Food Image to Recipe Generator

Chun Min Chen
Department of Physics
National Tsing Hua University
Hsinchu, Taiwan
vincent0110@gapp.nthu.edu.tw

Yu Chen Chen
Department of Phsyics
National Tsing Hua University
Hsinchu, Taiwan
victor50432@gmail.com

***Abstract*—In recent years more and more people favor building up their bodies and maintaining body shape. In order to achieve this goal, it's crucial to watch over nutrition intake. However, not many people have the leisure time to cook at home and prepare ingredients by themselves. Hence, foodies are becoming more common in our society, and they face a problem about identifying the ingredients of their daily meals to control diet well. This project is to provide support to identify the meals with convolutional neural network, comparing the performance between famous structures. Furthermore, correspond to the recipe of the particular food object and obtain the ingredients by extracting features through neural network and using machine learning algorithms to search linearly. Moreover, solving the problem of high dimensionality. And finally, building an interactive web application for users to utilize this service and successfully identifying their meal ingredients.**

Keywords—CNN, KNN, Food, Dimensionality.

I. Introduction

While we are eating our meals, we might be wondering what we are eating and what our meal is composed of. Furthermore, we might have interest how to replicate the same delicacy at home. Therefore, this project is to identify the images of foods and try to find the specific food types corresponding to the images. For example, the user would input a picture of a pizza, and would like to know what kind of pizza it is and what the ingredients are, then our generator could give the user the answer. Or while traveling to foreign countries, we might be curious about unfamiliar cuisines, our generator could also provide the correct answer. In Fig. 1. is our general procedure for the generator. It utilizes two datasets with same classes and classifies from one dataset and compare features with the other dataset to obtain specified food types, details will be provided in the following report.

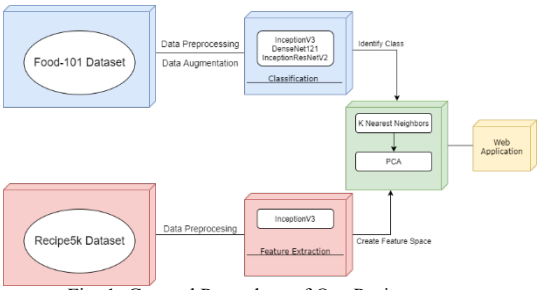

Fig. 1. General Procedure of Our Project

II. Dataset preparation

A. Acquiring Dataset

Since our goal is to identify ingredients from food images, our dataset requires food recipe data and food images. There was an ideal dataset provided by MIT. However, the dataset requires authorization and we did not have access to it. In order to solve this problem, there are two methods as follows:

- Scrape recipes and food images from cook book websites and label the data by hand.
- Utilize two datasets, one dataset to acquire labeled food images while the other provides food recipes with labeled food classes.

Our decision is to utilize two datasets. We acquire the labeled food image dataset Food101 from an 2014 ETHZ paper: Food-101- Mining Discriminative Components with Random Forests [1]. This dataset contains 101 food classes. In each class, it contains 1000 food images while 750 of the images goes to the train set and 250 images goes to the test set. In total, there are 101,000 images that are rescaled to have maximum side length of 512 pixels.

The other food recipe dataset is acquired from a 2017 ICIAP Workshops paper: Food Ingredients Recognition through Multi-label Learning [2]. This dataset contains 101 food classes, with each class containing about 50 recipes. In total, there are 4,826 unique recipes with 3,213 unique ingredients and each unique recipe corresponds to one food image.

B. Dataset preparation

Since the Food101 dataset is quite a large dataset, while experimenting with our model training the whole dataset would be quite inefficient. Therefore we randomly choose the number of classes to experiment on the model, creating a mini food dataset to train more quickly, saving us time and enabling us to tune the hyperparameters and experiment on more convolutional neural network (CNN) architectures.

About the Recipe5k dataset, it's recipes aren't aligned with the food classes. Hence, it requires some data cleaning in the food class names to correspond to the recipes perfectly.

In order to enhance image classification performance and prevent overfitting, data augmentation is necessary. I added some noise into the training images, containing image flips, shearing range and zooming range configuration. In this way,

we are able to prevent our model to overemphasize on our current training set and perform well on general datasets.

C. Feature Extraction

To perform feature extraction [3], we fed the Recipe5k images into the InceptionV3 [4] convolutional neural network architecture. InceptionV3 is the third version of the Inception family, the inception concept is to have filters with multiple sizes to operate on the same level. Since deeper network architectures are prone to overfit, this concept solves the problem by going wider. Moreover, to reduce expensive computation, dimension reduction is performed by adding a 1x1 convolution layer before other convolution layers. The main architecture of InceptionV3 is shown in Fig. 2.

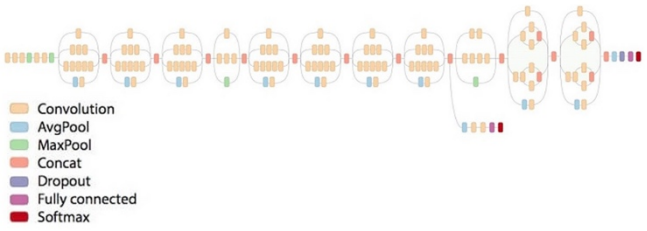

Fig. 2. Architecture of InceptionV3

After feeding the Recipe5k images into the CNN, it will give us the main features of the images for classification. To reduce dimensionality, I added an extra global average pooling layer and obtained images with 2048 dimensional features.

To visualize the feature space and perform dimensional reduction, I performed t-Distributed Stochastic Neighbor Embedding (t-SNE) [5]. It minimizes the divergence between two distributions, a distribution that measures pairwise similarities of the input objects and a distribution that measures pairwise similarities of the corresponding low-dimensional points in the embedding. Since it rescales quadratically in the number of objects, it can only be used as a visualization method as our dataset size is too large. The following Fig. 3. is a scatter plot of a sample of 10 classes from my original dataset.

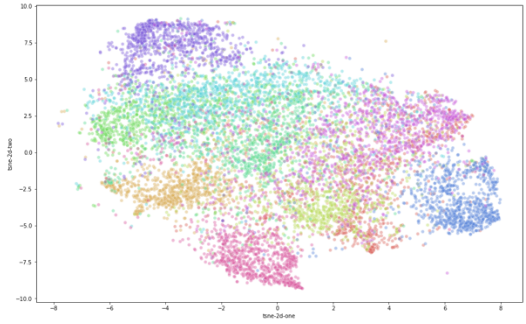

Fig. 3. Feature Visualization by t-SNE

III. Methods

We used different CNN architectures from recent Conference on Computer Vision and Pattern Recognition (CVPR) winners for our project to compare performances and boost accuracy. The CNN architectures are trained from ImageNet datasets. Since our dataset images are common photos in daily life, the coverage of ImageNet photos to our dataset shouldn't be too low. Hence, our method is to utilize the trained weights by transfer learning and adapt our model to it rather than freezing the convolution layers and fine tuning the model. We will briefly introduce the following CNN structures we have used and the K Nearest Neighbor algorithm.

A. InceptionV3

More than performing feature extraction, we also used InceptionV3 as convolutional base model. Its structure is also mentioned in the above section. The Inception family was introduced in 2015, and it performed exceptionally well compared to other models. In Fig. 4. is the performance plot of CNN models in recent years. The accuracy of Inception is close to the 2019 CVPR winner EfficientNet [8], which will be shortly introduced in the following section.

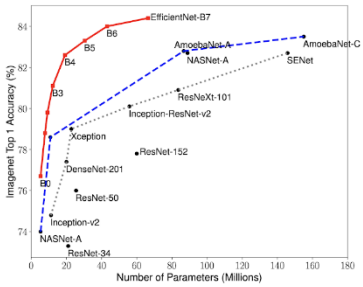

Fig. 4. Model Comparison Plot

B. DenseNet121

DenseNet [6] was developed specifically to improve the declined accuracy caused by the vanishing gradient in neural networks. Instead of drawing wider or deeper architectures, DenseNet exploits the potential of the network through feature reuse. In order to achieve this, DenseNet do not sum the output features of the layer with the incoming features but concatenate them.

$$x_l = H_l([x_0, x_1, \ldots, x_{l-1}]) \quad (1)$$

Because that grouping of feature maps can't be done when the sizes are different, DenseNet are divided into DenseBlocks which dimensions of the feature maps are the same. The number of filters changes between the DenseBlocks, increasing the dimensions of the channel. The growth rate k helps in generalizing the *l-th* layer, controlling the information added to each layer.

$$k_l = k_0 + k * (l - 1) \quad (2)$$

DenseNet has different versions according to the different layers in the neural network, consisting of transition layers, convolution layers, pooling layers, and DenseBlocks. The following Fig. 5. shows a 5-layer DenseBlock DenseNet structure.

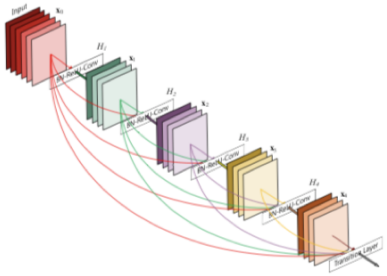

Fig. 5. Structure of DenseNet

C. *InceptionResNetV2*

InceptionResNetV2 is a CNN structure ensembled by Inception and ResNet structures. It is introduced along with InceptionV4, which also introduced specialized "Reduction Blocks" that were able to change the width and height of the grid. Residual connections were added to increase stability while training. However, with deeper residual units the network would destabilize, hence residual activations were also added. In Fig. 6. is the main structure for InceptionResNet.

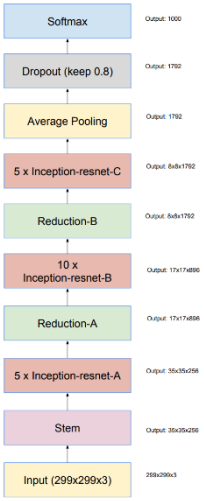

Fig. 6. Structure of InceptionResNet

D. *EfficientNet*

EfficientNet was developed using a multi-objective neural architecture search that optimizes operations. The search is conducted through a method named *compound scaling,* searching through model depth, width, and resolution to deliver optimal results. EfficientNet achieves state of art performance while being smaller and contains much less parameters. Hence, it achieved top scores for ImageNet recognition competitions. Other CNN architecture using the scaled method is greatly improved. However, I will not include the results of this architecture since its computational time is out of our expectation. It requires about 2 hours of time per epoch even for the smallest B0 version. It is still an exceptional architecture for us to experiment on.

E. *K Nearest Neighbors*

With our features of the images extracted, we needed to find a way to compare the input image features with the dataset image features. Hereby we use K Nearest Neighbors [9] to find the most similar images related to our input. We define the number of nearest neighbors as 3, but it is also available for users to define the number of recipes they want to correspond. With the feature vectors projected to the feature space, there are 2048 dimensions which is quite high. This could cause a problem because of the curse of dimensionality [12]. Hence, to use Euclidean distance to calculate the distances between feature vectors may not be precise since error may increase as dimension increases.

There are two ways to tackle this problem as follows:

- Changing the heuristic to Manhattan distance, since Euclidean distance tend to exaggerate discrepancy and is sensitive to extreme values.
- Perform Principal Component Analysis [10] to reduce dimensions. The t-SNE method isn't fit for our model since its more practical on smaller datasets.

Principle component analysis computes the principal components and using them to perform change of basis on the data. Principle components are identified by computing eigenvectors and the covariance matrix between features. In this way data is reduced to the most informative space, however nonlinear relations between the features may be lost in the preprocessing.

IV. Results

A. *Training Details*

Our model is trained on Google Colab Pro, which uses Nvdia T4 as GPU and TPUs are also available. To train with huger architectures like EfficientNetB3, it's neccesary to use Colab Pro to avoid memory limitation problems. The hyperparmeters are included in the following Table 1.

TABLE I. Model Hyperparameters and Details

Epochs	20
Batch Size	64
Optimizer	Stochastic Gradient Decent
Loss Function	Categorical Cross Entropy
Regularization	Ridge Regularization

Since our dataset is quite large, the training time for each epochs vary from 20 to 5 hours due to different architectures and GPUs. Therefore we only train 20 epochs for each model, which should be sufficient since the loss and accuracy curve is near convergence.

Batch size is being determined by the memory of our hardware, to achieve optimal results and compromise for computational time we choose the size 64.

Stochastic gradient decent [11] is performed by updating the direction on current batch. To avoid underfitting and

meeting plateaus, I chose a rather small learning rate of 0.0001 and momentum value as 0.9 to stablize updating process.

Cross entropy as loss function is being commonly use for categorical classification. Since all of our classes contain equal images, we don't need to worry about imbalanced classification problems and the loss function doesn't need weight adjustions. The equation for cross entropy is expressed as:

$$H(p, q) = -\sum_i p_i \log(q_i) \quad (3)$$

We also applied ridge regularization to prevent overfitting. It adds squared magnitude of coeffecients to the loss function, which is expressed as:

$$\sum_i^n (y_i - \sum_j^p x_{ij}\beta_j)^2 + \lambda \sum_j^p \beta_j^2 \quad (4)$$

B. *Model Results*

The model results of our 3 CNN models are as in Table 2.

TABLE II. MODEL RESULTS

Models	InceptionV3	DenseNet121	InceptionResNetV2
Train Accuracy	0.9555	0.8656	0.9127
Val Accuracy	0.9597	0.8972	0.9605
Val Loss	0.1907	0.3469	0.2382

As the results shows, the validation accuracy for food class classification is highest for InceptionResNetV2 while DenseNet121 performs worst. The convergence speed of the CNNs is worst for InceptionResNetV2 .Also, it's training time compared to the other two structures is 52 percent higher. Hence, I looked into the paper and realized InceptionV3 parallels better. The result plots are shown in the following Fig. 7. and Fig. 8.

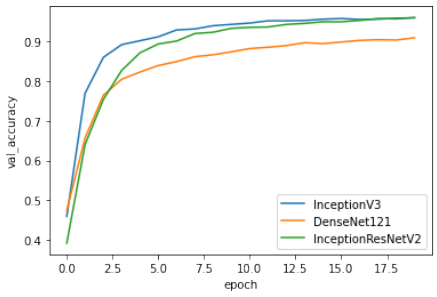

Fig. 7. Validation Accuracy Plot

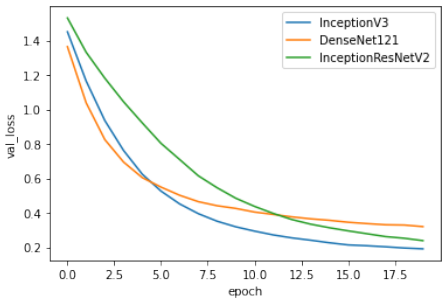

Fig. 8. Validation Loss Plot

After observing the training plots, I took a sample list of 10 classes using InceptionV3 to observe the heat map in Fig. 9. of the classification result. Since containing all 101 classes would be too large to observe. I noticed while classifying same categories such as " French Onion Soup" and " Hot Sour Soup", there would occur minor errors. Besides this minor error, our model seems to classify correctly shown in the diagonal columns.

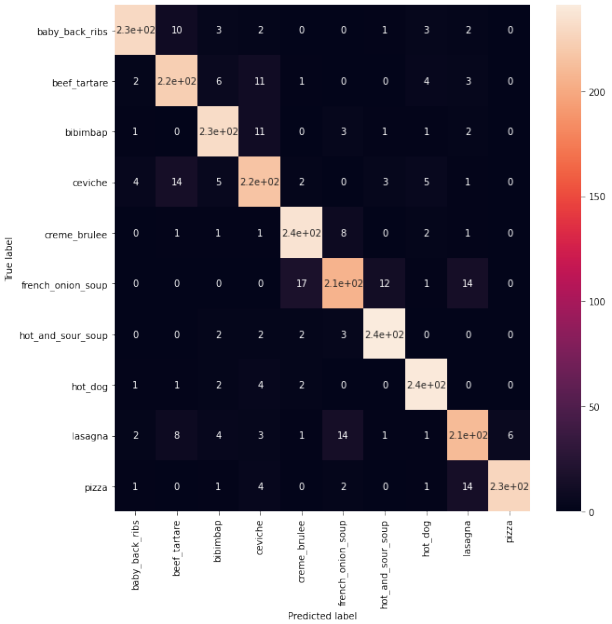

Fig. 8. Heat Map of 10 Sample Food Class

In Table 3. is the precision, recall and f1-scores of the ten food classes sampled the same as in Fig. 8. The overall average F1- score of all classes is 0.93, with "Soup" categories slightly lower than the other classes. To solve this problem the most straightforward idea is to add more data about soup to differentiate the dissimilarities between soups, or we could do more data augmentation in data preprocessing to increase our dataset size.

TABLE III. CLASSIFICATION REPORT

Classes	Precision	Recall	F1-score
Baby Back Ribs	0.94	0.95	0.94
Beef Tartare	0.90	0.94	0.92
Bibimbap	0.95	0.93	0.94
Ceviche	0.94	0.93	0.94
Crème Brulee	0.94	0.96	0.95
French Onion Soup	0.88	0.92	0.90
Hot and Sour Soup	0.90	0.87	0.89
Hot Dog	0.95	0.87	0.94
Lasagna	0.93	0.90	0.91
Pizza	0.93	0.92	0.92
Macro Average	0.93	0.93	0.93

C. Result Visualization

After successfully creating the model, I built a web application for users to interact with our project. The web application is built using StreamLit as API to package HTML and CSS language into python code, and the internet server was built on Ngrok. Our services are to input an image URL, and the application will connect to our computer server which runs the model to predict the food type and its ingredients. The demonstration is shown in Fig .11.

Fig. 11. Web Application Prediction Demo

After that, it will scrape 4 cafes or restaurants near Taipei City from the internet which might provide the food type that the user inputted to our model. Next, it will show the map locations of the four corresponding cafes or restaurants. The demonstration of the user interface is shown below in Fig. 12.

Fig. 12. Web Application Map Demo

V. Discussion and Conclusion

In this project, we have encountered some obstacles and managed to solve the difficulties. I will list them as follows.

- Our original objective was to utilize one dataset from the web but was denied access. However, we performed ensemble on two datasets and achieved our goal as proposed.
- Since computational time was expensive and we lack decent GPUs and memory space, we upgraded our hardware and preprocessed our data to be more efficient in computing.
- We still haven't been able to found datasets with full recipes included, so that our original future goal that we can generate recipes through natural language processing and transformer model was unable to continue.

Hence, our goal became to achieve higher accuracy in classifying foods, experimenting in smaller batches to find optimal hyperparameters and boost efficiency also. With this opportunity I am able to look further more into state of art CNN structures and study the innovations of recent structures. Also, learn more about dimension reduction and utilize this measure in real time data. This enables us to learn the crucial key to enhance our model and boost performance.

For future work, I wish to dig more into food and ingredients. Not only looking merely into the ingredients of the foods, but the nutrition of it also. Using ensemble methods to generate a meal that is essential to our body needs by nutrition information. Another direction of the future work is to generate recipes automatically or enabling neural networks to create new meals that are never seen before. And of course, we could always study more into CNN structures and try to keep boosting the accuracy of the model since there is still a lot of improving space.

Author Contribution Statements

Chun Min Chen (65%): Dataset Research and Survey, Model Coding, Dataset Preparation, Final Report Writing, Paper Researching, Model Experimenting and Analysis, Web Application Building

Yu Chen Chen (35%): Topic Development, Dataset Research and Survey, Proposal Presentation Preparation, Final Presentation Preparation

References

[1] Bossard L., Guillaumin M., Van Gool L. (2014) Food-101 – Mining Discriminative Components with Random Forests. In: Fleet D., Pajdla T., Schiele B., Tuytelaars T. (eds) Computer Vision – ECCV 2014. ECCV 2014. Lecture Notes in Computer Science, vol 8694. Springer, Cham. https://doi.org/10.1007/978-3-319-10599-4_29J.

[2] Marc Bolaños, Aina Ferrà and Petia Radeva. "Food Ingredients Recognition through Multi-label Learning" In Proceedings of the 3rd International Workshop on Multimedia Assisted Dietary Management (ICIAP Workshops), 2017.

[3] M. Jogin, Mohana, M. S. Madhulika, G. D. Divya, R. K. Meghana and S. Apoorva, "Feature Extraction using Convolution Neural Networks (CNN) and Deep Learning," *2018 3rd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*,Bangalore, India, 2018, pp. 2319-2323, doi: 10.1109/RTEICT42901.2018.9012507.

[4] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, and Jonathon Shlens. "Rethinking the Inception Architecture for Computer Vision"

[5] van der Maaten, Laurens & Hinton, Geoffrey. (2008). Viualizing data using t-SNE. Journal of Machine Learning Research. 9. 2579-2605.

[6] G. Huang, Z. Liu, L. Van Der Maaten and K. Q. Weinberger, "Densely Connected Convolutional Networks," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, 2017, pp. 2261-2269, doi: 10.1109/CVPR.2017.243.

[7] Szegedy, Christian & Ioffe, Sergey & Vanhoucke, Vincent & Alemi, Alexander. (2016). Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. AAAI Conference on Artificial Intelligence.

[8] Tan, Mingxing & Le, Quoc. (2019). EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks.

[9] Guo, Gongde & Wang, Hui & Bell, David & Bi, Yaxin. (2004). KNN Model-Based Approach in Classification.

[10] S. Sehgal, H. Singh, M. Agarwal, V. Bhasker and Shantanu, "Data analysis using principal component analysis," 2014 International Conference on Medical Imaging, m-Health and Emerging Communication Systems (MedCom), Greater Noida, 2014, pp. 45-48, doi: 10.1109/MedCom.2014.7005973.

[11] Ruder, Sebastian. (2016). An overview of gradient descent optimization algorithms.

[12] Keogh E., Mueen A. (2017) Curse of Dimensionality. In: Sammut C., Webb G.I. (eds) Encyclopedia of Machine Learning and Data Mining. Springer, Boston, MA. https://doi.org/10.1007/978-1-4899-7687-1_192