SELF-ATTENTION ARCHITECTURE FOR INGREDIENTS GENERATION FROM FOOD IMAGES

A Project Report

Submitted to the Faculty of Engineering of

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY KAKINADA, KAKINADA

In partial fulfilment of the requirements for the award of the Degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

By

CHAPARALA. JYOTHSNA	**BANDI. BHARGAVI**
(18481A0538)	**(18481A0522)**
AKURI. ESWAR SRAVANTH	**ATMURI. TRINADH KUMAR**
(18481A0504)	**(18481A0516)**

Under the guidance of

Dr. K. Srinivas, M. Tech, PhD

Associate Professor, Department of CSE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SESHADRI RAO GUDLAVALLERU ENGINEERING COLLEGE

(An Autonomous Institute with Permanent Affiliation to JNTUK, Kakinada)

SESHADRIRAO KNOWLEDGE VILLAGE

GUDLAVALLERU – 521356

ANDHRA PRADESH

2021-22

SELF-ATTENTION ARCHITECTURE FOR INGREDIENTS GENERATION FROM FOOD IMAGES

A Project Report

Submitted to the Faculty of Engineering of

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY KAKINADA, KAKINADA

In partial fulfilment of the requirements for the award of the Degree of

BACHELOR OF TECHNOLOGY
In
COMPUTER SCIENCE AND ENGINEERING

By

CHAPARALA. JYOTHSNA	**BANDI. BHARGAVI**
(18481A0538)	**(18481A0522)**
AKURI. ESWAR SRAVANTH	**ATMURI. TRINADH KUMAR**
(18481A0504)	**(18481A0516)**

Under the guidance of
Dr. K. Srinivas, M. Tech, PhD
Associate Professor, Department of CSE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SESHADRI RAO GUDLAVALLERU ENGINEERING COLLEGE
(An Autonomous Institute with Permanent Affiliation to JNTUK, Kakinada)
SESHADRIRAO KNOWLEDGE VILLAGE
GUDLAVALLERU – 521356
ANDHRA PRADESH
2021-22

SESHADRI RAO GUDLAVALLERU ENGINEERING COLLEGE

(An Autonomous Institute with Permanent Affiliation to JNTUK, Kakinada)

SESHADRI RAO KNOWLEDGE VILLAGE, GUDLAVALLERU

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project report entitled **"SELF-ATTENTION ARCHITECTURE FOR INGREDIENTS GENERATION FROM FOOD IMAGES"** is a bonafide record of work carried out by **CH. JYOTHSNA (18481A0538), B. BHARGAVI (18481A0522), A. ESWAR SRAVANTH (18481A0504), A. TRINADH KUMAR (18481A0516)** under the guidance and super vision of **Dr. K. SRINIVAS** in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering of Jawaharlal Nehru Technological University Kakinada, Kakinada during the academic year 2021-22.

Project Guide	**Head of the Department**
(Dr. K. SRINIVAS)	**(Dr. M. BABU RAO)**

External Examiner

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of people who made it possible and whose constant guidance and encouragements crown all the efforts with success.

We would like to express our deep sense of gratitude and sincere thanks to **Dr. K. Srinivas**, Associate Professor, Department of Computer Science and Engineering for his constant guidance, supervision and motivation in completing the project work.

We feel elated to express our floral gratitude and sincere thanks to **Dr. M. Babu Rao**, Head of the Department, Computer Science and Engineering for his encouragements all the way during analysis of the project. His annotations, insinuations and criticisms are the key behind the successful completion of the project work.

We would like to take this opportunity to thank our beloved principal **Dr. G.V.S.N.R.V Prasad** for providing a great support for us in completing our project and giving us the opportunity for doing project.

Our Special thanks to the faculty of our department and programmers of our computer lab. Finally, we thank our family members, non-teaching staff and our friends, who had directly or indirectly helped and supported us in completing our project in time.

Team members
CH. JYOTHSNA (18481A0538)
B. BHARAGAVI (18481A0522)
A. ESWAR SRAVANTH (18481A0504)
A. TRINADH KUMAR (18481A0516)

INDEX

LIST OF ABBREVIATIONS

Abbreviation	Explanation
ResNet	Residual Networks
eos	End of Sequence
IOU	Intersection over Union
TP	True Positives
FP	False Positives
FN	False Negatives
CNN	Convolutional Neural Networks
URL	Uniform Resource Locator

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Food filming is becoming more popular among food connoisseurs. Each meal has a narrative that is detailed in a lengthy recipe, and sadly, just looking at a dish provides no insight into the preparation of food. There are various websites that aid in recognizing a meal by its components, however, no method has been successful at forecasting the ingredients in a dish. As a consequence, this research proposes a method for automatically generating the dish's recipe. This approach estimates the image's title and ingredients and then generates the image's featured meal's specific cooking instructions. This investigation looked at a range of Indian cuisines, including lunch and breakfast. To enhance user-friendly, this paper provides a web application that displays the recipe with a photograph of the dish.

Keywords: Image Recognition, Multi-label Classification, Self-Attention, ResNet-50, Recipe Generation, Food Images, Feed Forward Network

CHAPTER-I
INTRODUCTION

1.1 INTRODUCTION

Cooking is a pleasurable art form. Nowadays, postings tagged with food and cooking have increased significantly on social media sites such as Instagram, Facebook, and Twitter. People found the effort of visualizing delectable meals and sharing them with the outside world to be enjoyable. Additionally, eating habits and culinary culture have evolved throughout time. Traditionally, meals were produced mostly at home, but people now regularly eat food prepared by third- parties (e.g., online ordering, and restaurants). As a result, a system is required that lengthens the road to heat the meal, just as it did in the past.

Constructing a recipe from a food photograph seems to be a hard process. Because the components transform the cooking process. Generally, people can recognize the visible chemicals, and cannot always anticipate the substances that are completely dissolved in the meal (e.g., salt, pepper, flour, sugar).

The last few years have witnessed outstanding improvements in visual recognition tasks such as natural image classification, object detection and semantic segmentation. However, when comparing to natural image understanding, food recognition poses additional challenges, since food and its components have high intra-class variability and present heavy deformations that occur during the cooking process. Ingredients are frequently occluded in a cooked dish and come in a variety of colours, forms and textures. Further, visual ingredient detection requires high level reasoning and prior knowledge (e.g., cake will likely contain sugar and not salt, while croissant will presumably include butter). Hence, food recognition challenges current computer vision systems to go beyond the merely visible, and to incorporate prior knowledge to enable high-quality structured food preparation descriptions.

Previous efforts on food understanding have mainly focused on food and ingredient categorization. However, a system for comprehensive visual food recog-nition should not only be able to recognize the type of meal or its ingredients, but also understand its preparation process. Traditionally, the image-to-recipe problem has been formulated as a retrieval task, where a recipe is retrieved from a fixed dataset based on the image similar-ity score in an embedding space. The performance of such systems highly depends on the dataset size and diversity, as well as on the quality of the learned embedding. Not sur-prisingly, these systems fail when a matching recipe for the image query does not exist in the static dataset.

An alternative to overcome the dataset constraints of retrieval systems is to formulate the image-to-recipe problem as a conditional generation one. Therefore, in this paper, we present a system that generates a cooking recipe containinga title, ingredients and cooking instructions directly from an image.

1.2 OBJECTIVES OF THE PROJECT

- To predict the title of the image
- To generate the list of the ingredients of the food
- To generate the recipe of the food by considering both the title and the ingredients
- User Interface through which the users can upload their food images

1.3 PROBLEM STATEMENT

Some several websites and applications provide assistance with the recipe and ingredients when the dish's title is searched. Numerous websites recommend a recipe based on the components that are given as input (e.g., dishes with leftover ingredients) and applications which classify the food using image recognition.

Previously, suggested image-to-instruction algorithms were constrained by the datasets on which they were tested. They generate titles, ingredients, and recipes for the images included in the dataset. Their precision is limited to that particular space. When these systems are presented with a picture that does not exist in the static dataset, they often fail.

CHAPTER-II

LITERATURE REVIEW

2.1. MULTI-LABEL CLASSIFICATION

The literature has made significant efforts to exploit deep neural networks in multi-label classification, including developing models and investigating loss functions that are well-suited for this job. One technique for extracting label dependencies is to use label powersets. Because powersets include all potential label combinations, they are unsuitable for solving large-scale issues. Another time-consuming alternative is to learn the labels' combined probability.

To address this problem, probabilistic classifier chains and associated recurrent neural network-based counterparts offer to split the joint distribution into conditions, therefore establishing intrinsic ordering. Take note that the majority of these models demand that a prediction be made for every one of the possible labels.

Additionally, joint embeddings of the input and label have been established to retain correlations and forecast label sets. The process of multilabel image classification. When multi-label classification goals are considered, binary logistic loss, target distribution cross-entropy, target distribution mean squared error, and ranking based losses have been examined and contrasted.

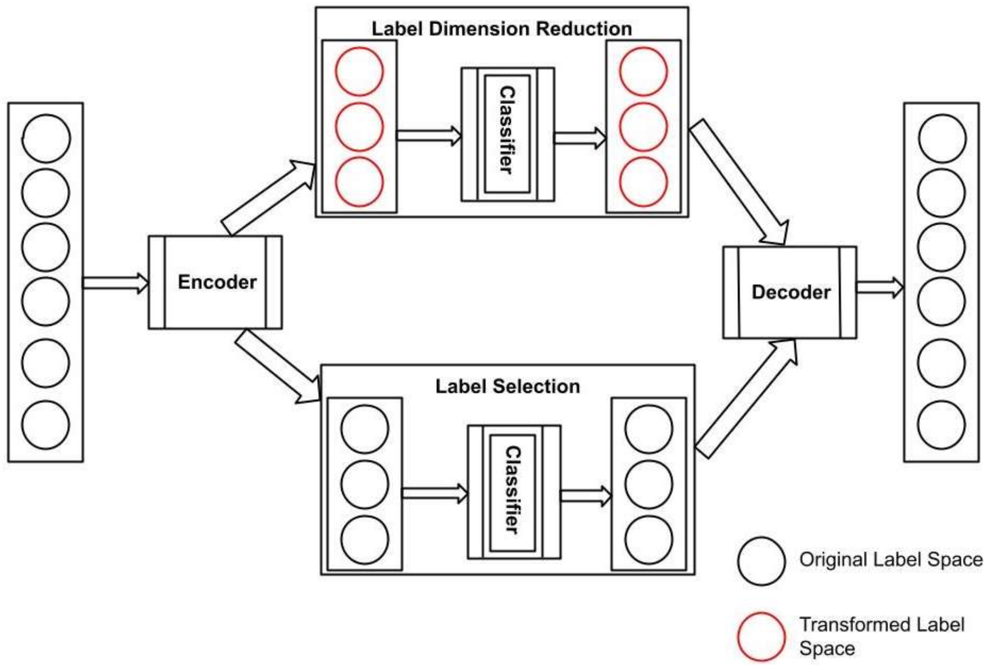


Figure 2.1: Multi-label Image Classification

2.2. CONDITIONAL TEXT GENERATION

Conditional text generation using auto-regressive algorithms has been extensively researched in the literature, with both text- and image-based conditionings being used. Different architectural designs have been researched in neural machine translation, where the objective is to anticipate the translation of a given source text into another language.

These designs include recurrent neural networks, convolutional models, and attention-based techniques. Recent work has extended sequence-to-sequence models to more fully accessible generation challenges, such as poems and narrative production. Following the trend in neural machine translation, auto- regressive models have demonstrated promising quality of image subtitles, where the main objective is to provide a brief description of the image's contents, opening the door to less restricted problems such as producing descriptive lines of text or story – telling.

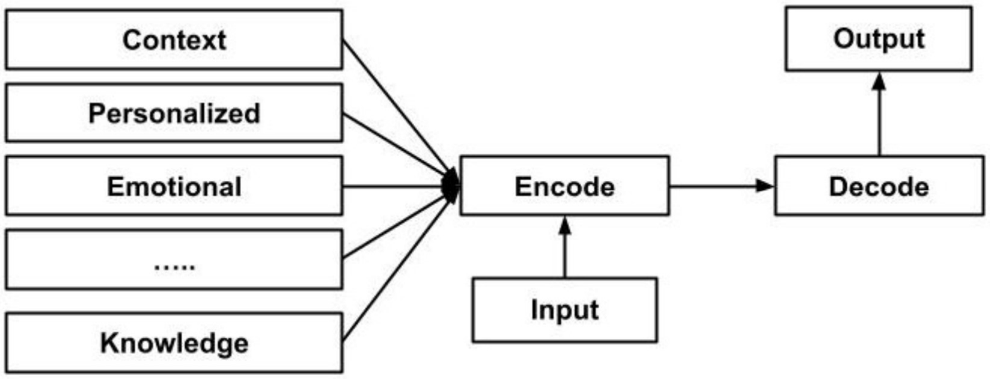


Figure 2.2: Conditional Text Generation

2.3. ENCODER-DECODER MODEL

The encoder-decoder model was presented in different papers. The difference between these two papers is on the basis of the relationship between input and output length. From a high-level, the model comprises two sub-models: an encoder and a decoder.

Encoder: The encoder is responsible for stepping through the input time steps and encoding the entire sequence into a fixed-length vector called a context vector.

Decoder: The decoder is responsible for stepping through the output time steps while reading from the context vector.

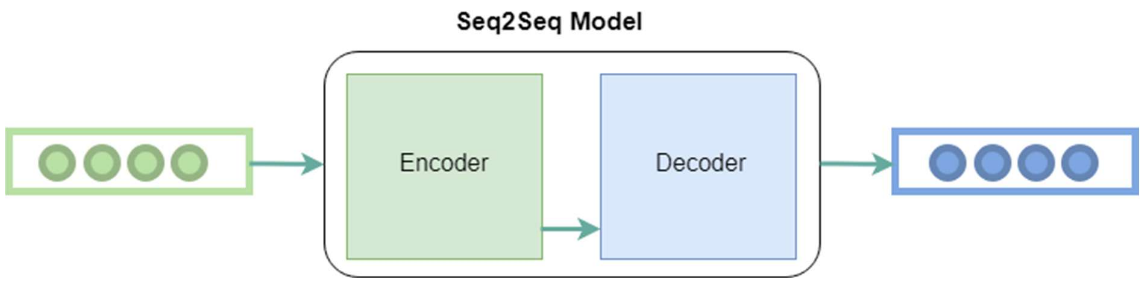


Figure 2.3: Encoder-Decoder Architecture

2.4. ATTENTION MECHANISM

Attention was proposed by authors of the Encoder-Decoder network as an extension of it. It is proposed to overcome the limitation of the Encoder-Decoder model encoding the input sequence to one fixed-length vector from which to decode each output time step. This issue is believed to be more of a problem when decoding long sequences.

2.5. SELF-ATTENTION MECHANISM

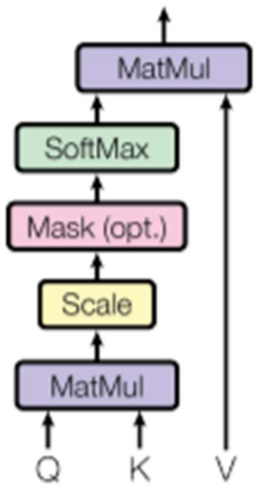


Figure 2.4: Self-Attention

The attention mechanism allows output to focus attention on input while producing output while the self-attention model allows inputs to interact with each other (i.e., calculate attention of all other inputs with respect to one input.

- The first step is multiplying each of the encoder input vectors with three weights matrices (W(Q), W(K), W(V)) that we trained during the training process. This matrix multiplication will give us three vectors for each of the input vector: the key vector, the query vector, and the value vector.

- The second step in calculating self-attention is to multiply the Query vector of the current input with the key vectors from other inputs.
- In the third step, we will divide the score by square root of dimensions of the key vector (dk). In the paper the dimension of the key vector is 64, so that will be 8. The reason behind that is if the dot products become large, this causes some self-attention scores to be very small after we apply softmax function in the future.
- In the fourth step, we will apply the softmax function on all self-attention scores we calculated with respect to the query word (here first word).
- In the fifth step, we multiply the value vector on the vector we calculated in the previous step.
- In the final step, we sum up the weighted value vectors that we got in the previous step, this will give us the self-attention output for the given word.

The above procedure is applied to all the input sequences. Mathematically, the self-attention matrix for input matrices (Q, K, V) is calculated as:

$$Attention\ (Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right) V$$

where Q, K, V are the concatenation of query, key, and value vectors.

2.6. MULTI-HEADED ATTENTION

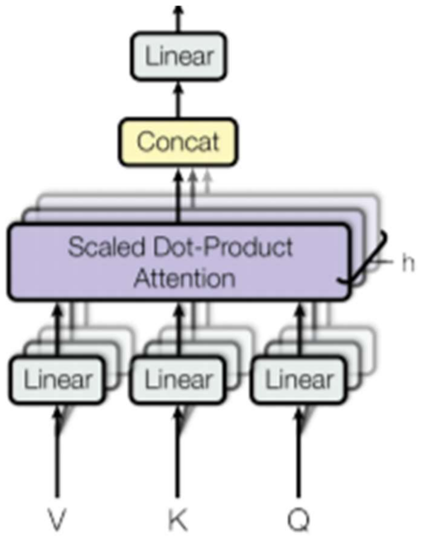


Figure 2.5: Multi-Headed Attention

In the attention paper, the authors proposed another type of attention mechanism called multi-headed attention. Below is the step-by-step process to calculate multi-headed self-attention:

- Take each word of input sentence and generate the embedding from it.
- In this mechanism, we created h (h = 8) different attention heads, each head has different weight matrices (W(Q), W(K), W(V)).
- In this step, we multiply the input matrix with each of the weight matrices (WQ, WK, WV) to produce the key, value, and query matrices for each attention head.
- Now, we apply the attention mechanism to these query, key, and value matrices, this gives us an output matrix from each attention head.
- In this step, we concatenate the output matrix obtained from each attention heads and dot product with the weight WO to generate the output of the multi-headed attention layer.

Mathematically multi-head attention can be represented by:

$$MultiHead\ (Q, K, V) = concat\ (head_1 head_2 head_3 \dots head_n)\ W_o$$

where,

$$head_i = Attention\ (QW_i^Q,\ KW_i^K,\ VW_i^V)$$

2.7. ATTENTION IN TRANSFORMER ARCHITECTURE

The transformer architecture uses attention model uses multi-headed attention at three steps:

- The first is encoder-decoder attention layers, in this type of layer the queries come from the previous decoder layer while the keys and values come from the encoder output. This allows each position in the decoder to give attention to all the positions of the input sequence.
- The second type is the self-attention layer contained in the encoder, this layer receives key, value, and query input from the output of the previous encoder layer. Each position in the encoder can get attention score from every position in the previous encoder layer.
- The third type is the self-attention in the decoder, this is similar to self-attention in encoder where all queries, keys, and values come from the previous layer. The self-attention decoder allows each position to attend each position up to and including that position. The future values are masked with (-Inf). This is known as masked-self attention.

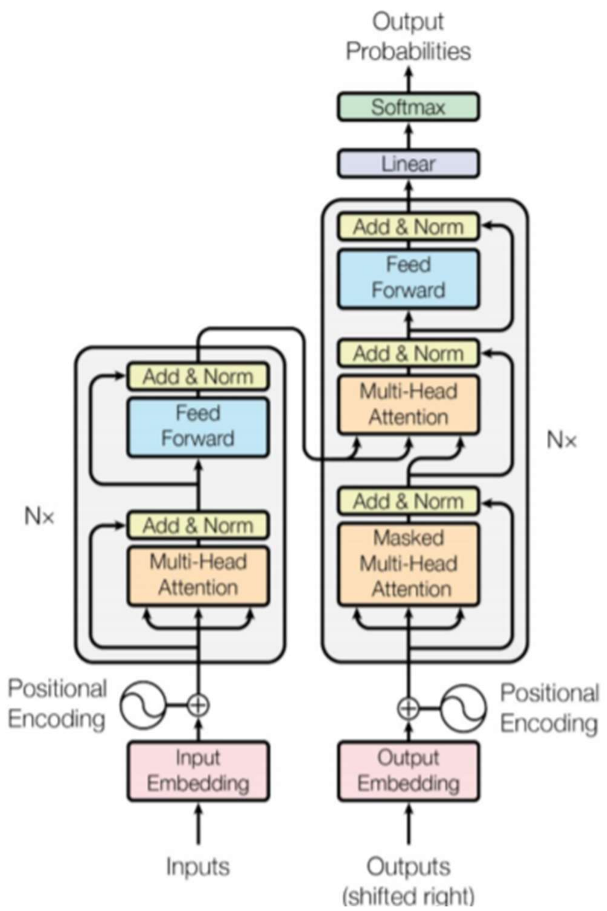


Figure 2.6: Attention in Transformer Architecture

CHAPTER-III
PROPOSED SYSTEM

3.1 PROPOSED SYSTEM

The proposed system in this research resolves the issue of using a static dataset to generate recipes for food images. This image-to-instruction generating method forecasts the dish's title, ingredients, and cooking directions. It predicts the ingredients from the picture and then generates instructions using both the image and the predicted ingredients. By examining the question does the order of the ingredients important, the anticipated ingredient is viewed as both a set and a list. Additionally, this paper demonstrates using a limited collection of photos that food image-to-ingredient prediction is a difficult problem for humans and that our technique is capable of outperforming them. This research contribution can be summarized as follows:

- It introduces an inverse cookery system that creates cooking instructions based on a picture and its ingredients, while experimenting with various attention methods for reasoning about both modalities concurrently.
- It examines ingredients comprehensively as well a list and a set, and proposes a prototype system for ingredient prediction that capitalizes on ingredient co-dependence without enforcing order.

3.2 METHODOLOGY

The system accepts a food picture as input and generates a recipe complete with title, ingredients, and cooking directions. Our technique begins by pretraining an image encoder and an ingredients decoder. The image encoder predicts a set of ingredients using visual characteristics taken from the input picture and ingredients. After that, train the ingredient encoder and decoder, which create title and instructions by putting the image's visual attributes and expected ingredients into a state-of-the-art sequence creation model.

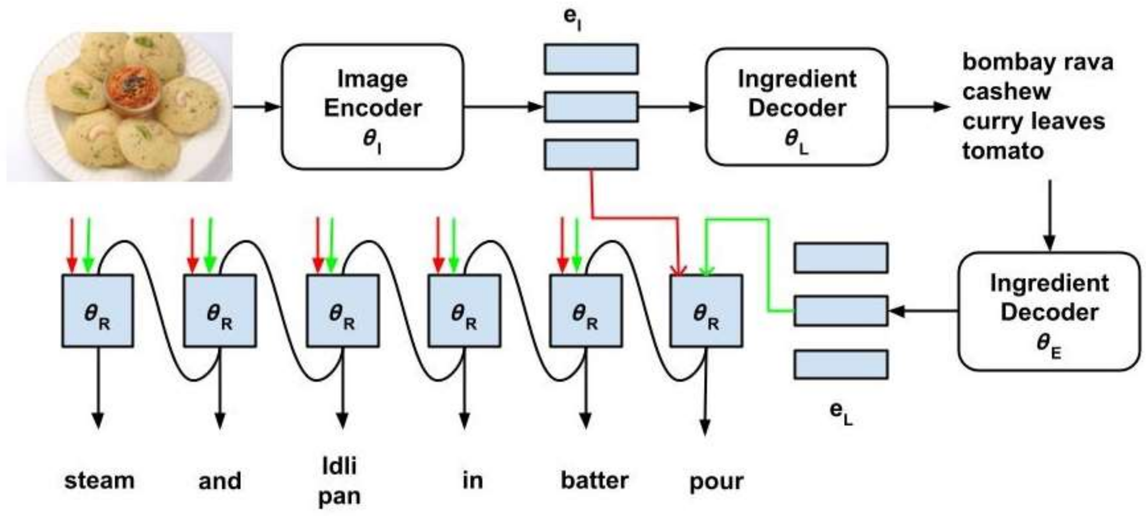


Figure 3.1: Methodology

The model considers a clip random 224 × 224 pixels for training and centre 224 x 224 pixels for assessment after resizing pictures to 256 pixels on their shortest side. Then employ a transformer with 16 blocks and 8 multi-head attentions, each with 64 dimensions, for the instruction decoder. Later, employ a transformer with four blocks and two multi-head attentions, each with 256 dimensions, for the ingredient decoder. The final convolutional layer of the ResNet-50 [24] model is used to produce picture embeddings. The dimensions of both the picture and the ingredients embeddings are 512.

The proposed model limits each dish to a maximum of 20 ingredients and maintain instructions to a total of 150 words. The models are trained using Adam optimizer until early-stopping conditions are fulfilled (with a probability of 50 and validation loss monitoring). PyTorch4 is used to implement all of the models. The extra information contains more implementation details.

3.2.1 ALGORITHM

Step 1: Input the image from the user.

Step 2: Extract the image features using ResNet. (Image encoder)

Step 3: Extract the ingredient using feed forward model. (Ingredient encoder and decoder)

Step 4: Generate the recipe using attention models (using the image and the predicted ingredients)

Step 5: Display the title, ingredients and the instructions.

3.3. IMPLEMENTATION

3.3.1 INGREDIENTS PREDICTION

As noted, before, the components prediction was accomplished in two methods. The first one stores the anticipated ingredients as a set, ensuring that no ingredient is duplicated in the final recipe formulation. The other is a list representation, which emphasizes the sequence of the elements.

3.3.2 LIST REPRESENTATION

1. Define a dictionary of N-dimensional ingredients as D

$$D = \{d_i\}_{i=0}^{N}$$

2. Create an ingredient list L by picking K components from D

$$L = [l_i]_{i=0}^{K}$$

3. Encrypt L as binary matrix of size K x N, with

$$L_{i,j} = \begin{cases} 1, & if\ d_j\ \in D \\ 0, & otherwise \end{cases}$$

4. Training process has M pictures, then the ingredient list pairing will be

$$\{(x^{(i)}, L^{(i)})\}_{i=0}^{M}$$

5. By optimizing the goal, predict $\hat{L}$ from a picture x:

$$argmax_{\theta_I,\theta_L} \sum_{i=0}^{M} log\ p(\hat{L}^{(i)} = L^{(i)}\ |x^{(i)};\ \theta_I, \theta_L)$$

here,

θ_I, θ_L denotes parameters that may be learned

6. Because L is a list, so divide $p(\hat{L}^{(i)} = L^{(i)}\ |\ x^{(i)})$ into K conditionals

$$\sum_{k=0}^{K} \log p(\hat{L}_k^{(i)} = L_k^{(i)} | x^{(i)}, \quad L_{<k}^{(i)})^3$$

3.3.3 SET REPRESENTATION

1. Create an ingredient list L by picking K components from S

$$S = [s_i]_{i=0}^{K}$$

2. Encrypt S as binary vector of size N, with

$$S_i = \begin{cases} 1, & if\ s_j\ \in S \\ 0, & otherwise \end{cases}$$

3. Training process has M pictures, then the ingredient set pairing will be

$$\{(x^{(i)}, S^{(i)})\}_{i=0}^{M}$$

4. By optimizing the goal, predict $\hat{L}$ from a picture x:

$$argmax_{\theta_I,\theta_L} \sum_{i=0}^{M} log\ p(\hat{S}^{(i)} = S^{(i)} | x^{(i)};\ \theta_I, \theta_L)$$

here,

θ_I, θ_L denotes parameters that may be learned

5. Considering that all components are independent, then quantize $p(\hat{S}^{(i)} = S^{(i)} \mid x^{(i)})$

$$\sum_{j=0}^{N} \log p\left(\hat{S}_j^{(i)} = S_j^{(i)} \middle| x^{(i)}\right)$$

Until the conclusion of the sequence (eos), the transformer anticipates the components in a list-like way. As previously stated, the disadvantage of this strategy is that it penalizes for order. To eliminate the order in which ingredients are anticipated, it is suggested that the results be pooled over many time steps via a max pooling operation.

Additionally, to guarantee that no component $\hat{L}^{(i)}$ is picked twice, it needs the pre-activation of $p\left(\hat{L}_k^{(i)} \middle| x^{(i)},\ L_k^{(i)}\right)$ to be -∞ all previously selected ingredients at time steps < k.

By reducing the binary cross-entropy between both the expected ingredients (after pooling) and the actual truth, train this model. By include the *eos* in the pooling procedure, the location of the token would be lost. As a result, include an extra loss accounting for it in order to learn the stopping criterion for the ingredient prediction. The *eos* loss is defined as the binary cross-entropy difference between the anticipated *eos* probability at all time stages and the ground truth probability at all time phases. Additionally, it integrates a cardinality l_i penalty that has found to be effective experimentally. Then sampled straight out from transformer's result during inference. This kind is referred to as a set transformer.

Alternatively, there is a chance to use the target distribution $p\left(s^{(i)} \middle| x^{(i)} = \frac{s^{(i)}}{\sum_j s_j^{(i)}}\right)$ to model the joint probability distribution of set elements and educate a system by minimising the cross-entropy loss between $p(s^{(i)}|x^{(i)})$ and the model's output distribution $p(\hat{s}^{(i)}|x^{(i)})$. Nonetheless, how to transform the target distribution back to the matching set of elements with adjustable cardinality remains unknown. Create a feed forward network in this situation and train it on the target distribution's cross-entropy loss. To retrieve the ingredient set, it is suggested to sample elements greedily from a cumulative distribution of ordered output probabilities $p\left(\hat{s}^{(i)}\middle|x^{(i)}\right)$ and to cease sampling when the sum of chosen elements' probabilities exceeds a threshold. This model is referred to as feed forward (target distribution).

3.3.4 RECIPE GENERATION

Using an instruction transformer, the model seeks to generate a series of instructions [3, 4] R = (r_1, r_2, …., r_T) (where r_T signifies a word in the sequence). Take note that the title is the first instruction [18, 19]. This transformer operates on two inputs simultaneously: the image representation e_I and the ingredient embedding e_L. The model extracts the picture representation using a ResNet-50 [25] encoder and acquire the ingredient embedding e_L using a decoder architecture for ingredient prediction, accompanied by a mono hidden layers assigning each ingredient to a fixed-size vector.

Each transformer block in the instruction decoder has two attention levels followed by a linear layer. The first layer of attention focuses on previously created outcomes, while the second layer focuses on model conditioning in order to improve the self-attention output. The transformer model is constructed using numerous transformer blocks, a linear layer, and a softmax nonlinearity that delivers a distribution across recipe words for every time step t.

The graphic depicts the transformer model, which is often restricted to a single modality. Our recipe generator, on the other hand, is constrained by two sources: picture features $e_I \in R^{P\text{x}\, d_e}$ and ingredient embeddings (where K is the amount of image and ingredient features, respectively, and is the embedding dimensionality). Thus, in the proposed system wants the attention to be able to reason concurrently about both modalities, leading the instruction generation process. To that purpose, this research has examined three distinct ways which are explained in the following.

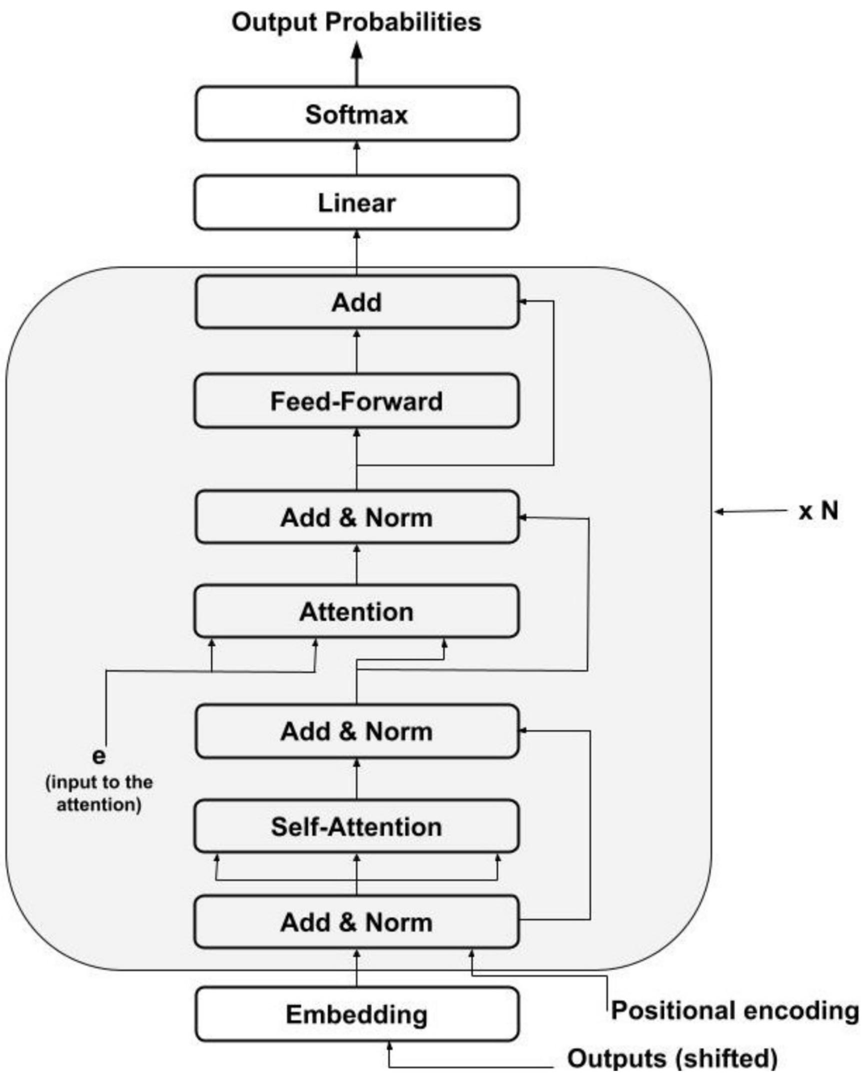


Figure 3.2: Transformer Model

3.3.4.1 CONCATENATION ATTENTION

This method combines the image e_I and the components that are predicted e_L in the very step of the dimension $e_{concat} \in R^{(K+P) \text{x}\, d_e}$. Then, these embeddings are applied to the attention.

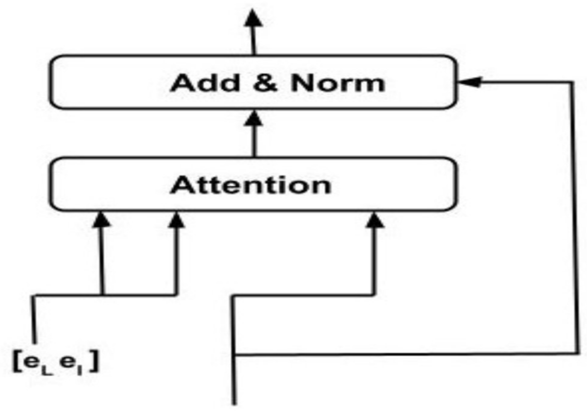


Figure 3.3: Concatenated Attention

3.3.4.2 INDEPENDENT ATTENTION

This method employs two levels of attention to cope with bimodal conditions. One layer is responsible for the picture embedding e_I, while the other is responsible for the ingredient embeddings e_L. Through a summing procedure, the result from both attention, layers are merged.

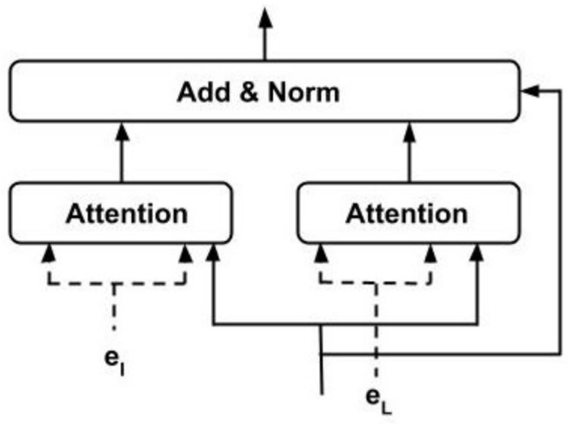


Figure 3.4: Independent Attention

3.3.4.3 SEQUENTIAL ATTENTION

This method alternates between the two conditioned modes progressively. We investigate two groupings in this design: (1) image first, where attention is calculated first over image embeddings e_I and then over ingredient embeddings e_L; and (2) ingredients first, in which the order is reversed and attention is computed first over ingredient embeddings e_L and then over image embeddings e_I.

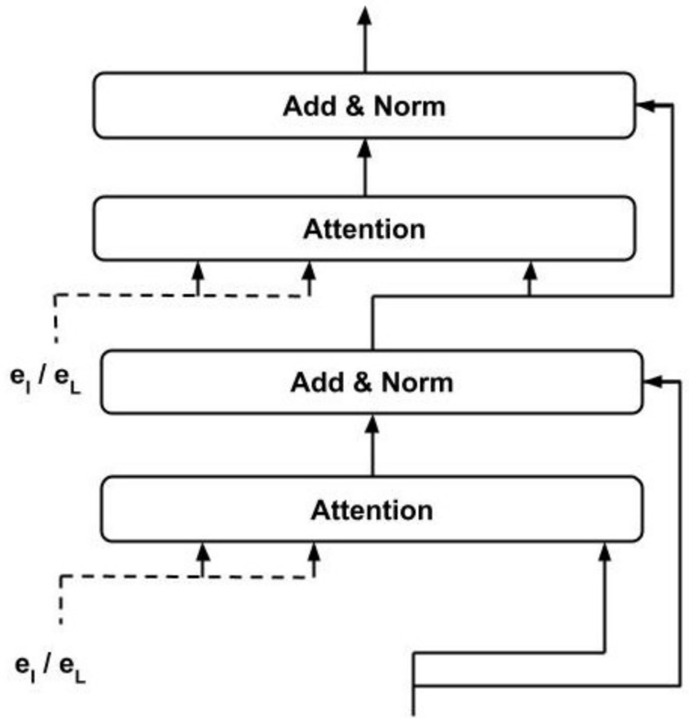


Figure 3.5: Sequential Attention

Independent focus produces the lowest outcomes, accompanied by both sequential and concurrent attention. Whereas the former is capable of successively refining the outcome using ingredient or visual information, independent attention could only do so in a single step. This also applies to concatenated attention, which generates the finest results. However, concatenated attention is adaptable enough to prioritize one modality over the other, while independent attention is compelled and included information from both modalities. As a consequence, this research uses the concatenated attention model to report on the test set's outcomes.

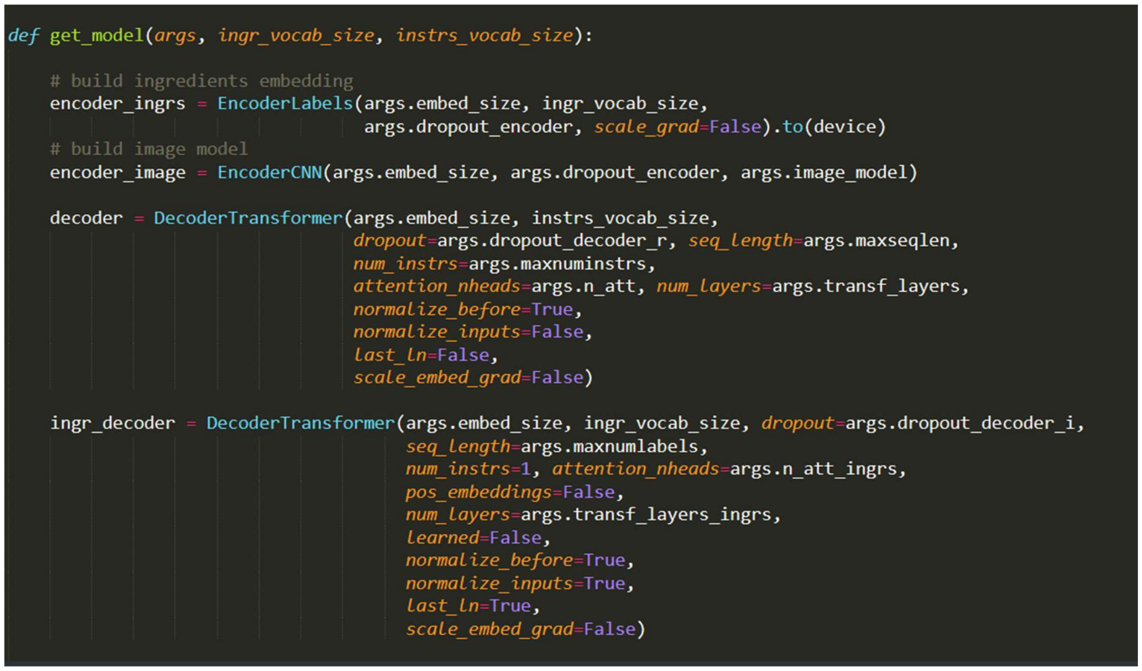
3.3.5. OPTIMIZATION

We train our recipe transformer in two stages. In the first stage, we pre-train the image encoder and ingredients de-coder. Then, in the second stage, we train the ingredient encoder and instruction de-coder by minimizing the neg-ative log-likelihood and adjusting θ_R and θ_E. Note that, while training, the instruction decoder takes as input the ground truth ingredients. All transformer models are trained with teacher forcing except for the set transformer.

3.3.6. CODE SNIPPETS

The following is the code snippet of the model for extracting image features, ingredient prediction.

File: model.py

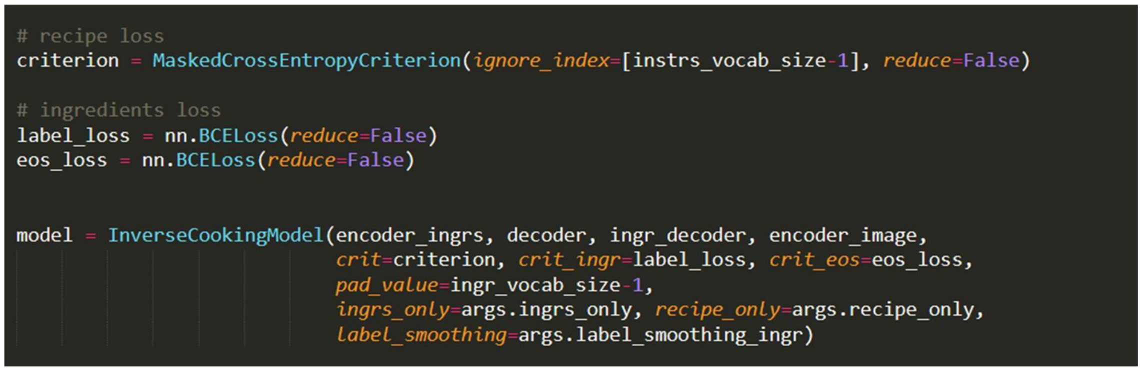


```python
def get_model(args, ingr_vocab_size, instrs_vocab_size):

    # build ingredients embedding
    encoder_ingrs = EncoderLabels(args.embed_size, ingr_vocab_size,
                                  args.dropout_encoder, scale_grad=False).to(device)
    # build image model
    encoder_image = EncoderCNN(args.embed_size, args.dropout_encoder, args.image_model)

    decoder = DecoderTransformer(args.embed_size, instrs_vocab_size,
                                 dropout=args.dropout_decoder_r, seq_length=args.maxseqlen,
                                 num_instrs=args.maxnuminstrs,
                                 attention_nheads=args.n_att, num_layers=args.transf_layers,
                                 normalize_before=True,
                                 normalize_inputs=False,
                                 last_ln=False,
                                 scale_embed_grad=False)

    ingr_decoder = DecoderTransformer(args.embed_size, ingr_vocab_size, dropout=args.dropout_decoder_i,
                                      seq_length=args.maxnumlabels,
                                      num_instrs=1, attention_nheads=args.n_att_ingrs,
                                      pos_embeddings=False,
                                      num_layers=args.transf_layers_ingrs,
                                      learned=False,
                                      normalize_before=True,
                                      normalize_inputs=True,
                                      last_ln=True,
                                      scale_embed_grad=False)
```

Figure 3.6: Code Snippet for title, ingredient prediction

The following is the code snippet for reducing the loss of predicting both the ingredients and the recipe.

File: model.py



```python
# recipe loss
criterion = MaskedCrossEntropyCriterion(ignore_index=[instrs_vocab_size-1], reduce=False)

# ingredients loss
label_loss = nn.BCELoss(reduce=False)
eos_loss = nn.BCELoss(reduce=False)


model = InverseCookingModel(encoder_ingrs, decoder, ingr_decoder, encoder_image,
                            crit=criterion, crit_ingr=label_loss, crit_eos=eos_loss,
                            pad_value=ingr_vocab_size-1,
                            ingrs_only=args.ingrs_only, recipe_only=args.recipe_only,
                            label_smoothing=args.label_smoothing_ingr)
```

Figure 3.7: Code snippet for reducing the prediction loss

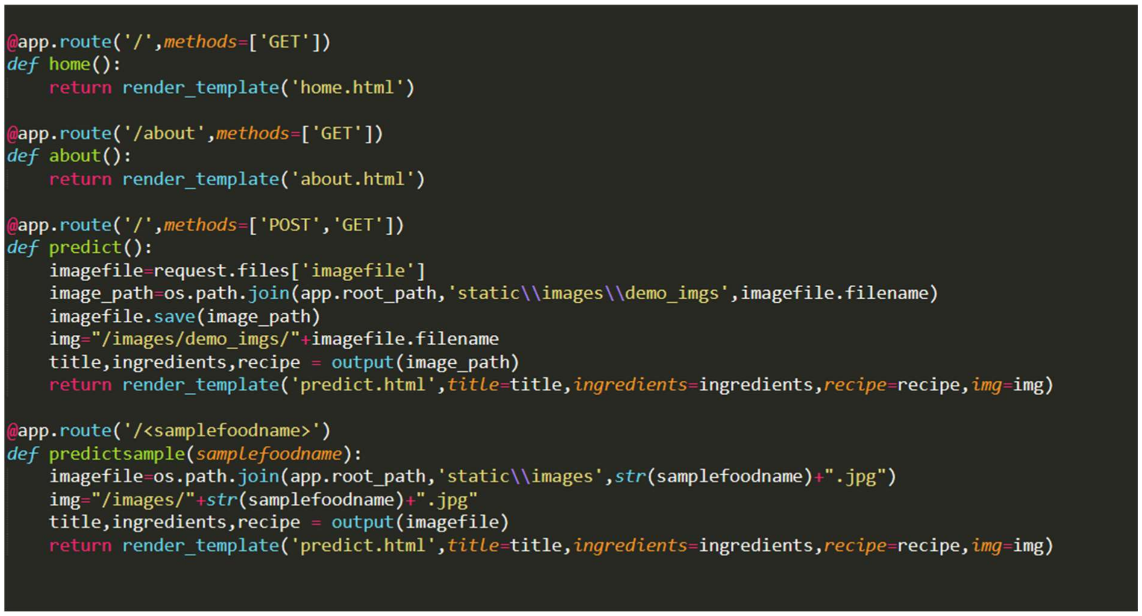
3.3.6. DEPLOYMENT

The model is integrated to a web application which helps in user-interface. The web application is build using Flask application.

3.3.6.1 FLASK

Flask is a web framework that provides libraries to build lightweight web applications in python. It is developed by Armin Ronacher who leads an international group of python enthusiasts (POCCO). It is based on WSGI toolkit and jinja2 template engine. Flask is considered as a micro framework. WSGI is an acronym for web server gateway interface which is a standard for python web application development. It is considered as the specification for the universal interface between the web server and web application. Jinja2 is a web template engine which combines a template with a certain data source to render the dynamic web pages.

The following code snippet represents the routes of the web application using flask.



```
@app.route('/',methods=['GET'])
def home():
    return render_template('home.html')

@app.route('/about',methods=['GET'])
def about():
    return render_template('about.html')

@app.route('/',methods=['POST','GET'])
def predict():
    imagefile=request.files['imagefile']
    image_path=os.path.join(app.root_path,'static\\images\\demo_imgs',imagefile.filename)
    imagefile.save(image_path)
    img="/images/demo_imgs/"+imagefile.filename
    title,ingredients,recipe = output(image_path)
    return render_template('predict.html',title=title,ingredients=ingredients,recipe=recipe,img=img)

@app.route('/<samplefoodname>')
def predictsample(samplefoodname):
    imagefile=os.path.join(app.root_path,'static\\images',str(samplefoodname)+".jpg")
    img="/images/"+str(samplefoodname)+".jpg"
    title,ingredients,recipe = output(imagefile)
    return render_template('predict.html',title=title,ingredients=ingredients,recipe=recipe,img=img)
```

Figure 3.8: Routes of web application using flask

3.3.7. EXECUTION

The code uses Python 3.6 version for execution. Make sure the system has python installed in it.

1. Install the required packages using the following commands

    a. Install PyTorch

        $ conda install pytorch=0.4.1 cuda90 -c pytorch

b. Install the dependencies

```
$ pip install -r requirements.txt
```

2. Download all the files (code) and run the code using the following command.

```
$ python run.py
```

3. It will display a localhost link and open that link in the browser.
4. The user can select default food images to see the output.
5. Or else, the user can choose random food image to see the output.
   (If the image is downloaded from the internet, place the image in the following folder and choose the second way to predict the recipe.
   Folder location:
   C:/Users/…/Self-Attention Architecture for Ingredients Generation from Food Images/Foodimg2Ing/static/images/demo_imgs

3.4 DATA PREPARATION

We train and evaluate our models on the RecipelM dataset, composed of 1,029,720 recipes scraped from cooking websites. The dataset contains 720,639 training, 155,036 validation and 154,045 test recipes, containing a title, a list of ingredients, a list of cooking instructions and (optionally) an image. In our experiments, we use only the recipes containing images, and remove recipes with less than 2 ingredients or 2 instructions, resulting in 252,547 training, 54,255 validation and 54,506 test samples.

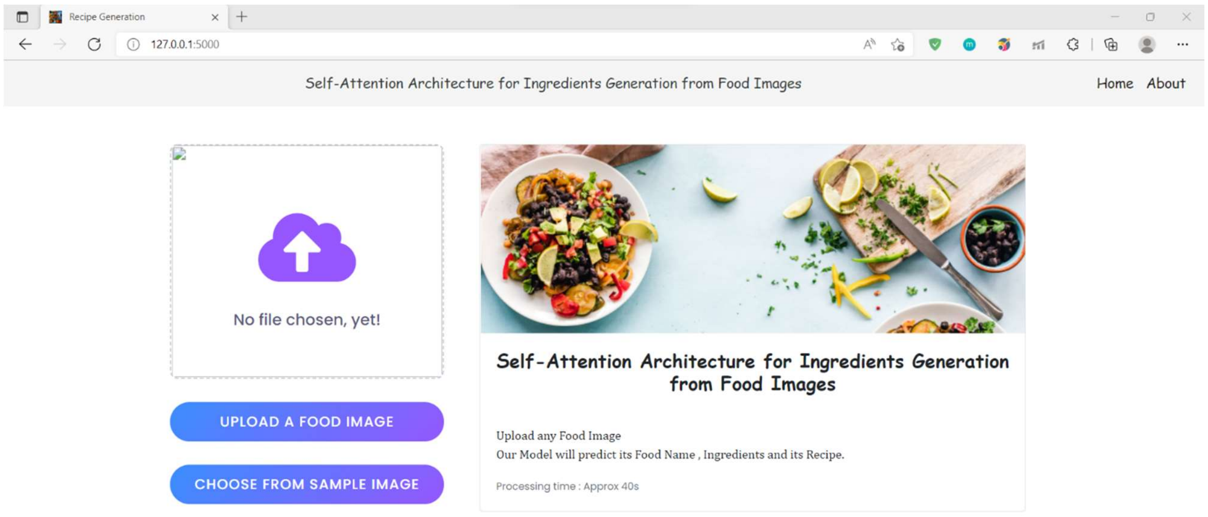
Since the dataset was obtained by scraping cooking web- sites, the resulting recipes are highly unstructured and contain frequently redundant or very narrowly defined cooking ingredients (e.g., *olive oil, virgin olive oil* and *spanish olive oil* are separate ingredients). Moreover, the ingredient vocabulary contains more than 400 different types of *cheese,* and more than 300 types of *pepper.* As a result, the original dataset contains 16,823 unique ingredients, which we pre- process to reduce its size and complexity.

First, we merge ingredients if they share the first or last two words (e.g., *bacon cheddar cheese* is merged into *cheddar cheese);* then, we cluster the ingredients that have same word in the first or in the last position (e.g., *gorgonzola cheese* or *cheese blend* are clustered together into the *cheese* category). Finally, we remove plurals and discard ingredients that appear less than10 times in the dataset. Altogether, we reduce the ingredient vocabulary from over 16k to 1488 unique ingredients. For the cooking instructions, we tokenize the raw text and remove words that appear less than 10 times in the dataset, and replace them with unknown word token. Moreover, we add special tokens for the start and the end of the recipe as well as the end of instruction. This process results in a recipe vocabulary of 23,231 unique words.

CHAPTER-IV
RESULTS AND DISCUSSIONS

4.1 RESULTS

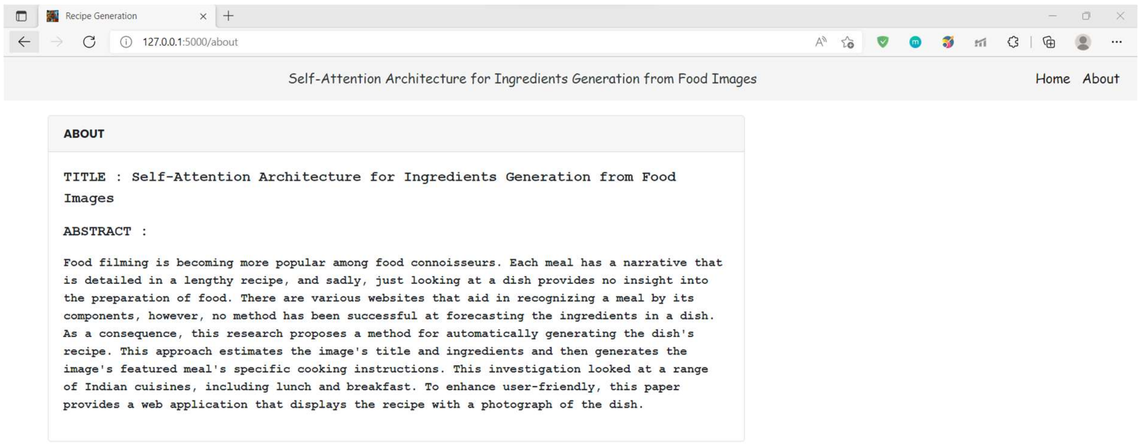
- When the generated URL is placed in the browser, it will lead to the below web application. This is the home page of the project.


Recipe Generation
127.0.0.1:5000
Self-Attention Architecture for Ingredients Generation from Food Images
Home
About
No file chosen, yet!
UPLOAD A FOOD IMAGE
CHOOSE FROM SAMPLE IMAGE
Self-Attention Architecture for Ingredients Generation from Food Images

Upload any Food Image
Our Model will predict its Food Name , Ingredients and its Recipe.
Processing time : Approx 40s

Figure 4.1: Home Screen of the Web application

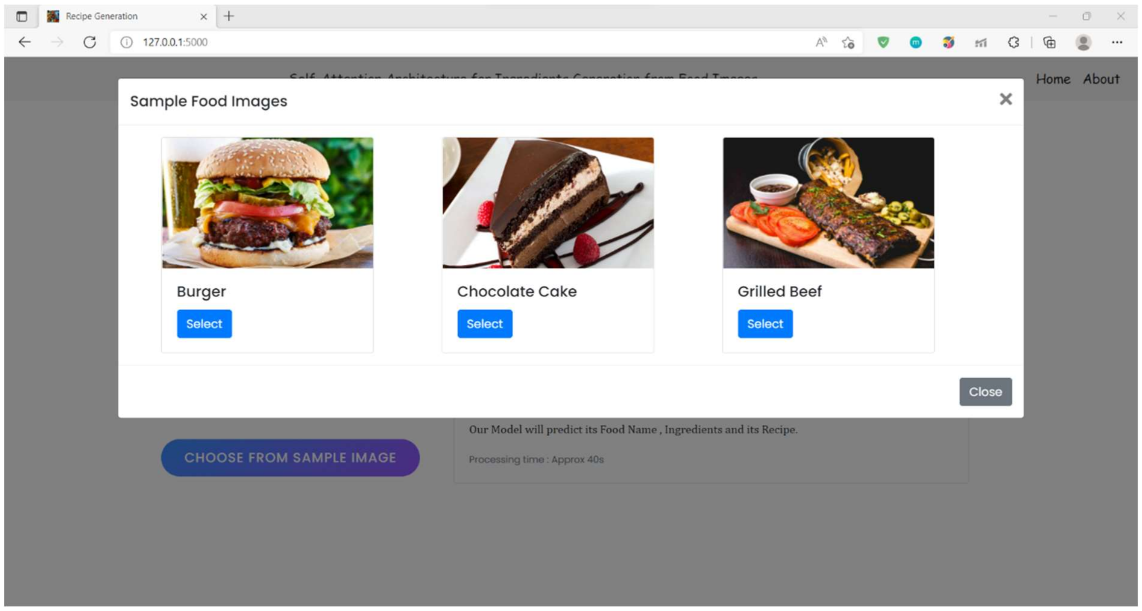
- By clicking on *"About"* in the home page, leads to about page of the application


Recipe Generation
127.0.0.1:5000/about
Self-Attention Architecture for Ingredients Generation from Food Images
Home
About
ABOUT

```
TITLE : Self-Attention Architecture for Ingredients Generation from Food
Images

ABSTRACT :

Food filming is becoming more popular among food connoisseurs. Each meal has a narrative that
is detailed in a lengthy recipe, and sadly, just looking at a dish provides no insight into
the preparation of food. There are various websites that aid in recognizing a meal by its
components, however, no method has been successful at forecasting the ingredients in a dish.
As a consequence, this research proposes a method for automatically generating the dish's
recipe. This approach estimates the image's title and ingredients and then generates the
image's featured meal's specific cooking instructions. This investigation looked at a range
of Indian cuisines, including lunch and breakfast. To enhance user-friendly, this paper
provides a web application that displays the recipe with a photograph of the dish.
```

Figure 4.2: About Screen of the Web Application

- The user has two ways to view the output of a food image
- Option 1: The user can choose a default food image by clicking on "*Choose from sample image*" button.



Our Model will predict its Food Name , Ingredients and its Recipe.

Processing time : Approx 40s

Figure 4.3: Option 1 (Choose the food image from the default ones)

- Option 2: The user can pick a random image either from the folder or download it from the internet.
- The image that is downloaded from the internet must be placed in a default location to view the output.

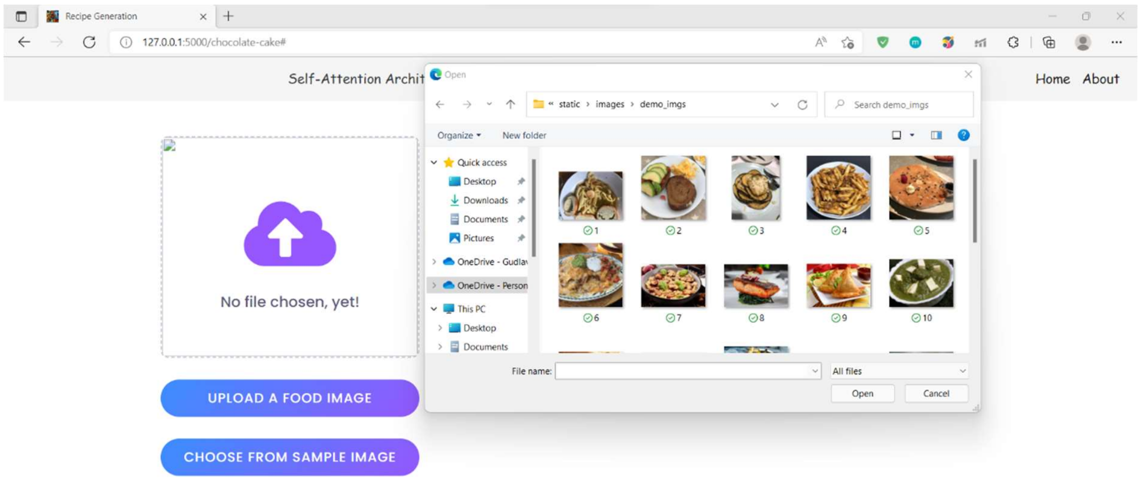


Figure 4.4: Option 2 (Choose any random food image)

- After choosing the image, the model starts to predict the results
- It will take approximately 40 seconds for the model to display the title, ingredients and the recipe of the image.

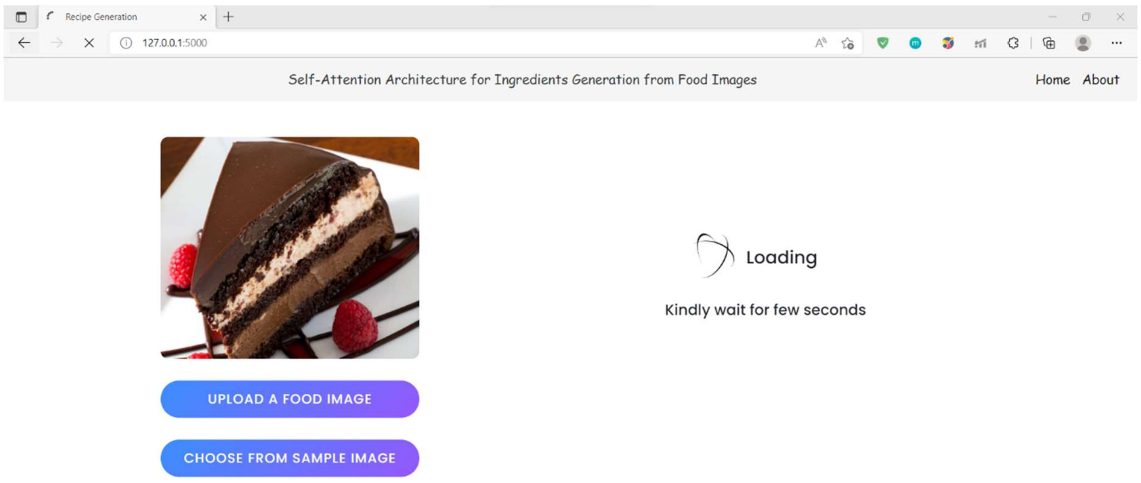
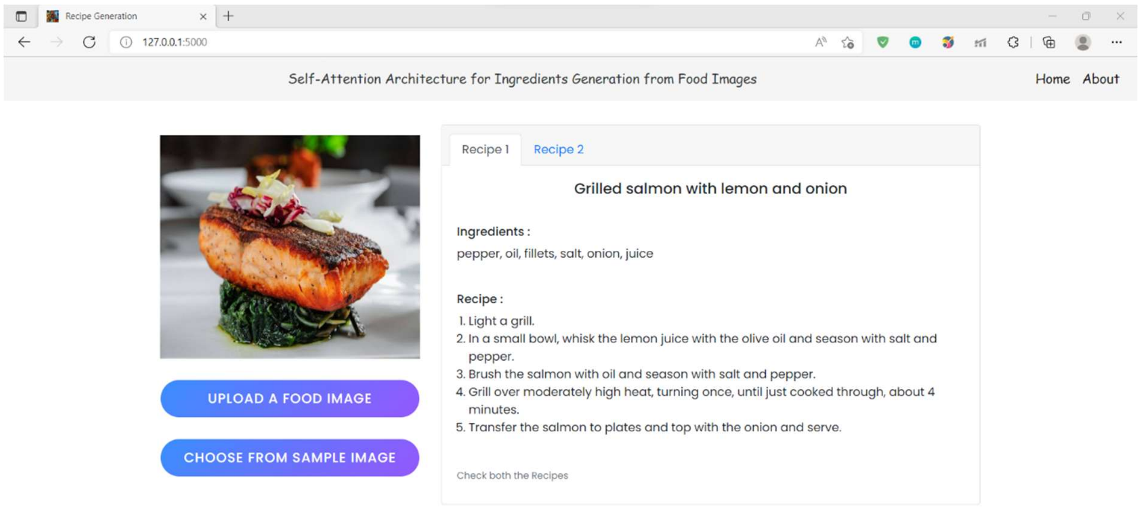

Recipe Generation
127.0.0.1:5000
Self-Attention Architecture for Ingredients Generation from Food Images
Home
About
Loading
Kindly wait for few seconds
UPLOAD A FOOD IMAGE
CHOOSE FROM SAMPLE IMAGE

Figure 4.5: Model predicting the results

- The models displays two recipes for the given food image



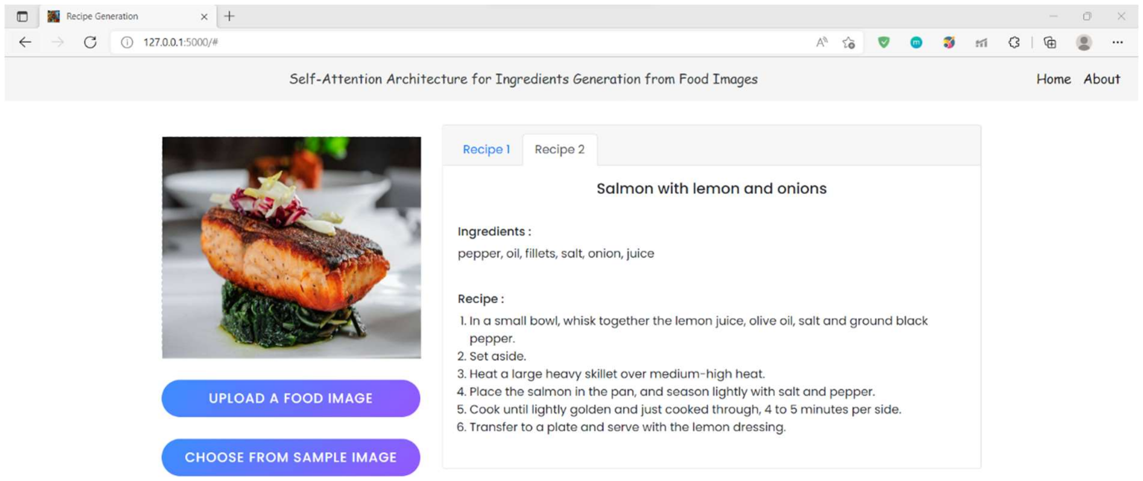
Grilled salmon with lemon and onion

Ingredients :
pepper, oil, fillets, salt, onion, juice

Recipe :
1. Light a grill.
2. In a small bowl, whisk the lemon juice with the olive oil and season with salt and pepper.
3. Brush the salmon with oil and season with salt and pepper.
4. Grill over moderately high heat, turning once, until just cooked through, about 4 minutes.
5. Transfer the salmon to plates and top with the onion and serve.

Check both the Recipes

Figure 4.6: Generated Recipe 1



Salmon with lemon and onions

Ingredients :

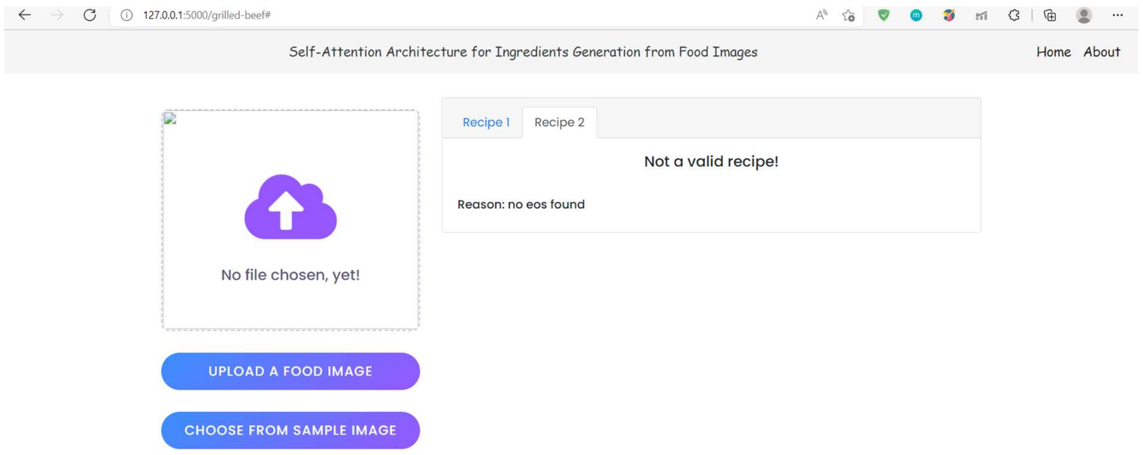
pepper, oil, fillets, salt, onion, juice

Recipe :

1. In a small bowl, whisk together the lemon juice, olive oil, salt and ground black pepper.
2. Set aside.
3. Heat a large heavy skillet over medium-high heat.
4. Place the salmon in the pan, and season lightly with salt and pepper.
5. Cook until lightly golden and just cooked through, 4 to 5 minutes per side.
6. Transfer to a plate and serve with the lemon dressing.

Figure 4.7: Generated Recipe 2

- If the predicted recipe 2 is similar to recipe 1 then the model will display the recipe 2 as following


127.0.0.1:5000/grilled-beef#
Self-Attention Architecture for Ingredients Generation from Food Images
Home
About
No file chosen, yet!
Recipe 1
Recipe 2
UPLOAD A FOOD IMAGE
CHOOSE FROM SAMPLE IMAGE

Not a valid recipe!

Reason: no eos found

Figure 4.8: Screenshot for "Not a valid Recipe"

4.2. DISCUSSIONS

4.2.1. MODEL EVALUATION

We assess models in this research using the Intersection over Union (IOU) and F1 scores, which are calculated for the aggregated values of True Positives (TP), False Negatives (FN), and False Positives (FP).

4.2.2.1 F1-SCORE

The F1-score is a metric that indicates the performance of the models on a given dataset. It is a method for combining the model's precision and recall, and is calculated as the harmonic mean of the model's precision and recall.

$$F1\ score = 2 \text{ x } \frac{precision \text{ x } recall}{precision + recall}$$

here,

$$precision = \frac{TP}{(TP+FP)}$$

$$recall = \frac{TP}{(TP+FN)}$$

From the above terms:

True Positives: Outcome in which the model forecasts the positive class properly.

False Positives: Outcome in which the model forecasts the positive class incorrectly.

False Negatives: Outcome in which the model forecasts the negative class incorrectly.

4.2.2.2 INTERSECTION OVER UNION

Intersection over Union is a performance measure that is used to determine an object detector's accuracy on a given dataset. To use Intersection over Union to analyse an object detector, it required bounding boxes for the ground truth (i.e., the original labelled bounding boxes from the dataset which specify the location of the object in the image) and the bounding boxes predicted by our model.

$$IOU = \frac{area\ of\ overlap}{area\ of\ union}$$

here,

$$area\ of\ overlap = A \cap B$$

$$area\ of\ union = A \cup B$$

A indicates the region of predicted bounded box

B indicates the region of original bounded box

The following table 4.1 depicts the accuracy of recipe generation (using IOU, F1-score) and table 4.2 shows the accuracy of ingredient prediction (using precision and recall).

Table 4.1: Accuracy of Recipe Generation

Model	**IOU**	**F1**
Recipe Generation	0.76	0.89

Table 4.2: Accuracy of Ingredient Prediction

Model	**Precision**	**Recall**
Ingredient Prediction	0.82	0.85

The accuracy of proposed model in this paper is compared with different convolutional neural network (CNN) models used for object recognition. The table 4.3 depicts the comparison of the model accuracies.

Table 4.3: Model Accuracy Comparison

Model	**F1-score**	**Precision**	**Recall**
Inception V3	0.47	0.45	0.50
VGG	0.54	0.52	0.58
ResNet-50	0.66	0.65	0.68
Our Model	0.81	0.80	0.83

Compared to other model, the proposed model outputs the highest accuracy when the terms like F1- score, precision and recall were considered.

CHAPTER -V

CONCLUSION AND FUTURE SCOPE

5.1 CONCLUSION

This project offers an image-to-recipe generation system that takes a food picture and generates a recipe with a title, ingredients, and cooking instructions sequence. It first predicts ingredients from image, demonstrating the need of modelling dependencies. The model then investigates instruction creation based on visuals and inferred ingredients, emphasising the need of thinking simultaneously about both modalities. Finally, user research findings corroborate the task's complexity and establish the system's superiority over state-of-the-art image-to-recipe retrieval systems.

Additionally, a web application is incorporated with the model, which aids in increasing the level of interaction between the user and the model. The outputs include a title, a list of ingredients, and the cooking directions for the food.

5.2 FUTURE SCOPE

Compared to present day, there will be a huge increase in tagging the food on the social media. Instead of downloading the food image from them and predicting their title, ingredients and the recipe will be a long process. The way to improve this project is to integrate the model with a lens that scans the food image and predicts output.

BIBILOGRAPHY

[1] A. Salvador, M. Drozdzal, X. Giro-i-Nieto and A. Romero, "Inverse Cooking: Recipe Generation from Food Images," 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 10445-10454, doi: 10.1109/CVPR.2019.01070.

[2] Micael Carvalho, Remi Cadene, David Picard, Laure Soulier, Nicolas Thome, and Matthieu Cord. "Cross-modal retrieval in the cooking context: Leaming semantic text-image embeddings". In SIGIR, 2018.

[3] J. Fujita, M. Sato and H. Nobuhara, "Model for Cooking Recipe Generation using Reinforcement Learning," 2021 IEEE 37th International Conference on Data Engineering Workshops (ICDEW), 2021, pp. 1-4, doi: 10.1109/ICDEW53142.2021.00007.

[4] Y. Pan, Q. Xu and Y. Li, "Food Recipe Alternation and Generation with Natural Language Processing Techniques," 2020 IEEE 36th International Conference on Data Engineering Workshops (ICDEW), 2020, pp. 94-97, doi: 10.1109/ICDEW49219.2020.000-1.

[5] A. Reusch, A. Weber, M. Thiele and W. Lehner, "RecipeGM: A Hierarchical Recipe Generation Model," 2021 IEEE 37th International Conference on Data Engineering Workshops (ICDEW), 2021, pp. 24-29, doi:10.1109/ICDEW53142.2021.00012.

[6] W. A. d. Santos, J. R. Bezerra, L. F. Wanderley Góes and F. M. F. Ferreira, "Creative Culinary Recipe Generation Based on Statistical Language Models," in IEEE Access, vol. 8, pp. 146263-146283, 2020, doi: 10.1109/ACCESS.2020.3013436.

[7] H. Jabeen, J. Weinz and J. Lehmann, "AutoChef: Automated Generation of Cooking Recipes," 2020 IEEE Congress on Evolutionary Computation (CEC), 2020, pp. 1-7, doi: 10.1109/CEC48606.2020.9185605.

[8] N. Hnoohom and S. Yuenyong, "Thai fast food image classification using deep learning," 2018 International ECTI Northern Section Conference on Electrical, Electronics, Computer and Telecommunications Engineering (ECTI-NCON), 2018, pp. 116-119, doi: 10.1109/ECTI-NCON.2018.8378293.

[9] G. G. Lee, C. Huang, J. Chen, S. Chen and H. Chen, "AIFood: A Large-Scale Food Images Dataset for Ingredient Recognition," TENCON 2019 - 2019 IEEE Region 10 Conference (TENCON), 2019, pp. 802-805, doi: 10.1109/TENCON.2019.8929715.

[10] W. Xu, H. Sun, C. Deng and Y. Tan, "TextDream: Conditional Text Generation by Searching in the Semantic Space," 2018 IEEE Congress on Evolutionary Computation (CEC), 2018, pp. 1-6, doi: 10.1109/CEC.2018.8477776.

[11] K. Singla and S. Biswas, "Machine learning explanability method for the multi-label classification
model," 2021 IEEE 15th International Conference on Semantic Computing (ICSC), 2021, pp. 337-340, doi: 10.1109/ICSC50631.2021.00063.

[12] B. Akhand and V. Susheela Devi, "Multi label classification of discrete data," 2013 IEEE International Conference on Fuzzy Systems (FUZZIEEE), 2013, pp. 1-5, doi: 10.1109/FUZZIEEE. 2013.6622574.

[13] M. Huang and P. Zhao, "Image multi-label learning algorithm based on label correlation," 2021 IEEE International Conference on Consumer Electronics and Computer Engineering (ICCECE), 2021, pp. 606-609, doi: 10.1109/ICCECE51280.2021.9342484.

[14] Y. Li and Y. Wang, "A Multi-label Image Classification Algorithm Based on Attention Model," 2018 IEEE/ACIS 17th International Conference on Computer and Information Science (ICIS), 2018, pp. 728-731, doi: 10.1109/ICIS.2018.8466472.

[15] B. Wang, Y. Liu, W. Xiao, Z. Xiong and M. Zhang, "Positive and negative max pooling for image classification," 2013 IEEE International Conference on Consumer Electronics (ICCE), 2013, pp. 278-279, doi: 10.1109/ICCE.2013.6486894.

SESHADRI RAO GUDLAVALLERU ENGINEERING COLLEGE

(An Autonomous Institute with Permanent Affiliation to JNTUK, Kakinada)
Seshadri Rao Knowledge Village, Gudlavalleru

Department of Computer Science and Engineering

Program Outcomes (POs)

Engineering Graduates will be able to:

1. **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions., component, or software to meet the desired needs.
5. **Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

10. **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Program Specific Outcomes (PSOs)

PSO1: Design, develop, test and maintain reliable software systems and intelligent systems.

PSO2: Design and develop web sites, web apps and mobile apps.

PROJECT PROFORMA

Classification of Project	Application	Product	Research	Review
	√			

Note: Tick Appropriate category

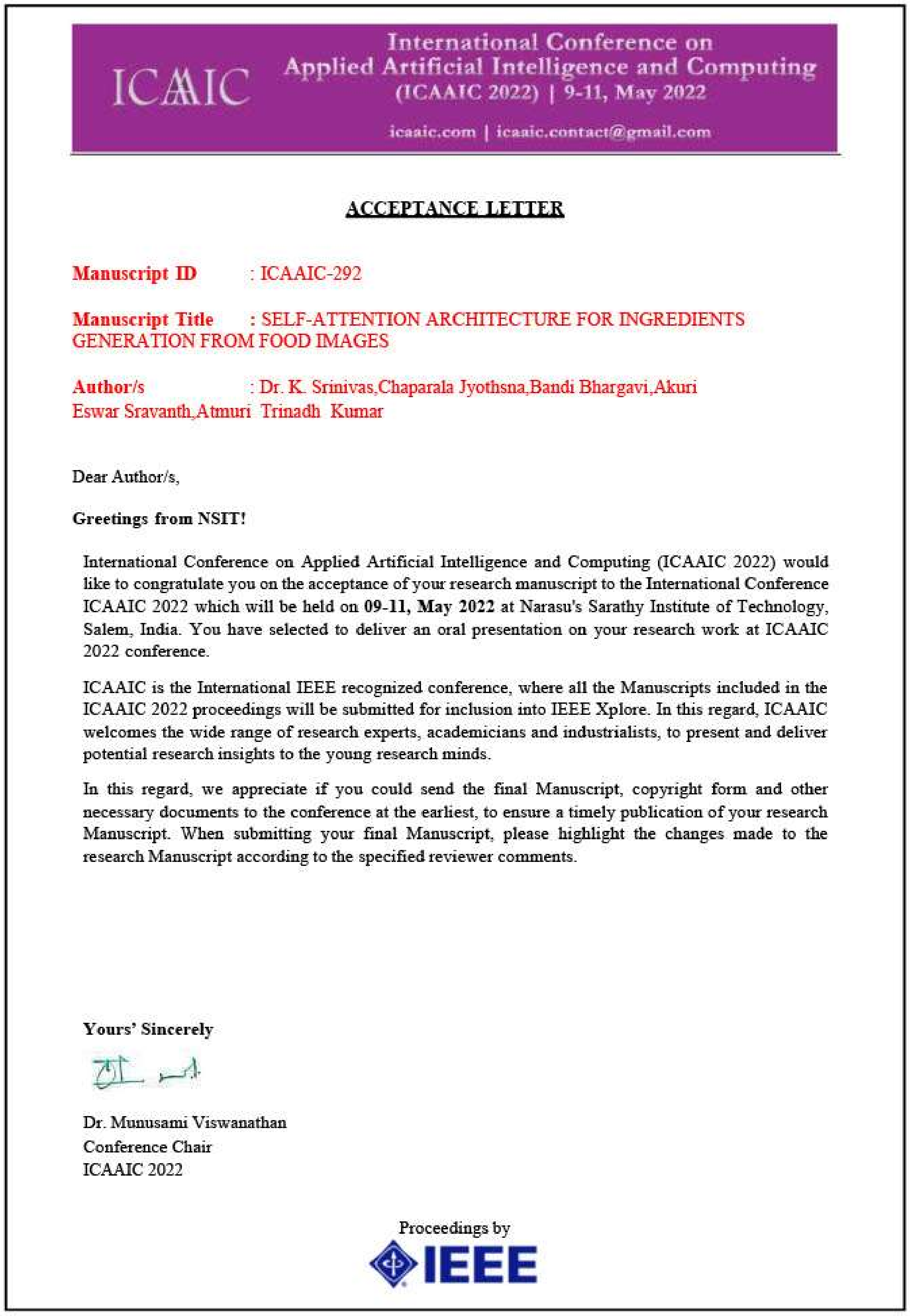
Project Outcomes	
Course Outcome (CO1)	Identify and analyze the problem statement using prior technical knowledge in the domain of interest.
Course Outcome (CO2)	Design and develop engineering solutions to complex problems by employing systematic approach.
Course Outcome (CO3)	Examine ethical, environmental, legal and security issues during project implementation.
Course Outcome (CO4)	Prepare and present technical reports by utilizing different visualization tools and evaluation metrics.

Mapping Table

CS1537: MAIN PROJECT														
Course Outcomes	Program Outcomes and Program Specific Outcome													
	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12		PSO 1	PSO 2
CO1	3	3	1					2	2	2				1	1
CO2	3	3	3	3	3			2	2	2		1		3	3
CO3	2	2	3	2	2	3	3	3	2	2	2			3	
CO4	2		1		3				3	3	2	2		2	2

Note: Map each project outcomes with POs and PSOs with either 1 or 2 or 3 based on level of mapping as follows:

1-Slightly (Low) mapped   2-Moderately (Medium) mapped   3-Substantially (High) mapped



ICAAIC International Conference on Applied Artificial Intelligence and Computing (ICAAIC 2022) | 9-11, May 2022

icaaic.com | icaaic.contact@gmail.com

ACCEPTANCE LETTER

Manuscript ID : ICAAIC-292

Manuscript Title : SELF-ATTENTION ARCHITECTURE FOR INGREDIENTS GENERATION FROM FOOD IMAGES

Author/s : Dr. K. Srinivas,Chaparala Jyothsna,Bandi Bhargavi,Akuri Eswar Sravanth,Atmuri Trinadh Kumar

Dear Author/s,

Greetings from NSIT!

International Conference on Applied Artificial Intelligence and Computing (ICAAIC 2022) would like to congratulate you on the acceptance of your research manuscript to the International Conference ICAAIC 2022 which will be held on **09-11, May 2022** at Narasu's Sarathy Institute of Technology, Salem, India. You have selected to deliver an oral presentation on your research work at ICAAIC 2022 conference.

ICAAIC is the International IEEE recognized conference, where all the Manuscripts included in the ICAAIC 2022 proceedings will be submitted for inclusion into IEEE Xplore. In this regard, ICAAIC welcomes the wide range of research experts, academicians and industrialists, to present and deliver potential research insights to the young research minds.

In this regard, we appreciate if you could send the final Manuscript, copyright form and other necessary documents to the conference at the earliest, to ensure a timely publication of your research Manuscript. When submitting your final Manuscript, please highlight the changes made to the research Manuscript according to the specified reviewer comments.

Yours' Sincerely

[illegible]

Dr. Munusami Viswanathan
Conference Chair
ICAAIC 2022

Proceedings by
IEEE