Fine-Grained Food Image Classification and Recipe Extraction using a Customised Deep Neural Network and NLP

Razia Sulthana Abdul Kareem[a], Timothy Tilford[b], Stoyan Stoyanov[c]

[a]*School of Computing and Mathematical Sciences, Faculty of Engineering and Science, University of Greenwich, London, SE10 9LS, , United Kingdom*
[b]*School of Computing and Mathematical Sciences, Faculty of Engineering and Science, University of Greenwich, London, SE10 9LS, , United Kingdom*
[c]*School of Computing and Mathematical Sciences, Faculty of Engineering and Science, University of Greenwich, London, SE10 9LS, , United Kingdom*

Abstract

Global eating habits cause health issues leading people to mindful eating. This has directed attention to applying deep learning to food-related data. The proposed work develops a new framework integrating neural network and natural language processing for classification of food images and automated recipe extraction. It address the challenges of intra-class variability and inter-class similarity in food images that have received shallow attention in the literature. Firstly, a customised lightweight deep convolution neural network model, MResNet-50 for classifying food images is proposed. Secondly, automated ingredient processing and recipe extraction is done using natural language processing algorithms: Word2Vec and Transformers in conjunction. Thirdly, a representational semi-structured domain ontology is built to store the relationship between cuisine, food item, and ingredients. The accuracy of the proposed framework on the Food-101 and UECFOOD256 datasets is increased by 2.4% and 7.5%, respectively, outperforming existing models in literature such as DeepFood, CNN-Food, Wiser, and other pre-trained neural networks.

Keywords: Image Classification, Ingredient Identification, Recipe Extraction, Deep Neural Networks, Domain Ontology, Natural Language Processing

1. Introduction

People have become more mindful about healthy eating. This has led to an increase in interest in healthy home cooking [1]. Culinary styles, known as cuisine, closely mirror individuals' eating habits, and there is a trend of people learning cooking through online resources. This trend has led to an increased circulation of food-related images on the internet and has opened up a lot of challenges in the field of Computer Vision and Artificial Intelligence [2]. Automated classification of diverse food images in different cuisines, estimating the platter nutritional quantity, identifying the ingredients from the food images, diet analysis, and food preference learning are a few areas of re-

(a) Inter Class Similarity (b) Intra Class variability

search attraction in recent days [3]. This demands training machines to understand the realm of culinary arts and hence can be suitably modified for specific applications.

Machine Learning inferential models plays a vital role in diet monitoring, calorie calculation, food weight estimation, and ingredient identification apps in day-to-day life [4]. However, these models often require extensive training and may be susceptible to errors due to human input. For instance, a user might upload an image of a food platter, hoping the application can identify the ingredients and provide a recipe for preparing it. Indeed, food images are examples of fine-grained visual recognition because they are non-rigid and have intrinsic properties.

Email addresses: razia.sulthana@greenwich.ac.uk (Razia Sulthana Abdul Kareem), t.tilford@greenwich.ac.uk (Timothy Tilford), s.stoyanov@greenwich.ac.uk (Stoyan Stoyanov)

Preprint submitted to Neurocomputing *April 27, 2024*

els are robust than human-validated ones, and exhibit superior performance in object identification, boundary detection, pattern/texture identification, etc.

A customised AlexNet Deep CNN (DCNN) model is introduced in [8], trained using the extensive ImageNet database and applied to the UECFOOD100 and UECFOOD256 datasets. The RootHoG feature extraction method was employed and the model's performance is compared against the Foodness Classifier (FC) model [9]. One of the main concerns in [8] DCNN model was the high number of parameters. The FC model in [9] is trained by tuning its hyperparameters over 470,000 food pictures scraped from Twitter to classify the image into 100 different classes. The model was trained with the UECFOOD256 databases and Food-101 databases [10] and recorded an accuracy of 94.3% on 'beef raman noodle' and 92.7% on curry.

The area of food image analysis faces several challenges regardless of whether a supervised learning algorithm, an unsupervised learning algorithm, or a deep neural network algorithm is used. Most existing research works apply these algorithms and models on the benchmarked datasets and medium-scale image datasets or endeavor to build a fully automated system without human intervention. Also, there exists an unintentional bias in classifying food images across different classes; heterogenous food image databases are another notable factor. To address these challenges and shortcomings of existing methods, the proposed approach focuses on building a comprehensive Food Classification and Recipe extraction by utilising an FC&R-CNN model with enhanced robustness to extract features from the food images. The utilization of Natural Language Processing (NLP) becomes essential for the efficient extraction and processing of recipes, cuisines, and food classes. Hence, NLP is introduced to discern and establish relationships within these elements, enhancing the effectiveness of recipe extraction and processing. This is achieved through the following novel developments:

Figure 1: Food Variants

Inter-class similarity [5] and intra-class variability [6] are the another characteristic of food images that render it hard to classify. The Fig. 1 (a) includes images of pasta, noodle, and salads, appear to be similar despite coming from different classes, demonstrating inter-class similarity. Alternately, the Fig. 1 (b) include images of different types of pizza and despite being in the same class, they look dissimilar demonstrating intra-class variability.

Most food categorization and ingredient detection systems rely on human validation to verify their predictions, often employing algorithms like k-Means, Support Vector Machines (SVM), and Vocabulary Trees [7]. Other studies focus on developing Convolution Neural Networks (CNN) for processing food images [8, 9, 10]. These self-learning CNN mod-

1. A custom-made CNN network architecture, MResNet-50 is built as a modified version of ResNet-50 and then is employed to classify images in UECFOOD256 and Food101 datasets. The MResNet-50 is applied to the UECFOOD256 dataset and Food-101 dataset to classify the images and it is found that MResNet-50 outperformed ResNet-50 by achieving higher accuracy on both datasets.
2. We propose to automatically identify and label the ingredients in the image using Natural Language Processing (NLP) techniques namely Word2Vec and Transformers and extract appropriate recipes from the Recipe1M+ and Recipe Ingredient datasets.
3. An ingredient-recipe hierarchical domain ontology is built to link ingredients, recipes, and cuisines facilitating structured knowledge organization. It automatically establishes relationships between ingredients, encompassing various food types and cuisines. Additionally, it identifies and categorizes food types containing allergens, promoting informed food choices.

The primary objective of the MResNet-50 model is to achieve fine-grained classification of food images by effectively discerning inter-class similarities and intra-class variabilities, utilizing its specifically designed layers for this purpose within the CNN model. Unlike existing literature models that rely on human intervention, the CNN model operates independently and does not necessitate human verification.

	Sections in the paper related to Objective 1	Sections in the paper related to Objective 2	Sections in the paper related to Objective 3
2. Related work	2.1 Food Image classification and Labelling	2.2 Recipe extraction	2.3 Domain ontology for food
3. Materials and Methods	3.1 Proposed DCNN Model	3.2 Automated Ingredient Identification using NLP	3.3 Food Domain Ontology
4. Experiments and Results	4.1.1 Food101 dataset 4.1.1 UECFOOD256 dataset	4.2 Results of Ingredient and Recipe extraction	4.3 Ontology validation

Figure 2: A schematic illustrating the organization of the paper and its associated objectives.

To assess its effectiveness, the model's performance is compared against those in the literature that require human intervention and struggle with identifying inter-class similarities and intra-class variabilities in images. This comparative analysis demonstrates that the proposed model achieves superior accuracy, confirming its capability in precisely classifying interclass similarities and intra-class variabilities.

As part of objective 3, a domain ontology specific to food images is constructed. Given its semi-supervised nature, the domain ontology allows for human intervention to evaluate classification quality and validate the model for potential future applications, if necessary.

The images depicted in Fig. 1 (a) and Fig. 1 (b) are sourced from the Food-101 and UECFOOD256 datasets, respectively which demonstrates inter-class similarity and intra-class variability. These images are considered as a basis for a brief case study analysis. In subsequent experiments, they are inputted into the proposed framework, the MResNet-50, which effectively classifies the images. Additionally, ingredients and cuisine are identified using the suggested NLP method, and the ontology is further updated with the pertinent information related to these images.

The rest of the paper is organized as follows. Section II reviews background research in food image classification, recipe extraction using NLP, and methods that use domain ontology for food-related applications. Section III provides a detailed explanation of the FC&R CNN which in three modules: the DCNN model (MResNet-50), recipe extraction algorithms in NLP, and food domain ontology. In Section IV, presents the implementation process and experimental findings. Section V concludes the paper by listing potential enhancements for future research.

2. Related work

This section explores the existing literature on classifying food images, identifying and labeling food items, extracting ingredients, and using ontologies for food-related applications corresponding to the objecctives. However the following sections and subsections are also framed each corresponding to one objective as shown in the Figure 2

Figure 3: A Collection of Food Image Datasets

2.1. Food Image Classification and Labelling

Research in food image classification has led to advancements in visual and feature-based recognition, making it possible to extract abstract information from food images using machine learning algorithms. Yang et al. [11] recognize the food by assigning a label to each of the pixels in the image which are then distributed over a histogram. The combined histogram from different portions of the image is taken as the feature vector and passed on to the discriminant classifier to recognize the food type. This method is termed Pairwise Feature Distribution (PFD). Although this method works effectively to detect food types in highly pixelated images, their performance tends to scale down in clustered images. The research presented in [12] uses Scale Invariant Feature Transformation (SIFT) to identify food types in images with variations in deformations and geometry. Despite being able to identify the food type in images with deformations and geometric variations, this approach has limitations as it cannot handle texture invariance in images. The list of food image datasets used in literature is shown in Fig 3. Following SIFT, a Non-redundant local binary pattern (NRLBP) is applied in [13] to extract the interest points in the image and and categorizes their appearance. However, there is room for analysing the structural invariance in the food images.

Alternately, a 2-step model is proposed in [14] on PFID dataset [15] to detect candidate regions in an image using region segmentation. DPM uses the sliding window to slide over the image making it computationally expensive and achieves a classification rate of 55.8%. A multi-ranking framework with modified growing region segmentation and SVM is proposed in [16]. It segments a food image into smaller regions and calculates similarity within pixels. The segmentation process is validated by human evaluation on 20 food images, achieving an accuracy of 61% for 17 food items. However, identifying tiny food ingredients with varying textures and colors would be more challenging and time-consuming. Another movable bounding box algorithm to identify the individual objects in an image is proposed in [17]. SVM with Linear kernel is applied to achieve a classification rate of 81.55%. The majority of recent work uses SVMs for classification and despite their popularity, SVMs perform poorly on large datasets and images with cluttered objects. Several deep learning algorithms are designed specifically for the classification of food images. The author in [18] proposes Im2Calories, which uses the GoogleNet CNN model on Food101 and MenuMatch dataset for calorie estimation of the food platter. This technique works effectively with raw, uncooked foods than cooked food because food change colour when they are cooked.

CNNs have been used extensively in the food image processing. Krizhevsky CNN [19] is trained with ImageNet to extract the colour features from the image. Comparing the Krizhevsky CNN's performance to that of the Spatial Pyramid

Matching (SPM) algorithm and the conventional SVM algorithm, the Krizhevsky CNN outperforms the other two benchmark techniques.

Bounding boxes are used to identify and categorise the objects in an image, based on which several DCNN networks are proposed in the literature [20]. These networks typically extract both high-level and pixel-level features from the image. The bounding boxes are formed by dividing the image into regions using: R-CNN [21], Fast R-CNN [22], Faster R-CNN [23], Mask R-CNN [24] models. Regression-based bounding boxes based on class probabilities are framed using YOLO [25], YOLOv2 [26], SSD [27], DSSD [28] CNN models. In General, CNN frameworks like AlexNet, VGGNet, ResNet, and DarkNet are used for object detection. Inspired by ResNet [29], a robust feature extractor DarkNet-53 [30], is developed, and almost all the variants of the ResNet models have succeeded in providing excellent results with many datasets and benchmark applications. Different ResNet models, including ResNet-18, ResNet-19, ResNet-20, and ResNet-34 using CIFAR datasets, are mentioned in a recent work [31, 32]. The reason the authors in [31] have decided to investigate ResNet models is that they utilise less energy than other pretrained networks, which motivated us to further investigate ResNet and introduce a modified version of it for the proposed study. A Wide Hierarchical Subnetwork-based Neural Network (Wi-HSNN) is applied in [13] which employs a subnet-based iterative training and a batch-by-batch parallel scheme. This approach is particularly useful for processing large-scale datasets but faces the drawbacks of subnetwork algorithms.

2.2. Recipe Extraction

The process of extracting a food's recipe based on its ingredients requires the application of NLP techniques and a detailed understanding of the semantic relationships between ingredients. Probabilistic models, Bayesian models, Neural network models, Text processing models, and several other models are proposed in the literature for recipe extraction.

A graphical model in NLP, Conditional Random Field (CRF) is applied to the VIREO Food-172 dataset and the corpus from the 'Xinshipu' website [33]. The graphical model, G is a collection of ingredients and contains N vertices representing the ingredients. The model generates a matching score that is matched against corresponding recipe's CRF. However, it is a computationally expensive process and demands more training time. A probabilistic Bayesian cuisine topic model is used for recipe identification from food images in [34]. It applies probability distribution using a Gaussian kernel over Yummily66K and also compares the result with the Bayesian model and Boltzmann machines. Nevertheless, the topics under which the cuisines are classified are statically fixed but have to vary dynamically as different food styles evolve.

A joint neural embedding with Long-Short Term Network (LSTM) is used for learning the recipe-image pair in Recipe1M corpus [35]. It trains the neural network by embedding the recipes with the food images. ResNet-50 and VGG-16 models are executed over the dataset in three versions: fixed vision, fine-tuning, and semantic regularization. The semantic embedding approach outperforms the other two in both the ResNet50 and VGG-16 CNN models. Although semantic embedding is taken into account, many of the foods prepared today use ingredients that are in new and unusual combinations, which makes the process challenging. An extended VGG and multitask CNN is applied in [36]. Term Frequency–Inverse Document Frequency (TF-IDF) and Word2Vec from NLP are applied for recipe extraction. The ingredient vector v_j corresponding to r_j recipe data is defined by $v_j = \sum_{k=1}^{N} tf - id\ f_{k,j} * Word2Vec(w_k)$, where N is the count of words corresponding to a recipe, Word2Vec() is the vector derived for the specific word w_k, and $tf - id\ f$ is the TF-IDF value of the ingredient in the recipe. VGG, on the other hand, is more computationally intensive than ResNet models.

2.3. Domain Ontology for food based applications

Domain ontology is a knowledge framework of specific concepts that are designed to a specific domain and is extensively applied in many different applications. However, little research is done in the area of food ingredient-recipe background. FoodOn [37] is a specific food ontology that provides the semantics of food nutritional facts, chemical ingredients, and nutritional components. But it contains a lot of information about food-related nutrition facts than food-recipe itself. FoodKG [38] proposes a food ontology graph that recommends food to the user based on nutritional facts on a day-to-day basis. Each of the ingredients, its corresponding recipes, and their nutritional composition is represented as entities and relationships in the ontology hierarchy. NLTK toolkit is also used to manage the queries raised by the users and nouns, verbs, adverbs are handled by WordNetLemmatizer. FoodKG is linked to FoodOn and semantic descriptions from USDA (the U.S. Department of Agriculture) but yet designed with very minimal ontological conceptualization.

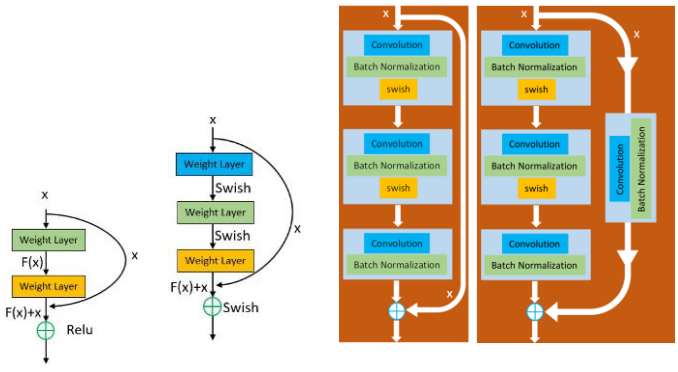
3. Materials and Methods

In this section, we discuss the proposed approach in three folds.

3.1. Proposed DCNN Model

CNNs are layered architecture that is helpful for image labeling, annotation, classification, etc. These multi-layered neural network process the input images with minimal preprocessing. Almost all the pre-trained CNN architectures are trained with ImageNet Dataset (The biggest repository of 1.2 million highresolution images). The ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) competition evaluates

neural network algorithms developed for image classification [19]. It is obligatory to report top-1 % and top-5 % error rates when the



(a) ResNet-34 (b) Proposed MResNet-50 (c) Identity block and Convolution block — MResNet-50 Proposed model

Figure 4: Building blocks

Imagenet dataset is used over any CNN architectures. Top-1 % accuracy is obtained when the image is rightly identified. Alternatively, Top-5% accuracy is achieved when the predicted result matches with the top 5 images that are identified by ImageNet. A CNN is trained using ImageNet to carry out a particular task, initial weights are assigned to the network links, and the transfer learning approach is used to use the learned information to complete another task.

CNN networks are modeled with several convolution layers followed by at least one fully connected layer with a set of nodes that corresponds to the number of image classes. ResNets, the residual network model was developed by He et al., in the year 2016 [29]. The ResNet-50 model is designed with 48 convolutional layers, one max pooling layer, and one fully connected layer. As compared to AlexNet and VGGNet, ResNet exhibits superior generalization performance with essentially fewer parameters compared to other models. Deeper CNNs like AlexNet would typically increase accuracy, but they would cause vanishing gradient problems [39]. ResNet, on the other hand, introduces a skip connection to solve the issue of vanishing gradients in deep layers

Fig. 4 (a) & (b) shows the building block of the ResNet model. The input x is passed into the first layer and is processed by a mapping function $H(x)$ to produce an output function $F(x)$, where $F(x) = H(x) - x$. The input x is again passed to layer 3 skipping layer 2. Hence layer 3 reads input x explicitly and draws an identical pattern from it. Although the layers are stacked, the input is skipped to the middle layer, referring to it as a ”skip connection,” and since identity mapping is carried out, neither the parameters nor the complexity is increased. The proposed MResNet-50 model is designed to use swish activation function in all its building blocks (Fig. 4 (b)).

The swish activation function [40] proposed by Google outperforms Relu activation function at many occasions during image classification. The function swish,$f(x) = x.sigmoid(x)$ i.e. $f(x) = f(x) = {}_{1+}{}^{x}{}_{e-x}$ as compared to relu, $f(x) = max(0, x)$, used in Inception-ResNet-v2 and Mobile NASNetA shows improved accuracy of 0.9% and 0.6%, respectively over ImageNet dataset. Inspired by the properties of swish being a non-monotonic, smooth, and self-gated function, the performance of MResNet-50 further increases with swish. The two building blocks of the MResNet-50: Identity block and Convolution block are shown in Fig. 4 (c). The identity block produces an output: $swish((block3(block2(block1(x))))+ x)$ and the output produced by convolution block would be: $swish((block3(block2(block1(x)))) + batch\_norm(conv2D(x)))$ with x as input.

The MResNet-50 model is designed in five stages with 50 layers. The MResNet-50 model is designed to be lightweight as the

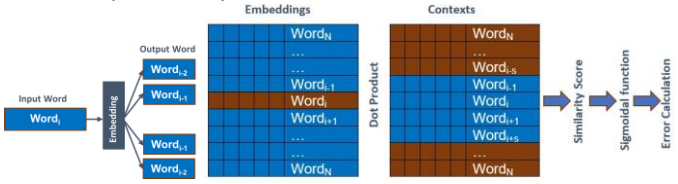
1. The first stage is designed with the convolution operation of 64 kernels each with a kernel size of 7*7 with a stride of 2. Following this, batch normalization, swish activation, and max pooling operation are done. This is the first layer of MResNet-50.
2. The second stage has one convolution block and two identity blocks. In each block three convolution operations are carried out with a kernel size of 1*1, 3*3, and 1*1 and a corresponding kernel count of 64, 64, and 256. This adds 9 layers to the MResNet-50.
3. The third stage has one convolution block and three identity blocks. In each block three convolution operations are carried out with a kernel size of 1*1, 3*3, and 1*1 and a corresponding kernel count of 128, 128, and 512. This adds 12 layers to the MResNet-50. 8% of the filters are reduced in this layer following the final identity block.
4. The fourth stage has one convolution block and five identity blocks. In each block three convolution operations are carried out with a kernel size of 1*1, 3*3, and 1*1 and a corresponding kernel count of 256, 256, and 1024. This adds 18 layers to the MResNet-50. 10% of the filters are reduced in this layer following the final identity block.
5. The fifth stage is similar to stage 2 but with a kernel count of 512, 512, and 2048 respectively for each layer. This adds 9 layers to the MResNet-50.
6. This is followed by the average pooling, layer flattening, fully connected layer, and a softmax function. It adds 1 layer to the MResNet-50.

On the whole, they all add up to 50 layers in the MResNet-50. The swish activation function is used in all the layers, and the model is trained and tested while the epoch and optimizer are changed and validated.

3.2. Automated Ingredient Identification using NLP

Following the food images classification in the Food101 and UECFOOD256 datasets, the Recipe1M+ dataset and the Recipe Ingredient datasets are processed using the Word2Vec algorithm to learn the dependencies between the ingredient and food classes.

The Word2Vec converts every word into a vector representation without losing its syntactic and semantic properties. It applies the Cosine function and finds the correlation between the words. Since Word2Vec is a two-layered network with one hidden and one output layer. The Word2Vec learns by speculating on the words around the input word of the food item using the formula: $Word2Vec(word_{in}) = word_{in} * P(^{word}_{word^{out}_{in}})$ i.e.



(a) skip-gram model (b) Word Embedding & similarity calculation

Figure 5: Word2Vec

$Word2Vec(fooditem) = fooditem * P(\forall_{wordsinsentence} fooditem)$.

Word2Vec uses the Skip-gram approach to predict the probability of the context given the name of the food item (Fig. 5(a)). For example, to search for the word 'mac and cheese' in a random sentence *one-half teaspoon of chitpole makes a spicy mac* taken from the Recipe1M+ dataset, the probable predictions are

$P(macandcheeseone-half)$, $P(macandcheeseteaspoon)$, $P(macandcheeseof)$, $P(macandcheesechitpole)$, $P(macandcheesemakes)$, $P(macandcheesea)$, $P(macandcheesespicy)$, $P(macandcheesemac)$.

For every word $word_{position} = [1,2,3,4,...P]$ in the randomly chosen sentence within a fixed window size of s, the context within the words is calculated. The likelihood of the prediction is given by

$likelihood(\Theta) = \prod_{i=1}^{P} \prod_{-s \leqslant j \leqslant s, j, 0} P(word_{i+j} | word_i; \theta)$ which is $likelihood(\Theta) = \prod_{i=1}^{P} \prod_{-s \leqslant j \leqslant s, j, 0} P(word_{out} | word_{in}; \theta)$. The *word out* slides every word in the sentence and the $P(word_{out} | word_{in})$ is calculated by

$$P(word_{out} | word_{in}) = \frac{}{\forall_{window} exp(word exp(word_{in} * word_{in} * word_{out})_{out})}.$$

The softmax function is applied on $word_{in} * word_{out}$ and for every food item that is passed into the untrained model, the embeddings are mapped, probability prediction and likelihood are calculated, and the output vocabulary $softmax(word_{in} * word_{out})$ is predicted (Fig. 5(b)). In simple words, the Skipgram algorithm, part of the Word2Vec framework, is extensively utilized for learning word embeddings. These embeddings represent words in a continuous vector space and are designed to predict surrounding context words based on a given target word. This approach effectively captures semantic relationships between words. The food item identified by the Word2Vec and MResNet-50 models is validated against various window sizes/embeddings during the training phase until the best embedding with the highest score that shows the highest similarity is found. The embeddings that generate a negative error score are ignored while training the next food item saving the execution time.

The transformers are another tool for finding the long-term dependencies between the text contents [41]. The transformer has two parts encoder and a decoder. The architecture of the transformer is shown in [42]. The encoder and decoder have multi-head attention and feed-forward blocks. In addition, the decoder has a masked multi-head attention block. The encoder maps the word representation $word = [w_1, w_2, w_3, w_4, ... w_P]$ to a sequence of continuous values $z = [v_1, v_2, v_3, v_4, ... v_P]$, the decoder utilises the results of the previous iteration and then generates output words in an autoregressive fashion. A pointwise convolution with fully connected layers and a stacked selfattention unit is used in both the encoder and decoder units.

The attention function takes the query and key-value pair as input and outputs a weighted sum of a value. The

$$attention(Query, Key, Value) = softmax\left(\frac{Query * Key_{Transpose}}{sqrt(Dimension_{Key})}\right) * Value$$ where $dimension_{key}$ is the dimension of the query and key. The multi-head attention function in the decoder instead of performing dot-product of queries with keys projects the querykey value N times, and the results of each of the projections are finally concatenated $multiheadattention(Query, Key, Value) = Concatenation(A_1, A_2, ..A_N)$ where A_i represents the $attention(Query, Key, Value)$ for every ith projection. In the proposed method, we train the model with sentences from Recipe1M+ and Recipe Ingredient dataset and each training batch includes a sentence with a word length of 100. The training period in our machine took 0.02 seconds and the overall training time was around 32 mins significantly smaller than the preprocessed dataset. Alternately during the initial execution of the transformer model architecture, it took 12 hrs for training 36 million sentences in 100,000 steps. One of the effective features of the transformer model is the attention mechanism and this capability allows transformers to handle all words or tokens simultaneously, enhancing processing

speed and facilitating the development of larger language models. The proposed approach integrates both Word2Vec and Transformers to effectively manage short words and lengthy sentences, particularly in reverse order. This integration results in the creation of a robust automated recipe extraction model.

3.3. Food Domain Ontology

A directed graphical ontology is developed using the Protege tool. The food items, ingredients, and cuisine are considered to be the main classes and each of their occurrence would be an instance. The relation between the food item and the ingredient would be a 'has ingredient' relationship, and similarly, there are many other relationships like 'use together', and 'check for allergen'. The items used in the ingredients may be a simple mixture or compound mixture and are also specified in the ontology. The proposed one is capable of accommodating many hierarchies and its semi-supervised. The hierarchy can be extended to make it deeper still keeping the model faster. Food items and ingredients identified by the model in the aforementioned section are stored in the ontology.

For a specific cuisine, there may be a number of food items, but a few might be prepared using the staple ingredients and many of them are made from the same ingredients but in a different form. For example, the dish Gratin (id 30 in UECFOOD256 category) is a French Cuisine and can be made with breadcrumbs, grated cheese, egg, butter with many other vegetables. Alternately, Gulai (id 231 in UECFOOD256 category) is Indonesian cuisine and is made with completely different types of ingredients. Though there are a number of cross relations between the cuisine, food items, and ingredients the ontology always organises to bring them into a structure.

4. Experiments and Results

The proposed framework is executed in a x86-64 machine (minimal 32GB) connected to NVIDIA TITAN X (32GB dedicated memory) GPU including a software stack comprising of GPU driver 352.68 or newer, a CUDA toolkit of 8.0 or newer, a python version 3.7 or compatible version beyond 2.7. The execution environment is pytorch build stable (1.13.1), package CONDA, language python and computation platform CUDA. We selected the specified hardware because the chosen hardware configuration possessed sufficient speed to manage the extensive datasets we were working with and the algorithms we designed. Above all it was readily accessible to us. The reason for creating a customised CNN architecture is that it is optimized and trained to excel at the detection of patterns and intricate features within and across food images. This customized model strikes a balance between the task's complexity and the computational resources at hand. Moreover, it integrates domain-specific insights by incorporating a domain ontology. The fine-tuning of hyperparameters is conducted systematically to enhance the model's performance for the targeted application.

4.1. MResNet-50 performance analysis

The MResNet-50 model is trained using ImageNet and the weights are reused for classifying images in Food-101, and UECFOOD256 datasets. The input image to the ResNet50 model is of size 256*256 with 3 slices of RGB image taken from both datasets. The model is trained and tested against three optimizers: Adaptive moment estimation (Adam), Stochastic Gradient Descent (SGD), and SGD with momentum. The performance of the model is analysed with each of the aforementioned optimizers and using three loss functions: (1) Angular Softmax, (2) Categorical Cross-Entropy, and (3) Large Margin Softmax Loss. In addition, the batch size, validation split, and epoch are varied, and the model's performance is measured using several metrics: training loss, test loss, training accuracy, test Accuracy, precision, and F1 score. A very detailed layer architecture of the proposed MResNet-50 is explained below and shown in Fig. 6 and the below steps shows how it is made lightweight by introducing the pruning operation at intermediary stages:

1) In the 1st layer-1st stage, the input image of shape (256*256*3) corresponding to (image height, image width, channels) is passed into the MResNet-50 model. The input image is zero-padded with (3*3) matrix and is then subjected to 2D convolution operation between input image (256,256,3) and 64, (7*7) kernel with stride 2. Both the input image and kernels are square in shape. The spatial dimension of the resultant feature map is calculated using $\frac{inputsize}{(height/width_{strides})+2*padding-kernelsize}k + 1$. This implies $\left\lfloor \frac{256+2*3-7}{2} \right\rfloor + 1$ giving 128 as the dimension of the new feature map. It produces a resultant feature map of size (128*128*64). Following this Batch Normalization is executed with axis value=3, swish activation function, and max pooling with (3*3) kernel size with a stride of 2. The shape of $\frac{128-3}{2}$ the new feature map after max pooling is calculated using $\frac{inputsize(height/width)-kernelsize}{strides}k\,jk + 1$ which implies +1 which gives a feature map with a dimension of (63*63*64). 2) In 2nd layer-2nd stage, the feature map (63*63*64) is subjected to a 2D convolution operation with 64, (1*1) kernel with a stride=1 and padding=0. The resultant feature map is of size (63,63,64). This is followed by Batch Normalization with axis value=3 and swish activation function.

3) In 3rd layer-2nd stage, the feature map (63*63*64) is subjected to a 2D convolution operation with 64, (3*3) kernel with a stride=1 and padding=1. The resultant feature map is of size (63,63,64). This is followed by Batch Normalization with axis value=3 and swish activation function.

4) In 4th layer-2nd stage, the feature map (63*63*64) is subjected to a 2D convolution operation with 256, (1*1) kernel with a stride=1 and padding=0. The resultant feature map is of size (63,63,256). This is followed by Batch Normalization with axis value=3 and swish activation function.

The feature map out of the 1st layer-1st stage is subjected to 2D convolution with 256, (1*1) kernel, stride=1, padding=0 followed by batch normalization and produces an output feature map (63,63,256) that is added to the output feature map from 4th layer-4th stage (63,63,256). This is subjected to the swish activation function and the (63*63*256) feature map is passed to the next layer.

5) The 5th, 6th and 7th layer in stage 2 performs the same functions similar to 2nd, 3rd and 4th layer in stage 2.

Alternately, the output feature map out of the 4th layer-2nd stage is added to the output feature map of the 7th layer-2nd stage and subjected to the swish activation function.

6) The 8th, 9th and 10th layer in stage 2 performs the same functions similar to 2nd, 3rd and 4th layer in stage 2. The output feature map out of the 7th layer-2nd stage is added to the output feature map of the 10th layer-2nd stage and subjected to the swish activation function. Following which 8% of the layers are pruned to ensure that the model is made lightweight.

7) In the 11th layer-3rd stage, the feature map (63*63*256) is subjected to a 2D convolution operation with 128, (1*1) kernel with a stride=2 and padding=0. The resultant feature map is of size (32,32,128). This is followed by Batch Normalization with axis value=3 and swish activation function.

8) In 12th layer-3rd stage, the feature map (32*32*128) is subjected to a 2D convolution operation with 128, (3*3) kernel with a stride=1 and padding=1. The resultant feature map is of size (32,32,128). This is followed by Batch Normalization with axis value=3 and swish activation function.

9) In the 13th layer-3rd stage, the feature map (32*32*128) is subjected to a 2D convolution operation with 512, (1*1) kernel with a stride=1 and padding=0. The resultant feature map is of size (32,32,512). This is followed by Batch Normalization with axis value=3 and swish activation function.

The feature map out of the 10th layer-2nd stage is subjected to 2D convolution with 512, (1*1) kernel, stride=2, padding=0 followed by batch normalization and is added to the output feature map from the 13th layer-3rd stage. This is subjected to the swish activation function and the (32*32*512) feature map is passed to the next layer.

10) The batch of (14th, 15th, 16th), (17th, 18th, 19th), (20th, 21st, 22nd) layer performs operation similar to (11th, 12th, 13th) layers. The resultant of it is a (32*32*512) feature map. Following this, 10% of the layers are pruned to ensure that the model is made lightweight.

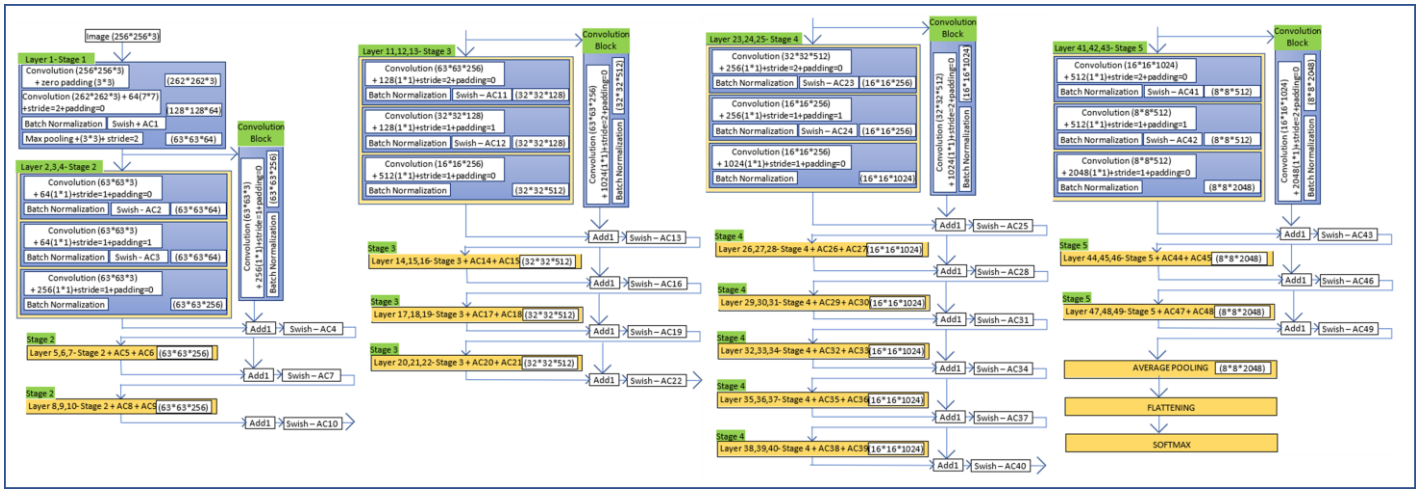
11) In 23rd layer-4th Stage the feature map (32*32*512) is subjected to a 2D convolution operation with 256, (1*1) kernel with a stride=2, padding=0. The resultant feature map is of size (16*16*256). This is followed by Batch Normalization with axis value=3 and swish activation function. Layers 24th, and 25th are then executed in the same way as followed in the 12th and 13th layers but with a kernel count of 256 and 1024. Following this, the convolution block is applied. The resultant of this layer would be a (16*16*1024) feature map. 12) The batch of (26th, 27th 28th), (29th, 30th, 31st), (32nd, 33rd, 34th), (35th, 36th, 37th), (38th, 39th, 40th) layer performs operation similar to (23rd, 24th 25th) layers. However, following each of these batches, the identity block is applied. The resultant of this layer would be a (16*16*1024) feature map 13) In 41st layer-5th Stage the feature map (16*16*1024) is subjected to a 2D convolution operation with 512, (1*1) kernel with a stride=2, padding=0. The resultant feature map is of size (8*8*512). This is followed by Batch Normalization with axis value=3 and swish activation function. Layers 42nd, and 43rd are then executed in the same way as followed in the 24th and 25th layers but with a kernel count of 512 and 2048. Following this, the convolution block is applied. The resultant of this layer would be an (8*8*2048) feature map

14) The batch of (44th, 45th, 46th), (47th, 48th, 49th) layer performs operation similar to (41st, 42nd, 43rd) layers. However, following each of these batches, the identity block is applied. The resultant of it is an (8*8*2048) feature map.

15) The 50th layer performs average pooling. In average pooling the (8*8*2048) feature map is reduced to (4*4(2048). This is followed by a flattening and a softmax layer.

The results of the MResNet-50 model are applied to the Food-101 dataset Table 1 with a split ratio of 80:20 for training and testing. The experimental analysis is carried out with batch sizes of 16 and 32 and alternating with changing epochs of 10, 35, and 50 respectively for each of the optimizers Adam, SGD, and SGD-w-M (SGD with Momentum).

It is observed that the batch size and the number of epochs have an impact on the results produced by the three optimizers. During the testing and training phases of the MResNet-50, the loss values and accuracy values for each of the three loss functions are measured. The accuracy value achieved during testing for all three optimizers is slightly less than training, although the difference is not exceptionally significant. Hence, the MResNet-50 model is neither underfitting nor overfitting, as evidenced by the results. The model can predict accurately for testing data since it has learned adequately from the training data. The subsubsections 4.1.1 and 4.1.2, address the results of the FOOD-101 dataset and the UECFOOD256 dataset, respectively. The results obtained on the FOOD-101 dataset and the UECFOOD256 dataset are separately mentioned in Table 1



Batch Size	Optimizer	Epochs	Angular Softmax loss								Categorical Cross entropy loss				Large Margin Softmax Loss			
			Loss		Accuracy						Loss		Accuracy		Loss		Accuracy	
			Train	Test	Train	Test					Train	Test	Train	Test	Train	Test	Train	Test
16	Adam	10	0.0881	0.0954	0.8331	0.8040					0.0874	0.0929	0.8273	0.7982	0.0886	0.0941	0.8285	0.7994
16	Adam	35	0.0697	0.0790	0.8719	0.8346					0.0924	0.0983	0.8658	0.8284	0.0932	0.0991	0.8665	0.8292
16	Adam	50	0.0511	0.0601	0.9140	0.8779					0.1141	0.1202	0.8108	0.8704	0.1149	0.1209	0.9096	0.8947
32	Adam	10	0.0313	0.0426	0.9068	0.8915					0.1084	0.1143	0.8397	0.8543	0.1091	0.1150	0.9001	0.8550
32	Adam	35	0.0839	0.0939	0.9135	0.8635					0.1081	0.1143	0.8901	0.8564	0.1093	0.1155	0.8751	0.8575
			0.9132	0.1023	0.1077	**0.9197**	**0.9065**	0.1032	0.1086	0.9043								
			0.8175	0.1149	0.1204	0.8408	0.8100	0.1157	0.1212	0.8416								
16	SGD	35	0.0783	0.0868	0.8942	0.8602					0.1075	0.1146	0.8871	0.8530	0.1082	0.1153	0.8878	0.8538
16	SGD	50	0.0715	0.0806	0.8240	0.7875					0.0859	0.0913	0.8183	0.7818	0.0871	0.0925	0.8195	0.7830

32	SGD	10	0.0518	0.061	0.8895	0.8524								0.0900	0.0965	0.9063	0.8463	0.0907	0.0972	0.8842	0.8471
32	SGD	35	0.0557	0.064	0.9042	0.8696	0.8618	0.1084	0.1138	**0.9138**	0.8547	0.1096	0.1149	**0.9009**	0.1137	0.1203	0.8739	0.8621	0.1146	0.1216	0.8980	0.8630
							0.8394	0.1065	0.1128	0.8756	0.8323	0.1077	0.1139	0.8768								
16	SGD-w-M	35	0.0531	0.061	0.8427	0.8093									0.0907	0.0978	0.8366	0.8032	0.0919	0.0990	0.8378	0.8043
16	SGD-w-M	50	0.0777	0.084	0.9020	0.8761									0.0884	0.0952	0.8261	0.8701	0.0894	0.0963	0.8970	0.8711
32	SGD-w-M	10	0.0619	0.069	0.9030	0.8689									0.0884	0.0943	0.8636	0.8630	0.0891	0.0955	0.8756	0.8636
32	SGD-w-M	35	0.0635	0.071	0.8798	0.8478									0.0925	0.0987	0.8900	0.8416	0.0936	0.0997	0.8647	0.8427
32	SGD-w-M	50	0.0844	0.090	0.8966	0.9155									0.0986	0.1056	0.8971	0.9089	0.0995	0.1064	0.8909	0.9097

Figure 6: MResNet-50 architecture diagram

Table 1: Performance of the MResNet-50 model using Angular Softmax Loss, Categorical cross Entropy Loss, Large Margin Softmax Loss Function over Food 101 dataset

Table 2: Performance of the MResNet-50 model using Angular Softmax Loss, Categorical cross Entropy Loss, Large Margin Softmax Loss Function over UEC-Food256

Batch Size	Optimizer	Epochs	Angular Softmax loss				Categorical Cross entropy loss				Large Margin Softmax Loss			
			Loss		Accuracy		Loss		Accuracy		Loss		Accuracy	
			Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
16	Adam	10	0.0978	0.1002	0.8440	0.8155	0.0942	0.0973	0.8510	0.7897	0.0945	0.0969	0.8127	0.7937
16	Adam	35	0.0804	0.0824	0.8810	0.8644	0.0656	0.0682	0.8508	0.8389	0.0728	0.0740	0.8280	0.8143
16	Adam	50	0.0725	0.0743	0.8883	0.8051	0.0910	0.0938	0.8400	0.8090	0.0763	0.0786	0.8220	0.8050
32	Adam	10	0.0570	0.0584	0.8460	0.8168	0.0810	0.0836	0.8237	0.7468	0.1082	0.1100	0.8044	0.7869
32	Adam	35	0.1053	0.1079	0.8922	0.8616	0.0530	0.0560	0.8211	0.8317	0.0642	0.0655	0.8524	0.8175
32	Adam	50	0.0924	0.0947	0.8700	0.8270	0.0680	0.0713	0.8089	0.8169	0.0702	0.0719	0.8265	0.8124
16	SGD	10	0.1061	0.1085	0.8125	0.7545	0.1056	0.1090	0.8034	0.7624	0.0916	0.0944	0.7765	0.7565
16	SGD	35	0.0930	0.0953	0.8213	0.8005	0.0856	0.0886	0.8903	0.8681	0.0736	0.0750	0.9021	0.8436
16	SGD	50	0.0747	0.0766	0.8567	0.8087	0.0856	0.0880	0.8769	0.8531	0.0510	0.0530	0.8947	0.8697

32	SGD	10	0.0710	0.0720	0.8660	0.7859											0.0520	0.0550	0.8300	0.8090	0.0950	0.0970	0.8890	0.8746
32	SGD	35	0.0751	0.0770	0.8910	0.8557											0.0824	0.0840	0.8700	0.8790	0.0570	0.0590	0.8380	0.8251
32	SGD	50	0.1042	0.1068	0.8855		0.0722	0.0755	0.8844	**0.8863**	0.0680	0.0698	0.87											
									0.8610	0.8602	0.0807	0.0830	0.87											
16	SGD-w-M	10	0.0511	0.0624	0.8620	0.8339											0.0920	0.094						
16	SGD-w-M	35	0.0565	0.0679	0.8590	0.7910											0.0894	0.0918	0.8428	0.8121	0.0670	0.0700	0.8240	0.8077
16	SGD-w-M	50	0.0925	0.0948	0.8923	0.9220											0.1042	0.1072	0.8153	0.8809	0.0870	0.0904	0.8950	0.8753
32	SGD-w-M	10	0.0558	0.0870	0.8770	0.8676											0.0740	0.0760	0.7420	0.7421	0.0970	0.0990	0.8110	0.7946
32	SGD-w-M	35	0.0982	0.1000	0.8950	0.8420											0.0660	0.0690	0.7110	0.7305	0.0660	0.0682	0.7790	0.7659
32	SGD-w-M	50	0.0885	0.0907	0.9065	0.8315											0.0810	0.0841	0.7955	0.8762	0.0850	0.0870	0.8680	0.8519

and Table 2 respectively. The significant findings and the accuracy associated with each optimizer are delineated using red and blue font colors to facilitate quick referencing. The highest accuracy achieved for each optimizer is represented in the font color red.

4.1.1. Food101 dataset- Inferences on the Results-Table 1

With angular softmax loss function, the optimizer Adam achieves the maximum training accuracy as compared to the other two optimizers for a batch size of 32, and after training the model with 50 epochs produces a 92.67% accuracy. Alternatively, this could be overfitting as with the same batch size, and in 35 epochs, Adam produces an accuracy of 91.35%. During training on the ImageNet Dataset, ResNet-50 produced a greater accuracy overall in 35 iterations [43], while MResNet50 likewise delivers the greatest accuracy in 35 iterations. Similarly, the accuracy produced by the optimizer SGD with a batch size of 32 and 35, and 50 epochs, respectively, is 90.42% and 90.96%, and there is no noticeable difference between these numbers. The accuracy produced by SGD with Momentum optimizer seems to be fluctuating between 89% and 91% with batch sizes of 16 and 32.

With Categorical cross-entropy loss, the optimizer Adam produces the highest training accuracy of 91.97% in 50 epochs with a batch size of 32 and 89.01% in 35 epochs with the same batch size. Also, these two numbers do not significantly differ from one another. Hence 35 epochs are again considered to be the optimal number with MResNet-50. The SGD optimizer slightly fluctuates in the accuracy value with 16 and 32 batch sizes with 10, 35, and 50 epochs. SGD with Momentum optimizer produces an accuracy of 89.00% and 89.71% with a batch size of 32 and in 35 and 50 epochs with no discernable difference between them.

With Large Margin Softmax loss, the optimizer Adam produces the highest training accuracy of 90.96% with a batch size of 16 in 50 epochs. The optimizer SGD with batch size 32 produces an accuracy of 90.09% and 89.80% in 35 and 50 epochs respectively and there is no major noticeable difference between the numbers. Surprisingly, in batch size 16, SGD with Momentum optimizer produces an accuracy of 89.09%, and 86.47% for 50 and 35 epochs, and a slight difference between them is seen. Overall, the Adam optimizer with MResNet-50 produces good results on the Food101 dataset across all three optimizers and the accuracy produced by all three optimizers is higher for batch size 32 than for 16.

4.1.2. UECFOOD256 dataset- Inferences on the Results-Table 2

With Angular softmax loss the SGD with Momentum optimizer produces a higher accuracy of 90.65% in 50 epochs with batch size 32 and 89.51% in 35 epochs with batch size 32 and there is no discernible difference between them. Adam produces the second-highest accuracy of 89.22% in 35 epochs and SGD produces 89.13% in 35 epochs with batch size 32. As such in both Adam and SGD, there is no major difference in the accuracy.

With Categorical Cross entropy loss, the SGD optimizer produces a higher accuracy of 89.03% in 35 epochs with a batch size of 16 and in 50 epochs the accuracy is slightly getting reduced for the same batch size hence 35 epochs with 16 batch size is considered an optimal tuning. The Adam optimizer produces an accuracy of 85.10% and 85.08% with 16 batch sizes but with 10 and 35 epochs respectively. The SGD with Momentum optimizer produces an 86.10% and 84.28% with 10

and 35 epochs with batch size 16. As a unique feature, the batch size of 16 is well suited for Categorical cross-entropy loss with the UECFOOD256 dataset.

With the large margin softmax loss for the SGD produces a higher accuracy of 90.21% for a batch size of 16 and in 35 epochs and 89.47% for the same batch size in 50 epochs. The SGD with momentum also produces an accuracy of 89.50% with batch size 16 for 50 epochs and 82.50% with the batch size but for 35 epochs. The difference seems to be a bit higher in this case.

Overall for the UECFOOD256 dataset the accuracy produced by all three optimizers is higher for batch size 16 than for 32 and the very highest is produced by SGD with momentum. The fluctuations observed in the results of implementing MResNet50 on various optimizers for both datasets, where the highest accuracy obtained exceeds 85%, might stem from several factors. These include the diversity of images within the datasets, variations in optimizers and loss functions utilized, as well as differences in hyperparameter tuning.

Comparative analysis of the proposed MResNet-50's Top1% and Top-5% accuracy values to other models in existing literature are shown in Table 3. Top-1% accuracy refers to the model accurately predicting the food item to its corresponding class, while top-5% accuracy refers to the predicted class matching any of the top five predictions. Additionally, the proposed model proves that the prediction accuracy of the Top5% is greater than the prediction accuracy of the Top-1%. The execution time taken by MResNet-50 for Food101 dataset using Adam optimizer is 29.2 min, 30.1 min and 38.6 min and for UECFOOD256 dataset using SGD optimizer it is 19.8 min, 18.4 min and 23.1 min for each of the models as shown in Table 3. The reduced execution time for the UECFOOD256 dataset compared to the FOOD-101 dataset is attributed to the smaller size of the UECFOOD256 than the FOOD-101 dataset.

The results in Table 3 show that the MResNet-50 model proposed demonstrates superior performance compared to existing models that required human intervention and those that struggle with intra-class similarity and inter-class variability issues. This achievement substantiates the successful attainment of our objective. Moreover, we have augmented the test image set with intricate images to evaluate the model's ability to handle diverse scenarios. The higher accuracy observed in handling these images further validates the effectiveness of our approach. As part of the case study, food items mentioned in Fig. 1 (a) and Fig. 1 (b) are extracted from the FOOD101 and UECFOOD256 datasets and the proposed MResNet-50 model accurately classifies the food images.

4.2. Results of Ingredient and Recipe extraction

The ingredient identification module runs on the Recipe1M+ and recipe ingredients dataset using two algorithms. The results of the Word2Vec are shown in Table 4. Following the preprocessing, the skip-gram algorithm in Word2Vec is applied with the vector length of 20 and 40 to identify the target ingredients and thereby extract them. The lower error means that the vector and the food item are closely related to each other whereas if higher it means they are far away from each other and that specific recipe/ingredient is not to the food item. However, the results of applying Word2Vec before and after stemming show a marginal difference because stemming changes the meaning of the sentence. Hence the results shown in the table is the one that is obtained before applying the stemming. We train the 256 and 101-dimensional models on more than a million words and

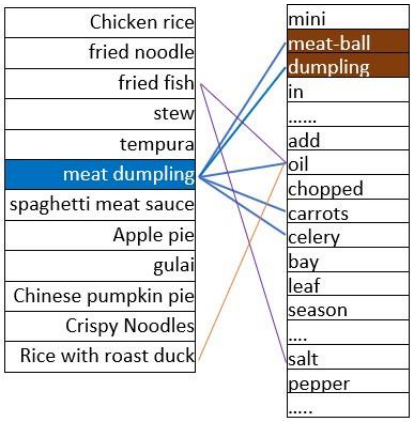


Figure 7: Patterns learnt by the Transformer

compute the score of every sentence and try to find the appropriate sentence with the highest computing score. At the end of the iteration, the final score will be added to the sum of all the individual predictions, and using the likelihood factor the relevant sentence is identified and extracted.

The Transformer model is executed on both datasets. As an initial step, they are subjected to byte-pair encoding. The millions of sentence pairs are divided into tokens i.e. word-piece vocabulary. The sentence pairs are then grouped into a specific lengths of 20 and 40. The Recipe1M+ dataset has two layers: layer 1 and layer 2. The BLEU (BiLingual Evaluation Understudy) score is used for measuring the similarity of prediction in machine translation of searching. The results of applying transformers and regular baseline LSTM (Long short-term Memory) [41] are shown in Table 5. The LSTM approaches using single forward and single reversed methods followed in [41] are applied to both datasets. The training BLEU score produced by Transformers on both datasets is recorded. Fig. 7 shows the pattern learned by the Transformer.

The LSTM is very effective in handling short sentences but not efficient in handling reversed sentences because it fails to understand the reversed sequences. On the contrary, transformers are very good at handling both of them. Most importantly, we apply the unoptimized, simple, and straightforward transformer function for handling both the long and short sequences.

4.3. Ontology validation

The proposed semi-structured ontology shown in Fig. 8 can be validated using descriptive information [50] which tells us how well the ontology can provide us with the specified information for the query that is asked of it.

An ingredient name when passed onto the ontology will identify all the food types that use it. The ontology can be used to filter the food types that have a specific allergen. The advantage of ontology over using documenting software is that ontology reunites the food types that share similar ingredients near to each other in space thereby reducing the entropy value. The ontology that is built for this application has one-to-one, one-tomany, and many to one and many to many relationships. Cer-

Method	Training score 1M+	BLEU Recipe	Training score Recipe ingredients dataset	BLEU
LSTM single forward [41]	28.97		25.23	
LSTM single reversed [41]	35.28		29.01	
Transformers	45.12		24.19	

3: Comparison of the performance of the Top-1% Top-5% of the proposed MResNet-50 model with other existing deep CNN models over Food-101 and)OD256 dataset

CNN approaches	Food-101 Dataset		CNN approaches	UECFOOD256	
	Top-1%	Top-5%		Top-1%	Top-5%
Deep Food [44]	77.4	93.7	DeepFood [44]	54.7	81.5
CNN-FOOD(ft) [8]	70.41	-	CNN-FOOD(ft) [8]	67.57	88.97
ResNet(APL) [45]	78.5	94.1	ResNet(APL) [45]	71.2	91.1
Inception-V3 [46]	88.28	96.88	Inception-V3 [46]	76.17	92.58
WISeR [47]	90.27	98.71	WiSer [47]	83.15	95.45
AlexNet-CNN [10]	56.4	-	DeepFoodCam [48]	63.77	85.82
Inception+Wi-HSNN [49]	84.7	-	Inception+Wi-HSNN [49]	77.7	-
(Combined results of ResNet, DenseNet, and InceptionNet) +WiHSNN [49]	90.8	-	(Combinedresults of ResNet, DenseNet, and InceptionNet) +WiHSNN [49]	83.1	-
ResNet-50 [3]	82.54	95.79	ResNet-50 [3]	71.7	91.33
MResNet-50 - Angular Softmax Loss (32, Adam, 50) (29.2 min)	92.67	96.43	MResNet-50 - Angular Softmax Loss (32, SGD-w-M, 50) (19.8 min)	90.65	95.47
MResNet-50 - Categorical Cross Entropy Loss (32, Adam, 50) (30.1 min)	91.97	97.29	MResNet-50 - Categorical Cross Entropy Loss (16, SGD, 35) (18.4 min)	89.03	94.35
MResNet-50 - Largin Margin Softmax Loss (16, Adam, 50) (38.6 min)	90.96	95.79	MResNet-50 - Largin Margin Softmax Loss (16, SGD, 35) (23.1 min)	90.21	96.59

Table 4: Word2Vec SkipGram Training Error on both the datasets

Input word	Output vector length	Target	Recipe 1M+ Training Error	Recipe Ingredient dataset
Food item 1	20	0	-0.91	-0.82
Food item 1	40	1	0.64	0.58
Food item 2	20	1	0.18	0.17
Food item 2	40	1	-0.83	-0.77
Food item 3	20	0	-0.08	-0.03
Food item 3	40	0	0.72	0.76
Food item 4	20	1	0.32	0.28
Food item 4	40	0	0.59	0.58
Food item 5	20	1	-0.89	-0.87
Food item 5	40	1	-0.12	0.78
Food item 6	20	0	0.90	0.81
Food item 6	40	0	-0.43	-0.65

Table 5: Transformers BLEU score on both the datasets

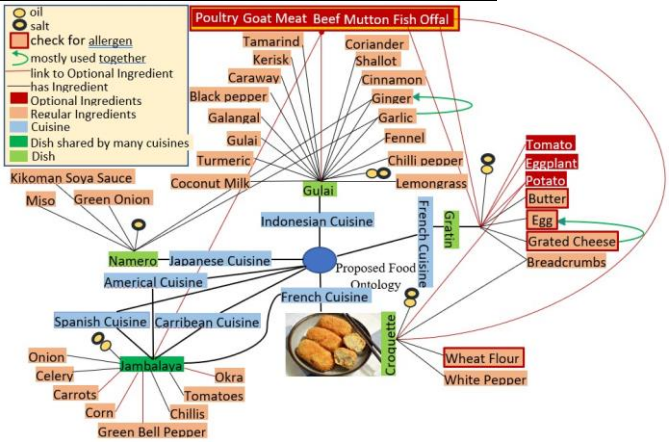


Figure 8: Ontology picture of cuisine-food type relationship

tain dishes are related to specific cuisine which signifies oneto-one relation and a few food items are shared across many cuisines showing one-to-many relations, many ingredients are used in one food item showing many relations, and there are a lot many other relations which are indirectly seen between food items, cuisines, ingredients, etc. The interrelation between food and cuisine can be quickly identified in ontology than the conventional search techniques.

In summary, the application of MResNet-50 to the Food-101 and UECFOOD256 datasets yielded accuracies of 92.67% and 90.65%, respectively compared to existing models. Additionally, employing the transformer model for ingredient and recipe extraction resulted in BLEU scores of 45.12 and 24.19 on the Recipe 1M+ and Recipe Ingredients datasets compared to other techniques. The domain ontology model integrates the architecture of food-cuisine and ingredients, establishing a comprehensive framework for Fine-Grained Food Image Classification and Recipe Extraction through a tailored combination of Deep
Neural Networks and Natural Language Processing techniques.

Like other research endeavors, this project faces limitations from two distinct perspectives: the deep learning framework and the chosen application. In terms of model development, one of the main concern involves fine-tuning hyperparameters in ResNet models, given their sensitivity. This sensitivity may hinder the model's ability to capture the intricate longrange dependencies within images during training. However, this is effectively addressed in the proposed work by leveraging a domain ontology, which documents relationships between cuisines and ingredients. Nevertheless, this necessitates the creation of different domain ontology for various applications. Regarding the food image application, a limitation stems from the diversity of cultures and regions, each with its unique cuisine, which are not all recorded or the images captured into available datasets. Additionally, to personalize these applications for individual users, their dietary preferences and allergens need to be captured and by cross-referencing these preferences, a holistic application can be developed to meet the diverse needs of users.

5. Conclusion

This research paper presents a novel Fine-Grained Food Image Classification and Recipe Extraction framework, integrating a Customized Deep Neural Network and NLP. The framework achieves its objectives through three key innovations. Firstly, a tailored MResNet-50 model is developed, enhancing food item identification accuracy, surpassing traditional ResNet-50 models and other similar models in literature. The MResNet-50 model is optimized for a lightweight design through layer modifications, feature pruning, and the adoption of the swish activation function. Secondly, an NLP module, employing Recipe1M+ and Recipe Ingredient datasets, utilizes Word2Vec and transformer algorithms to identify ingredients, demonstrating superior performance compared to LSTM approaches. Thirdly, an ontology structured around cuisine, food type, and ingredients is established, facilitating rapid allergen identification through the Protege ontology model.

The combined framework of CNN, NLP, and Ontology exhibits robust and precise performance in food image processing applications, achieving notable accuracies of 92.67% and 90.65% on the Food-101 and UECFOOD256 datasets, respectively. Furthermore, employing the transformer model for ingredient and recipe extraction yields significant BLEU scores of 45.12 and 24.19 on the Recipe 1M+ and Recipe Ingredients datasets, outperforming alternative techniques. The developed ontology enhances the framework's versatility for future applications and accelerates data processing. The framework enables accurate food image classification while automating recipe extraction through NLP and ontology. Considering its potential for future extensions, this solution stands as a viable option capable of delivering commendable results within a reasonable timeframe.

The impact of this work extends to the healthcare domain, empowering vulnerable patients to manage ingredient labels, allergens, and overall health, while aiding food safety inspection operators in allergen detection and advisory tasks. Additionally, the lightweight design of the framework enables integration with wearable smart devices, facilitating various healthcare applications such as nutrition tracking, calorie computation, remote monitoring, and telehealth services. Its lightweight nature ensures swift real-time execution especially in mobile applications, offering quick outputs, which could be explored as a potential avenue for future expansion.

References

[1] M. T. Gorski, C. A. Roberto, Public health policies to encourage healthy eating habits: recent perspectives, Journal of healthcare leadership 7 (2015) 81.

[2] V. Kakani, V. H. Nguyen, B. P. Kumar, H. Kim, V. R. Pasupuleti, A critical review on computer vision and artificial intelligence in food industry, Journal of Agriculture and Food Research 2 (2020) 100033.

[3] G. Ciocca, P. Napoletano, R. Schettini, Cnn-based features for retrieval and classification of food images, Computer Vision and Image Understanding 176 (2018) 70–77.

[4] D. Sahoo, W. Hao, S. Ke, W. Xiongwei, H. Le, P. Achananuparp, E.-P. Lim, S. C. Hoi, Foodai: Food image recognition via deep learning for smart food logging, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 2260–2268.

[5] S. Horiguchi, S. Amano, M. Ogawa, K. Aizawa, Personalized classifier for food image recognition, IEEE Transactions on Multimedia 20 (10) (2018) 2836–2848.

[6] M. Taskiran, N. Kahraman, Comparison of cnn tolerances to intra class variety in food recognition, in: 2019 IEEE International Symposium on Innovations in Intelligent Systems and Applications (INISTA), IEEE, 2019, pp. 1–5.

[7] Y. He, C. Xu, N. Khanna, C. J. Boushey, E. J. Delp, Analysis of food images: Features and classification, in: 2014 IEEE international conference on image processing (ICIP), IEEE, 2014, pp. 2744–2748.

[8] K. Yanai, Y. Kawano, Food image recognition using deep convolutional network with pre-training and fine-tuning, in: 2015 IEEE International Conference on Multimedia & Expo Workshops (ICMEW), IEEE, 2015, pp. 1–6.

[9] K. Yanai, Y. Kawano, Twitter food photo mining and analysis for one hundred kinds of foods, in: Pacific Rim Conference on Multimedia, Springer, 2014, pp. 22–32.

[10] L. Bossard, M. Guillaumin, L. V. Gool, Food-101–mining discriminative components with random forests, in: European conference on computer vision, Springer, 2014, pp. 446–461.

[11] S. Yang, M. Chen, D. Pomerleau, R. Sukthankar, Food recognition using statistics of pairwise local features, in: 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, IEEE, 2010, pp. 2249–2256.

[12] Z. Zong, D. T. Nguyen, P. Ogunbona, W. Li, On the combination of local texture and global structure for food classification, in: 2010 IEEE International Symposium on Multimedia, IEEE, 2010, pp. 204–211.

[13] D. T. Nguyen, Z. Zong, P. O. Ogunbona, Y. Probst, W. Li, Food image classification using local appearance and global structural information, Neurocomputing 140 (2014) 242–251.

[14] Y. Matsuda, H. Hoashi, K. Yanai, Recognition of multiple-food images by detecting candidate regions, in: 2012 IEEE International Conference on Multimedia and Expo, IEEE, 2012, pp. 25–30.

[15] M. Chen, K. Dhingra, W. Wu, L. Yang, R. Sukthankar, J. Yang, Pfid: Pittsburgh fast-food image dataset, in: 2009 16th IEEE International Conference on Image Processing (ICIP), IEEE, 2009, pp. 289–292.

[16] L. Oliveira, V. Costa, G. Neves, T. Oliveira, E. Jorge, M. Lizarraga, A mobile, lightweight, poll-based food identification system, Pattern Recognition 47 (5) (2014) 1941–1952.

[17] Y. Kawano, K. Yanai, Real-time mobile food recognition system, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2013, pp. 1–7.

[18] A. Meyers, N. Johnston, V. Rathod, A. Korattikara, A. Gorban, N. Silberman, S. Guadarrama, G. Papandreou, J. Huang, K. P. Murphy, Im2calories: towards an automated mobile vision food diary, in: Proceedings of the IEEE international conference on computer vision, 2015, pp. 1233–1241.

[19] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, Communications of the ACM 60 (6) (2017) 84–90.

[20] L. Aziz, M. S. B. H. Salam, U. U. Sheikh, S. Ayub, Exploring deep learning-based architecture, strategies, applications and current trends in generic object detection: A comprehensive review, IEEE Access 8 (2020) 170461–170495.

[21] R. Girshick, J. Donahue, T. Darrell, J. Malik, Rich feature hierarchies for accurate object detection and semantic segmentation, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2014, pp. 580–587.

[22] R. Girshick, Fast r-cnn, in: Proceedings of the IEEE international conference on computer vision, 2015, pp. 1440–1448.

[23] S. Ren, K. He, R. Girshick, J. Sun, Faster r-cnn: Towards real-time object detection with region proposal networks, Advances in neural information processing systems 28 (2015).

[24] K. He, G. Gkioxari, P. Dollar, R. Girshick, Mask r-cnn, in: Proceedings of´ the IEEE international conference on computer vision, 2017, pp. 2961–2969.

[25] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, You only look once: Unified, real-time object detection, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 779–788.

[26] J. Redmon, A. Farhadi, Yolo9000: better, faster, stronger, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 7263–7271.

[27] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, A. C. Berg, Ssd: Single shot multibox detector, in: European conference on computer vision, Springer, 2016, pp. 21–37.

[28] C.-Y. Fu, W. Liu, A. Ranga, A. Tyagi, A. C. Berg, Dssd: Deconvolutional single shot detector, arXiv preprint arXiv:1701.06659 (2017).

[29] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

[30] Q.-C. Mao, H.-M. Sun, Y.-B. Liu, R.-S. Jia, Mini-yolov3: real-time object detector for embedded applications, Ieee Access 7 (2019) 133529–133538.

[31] Y. Guo, W. Peng, Y. Chen, L. Zhang, X. Liu, X. Huang, Z. Ma, Joint a-snn: Joint training of artificial and spiking neural networks via selfdistillation and weight factorization, Pattern Recognition 142 (2023) 109639.

[32] S. Zhang, M. Gao, Q. Ni, J. Han, Filter pruning with uniqueness mechanism in the frequency domain for efficient neural networks, Neurocomputing 530 (2023) 116–124.

[33] J. Chen, C.-W. Ngo, Deep-based ingredient recognition for cooking recipe retrieval, in: Proceedings of the 24th ACM international conference on Multimedia, 2016, pp. 32–41.

[34] W. Min, B.-K. Bao, S. Mei, Y. Zhu, Y. Rui, S. Jiang, You are what you eat: Exploring rich recipe information for cross-region food analysis, IEEE Transactions on Multimedia 20 (4) (2017) 950–964.

[35] A. Salvador, N. Hynes, Y. Aytar, J. Marin, F. Ofli, I. Weber, A. Torralba, Learning cross-modal embeddings for cooking recipes and food images, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 3020–3028.

[36] T. Ege, K. Yanai, Image-based food calorie estimation using recipe information, IEICE TRANSACTIONS on Information and Systems 101 (5) (2018) 1333–1341.

[37] D. M. Dooley, E. J. Griffiths, G. S. Gosal, P. L. Buttigieg, R. Hoehndorf, M. C. Lange, L. M. Schriml, F. S. Brinkman, W. W. Hsiao, Foodon: a harmonized food ontology to increase global food traceability, quality control and data integration, npj Science of Food 2 (1) (2018) 1–10.

[38] S. Haussmann, O. Seneviratne, Y. Chen, Y. Ne'eman, J. Codella, C.-H. Chen, D. L. McGuinness, M. J. Zaki, Foodkg: a semantics-driven knowledge graph for food recommendation, in: International Semantic Web Conference, Springer, 2019, pp. 146–162.

[39] S. Hochreiter, The vanishing gradient problem during learning recurrent neural nets and problem solutions, International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 6 (02) (1998) 107–116.

[40] P. Ramachandran, B. Zoph, Q. V. Le, Searching for activation functions, arXiv preprint arXiv:1710.05941 (2017).

[41] I. Sutskever, O. Vinyals, Q. V. Le, Sequence to sequence learning with neural networks, Advances in neural information processing systems 27 (2014).

[42] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, Advances in neural information processing systems 30 (2017).

[43] K. Osawa, Y. Tsuji, Y. Ueno, A. Naruse, R. Yokota, S. Matsuoka, Secondorder optimization method for large mini-batch: Training resnet-50 on imagenet in 35 epochs, arXiv preprint arXiv:1811.12019 1 (2018) 2.

[44] C. Liu, Y. Cao, Y. Luo, G. Chen, V. Vokkarane, Y. Ma, Deepfood: Deep learning-based food image recognition for computer-aided dietary assessment, in: Inclusive Smart Cities and Digital Health: 14th International Conference on Smart Homes and Health Telematics, ICOST 2016, Wuhan, China, May 25-27, 2016. Proceedings 14, Springer, 2016, pp. 37–48.

[45] Z. Fu, D. Chen, H. Li, Chinfood1000: A large benchmark dataset for chinese food recognition, in: Intelligent Computing Theories and Application: 13th International Conference, ICIC 2017, Liverpool, UK, August 7-10, 2017, Proceedings, Part I 13, Springer, 2017, pp. 273–281.

[46] H. Hassannejad, G. Matrella, P. Ciampolini, I. De Munari, M. Mordonini, S. Cagnoni, Food image recognition using very deep convolutional networks, in: Proceedings of the 2nd international workshop on multimedia assisted dietary management, 2016, pp. 41–49.

[47] N. Martinel, G. L. Foresti, C. Micheloni, Wide-slice residual networks for food recognition, in: 2018 IEEE Winter Conference on applications of computer vision (WACV), IEEE, 2018, pp. 567–576.

[48] Y. Kawano, K. Yanai, Food image recognition with deep convolutional features, in: Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication, 2014, pp. 589–593.

[49] W. Zhang, J. Wu, Y. Yang, Wi-hsnn: A subnetwork-based encoding structure for dimension reduction and food classification via harnessing multicnn model high-level features, Neurocomputing 414 (2020) 57–66.

[50] C. Reyes-Pena, M. Tovar-Vidal, Ontology: components and evaluation, a review, Research in Computing Science 148 (3) (2019) 257–265.