Impact of a Small Dataset on Transfer Learning for Food Detection Task

Nilay Bhatt
nilaybhatt@gatech.edu

Vijay Gentyala
vgentyala3@gatech.edu

Shruti Pohane
spohane6@gatech.edu

Abstract

In this paper we aim to quantify the least amount of data required to effectively train specific layers in the process of transfer learning. After freezing majority of the layers of the pre-trained models, VGG and ResNet respectively, a few selected layer will be unfrozen and re-trained. We aim to answer how much data is realistically needed to see an appropriate improvement in performance. The environment to test these ideas and questions will be a food image to recipe neural network.

1. Introduction

The purpose of this paper is to understand the effects of using a small data set to fine-tune our model in this food detection task. Specifically, the task is to classify a food image and return the appropriate recipe. There are two key questions that we would like to answer. The first question we would like to research is how much data is truly needed to fine-tune a model during transfer learning. The second question revolves around how does the architecture of the chosen neural network affect the amount of data needed to fine-tune specific layers. Note, the goal is not to retrain every layer in the neural network but re-train specific layers. During these steps, we will also note how the data size affects the hyper-parameters of the tuned layer in the pre-built model. We will answer the above questions in a practical setting through an Image-to-Recipe solution. The Image-to-Recipe solution will test whether a neural network can effectively retrieve related recipes given a specific food.

2. Background

This implementation of generating a recipe from image recognition has also been attempted before with different types of strategies. In the initial blog [2], the author trains convolution neural networks to classify images into categories and match to a recipe using various topic modeling, feature extraction, and transfer learning algorithms. The biggest limitation of the original blog post is that they don't evaluate how well their models perform. Another article [5], mentions algorithms for reinforcement learning to generate recipes from images. The authors in this article encode the images and extract features to generate ingredients and then recipes. However, one limitation of what they propose is that they do not categorize the images to propose similar recipes which we aim to do. One article [4] looks into image and text classification using the images and html tags of the page to use models like Bag of Words or CNN to generate the recipe. A final article [3], takes the pixels of the image to extract features and feed it through a ResNet encoder to predict the ingredients and train with an Adam optimizer. While we have seen completely different types of implementations, each approach uses the same underlying feature extraction and CNN models for training. With these current implementations of food image to recipe generation, some of the major limitations are how to experiment and evaluate the translations. It can be very subjective to evaluate the performance of all of these models because the recipe titles or ingredients differ from dataset to dataset and can require human judgment to evaluate the end result. Furthermore, there can be many food items in an image and multiple many different ways to cook a certain item. The size of the food dataset also makes an impact on the experimentation and evaluation. That is what makes this image to recipe translation such a hard task to implement.

3. Motivation

The process of image to recipe translation is a difficult one. The dataset is extensive and there are multiple ways to prepare any particular dish. If we are successful in appropriately evaluating the performance of the image translations, our experiments can be very useful for further research and implementations. The results of this experiment will give researchers and practitioners a rule of thumb of how much data is needed to fine tune specific layers. If researchers and practitioners have a lower bound on the amount of data needed to train certain layers of a pre-trained model, then they can make more informative decisions on the overall modeling process.

4. Data

Our dataset is downloaded from two places: Kaggle and an archived repository. We were able to merge

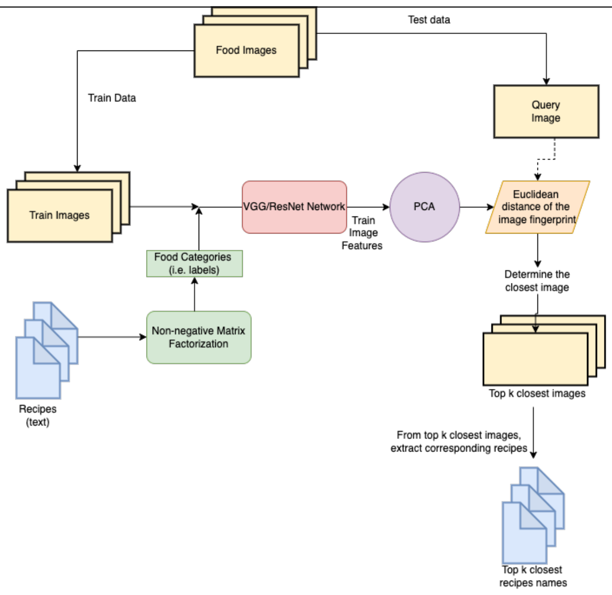

Food Images
Test data
Train Data
Query Image
Train Images
VGG/ResNet Network
PCA
Euclidean distance of the image fingerprint
Train Image Features
Food Categories (i.e. labels)
Determine the closest image
Non-negative Matrix Factorization
Recipes (text)
Top k closest images
From top k closest images, extract corresponding recipes
Top k closest recipes names

Figure 1. Project Architecture

the two datasets together and the final dataset contains 85,791 recipes that have associated pictures. All the data was scraped from various websites such as Epicurious, BBC UK, and CookStr, but the final dataset itself is self-contained into two folders. The first part of the dataset is a csv table that labels the ingredients, instructions, title of the food, and name of the corresponding images. The dataset also contains different folders of images, the folders even further categorized by the size of the images. Overall we were interested in the ingredients, the name of the recipe, and the instructions columns that are represented as strings. The images are also jpg images that correspond to each row of the csv file. Overall most rows of the csv file have a corresponding image, but there are some rows of the initial dataset that were tagged with a generic photo which limited our dataset to 85,791 rows with actual food images. In order to use the dataset for topic modeling, we needed to clean the dataset. It was a two step process. In the first step, any rows that contained lists or list of lists were converted into strings, the words were converted to lowercase, and any descriptive words (i.e. easy or best) were removed from the food name. In the second step of the clean up, we used a tokenizer to create tokens of the food name and removed all English stop words. Finally, the images dataset was categorized into different folders based off of the different topics we got from topic modeling. Overall we noticed that categories like chicken, chocolate chip cookies, or salad had well over 3000 images. While other categories like casserole, zucchini salad, or potatoes had less than 1000 images.

5. Approach

After cleaning up the data, we applied different topic modeling techniques to divide the images into separate categories or topics. In this section of our project, we used different techniques, like Latent Dirichlet Allocation (LDA) and Non-negative Matrix Factorization (NMF), to create multiple categories for the images. We also gathered different statistics, like term frequency-inverse document frequency (TF-IDF), coherence, and log perplexity scores to measure how well the topics modeling performed. There was also some degree of human intervention involved to evaluate the accuracy of the models. The first method that was implemented was LDA to start the topic modeling process and get an understanding of the various categories and number of topics. Training the LDA model involved two parts. First, we tuned the hyperparameters (described in depth in Section 6) and gathered various metrics like jaccard similarity, coherence, and perplexity scores. The second part involved building a corpus and dictionary out of the recipe titles to understand the various topics and see any similarities. Visualizing the top words in each topic through LDA also gave us a better understanding to see if any common words that don't have any importance needed to be eliminated in the preprocessing step. The idea behind LDA is to iterate through all the words in each document and assign probability scores to each word that it belongs to a certain topic. Essentially, the model has the important task of classifying words from a document in particular topics and we have to provide the model the number of topics we want. After understanding the various categories and topics these recipe images can belong to, we used the NMF model to predict categories for the image. First, we calculated tfi-df scores and extracted the top feature names to feed into the NMF model. Using our dataset and TF-IDF weights, the NMF model assigns each recipe name to a category. From there, we created folders for each category and used the initial dataset to move the image from its original folder into its respective category. The purpose of TF-IDF is to calculate how many times a word appears in a document and see how important that word is in the recipe name. We used NMF to perform dimensionality reduction and feature extraction through a low-rank non-negative matrix approximation. The idea behind using TF-IDF weights with the NMF model is so that we can reduce dimensionality and perform clustering on the recipe names to sort them into different categories.

Next, we sorted the food images corresponding to its relative topic label. The images are then pre-processed and modeled using the Keras library before inputting them into a version of VGG/ResNet structure. For this portion of the assignment, we use the images in the sorted categorized folders and translate them into arrays. Note that these prebuilt neural networks are modified such that the last fully connected layer is removed and the output of the second to last and third to last dense layer is used as the image's fingerprint. The image's fingerprint is a unique tensor that represents the image's structure and the data that the model understands from the image. We split the image data into train and test. For this, we ran through different cuts of train and test percentages, as well as different variations with the choice of the neural network layer. We also iterated through different methods of freezing and unfreezing certain layers to attain more accuracy, more of this is described in other sections. We extract different image fingerprints for the train set. These features were then transferred to do a PCA analysis with 300 components. We kept the number of components the same due to the amount of pixels that these images have. These features that were part of the training dataset were transformed using PCA. We then used these PCA transformed features to find the result images for which we can find closest fingerprints to. We use euclidean distance to find a fingerprint of an image which is closest to the one that we queried upon and return the closest images. We then use these predicted images and then curate different recipes using the label of the found image.

Our basis and inspiration for this project came from a blog post, where the author created an application that classified images into various food categories and outputted a matching recipe. We anticipated running into a couple of issues, but the severity of these issues were slightly underestimated. First, the author scraped the recipes in a non-English language. Our initial solution was to leverage a language translation API to translate these recipes into English. However, this solution proved to be too expensive and time-consuming. In addition, we underestimated the amount of time it would take to scrape the recipes from various websites. We implemented and tested a prebuilt recipe-scraper from a python library and discovered that scraping a large dataset was computationally heavy and time consuming. These issues led us to search for smaller and handy datasets. In addition, we initially believed that code from the blog post was plug and play but we ran into a lot of issues, from unsupported libraries to broken/unintuitive logic in the web scraper we were unable to use the original dataset. Thus, instead of fixing these issues, the team decided to re-create the architecture from scratch and use a smaller combined dataset from various sources. A minor issue was that one of the csv files we used were tagged with images that did not exist in the image dataset, so we used another dataset to increase the size of our overall dataset.

Another problem that we faced was during the data preprocessing and topic modeling step. Since the dataset that we used was smaller and the recipe names included a lot more details than the original dataset, we had to adjust our models to avoid overfitting and focusing on those unnecessary words. For example, the food names in the original recipe were as generic as "Italian Mediterranean Pasta", but in our dataset the food names were more detailed like "Grandma VanDoren's White Bread". This issue was not anticipated and we needed to go through a fair share of hyperparameter tuning to ensure that the final categories were generic enough to not pick up on some of these extra words, but maybe that made us lose out on the accuracy of the categorizations. We definitely tried preprocessing the data on our end to remove common unimportant words, but we could not remove all of them as the dataset was too big to go through manually. Also, removing those extra words makes a difference in how the food is categorized. Those words are equally necessary to place importance on the big topic words. This was a known factor that we moved forward with in our project.

For the image recognition step, we faced computation issues due to the memory sizes of our dataset. The dataset was upwards to 9 gigabytes, making local computation ineffective. It was also difficult transferring the data to Google Cloud due to the size and difficult mounting the dataset to our requested VM environment. In addition, requesting powerful T4 GPUs through the Google Cloud Platform was very difficult as there were only a limited quantity and had a dollar component attached to them. We did not anticipate these issues as we originally believed that Google Cloud is able to handle computationally heavy tasks. We encountered various GPU crashes both locally as well as in the GCP environment as well as memory errors despite having large memories (RAM) available locally, i.e. 16 GB and 32 GB. During one such crash, we lost the notebook that was running the code due to a long output message from tqdm.

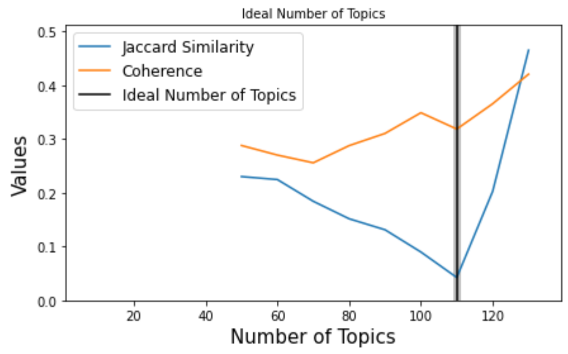


Figure 2. Ideal Number of Topics

This was addressed by retrieving an older copy from version control and re-doing the work. This rendered hours of work done useless as we ended up restarting our process. Another approach we tried to address these crashes, was to reduce the batch sizes for the ResNet model which ensured we do not crash or freeze up our PCs, but also increased our computational times. Next, we pickled the features and test data so we are not spending the time rerunning the models again and again. This helped reduce the pressure off the GPUs. Two of the three authors also used their local GPUs NVIDIA 1070 and 2080, to accomplish this task

6. Experiments and Results

For topic modeling, we conducted different experiments to tune various hyperparameters and classify the food name into separate categories. For the LDA model, the biggest hyperparameter to tune was the number of topics. Different values for the number of topics yielded different probability scores and different words. In order to decide the optimal number of topics for our dataset, we gathered a bunch of metrics like coherence, perplexity, and Jaccard similarity scores. The perplexity values gave us an idea of how well the LDA model predicts against a data sample, even the ones it has not seen before. It is measured as a log-likelihood value and generally a lower perplexity score means that the model is performing better. However, this metric is not enough to evaluate the LDA model by itself. We also used the coherence scores to measure the similarity of the top words within a particular topic. The higher the score, the better the model performs in extracting the topics. The Jaccard similarity coefficient also helped us measure the distance between the words in a topic. The lower the score, the better the predictions. Using all three of these scores, we tried different number of topics in the LDA model and visualized the results in a chart seen in Figure 2 and A to find the optimal number of topics for LDA.

We ended up testing LDA with 110 topics since that was the point where Coherence was high, the topic overlap using Jaccard Similarity scores were the lowest, and the log perplexity score was relatively low. Using the extracted topics from LDA, we got a general idea of the different categories and what adjective or stop words can be removed in the preprocessing step. In the next step in our topic model process, we calculated TF-IDF scores and used them in the NMF model. For calculating TF-IDF, there were not many hyperparameters that could have been tuned. But we got a general idea of what top categories has higher TF-IDF scores, like chicken, salad, or chocolate as seen in C. However, for NMF, there was some experimentation and hyperparameter tuning we needed to do to get the categories for the food images. For NMF, we tried various values for the l1_ratio, but that did not make much of a difference in the final results, we also tried changing the alpha score but noticed no significant improvements either. Finally, we just ended up changing the number of n_components. The n_components is how many features or topics that dataset will be divided into. Arriving at the optimal number of n_components for our NMF model required a bit of trial and error and human evaluation to see which categories fit the dataset the best. Overall, the topics given by NMF (D) are better than those given by LDA (B). For LDA, the top 5 topics contains words that are more random and generic like three, style, warm, and shoot. On the other hand, the top 10 topics for NMF contains less irrelevant words like layer or mix. It also seems like NMF performs better than LDA. This is probably because NMF does not require any knowledge of past data, runs faster on a smaller subset of data (like recipe titles), and takes the TF-IDF scores into consideration instead of raw data [1]. All of these various experiments and hyperparameter tuning done for the topic modeling was somewhat successful. While the category names were not fully accurate, similar images were definitely grouped together in the same category.

For the image prediction model, we ran six different experiments to measure the performance of the model both quantitatively as well as qualitatively. We conducted these experiments by splitting the initial dataset into training and testing. By varying the size of the training set, we are effectively changing the number of ground-truth image fingerprints that the model sees. Therefore, if the size of the training set is small then the number of digital fingerprints it sees are small, which makes it harder for the model to determine the closest fingerprint of a query image to a fingerprint of a train image. The six experiments are as follows:

1. 90% of the data is train & 10% of the data is our query images. We used a VGG network that outputted an image's fingerprint from the first dense layer, which was of size 1x4096. (K) (L)

2. 90% of the data is train & 10% of the data is our query images. We used a VGG network that outputted an image's fingerprint from the second dense layer, which

was of size 1x4096. (O) (P)

3. 60% of the data is train & 40% of the data is our query images. We used a VGG network that outputted an image's fingerprint from the first dense layer, which was of size 1x4096. (I) (J)

4. 60% of the data is train & 40% of the data is our query images. We used a VGG network that outputted an image's fingerprint from the first dense layer, which was of size 1x4096. (M) (N)

5. 90% of the data is train & 10% of the data is our query images. We used a ResNet50 network that outputted an image's fingerprint from the last dense layer. (G) (H)

6. 60% of the data is train & 40% of the data is our query images. We used a ResNet50 network that outputted an image's fingerprint from the last dense layer. (E) (F)

Please note that all models required an input size of 224x224. The output size (i.e. the digital fingerprint) for all the models is nx4096, where n is the number of query images. The optimizer and loss function for these models were Adam and categorical-cross-entropy loss respectively. We kept these functions the same to test the capabilities of transfer learning. If we changed the optimizer and loss functions then we would have to retrain the model, thereby defeating the purpose of transfer learning.

We hypertuned the PCA model on the number of principal components to use and determined that 300 was the ideal number of principal components. We chose these values randomly through trial and error. We saw that when we increased the number of components the computation time was too long and when the number of components were too small, the accuracy dipped (the ability to retrieve the correct recipe). In terms of the VGG model, we tuned the layers, which are part of the model, and ran the VGG model with the last dense layer(FC2), removing the last linear layer as well as only keeping the last but one dense layer (FC1). We chose these two layers because these layers tend to capture most of the finer details of the image. Therefore, the output of the matrix of these layers can be a strong identifier. For the ResNet model, we hypertuned the last dense layer so that the output of it is the same as the VGG model i.e. 1x4096. This was done in order to keep the comparisons of the two models the same.

6.1. Measurements

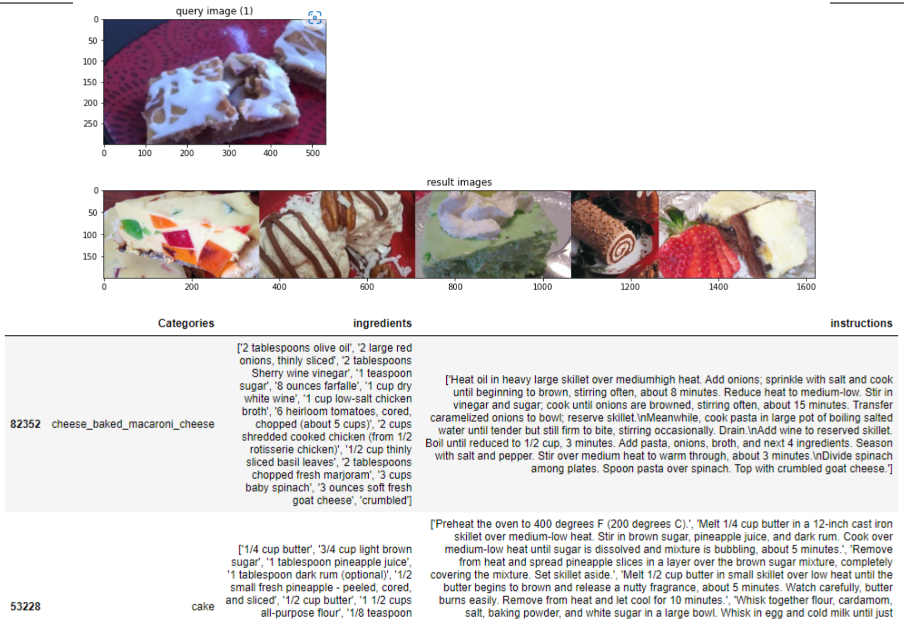
As discussed in our paper before, our goal was to understand how these models work when we change the size of the datasets, specifically to understand how well these deep learning models perform when we reduce the size of the datasets. In order to measure accuracy of the models, we found N closest images to our query, we then found their corresponding labels (high level categories) for each of the resultant images found. We then measured success by looking at how many images out of the N closest images found were part of the same label as well as if there existed at least one image that corresponded to the same high level category of the input/test image. For these experiments we had the method return the top 5 closest images.

6.2. Quantitative Results

When comparing graph (J) with graph (N), you can see that when we extract the digital fingerprints from the second fully connected layer in VGG, it performs much better than the first dense layer (FC1). The reason why this occurs is because the second dense layer (FC2) captures more compositional components than FC1, as FC2 is deeper in the network. When comparing the VGG model with the FC2 layer (L) versus the VGG model with the FC1 layer (P), they surprisingly performed the same. It seems like adding more images to your training data, essentially adds more noise. This noise degrades the model's performance severely. The noise is due primarily to classes with low amounts of images. These are the hard to classify images. When ResNet50 is compared to the performance of VGG16, the ResNet model consistently does better across all conditions (F)(E) (H)(G). This should not be surprising. The ResNet50 performs better because the model is much deeper than the VGG16 model. Thus, the ResNet50 model can capture finer compositional features than the VGG16 model. The experiments seem to be inconclusive. While there is a correlation that when there are fewer images, the performance increases, we cannot prove that there is a causation. We cannot prove that a low amount of data causes the model to perform better. In addition, there maybe a relationship network chosen plays an affect on the performance of the task. Most of these images had 0% correct prediction in terms of their high level category. Whereas very few of the images had a higher percentage (greater than 60%) of the number of correctly identified labeled images. This does suggest two things, there is class imbalance when we split our image data for training and testing and secondly, the topic modeling part would need further refining to be able to associate these images in a more correct manner.

6.3. Qualitative Results

We saw that the model failed to classify many of the images properly. For example, food images that had the same texture with other images were difficult for both VGG and ResNet50 neural networks to differentiate. If the neural networks saw an image of an “oatmeal cookie”, it would often bring back results of bread or even a burger due to the model failing to recognize the similar texture of bread buns



	Categories	ingredients	instructions
82352	cheese_baked_macaroni_cheese	['2 tablespoons olive oil', '2 large red onions, thinly sliced', '2 tablespoons Sherry wine vinegar', '1 teaspoon sugar', '8 ounces farfalle', '1 cup dry white wine', '1 cup low-salt chicken broth', '6 heirloom tomatoes, cored, chopped (about 5 cups)', '2 cups shredded cooked chicken (from 1/2 rotisserie chicken)', '1/2 cup thinly sliced basil leaves', '2 tablespoons chopped fresh marjoram', '3 cups baby spinach', '3 ounces soft fresh goat cheese', 'crumbled']	['Heat oil in heavy large skillet over mediumhigh heat. Add onions; sprinkle with salt and cook until beginning to brown, stirring often, about 8 minutes. Reduce heat to medium-low. Stir in vinegar and sugar; cook until onions are browned, stirring often, about 15 minutes. Transfer caramelized onions to bowl; reserve skillet.\nMeanwhile, cook pasta in large pot of boiling salted water until tender but still firm to bite, stirring occasionally. Drain.\nAdd wine to reserved skillet. Boil until reduced to 1/2 cup, 3 minutes. Add pasta, onions, broth, and next 4 ingredients. Season with salt and pepper. Stir over medium heat to warm through, about 3 minutes.\nDivide spinach among plates. Spoon pasta over spinach. Top with crumbled goat cheese.']
53228	cake	['1/4 cup butter', '3/4 cup light brown sugar', '1 tablespoon pineapple juice', '1 tablespoon dark rum (optional)', '1/2 small fresh pineapple - peeled, cored, and sliced', '1/2 cup butter', '1 1/2 cups all-purpose flour', '1/8 teaspoon	['Preheat the oven to 400 degrees F (200 degrees C).', 'Melt 1/4 cup butter in a 12-inch cast iron skillet over medium-low heat. Stir in brown sugar, pineapple juice, and dark rum. Cook over medium-low heat until sugar is dissolved and mixture is bubbling, about 5 minutes.', 'Remove from heat and spread pineapple slices in a layer over the brown sugar mixture, completely covering the mixture. Set skillet aside.', 'Melt 1/2 cup butter in small skillet over low heat until the butter begins to brown and release a nutty fragrance, about 5 minutes. Watch carefully, butter burns easily. Remove from heat and let cool for 10 minutes.', 'Whisk together flour, cardamom, salt, baking powder, and white sugar in a large bowl. Whisk in egg and cold milk until just

Figure 3. Result of a test image

Student Name	Contributed Aspects	Details
Nilay Bhatt	Image Detection/Recipe Retrieval	Implemented a VGG network to identify and retrieve matching recipes for a query image.
Vijay Gentyala	Data Analysis/Image Detection/Recipe Retrieval	Implemented a VGG network to identify and retrieve matching recipes for a query image. Analyzed the datasets to see if there are any interesting patterns or any issues.
Shruti Pohane	Data Preprocessing/Topic Modeling	Creating and hypertuning LDA model and NMF model for topic modeling on ingredients/instructions to create appropriate labels for our image dataset.

Table 1. Contributions of team members.

and cookie surface. Similarly, the model over-compensated for colors that were high in contrast, for example, it classified a M&M cookie for a garden salad which had red cherry tomatoes and purple onion. In addition, the neural networks often mistook the silverware (i.e. plates, spoons or forks) as part of the food. Thus, the neural network believed distinct foods with similar silverware were indeed similar. Below is an example of how the model performs on an image of a pastry. We observe that the model does a great job of connecting it to other pastries as well as having the ability to recommend multiple cake recipe !

7. Conclusion

In conclusion, we focused on evaluating and experimenting with translated images to closest recipes. Some points for future improvements could be that we use the recipes and ingredients to generate a new recipe for the images instead of using just recipe names. Our goal was to make an environment that does not have enough data to test the capabilities of transfer learning. Researchers can improve on our project by creating an architecture that can identify multiple foods in an image while decreasing the training set size. In addition, within our experiments, we did not measure the correctness of a recipe (i.e too much salt). Removing these incorrect recipes can further reduce our dataset and further challenge the concepts of transfer learning.

8. Work Division/Project Code Repositories

Work division tasks are detailed in Table 1. The code repositories and changes made to the code are referred to in Table 2.

Link	Changes Made
Image-to-Recipe Translation with Deep Convolutional Neural Networks	Initial blog post given in the assignment, changes made to topic modeling and transfer learning to fit our dataset for testing and experimentation.
Jaccard's Similarity	Used as a reference and calculated other metrics for LDA on our dataset.
Optimal Topics For LDA	Used as a reference for visualization and getting various scores for LDA. Updated the code to get perplexity scores as well.

Table 2. Code Repositories Referred.

References

[1] Egger. A topic modeling comparison between lda, nmf, top2vec, and bertopic to demystify twitter posts. *Front Sociol*, 1(1), 2022. 4

[2] Muriz. Image-to-recipe translation with deep convolutional neural networks. `https://towardsdatascience.com/this-ai-is-hungry-b2a8655528be`, 2019. 1

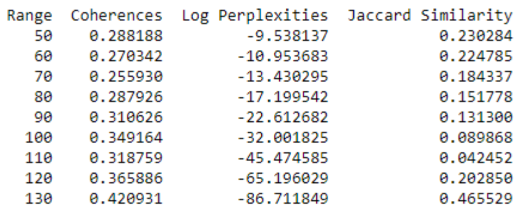
[3] A Salvador, M Drozdzal, X Giro i Nieto, and A Romero. Inverse cooking: Recipe generation from food images. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 1(1):10453–10462, 2019. 1

[4] X Wang, D Kumar, M Thome, M Cord, and F Precioso. Recipe recognition with large multimodal food dataset. *IEEE International Conference on Multimedia amp; Expo Workshops (ICMEW)*, 1(1):234–778, 2015. 1

[5] M Zhang, G Tian, Y Zhang, and P Duan. Reinforcement learning for logic recipe generation: Bridging gaps from images to plans. *IEEE Transactions on Multimedia*, 24(1):352–365, 2022. 1

9. Appendix

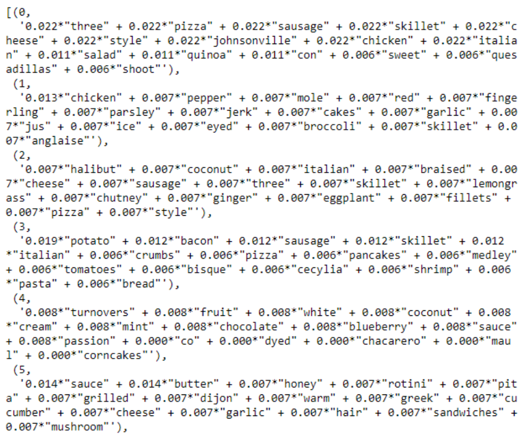
A. Various Metrics for LDA



```
Range  Coherences  Log Perplexities  Jaccard Similarity
   50    0.288188         -9.538137            0.230284
   60    0.270342        -10.953683            0.224785
   70    0.255930        -13.430295            0.184337
   80    0.287926        -17.199542            0.151778
   90    0.310626        -22.612682            0.131300
  100    0.349164        -32.001825            0.089868
  110    0.318759        -45.474585            0.042452
  120    0.365886        -65.196029            0.202850
  130    0.420931        -86.711849            0.465529
```

Figure 4. Metrics

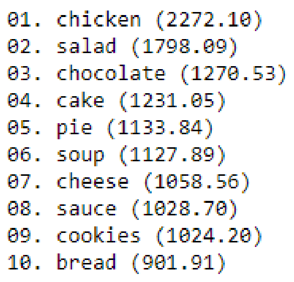
B. Top 5 Topics for LDA



```
[(0,
  '0.022*"three" + 0.022*"pizza" + 0.022*"sausage" + 0.022*"skillet" + 0.022*"c
heese" + 0.022*"style" + 0.022*"johnsonville" + 0.022*"chicken" + 0.022*"italia
n" + 0.011*"salad" + 0.011*"quinoa" + 0.011*"con" + 0.006*"sweet" + 0.006*"ques
adillas" + 0.006*"shoot"'),
 (1,
  '0.013*"chicken" + 0.007*"pepper" + 0.007*"mole" + 0.007*"red" + 0.007*"finge
rling" + 0.007*"parsley" + 0.007*"jerk" + 0.007*"cakes" + 0.007*"garlic" + 0.00
7*"jus" + 0.007*"ice" + 0.007*"eyed" + 0.007*"broccoli" + 0.007*"skillet" + 0.0
07*"anglaise"'),
 (2,
  '0.007*"halibut" + 0.007*"coconut" + 0.007*"italian" + 0.007*"braised" + 0.00
7*"cheese" + 0.007*"sausage" + 0.007*"three" + 0.007*"skillet" + 0.007*"lemongr
ass" + 0.007*"chutney" + 0.007*"ginger" + 0.007*"eggplant" + 0.007*"fillets" +
0.007*"pizza" + 0.007*"style"'),
 (3,
  '0.019*"potato" + 0.012*"bacon" + 0.012*"sausage" + 0.012*"skillet" + 0.012
*"italian" + 0.006*"crumbs" + 0.006*"pizza" + 0.006*"pancakes" + 0.006*"medley"
+ 0.006*"tomatoes" + 0.006*"bisque" + 0.006*"cecylia" + 0.006*"shrimp" + 0.006
*"pasta" + 0.006*"bread"'),
 (4,
  '0.008*"turnovers" + 0.008*"fruit" + 0.008*"white" + 0.008*"coconut" + 0.008
*"cream" + 0.008*"mint" + 0.008*"chocolate" + 0.008*"blueberry" + 0.008*"sauce"
+ 0.008*"passion" + 0.000*"co" + 0.000*"dyed" + 0.000*"chacarero" + 0.000*"mau
l" + 0.000*"corncakes"'),
 (5,
  '0.014*"sauce" + 0.014*"butter" + 0.007*"honey" + 0.007*"rotini" + 0.007*"pit
a" + 0.007*"grilled" + 0.007*"dijon" + 0.007*"warm" + 0.007*"greek" + 0.007*"cu
cumber" + 0.007*"cheese" + 0.007*"garlic" + 0.007*"hair" + 0.007*"sandwiches" +
0.007*"mushroom"'),
```

Figure 5. LDA

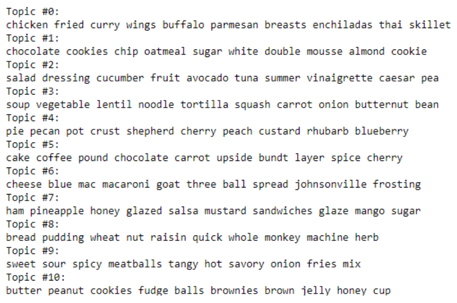
C. Top 10 TF-IDF Scores



```
01. chicken (2272.10)
02. salad (1798.09)
03. chocolate (1270.53)
04. cake (1231.05)
05. pie (1133.84)
06. soup (1127.89)
07. cheese (1058.56)
08. sauce (1028.70)
09. cookies (1024.20)
10. bread (901.91)
```

Figure 6. TF-IDF

D. Top 10 Topics for NMF



```
Topic #0:
chicken fried curry wings buffalo parmesan breasts enchiladas thai skillet
Topic #1:
chocolate cookies chip oatmeal sugar white double mousse almond cookie
Topic #2:
salad dressing cucumber fruit avocado tuna summer vinaigrette caesar pea
Topic #3:
soup vegetable lentil noodle tortilla squash carrot onion butternut bean
Topic #4:
pie pecan pot crust shepherd cherry peach custard rhubarb blueberry
Topic #5:
cake coffee pound chocolate carrot upside bundt layer spice cherry
Topic #6:
cheese blue mac macaroni goat three ball spread johnsonville frosting
Topic #7:
ham pineapple honey glazed salsa mustard sandwiches glaze mango sugar
Topic #8:
bread pudding wheat nut raisin quick whole monkey machine herb
Topic #9:
sweet sour spicy meatballs tangy hot savory onion fries mix
Topic #10:
butter peanut cookies fudge balls brownies brown jelly honey cup
```

Figure 7. NMF

E. Resnet50 Trained on 60% of the Data

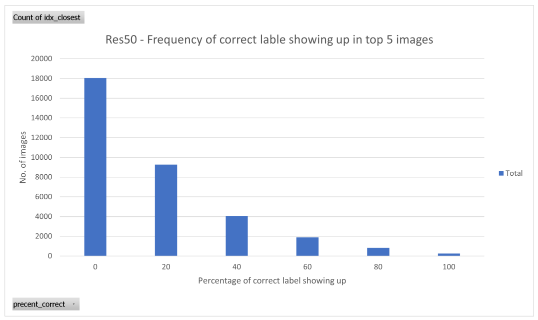


Figure 8. ResNet50's Performance (60-40 split)

F. Resnet50 Trained on 60% of the Data At Least 1 Correct Label

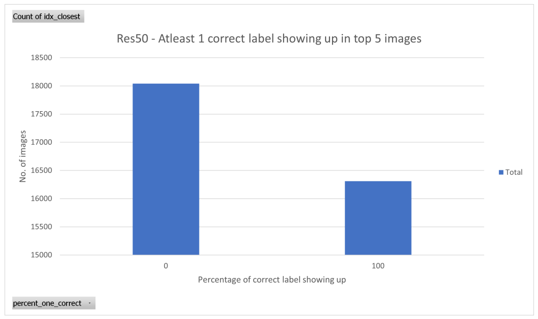


Figure 9. ResNet50's Performance with at least 1 correct label (60-40 split)

G. Resnet50 Trained on 90% of the Data

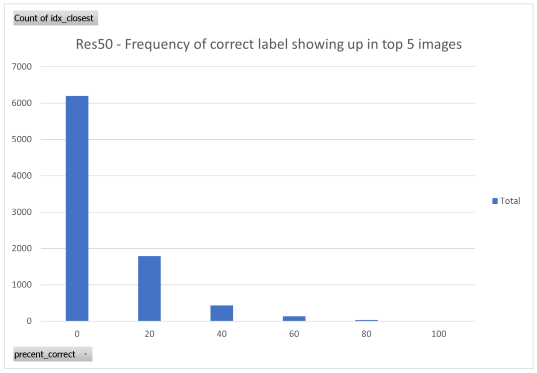


Figure 10. ResNet50's Performance (90-10 split)

H. Resnet50 Trained on 90% of the Data At Least 1 Correct Label

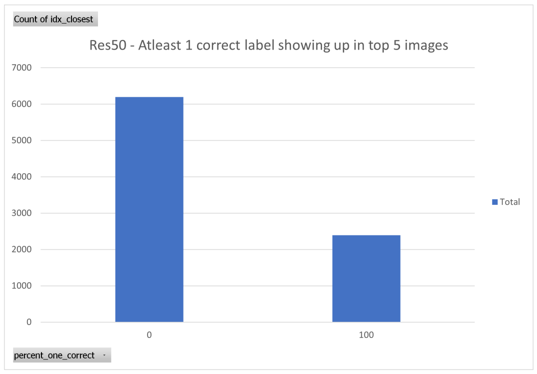


Figure 11. ResNet50's Performance with at least 1 correct label (90-10 split)

I. VGG FC1 60-40

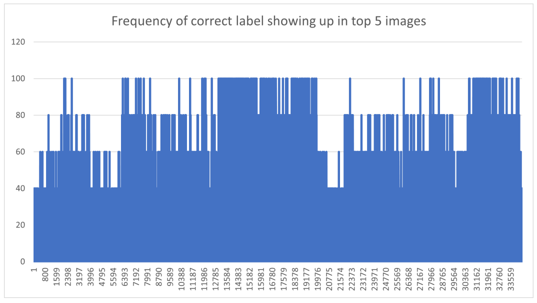


Figure 12. FC1(60-40 split)

J. VGG FC1 60-40 At Least 1 Correct Label

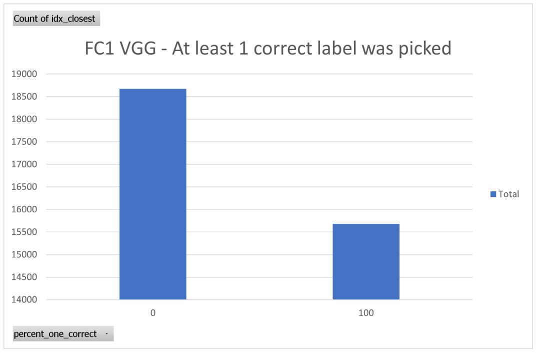


Figure 13. FC1(60-40 split) with at least 1 correct label

K. VGG FC1 90-10

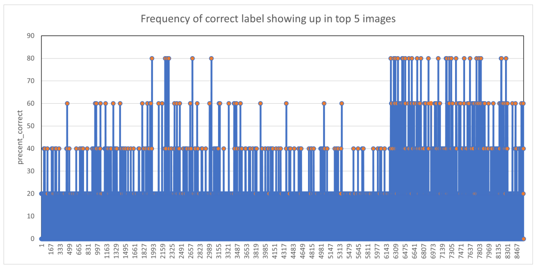


Figure 14. FC1(90-10 split)

L. VGG FC1 90-10 At Least 1 Correct Label

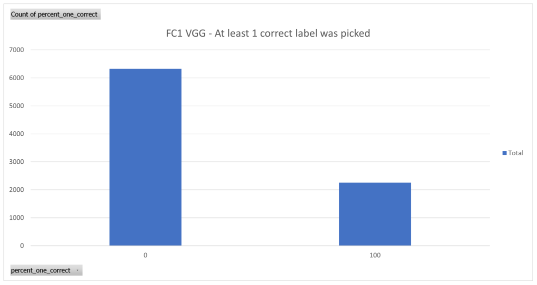


Figure 15. FC1(90-10 split) with at least 1 correct label

M. VGG FC2 60-40

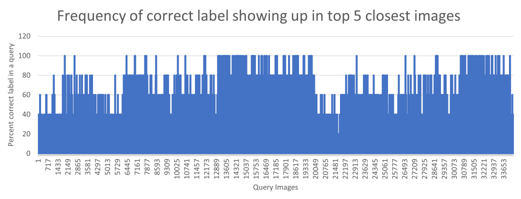


Figure 16. FC2(60-40 split)

N. VGG FC2 60-40 At Least 1 Correct Label

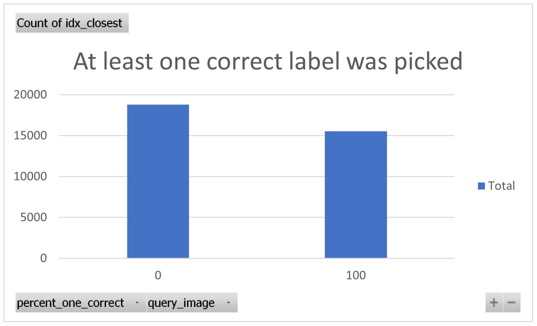


Figure 17. FC2(60-40 split) with at least 1 correct label

O. VGG FC2 90-10

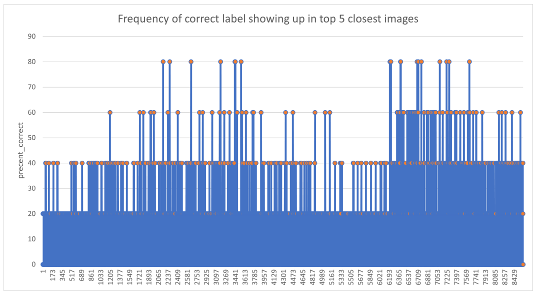


Figure 18. FC2(90-10 split)

P. VGG FC2 90-10 At Least 1 Correct Label

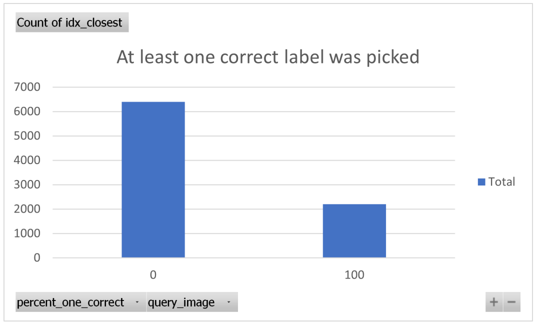


Figure 19. FC2(90-10 split) with at least 1 correct label