

Transforming the Image2Recipe Domain

Text-Image Joint Embeddings Leveraging Transformers

Porter Bagley
Georgia Institute of Technology
pbagley6@gatech.edu

Bailey Russo
Georgia Institute of Technology
brusso6@gatech.edu

Tyler DiPentima
Georgia Institute of Technology
tdipentima3@gatech.edu

Scott Von Stein
Georgia Institute of Technology
ssstein31@gatech.edu

Abstract

In this paper, we build upon prior research regarding Image-to-Recipe models. This model type takes an image as input and outputs a ranking of which recipes are most likely to be related to an image. Model success was evaluated using median rankings and recall scores calculated based on if this prediction ranking contains the known recipe for an image, looking for the target recipe’s inclusion in the top-1, top-5, and top-10 of rankings. Our primary contributions are the introduction of pre-trained transformer modules for both the text and vision segments of the network resulting in a 2.86 times improvement in the top-1 recall score and a 33% MedR improvement over the benchmark model trained on a subset of data.

1. Introduction/Background/Motivation

Focusing in on a previous model created to accomplish this task trained and distributed by an MIT research team [6], the team used this as our benchmark for success. With the goal of improving recall scores for the top-n rankings as well as reducing the computational overhead compared to this model, experiments were conducted to explore novel loss functions as well as module replacements or additions for either the textual or image processing. These objectives were motivated by the benchmark model leaving room for improvement correctly identifying recipes in the top-10 prediction rankings for image samples about 70% of the time and taking 3 days to train the model despite having 4 machines with 12GB of memory each [9].

The benchmark model uses a dataset of consisting of text and image data fed into the model at different endpoints to create a joint embedding. The text is processed through encoders for the instructions and ingredients data using skip-instructions and word2vec while the images are trained

through a ResNet-50 module. The outputs and losses of these separate model branches are then combined into a semantic regularization loss used to train the full model. One limitation of this model includes its complexity, specifically pertaining to the research team’s limited time frame and computational resources available to us. A more global limitation is that text & image analysis in deep learning has evolved since the time of this model’s creation with transformer modules becoming the new state-of-the-art [11] [12]. We believe replacing the current model’s encodings with transformer encoding will yield improved recall.

The purpose of this research is increasing the reliability of Image-to-Recipe models. This has societal benefit in its applications including allergen detection, recipe reverse-engineering, and nutrition tracking. Successful and reliable applications to these fields would provide health benefits whether short or long term or help inspire someone to recreate their favorite meal. Academically, these advancements could also be extrapolated to new areas that use similar text and image embeddings to improve performance on unrelated tasks.

The data-set used was the Recipe1M data-set produced and used by MIT [9] which consists of 1 million recipes and over 800,000 food images. This data-set is not publicly available and was received through a direct inquiry to the authors of [6]. The data-set was web-scraped from popular online cooking site’s HTMLs and is one of the few data-sets that contains ample text and image food representations. This exact gap in multi-modal recipe data inspired its creation. Each instance is accompanied by an image class, a recipe class, and a target class. The recipes and images are also associated with divided into classes to facilitate a regularization objective. The data-set can be seen as two aspects where the first is the textual recipe information while the second is the images associated with the same recipes if applicable. Some recipes were not found to have accompany-

ing pictures hence the 200,000 instance disparity between the two modes. A 70-15-15 split between training, validation, and testing is recommended. This data-set does not contain any sensitive or identifying information. [3]

2. Approach

Overall, we are interested to see if we can improve the performance on the Image-to-Recipe task. We are specifically interested in discovering methods that outperform the original model from [9] in a low-data context, and that are able to achieve good performance without lengthy training times. We experiment with few-shot learning, alternative loss functions, and the introduction of pre-trained transformers in both the text and vision modules of the network. We fork the original code repository¹ from [9] and build off of it.

2.1. Few-Shot VGG-16

Initially, we attempted to combat the first model limitation of complexity by testing the effectiveness of a purely vision-based model. A pre-trained VGG-16 model [10] was loaded and few-shot learned to tune it to the Image to Recipe task. This was done using 5 examples per ingredient class and training for 10 epochs. This is novel compared to the benchmark architecture, pulling out the vision modules and training with less data. With the given project time and computation constraints, it was clear that only a subset of the data-set could be trained on. Treating this as a scenario with a low amount of labelled data, few-shot learning offered a chance at achieving similar recall scores as the benchmark model without having to train on the full million data-points as shown in previous work [7] [1].

Due to the structure of the data-set expecting both the image and text data to be utilized, there were instances where the image labels were zero, or unlabelled. This caused issues of over-fitting with initial models as the zero class dominated the image labels occurring at around 30% of the data-points. To combat this, we treated this as a semi-supervised scenario and leveraged prior work on the effectiveness of pseudo-labelling [5]. The data-points labelled as the zero class received a new label that was determined by the predictions of the failed model that over-fit on the zero class. The new label was the most likely non-zero class as predicted by the failed model. A new model was then few-shot learned using this fully labeled data-set.

2.2. Dataset Reduction

Due to limitations on our experimentation time, we anticipated that the size of the original dataset, and the training time required for the original model in [9] would create

a problem. We first determined to use a subset representing 1% of the original dataset, and training for about 3% as long. The original dataset includes 1047 classes of recipes. After encountering issues getting a working baseline, we further simplify the dataset by using only 10 classes, still with a subset that was 1% the size of the original dataset. The re-trained original model acted as our benchmark for experimentation and measuring success.

2.3. Model Effectiveness Plan

In order to improve the effectiveness of this model, we examined what could be changed about its architecture. The model operates using a core loss function and three encoders, one for each the instructions text, the ingredients text, and the image. We determined three main components that may benefit from changes: the loss function, the methods used to generate the text embeddings (for both instructions and ingredients), and the methods used to generate the image embeddings.

2.4. Loss Functions

Within the existing model architecture, the core loss is a summation of three separate loss functions, those being the cosine similarity loss between the text vector and the image vector, the cross entropy loss of the image, and the cross entropy loss of the recipe embeddings. We looked at each of these losses to determine if there are other loss functions we might be able to use in their stead.

We first examined the cosine similarity loss. However, we quickly realized that theoretically, this loss function is a strong performer and we therefore did not perform further experimentation on it. We chose to not do this due to a few reasons. One reason is that the cosine loss function is a commonly accepted and tested loss function in text-to-image vector comparison. Other reasons include that it is suitable for high-dimensional spaces, it's robust for different text lengths, and only takes into consideration non-zero vectors.

We also examined the cross entropy loss and determined that there was a possibility to improve performance by using mean squared error loss. Since the cross entropy loss function is applied to both the image and the recipe embeddings, with the intention of classification into a food-related semantic category, we wanted to try moving this loss problem from that of classification to that of difference between actual and expected values. The reason we believed this would improve performance is due to the fact that this overarching image-to-recipe problem is not as black and white as other classification problems. While objects in classification problems can sometimes be very cleanly broken out into classes (example: a dog and a truck), food recipes have a lot of overlap. While it may be accurate to call a dish macaroni and cheese, it may also be adequate to call the same

¹<https://github.com/torralba-lab/im2recipe>

dish pasta casserole, with both dishes only having slightly different ingredients, or the same ingredients but a different recipe. As such, we wanted to experiment with making this problem a regression one.

In order to implement a mean square error loss, we changed the model slightly to account for the fact that the loss is no longer being calculated based on class probabilities but rather actual image vectors. This meant changing the code mentioned in [6] to use a MSELoss function with the inputs being the image and recipe vectors themselves. The resulting losses were then added together to get the core loss used for model training.

2.5. Recipe Embeddings

The recipe can be viewed as consisting of two parts: the ingredients and the instructions. The benchmark model encodes the ingredients by passing word2vec representations for each ingredient into a bidirectional LSTM. The benchmark model encodes the instructions by passing vector representations for each instruction into a forward LSTM. Vector representations for each instruction are created using an approach similar to how word vectors are created in a skip-gram model. The resulting encodings for the ingredients and instructions are then concatenated and passed through a linear layer to create the final embedding for a recipe.

Rather than using an LSTM to encode the ingredients and instructions, we hypothesized that generating encodings by passing them through a pretrained transformer model could help to improve the representation of the recipe embeddings as well as the overall performance of our model. Given that transformers utilize an attention mechanism and pretrained models are often trained on text corpora containing hundreds of millions of tokens, we believed the resulting embeddings from such models would produce vector representations that were semantically more meaningful. The exact procedure we performed involved concatenating the list of instructions for a recipe, and passing this concatenated string of instructions into a pretrained transformer, which produced an embedding representing the entire sequence of instructions. This exact process was also used for the ingredients. Similar to the benchmark model, the encoded outputs for the ingredients and instructions were then concatenated and passed through a linear layer.

Using a pretrained model allowed us to remove the LSTMs that were originally used to do the encoding, and ultimately reduce training time as the number of parameters to learn was now smaller. Because we used only 1% of the total data set, a pretrained model would likely provide better results on this small subset of the data. A model that is not pretrained would not have the ability to learn something as complex as recipe instructions with such few training examples.

2.6. Image Embeddings

The original model from [9] uses a pre-trained ResNet-50 as the backbone for the vision module, which is then fine-tuned for the Image-to-Recipe task. Due to the impressive performance of transformers in vision tasks in recent years, we decided to experiment with swapping out the ResNet-50 for a pre-trained Vision Transformer (ViT) [2] as the backbone of the vision module.

Previous work [11] has shown Vision Transformers to perform very well on transfer learning tasks, even exceeding the performance of CNNs. Our hypothesis is that a ViT backbone will outperform the ResNet-50 backbone in this transfer learning task as well.

3. Experiments and Results

3.1. Few-Shot VGG-16

While the initial model over-fit on the zero class achieved recall scores as high as 50% it was clear that these results were inflated and held little meaning. Through pseudo-labelling and removal of the zero-class from the label pool, a more meaningful experiment was run over 10 epochs using around 5 labels per class using all 1047 classes. This was done intentionally to preserve class balance as well as provide enough samples for the few-shot learning process. Success was measured by comparing the recall scores to that of the original model trained on the entire data-set provided by [9].

Table 1. Comparison between benchmark model and the few-shot learned pure vision model

	Original	Pure Vision Model
Top-1 Recall	.2749	.0039
Top-5 Recall	.5778	.0756
Top-10 Recall	.6962	.1224

As seen in Table 1, the pure vision model failed to come close to recall scores of the best original model. This could be due to any number of things such as: Few-shot learning is typically performed for a much smaller pool of new classes [7] given VGG-16 has 1000 classes [10], the epoch disparity between the few-shot learned model and [6]’s best model is large with theirs being trained for 220 epochs, and the context from the recipe embeddings may be crucial for high recall. With our constraints, further testing to rule out the epoch disparity or few-shot learn smaller subsets of new classes at a time was unrealistic, so we proceed with the data-set reduction backup plan.

3.2. Dataset Reduction

In order to establish a baseline, we took the original model from [9], and looked at performance on 1% of the data, and 1047 classes. As seen in Fig. 1, the original model

was able to learn a small amount initially, but improvements to the validation score leveled off after 3 epochs. This isn't surprising considering how many classes there are, and how much data the model was originally trained with to achieve good performance.

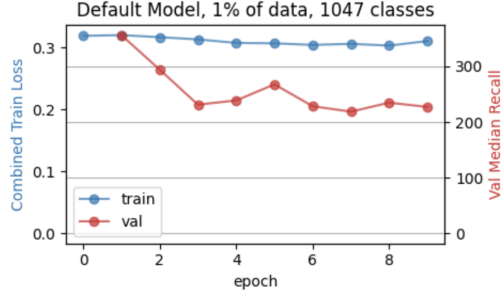


Figure 1. Default model training curve, 1% of data, all classes

We wanted to ensure that learning on the dataset was at least feasible by the original model before continuing on to our own improvements. After reducing the number of classes down to 10, we saw more learning from the original model (see Fig. 2). This demonstrated that the task was possible with the limited data size and training time. We use this model as our baseline in our experiments. On a hold-out dataset, this model achieved a median recall ranking of 113.5, and the following top-k recall scores: [1: 0.0049, 5: 0.0335, 10: 0.0689].

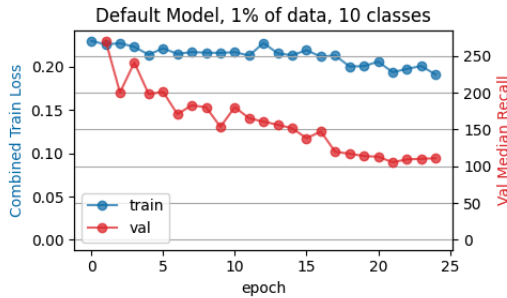


Figure 2. Default model training curve, 10 classes

3.3. Loss Functions

After replacing the cross entropy loss functions for the image and recipe embeddings with mean squared error loss functions, we trained the model on the limited data set. The loss curve that we obtained can be seen in Fig. 3. Looking at these curves, the model did not perform as well as the original model. Even though loss was lower for these original epochs, the median recall was much worse. Since we were limited in how many epochs we can train on, due to computational and timing limitations, we can't put an over reliance on the loss curves, and are therefore putting a higher importance on the recall scores.

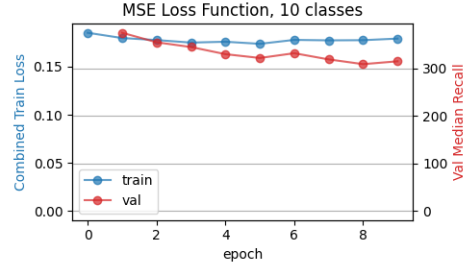


Figure 3. Training curve loss with MSE Loss function instead of Cross Entropy Loss

From these results, we can conclude that the categories defined by the recipes in the dataset do not overlap enough to make this a stronger regression problem than it is a classification problem. Practically, this means that the dataset is made up of recipes that are more akin to be broken up into their own classes, as opposed to having multiple classes that can be merged into one uber class. Outside the scope of this research, but a potential area for further investigation, is training a model on images and recipes that are all very similar to each other. Theoretically, this could be an area where a mean squared error loss function could perform well. However, since our dataset has a wide range of recipes, this clearly is not the case.

3.4. Recipe Embeddings

Because we aimed to have a single encoding representing a sequence of ingredients or a sequence of instructions, and the concatenation of instructions or ingredients could effectively be looked at as a short paragraph, we believed that a sentence transformer was the most appropriate type of pretrained transformer model to use. Sentence transformers take as input a sentence or short paragraph and output an embedding vector that captures the text's semantic meaning. We used the "all-MiniLM-L6-v2" transformer from the sentence transformers library. This pretrained sentence transformer was chosen due to its ability to produce high quality embeddings while also having a fast encoding speed [8]. We compute the embeddings for instructions and recipes for all observations prior to training. This saves training time as we are essentially freezing all layers of the pretrained transformer.

In Fig. 4, we can visualize the embeddings produced from our pretrained transformer model. The points are colored by the food semantic category to which they belong. We can see that the pretrained model produces good quality embeddings as semantically similar vectors are closer to one another in vector space.

We generate the recipe embedding using this pretrained transformer model and keep all other variables in the benchmark model the same during training. We train on the 1%

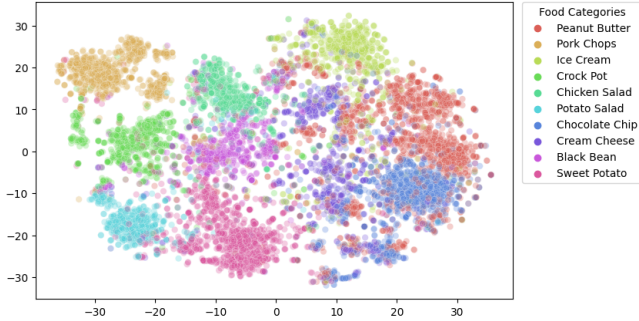


Figure 4. A t-SNE visualization of recipe instructions encoded using pretrained sentence transformer. Points are colored by food category.

subset of data containing only 10 classes. As Fig. 5 shows, the validation median recall after 25 epochs is worse than the validation median recall for the default model. The training loss also appears to be slightly higher than the training loss for the default model. It appears that the original model that uses LSTMs to encode the ingredients and instructions performs better.

The pretrained model may not perform as well as the LSTM due to the fact that its weights are frozen and it does not have the ability learning anything during training. The pretrained model was not originally trained on a dataset that is specific to the domain we are in [8]. Thus, it is not really optimized to understand the language of cooking instructions and food ingredients. This may lead to the resulting embeddings for the ingredients and instructions to be sub-optimal. Alternatively, the LSTM is directly learning on the text data that we are interested in. This may allow for the LSTM to produce more relevant encodings for the task at hand.

If we reexamine the learning curve in Fig. 5, we can actually see that after 10 epochs the model using the pretrained sentence transformer is giving better results for the validation median recall than the original model. Since the sentence transformer is pretrained, it does not need to see that many training examples for it to produce good quality embeddings. Since the LSTM is learning its weights from scratch, we are only able to see good performance after a greater number of epochs. Once the LSTM has seen enough data, it is able to produce encodings for ingredients and instructions that are more relevant.

3.5. Image Embeddings

Using a ViT backbone for the image module was quite effective for the first several epochs. After the first epoch, the validation score was already better than the baseline model. However, around epoch 7, the validation score took a massive hit. This occurs because of the modality-switching built into the architecture. The authors included

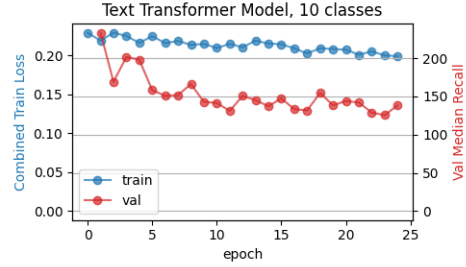


Figure 5. Training learning curve for model using pretrained sentence transformer

modality switching in the architecture where only one modality is trained at a time to help with convergence. This is controlled by the ‘patience’ hyperparameter which determines how many epochs of a non-improving validation score before switching modalities. Therefore a higher ‘patience’ means less frequent modality switching.

Because the modality switching was hurting performance, we explored different values for ‘patience’. Results from this can be seen in Table 2. The table only shows the best results after 25 epochs, but later in training, ‘patience=5’ diverged, while ‘patience=100’ continued to converge. We decided to turn off modality switching by using ‘patience=100’.

Table 2. Validation scores for ‘patience’ hyperparameter search (lower is better). 25 epochs.

	ViT	Original
patience = 1	65.50	105.45
patience = 5	47.15	N/A
patience = 100	47.15	71.90

We see the performance of the tuned ViT model in Fig. 6.

Why does using the pretrained ViT perform so much better than the ResNet? We suspect that the attention mechanism including in the ViT allows for better transfer learning than a CNN, because the transformer layers are better able to focus on individual parts of the image and make connections across different parts. This is especially important when trying to make sense of a picture of food, where high-level features, fine-grained features, and context are all important for determining the recipe.

3.6. Combined Model

For the last experiment, we combined the two pre-trained transformer enhancements into one model to see if they could synergize and provide increased benefits. This new model therefore utilized transformers for both the text and image embeddings and was trained on a subset of the dataset using the hyper-tuned parameters from section 3.5.1 for 30 epochs.

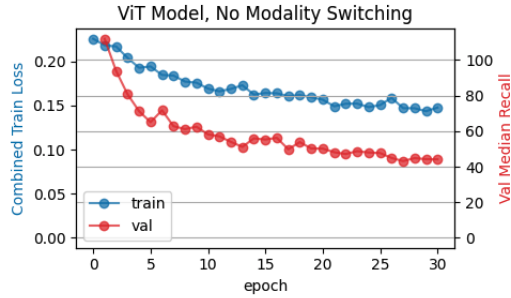


Figure 6. ViT model with patience=100, meaning no modality switching occurred.

Comparing to the benchmark model at 30 epochs, our combined model showed significant improvement performing almost 3 times better on top-1 recall and nearly 33% better in terms of MedR as seen in Table 3.

Similar success with using transformers for a text-image joint embedding has been seen in [4], but this may indicate that the paper is transferable to new task domains where both image and text data is relevant. Here we reap the benefits of self-attention [12] to provide a more refined, focused embedding.

Table 3. Comparison between benchmark model and our combined model at 30 epochs. Higher recall is better, lower median rank is better.

	Original	Combined Model
Top-1 Recall (\uparrow)	.0147	.0420
Top-5 Recall (\uparrow)	.0678	.1308
Top-10 Recall (\uparrow)	.1205	.2074
MedR (\downarrow)	72.05	47.6

3.7. Inference Comparison

We qualitatively evaluated our best model by examining the retrieved ingredients for different image queries. The query image used and the ingredients retrieved by the model are shown in Tables 4-6. We can see the retrieved ingredients are what one might expect each food dish to contain. For example, in Table 6 we see an image of ice cream and the retrieved ingredients include cream, milk, vanilla extract, and sugar. While the retrieved ingredients may not contain every ingredient that is present in the image - for example, some sort of nuts appear to be present in the ice cream - the model is retrieving the ingredients which make up the essence of the dish. This indicates to us that our qualitative analysis of the model matches the results from our quantitative analysis.

4. Future Work

Given the success we had with the addition of pre-trained transformers for the visual and textual embeddings it would

Table 4. Inference images


Query Image	Retrieved ingr.
	8 eggs, 2 lbs potatoes, 1 c Miracle Whip or mayonnaise, 1/4 c mustard, 3 tsp dill, 2 tsp parsley, 4 tsp chives, 3/4 tsp salt, 1/2 tsp black pepper

Table 5. Inference images



Query Image	Retrieved ingr.
	6 pork chops, 1 tbsp oil, 1 can cream of mushroom soup, 12 c sour cream, 12 c water, 1 tsp parsley, salt and pepper, 1 can French-fried onions

Table 6. Inference images

Query Image	Retrieved ingr.
	70 ml single cream, 60 ml buttermilk, 1 pints full fat milk, 1 tsp vanilla extract, 150 grams caster sugar, 3 egg yolks

be interesting to see this new model trained on the full dataset under the same conditions as the MIT model [6]. To further build upon the findings of this paper, one could try and utilize few-shot learning to adapt the pre-trained transformers to this specific task domain before inserting them into the benchmark model. This would address the concerns mentioned in Section 3.4.

Additionally, if we were to unfreeze some of the deeper layers of the pretrained sentence transformer from Section 3.4 during training, the model would still retain all of the previous knowledge it had gained from the domain it was initially trained on, while also being able to learn new knowledge relating to our specific task domain.

References

- [1] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *CoRR*, abs/2005.14165, 2020. 2

- [2] 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, 2021. [3](#)
- [3] Timnit Gebru, Jamie Morgenstern, Briana Vecchione, Jennifer Wortman Vaughan, Hanna M. Wallach, Hal Daumé III, and Kate Crawford. Datasheets for datasets. *CoRR*, abs/1803.09010, 2018. [2](#)
- [4] Varuna Krishna, S Suryavardan, Shreyash Mishra, Sathyanarayanan Ramamoorthy, Parth Patwa, Megha Chakraborty, Aman Chadha, Amitava Das, and Amit Sheth. Imaginator: Pre-trained image+text joint embeddings using word-level grounding of images, 2023. [6](#)
- [5] Dong-Hyun Lee. Pseudo-label : The simple and efficient semi-supervised learning method for deep neural networks. *ICML 2013 Workshop : Challenges in Representation Learning (WREPL)*, 07 2013. [2](#)
- [6] Javier Marin, Aritro Biswas, Ferda Ofli, Nicholas Hynes, Amaia Salvador, Yusuf Aytar, Ingmar Weber, and Antonio Torralba. Recipe1m+: A dataset for learning cross-modal embeddings for cooking recipes and food images. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2019. [1](#), [3](#), [6](#)
- [7] Archit Parnami and Minwoo Lee. Learning from few examples: A summary of approaches to few-shot learning, 2022. [2](#), [3](#)
- [8] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019. [4](#), [5](#)
- [9] Amaia Salvador, Nicholas Hynes, Yusuf Aytar, Javier Marin, Ferda Ofli, Ingmar Weber, and Antonio Torralba. Learning cross-modal embeddings for cooking recipes and food images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017. [1](#), [2](#), [3](#)
- [10] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. [2](#), [3](#)
- [11] M. Usman, T. Zia, and A. Tariq. Analyzing transfer learning of vision transformers for interpreting chest radiography. *J Digit Imaging*, 35:1445–1462, 2022. [1](#), [3](#)
- [12] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017. [1](#), [6](#)