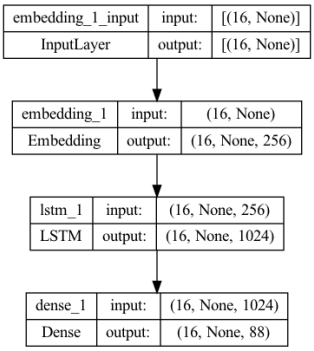

Figure 11: LSTM model configuration

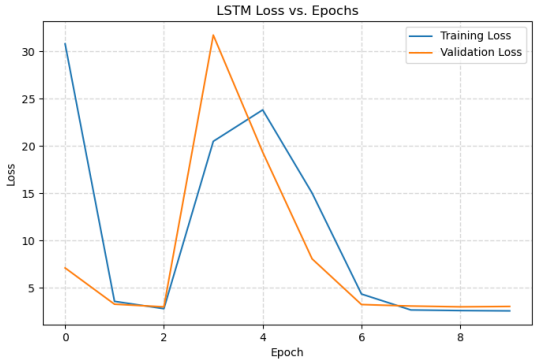

Figure 12: LSTM Loss vs Epoch

cessing. Recipe sequences are capped at 500 characters. Shorter sequences are padded with <_> while longer ones are truncated, maintaining consistent batch sizes.

5.3.2 Model Configuration and Training

The model was built using the TensorFlow framework and includes multiple stacked LSTM layers to enhance learning capacity (see Figure 11). It processes each character by retrieving its embedding, feeding it through the LSTM layers, and then using a dense layer to generate logits that predict the next character's likelihood.
The model is trained for 10 epochs with 200 steps each, using batched sequence of size 16. Categorical cross-entropy loss function is applied for multi-class classification, predicting the next character. Adam optimizer with a learning rate of 0.001 is used for minimizing loss, balancing speed and convergence stability. Early Stopping prevents overfitting by monitoring performance and halting training if improvement stalls, preserving the model's generalization ability and saving computational resources.

5.3.3 Evaluation and Results

The validation loss on Figure 12 suggest that the model may be overfitting due to complexity or inadequate regularization. Halting training at the second epoch could have enhanced generalization, as evidenced by optimal validation loss and accuracy observed at that point.

Input: *eggs, flour, butter, sugar, vanilla*

Output: *eggs, flour, butter, sugar, vanillaiinnin:i¿Die: ¿Die: ¿Die:i¿D¡i¿D¡i¿D¡i¿D¡i¿D¡i¿i¿D¡i¿t¿e: ¿D¡i¿e:t¿e:t¿e: ¿Die:i¿D¡i¿e¿e¿D¡i¿t¿e¿...*

Figure 13: Example of LSTM model output.

6. Anticipated Challenge

We anticipated challenges primarily related to data management and computational resources. The large sizes of the RecipeNLG and Food-101 datasets posed potential issues, as handling and processing such volumes of data requires significant memory and processing power. Additionally, we were concerned about the adequacy of our computational resources, especially given the complexity involved in training an advanced model like GPT-2

7. Discussion

The main objective of our project was to explore various architectures and assess their compatibility and performance when used together. We encountered numerous challenges with fine-tuning the models, which compelled us to investigate the reasons behind the discrepancy between the excellent qualitative results and the poor quantitative performance. Although we do not yet have a complete recipe generator, we have established building blocks that can be further optimized.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] Michał Bień, Michał Gilski, Martyna Maciejewska, Wojciech Taisner, Dawid Wisniewski, and Agnieszka Lawrynowicz. RecipeNLG: A cooking recipes dataset for semi-structured text generation. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 22–28, Dublin, Ireland, Dec. 2020. Association for Computational Linguistics.

[3] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101 – mining discriminative components with random forests, 2014.

[4] Prateek Chhikara, Dhiraj Chaurasia, Yifan Jiang, Omkar Masur, and Filip Ilievski. Fire: Food image to recipe generation, 2023.

[5] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2020.

[6] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[7] A. A. Russo, B. Hurst, and T. Weber. Tastifynet: Leveraging adversarial examples for generating improved recipes. `http://cs230.stanford.edu/projects_winter_2021/reports/70609760.pdf`, 2021. Accessed: 2024-04-20.

[8] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[9] M. Vélez-Toral, C. Rodríguez-Reinado, A. Ramallo-Espinosa, and M. Andrés-Villas. "it's important but, on what level?": Healthy cooking meanings and barriers to healthy eating among university students. *Nutrients*, 12(8):2309, 2020.