Engineering Journal issn: 0125-8281

Article

Pushing the Accuracy of Thai Food Image Classification with Transfer Learning

Sirawit Ittisoponpisan[1,a], Chayanat Kaipan[2,b], Somporn Ruang-on[2,c], Rattayagon Thaiphan[2,d] and Kritaphat Songsri-in[2,e,*]

1 Division of Biological Science, Faculty of Science, Prince of Songkla University, Songkhla, Thailand
2 Department of Computer Science, Faculty of Science and Technology, Nakhon Si Thammarat Rajabhat University, Thailand
E-mail: [a]sirawit.i@psu.ac.th, [b]6111425006@nstru.ac.th, [c]somporn_rua@nstru.ac.th, [d]rattayagon_tha@nstru.ac.th, [e]kritaphat_son@nstru.ac.th (Corresponding author)

Abstract. Food image classification is a challenging problem, the solution of which can be of great benefit to many real-world applications such as nutrition and allergy estimation. Most of the previous studies proposed to use variations of convolutional neural networks to tackle the problem. However, due to the limited number of annotated food image datasets, there is still some room for improvement, especially in terms of accuracy and speed. Generally speaking, neural networks trained to solve image classification problems on a small dataset benefit from utilizing the weights of the networks that have been pre-trained on a large image classification dataset such as ImageNet. In this paper, we compare the trade-offs between training networks from scratch, deploying pre-trained networks as feature extractors, and fine-tuning the networks for Thai food image classification. By utilizing Transfer Learning with EfficientNetV1, we were able to achieve higher accuracy for Thai Food Image Classification on the largest publicly available Thai food image dataset, THFOOD-50. In particular, our proposed method improves upon the accuracy of the previous state-of-the-art method from 84.06% to 91.49% while maintaining the speed for the prediction at 103 ms and 1205 ms for GPU and CPU, respectively.

Keywords: Thai food image classification, food image classification, deep learning, neural networks, transfer learning.

ENGINEERING JOURNAL Volume 26 Issue 10
Received 26 December 2021
Accepted 18 October 2022
Published 31 October 2022
Online at https://engj.org/
DOI:10.4186/ej.2022.26.10.57

DOI:10.4186/ej.2022.26.10.57

1. Introduction

Thai cuisine is one of the most well-known in the world. In 2017, Seven Thai dishes made it onto CNN Travel's list of the "World's 50 Best Foods", and Thailand was the country with the most dishes included in the top 100. These famous dishes included "Tom Yam Goong", "Pad Thai", and "Som Tam". According to MasterCard's Global Destination Cities Index, Bangkok, the capital of Thailand, has been the most visited city since 2016. More than 38 million international travelers visit Thailand each year, and most of them are not native Thai speakers. Communicating and ordering food for them can be troublesome. Hence, a model that can accurately classify Thai food images will be very useful. Similar to the general image classification problem, Thai food image recognition has been solved by using convolutional neural networks [1, 2, 3, 4]. In this work, we propose to further improve the performance by employing Transfer Learning.

Since the introduction of AlexNet [5] in 2012, Deep Convolutional Neural Networks (DCNNs) have been the predominant method for image classification problems because they can achieve superior accuracy by exploiting the availability of large labeled datasets with their millions of parameters. Following the success, many subsequent works such as VGG16 [6], InceptionNet [7], and ResNet [8] explored different network architectures to further improve the performance in terms of accuracy, memory usage, and inference speed. Particularly, DCNNs refer to artificial neural networks containing multiple convolutional and other linear and non-linear layers. They are commonly applied to computer vision problems due to their translation equivariant property from the convolution operation. Additionally, DCNNs have also been adopted to solve many other machine learning problems such as Image Detection [9, 10, 11, 12], Voice Recognition [13, 14, 15, 16], and Natural Language Processing [17, 18, 19].

Recently, food image classification has gained more attention from researchers because they have many real-world applications including nutrition estimation, allergy trace recommendation, foreign food prediction, and automatic food logging and searching. Due to the ubiquity of smartphones and social media, millions of food images are easily available and the need for automatic labeling is in high demand. Multiple food image datasets have been collected in the past few years. For instance, Food-11 [20] contains 16,643 images labeled into 11 major food categories such as bread, meat, and dessert. Food-101 [21] consists of 101 food categories and contain 101,000 images. FoodX-251 [22] is a dataset of 251 fine-grained food categories that has more than 150,000 images. ISIA Food-500 [23] is a large dataset with 399,726 images annotated into 500 classes. In addition, many local food datasets were studied. For example, KenyanFood13 [24] is a Kenyan food dataset that contains 8,174 images labeled into 10 classes. ChineseFoodNet [25] contains more than 180,000 images of 208 categories of Chinese food images. FOOD-AI [26] consists of 400,000 Singaporean food images, which are annotated into 756 classes. As for our focus, Thai Food image classification, THFOOD-50 [1] is the largest publicly available dataset which contains 15,770 images of 50 kinds of famous Thai food. These large food image datasets help speed up the development of food recognition models tremendously. Compared to general image classification, food image classification can be more challenging as there is great variation among food images from the same class. Conversely, food images that belong to different classes may appear similar or contain common ingredients that can be hard even for a person to distinguish between them.

Typically, solving image classification problems on smaller datasets benefits from Transfer Learning, in which the weights of the networks that were pre-trained on a large image classification dataset such as ImageNet [27] are reused. In this paper, we studied the trade-off in terms of accuracy, training time, and inference time for MobileNets [28, 29, 30], EfficientNets [31, 32], and ResNets [8, 33]. Since Thai foods consist of many cultural dishes, and the size of the biggest data set, THFOOD50 is still lack behind other food image datasets, we propose to explore training the networks from scratch, deploying their pre-trained version as feature extractors, and fine-tuning them for Thai food image classification.

This paper is organized as follows. Related works are summarized in Section 2. Section 3 presents a Thai food image dataset used in our experiments. Pre-trained networks' architectures and their training procedure are explained in Section 4. Experiments and results are explained in Section 5. Finally, the conclusion and discussions are summarized in Section 6.

2. Related Works

The first paper that pioneered Thai food image recognition is [1] in 2017. It introduced THFOOD-50, the first and currently the largest Thai food image dataset. They proposed a novel network called NU-InNet1.1 which adopts the Inception module [7] with appropriate depth in order to maintain the model's accuracy while reducing the processing time and model size. The proposed model exceeds the performance of the GoogLeNet [7] and reaches an accuracy of 69.8% while reducing the prediction time to 18.16 ms.

An updated version of NU-InNet1.1 was proposed in [34]. The paper explored stacking multiple adjusted inception modules and found that 4 stacks performed best, reaching the accuracy of 80.34% on the THFOOD-50 dataset.

The NU-InNet1.1 Depth 4 was later upgraded into a NU-ResNet1.1 Depth 4 in [35] by adopting the concept of residual layers from ResNet [8]. Skip layers were applied between the beginning and the end of each inception module. It became the best classification model for the THFOOD-50 dataset at 83.07% accuracy with an inference time of 44.60 ms.

There are also other works on Thai food image classification. For instance, [2] focused on classifying images of small Thai desserts and snacks with a simple 7-layer CNN and was able to achieve 86.00 % accuracy.

[3] proposed another Thai food image dataset with 3,961 images that were grouped into 11 classes. They fine-tuned a GoogLeNet, which had been pre-trained on an ImageNet dataset, and reached an accuracy of 88.33%.

A food recognition application was developed by [4] to classify Thai food images into 13 kinds. Based on their own Thai food image dataset of 7,632 images, they deployed Transfer learning with a VGG19 network [6] and were able to reach an accuracy of 82.00%. Since the VGG19 network was quite big (144×10^6 parameters), the predictions were performed on the server and the results were sent back to clients' devices.

From these previous works, we can gather some key information here. First, all of them rely on training neural networks in order to achieve state-of-the-art results. Secondly, Transfer learning can boost the performance of classifiers on a small dataset significantly. However, there has never been an extensive study of applying Transfer Learning for the largest Thai food dataset, THFOOD-50. This research objective is to compare the results of Transfer Learning with different networks, training strategies, and augmentations.

3. Dataset Analysis

3.1. THFOOD-50 Dataset

The THFOOD-50 dataset [1] is the largest publicly available dataset for Thai food image classification with images collected from Google, Bing, and Flickr. The dataset contains 15,770 images of 50 famous Thai dishes. Most of the dishes are starters or main courses such as "Tom Yum Kung", "Pad Thai", and "Thai Green Curry". There is only one dessert class, which is "Khao Niew Ma Muang" (Mango sticky rice). The dataset is separated into two parts, 90% for the training set and 10 % for the test set. A sample image from each kind of the dishes is shown in Fig. 1. The images in the dataset are collected at different sizes and ratios. They are also captured under various lighting and coloring conditions. The proportion of each class in the training set is depicted in Fig. 2. We can notice that the numbers of each type of food are quite balanced, with the greatest imbalance being "Kuay Jab" (Chinese roll noodle soup) and "Kor Moo Yang" (Grilled Pork Neck) which are in the ratio 609:164 (3.71:1).

Fig. 1. Sample images from 50 famous Thai dishes from the THFOOD-50 dataset [1].

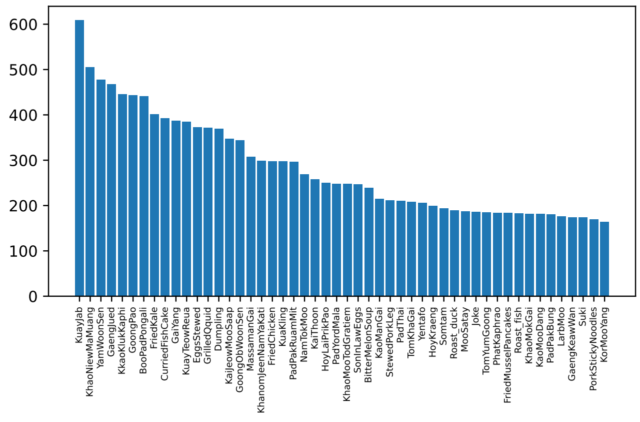

Fig. 2. Distributions of all 50 Thai dishes in the THFOOD-50 dataset. The numbers of each type of food are quite balanced with the greatest ratio between different dishes being 609:164 (3.71:1) for KuayJab (Chinese roll noodle soup) and KorMooYang (Grilled Pork Neck).

4. Methods

In this section, we describe networks that are used in our experiments, namely MobileNet and ResNet. We then explain the training procedure used in our experiments such as training networks from scratch and Transfer Learning. Lastly, we extensively describe the implementation details of our experiments.

4.1. MobileNet

MobileNet [28] was originally proposed in 2017, which helped kick-start research on neural networks for embedded systems. Owing to its success, MobileNetV2 [29] and MobileNetV3 [30] were subsequently studied.

4.1.1. MobileNetV1

MobileNetV1 replaced regular convolutional layers with depthwise separable convolutions layers, which were proven to be highly efficient in terms of the produced accuracy compared to the number of required parameters. Specifically, a standard convolutional layer, which takes a $D_F \times D_F \times M$ input and produces an output with shape $D_F \times D_F \times N$, will require the number of parameters as

$$D_K \times D_K \times M \times N \tag{1}$$

where D_F and D_K indicate the spatial size of a square input and a convolutional kernel size, and M and N refer to the feature size of the input and output, respectively. The standard convolutional layers would also have a computational cost of

$$D_K \times D_K \times M \times N \times D_F \times D_F \tag{2}$$

Whereas, a depthwise separable convolutional layer would require the number of parameters as small as

$$D_K \times D_K \times M + M \times N \tag{3}$$

with the computational cost as

$$D_K \times D_K \times M \times D_F \times D_F + M \times N \times D_F \times D_F \tag{4}$$

The paper also introduced two simple strategies to further reduce the model's size and computational cost. First, each intermediate feature can be shrunk by a factor of $\alpha \in (0, 1]$. Secondly, the input image can be spatially reduced by a factor of $\rho \in (0, 1]$. In our experiments, we simply set both α and ρ to 1. More details about the full network architecture can be found in [28].

4.1.2. MobileNetV2

MobileNetV2 modified the way a depthwise separable convolutional layer in MobileNetV1 worked and further improved the efficiency and accuracy of the networks. Firstly, it introduced two types of layer blocks: a residual block and a downsizing block. Both of these blocks contain 3 layers. Unlike the MobileNetV1, the first layer is a 1×1 convolutional layer with RELU [36] as a non-linear activation function. The second layer is a depthwise convolutional layer. Lastly, the third layer is another 1×1 convolutional layer with a linear activation function. The residual block additionally contains a residual skip connection within the block. The full architecture is clearly explained in [29].

4.1.3. MobileNetV3

MobileNetV3 is the most recent enhancement to the MobileNets architecture. Compared to MobileNetV1 and MobileNetV2 whose architectures were manually developed, the architecture of MobileNetV3 were automatically searched by using reinforcement learning methods MnasNet [37] and NetAdapt [38]. First, the global structure of the networks' blocks was optimized with the MnasNet. Then, each layer was sequentially fine-tuned with the NetAdapt. Additionally, the inclusion of a squeeze-and-excite inside the residual blocks improved the accuracy while reducing the number of parameters. Lastly, the last stage of the networks was manually optimized to reduce computational time and parameters while maintaining similar accuracy. Details of the whole architecture can be found at [30].

4.2. EfficientNet

4.2.1. EfficientNetV1

EfficientNet is a convolutional neural network designed to proportionally scale a network's depth, width, and resolution dimensions with a compound coefficient. Particularly, if more computational resources with coefficient ϕ are allowed, we can increase the network depth by α^ϕ, width by β^ϕ, and image size by γ^ϕ, where α, β, and γ are constant coefficients determined by a small grid search. The original paper [31] proposed 8 different scale networks. In this paper, we only focus on the one whose image input size is 224×224 in order to make a fair comparison with other methods.

4.2.2. EfficientNetV2

EfficientNetV2 improves EfficientNetV1 in terms of computational time, network size, and prediction accuracy. To speed up training time and reduce the network's size, it restricts the maximum image scaling size to 480×480 and skips unnecessary search options discovered by the original EfficientNetsV1 such as pooling skip ops. The paper [32] also introduced a concept of progressive learning, in which the networks progressively increase regularization as the network learns longer. Similarly to EfficientNetsV1, multiple networks size were proposed, but we only keep the network whose image size is 224×224.

4.3. ResNet

4.3.1. ResNetV1

ResNet is the most highly-cited paper between 2015 and 2019. The proposed models won the ILSVRC and COCO competitions in 2015. The paper introduced the idea of a residual skip connection between layers which was experimentally shown to have a few benefits. First, the skip connection alleviates the issue of the vanishing gradient. Secondly, it enables layers to keep the identity function rather than being forced to transform the features. These effects allow a deeper network to be trained effectively. Multiple versions with different depths were introduced in the paper. Namely, ResNet18, ResNest50, and ResNet152. In this paper, we focused on exploring the mid-size network i.e. ResNet50V1.

4.3.2. ResNetV2

ResNetV2 extensively studied the residual building block proposed in ResNetV1 and proposed a novel residual unit that made the training even more stable and improved accuracy. Particularly, non-linear activation functions such as RELU were moved to the beginning of the residual block. The last non-linear activation layer was also removed in the second version. The full details of the residual block and architecture can be found at [33]. In order to allow for network comparison, we also select ResNet50V2 as a model of choice.

4.4. Training Networks from Scratch

When a deep neural network is trained from scratch, its weights from each layer are initialized randomly, and they are updated iteratively using the stochastic gradient descent method and back-propagation. The weight optimization process is demonstrated in Fig. 3. From the figure, we can see that we separated the networks into two parts: a feature extractor and a classifier. The feature extractors usually refer to the convolutional layers at the beginning of the networks while the classifiers are normally implemented using fully connected layers. Additionally, we highlighted the whole network in red to represent the fact that every layer's parameters are updated during training.

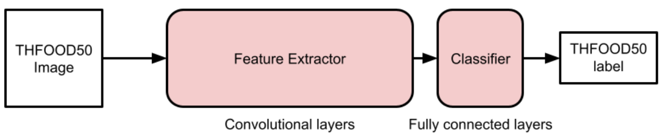

Fig. 3. When a network is trained from scratch, the parameters in all layers including a feature extractor and a classifier are updated. This is represented by the highlighted red blocks.

4.5. Transfer Learning

In general, training a deep neural network from scratch takes a long time and requires high computational resources. Additionally, a large amount of data is required to achieve good results. These restrictions can be overcome by Transfer Learning, an alternative approach for training deep neural networks that offer good results even with small datasets.

Transfer learning in the context of deep learning refers to utilizing model weights obtained from a network pre-trained on relatively large datasets to help solve new tasks for which little training data is available. For instance, ImageNet-1k [27] and ImageNet-21k [39] are extremely large image recognition datasets that contain millions of images, each of which is annotated with one of the thousands of labels. As a result, they are commonly used to pre-train neural networks for computer vision problems. For image classification, deep networks usually consist of many convolution layers followed by a few fully connected layers. Correspondingly, early layers of the pre-trained networks are known to extract generic low-level features from the photographs. As the layers progress, the networks tend to learn more abstract representations of the images. Hence, the weights of the network trained on large image classification datasets usually give a good starting point for training networks on a similar dataset with a smaller size. There have been many studies [40, 41, 42] that tried to improve the ways transfer learning can optimally benefit smaller problems. In this paper, we consider the following setups.

4.5.1. Features extractors

The simplest way that a pre-trained network can be utilized for a smaller dataset is to use them as feature extractors. Particularly, images from the smaller dataset will be passed through the pre-trained networks once, and the output from the last convolutional layer is typically used as intermediate features. These features can then be used as input for any machine-learning model. In this paper, we simply used a fully connected layer to make the predictions in order to compare the results with other training strategies. With this approach, the computational cost is much less expensive as each image in the small dataset can be passed through the pre-trained network once. The extracted feature of each image then can be trained with smaller machine-learning models. The training strategy is depicted in Fig. 4. The top part of the figure represents a deep neural network that was pre-trained on the ImageNet dataset. The convolutional layers of the network are reused as depicted in the bottom part of the figure. We can see that the weight of the feature extractor is kept frozen during training, and only the classifier is updated. In the figure, the blue blocks represent frozen weights, and the red blocks mean that they are being trained.

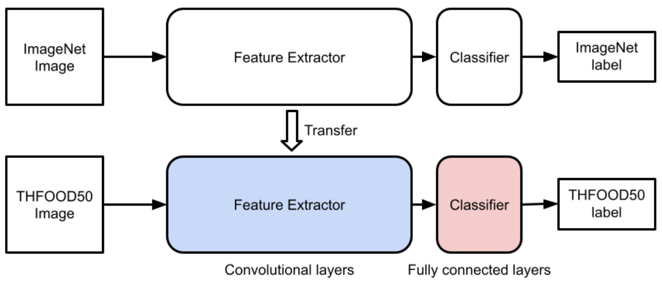

Fig. 4. When a pre-trained network is used as a feature extractor, the parameters of the feature extractors layers are kept frozen while the weight of the classifier is updated during training. This is demonstrated using the colors blue and red, respectively.

4.5.2. Networks fine-tuning

Typically, the numbers of labeled classes of a large dataset and the smaller dataset are different for image classification. Hence, the first step of fine-tuning a pre-trained deep neural network is to adjust the last output layer to have the same size as the available class in the target dataset. This first step is essentially the same as using the convolutional layers as a feature extractor, and only the fully connected layer is updated. This initial classifier training is necessary for stable training in the next step where the whole network is fine-tuned. The training procedure is demonstrated in Fig. 5. The top network depicts pre-training a network on the ImageNet dataset. The weights of the parameters from the convolutional layers are reused in the second network in the middle, where only the fully connected layer is trained. The network at the bottom demonstrates the whole network being fine-tuned. Both convolutional layers and the fully connected layer are trained end-to-end. Normally, a relatively lower learning rate is used during fine-tuning.

4.6. Implementation

In this session, we discuss the implementation details regarding training the neural networks from scratch as well as fine-tuning the networks. Our experiments are implemented entirely with Python 3 [43] using the framework Keras [44].

4.6.1. Image pre-processing

To allow for model comparison, we followed the image pre-processing setup used in the networks pre-trained on the ImageNet dataset, in which image pixels are re-scaled to have values between 0 and 1. This is simply done by dividing each image pixel by 255. Specifically, let X be the pixel values of a given image and X' be the normalized pixel values, then the normalization is performed as

$$X' = \frac{X}{255} \quad (5)$$

Since the dataset contained images of different sizes and in different ratios, we adjusted all of the input images for all of the networks in our experiments to be 224×224, which was achieved by center cropping and bi-linear interpolation.

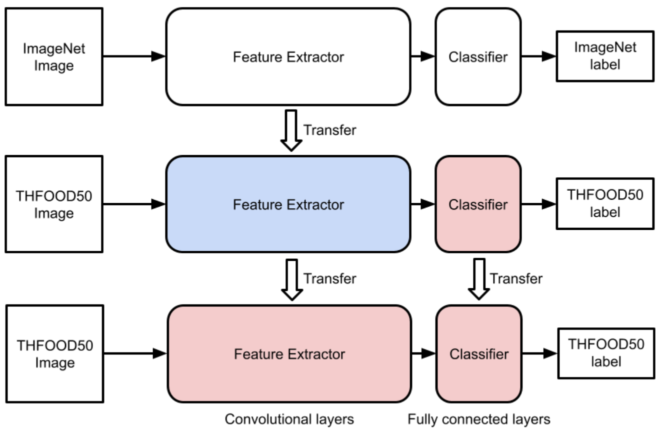

Fig. 5. In order to fine-tune a pre-trained deep neural network, the fully connected layer must first be trained from scratch. Then the whole network can be fine-tuned with a much lower learning rate. The top network depicts pre-training a network on the ImageNet dataset. The second network in the middle represents training the fully connected layer. The network at the bottom demonstrates fine-tuning the whole network. In this figure, the blue blocks represent frozen weights, and the red blocks mean that they are being trained.

4.6.2. Image augmentations

Image augmentations [45, 46, 47, 48] have been proven to enhance the effectiveness of training deep neural networks by increasing the amount of training data by adjusting the images geometrically or coloristically. In addition, when they are deployed in an online manner during training, they can also prevent the networks from over-fitting on the small datasets. For our experiments, we only performed geometric augmentations to the images during training because any color augmentation could interfere with the semantics of the Thai food images. In particular, we explored random horizontal flipping, random rotation, random zoom, and random translation. Each of these random augmentation intensities can be adjusted using a parameter “ratio” in the framework Keras. For simplicity, we only explored the same value of ratio among these augmentations at four different levels of 0.05, 0.10, 0.15, and 0.20.

4.6.3. Hyper-parameters setting

Training deep networks tends to be time and resource-expensive, and one of the causes is hyper-parameters tuning. We avoided a few of these hyper-parameter searches by setting them to sensible values. Specifically, we set the batch size to 64 and used Adam optimizer [49] as our stochastic optimizer due to its robustness for training deep neural networks. For training from scratch and training the last fully connected layer, the learning rates were set to 1×10^{-3}, and the β_1 and β_2 were set to 0.9 and 0.999, respectively. For fine-tuning the networks, we only adjusted the learning rates to be lower at 1×10^{-5} in order to prevent the networks from making any drastic changes to the weights. For all the training, we let the networks learn for 100 epochs. Lastly, the weight decay was set to 5×10^{-4}.

4.6.4. Loss function

Let y be a one-hot ground-truth label vector, $\hat{y}$ be the predicted probabilities of each class from the networks, and let c refer to each possible class. We utilized the commonly-used multi-class cross entropy as our loss function. Mathematically, it is defined as

$$L(y, \hat{y}) = -\sum_{c} y_c \log \hat{y_c} \tag{6}$$

5. Results and Discussion

5.1. Evaluation of Models

Since the proportion of each label is quite balanced, we measured the performances of the networks using accuracy. Specifically,

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{7}$$

where TP is the number of true positive examples, TN is the number of true negative examples, FP is the number of false positive examples, and FN is the number of false negative examples.

5.2. Internal Results

We conducted multiple experiments and reported the results on the THFOOD-50 dataset. Specifically, we compared the performances of different models when they were trained with each strategy. Various levels of image augmentations were examined. Finally, the comparison between the accuracy against inference time of all models is reported.

5.2.1. Networks comparison

In total, the seven aforementioned networks were trained on the THFOOD-50 dataset. In particular, they were MobileNetV1, MobileNetV2, MobileNetV3, EfficientNetV1, EfficientNetV2, ResNetV1, and ResNetV2. Each of these networks was trained with three strategies. Firstly, they were trained from scratch. Then, two procedures were utilized for transfer learning where only the last fully-connected layer or the whole network was fine-tuned as described in Section 4.5. The performances in terms of the number of trained parameters, training time, and accuracy are reported in Table 1. Generally, we can notice that the size of the MobileNets, EfficientNet, and ResNets are significantly different based on the number of parameters and the training time. Comparing the training procedures, fine-tuning only the last layer took the least amount of time for training, and the MobileNetV1 took the shortest time at 41 milliseconds per iteration. Training the whole network both from scratch or fine-tuning took around 300 milliseconds per iteration for most of the networks. In terms of accuracy, the performance of EfficientNets is always better than those of MobileNets and ResNets. This may be because MobileNets are too small and underfit the dataset while ResNets are too deep and contain a much larger number of parameters, allowing for overfitting the THFOOD-50 dataset. This phenomenon is consistent with the results reported in [1, 34, 35], in which larger networks seemed to perform worse on the THFOOD-50 dataset. When the networks were trained differently, the accuracy of fine-tuning only the last layer was the worst, followed by training the whole network from scratch and fine-tuning the whole network, respectively. The validation accuracies during training of these seven networks trained with different strategies were shown in Fig. 6. From the experiments, the best model is the fully fine-tuned EfficientNetV1 reaching an accuracy of 90.87%. The validation accuracies during training of these networks when they were fully trained were explicitly compared in Fig. 7.

5.2.2. Image augmentations

We conducted four levels of image augmentations at 0%, 5%, 10%, and 15%. All the models were fully fine-tuned and their accuracies are presented in Table 2. From the table, we can see that augmentations had a positive effect on the accuracies. For MobileNets and EfficientNets, the benefits are most notable at intermediate levels of augmentations either at 5% or 10 %. The best performing model reached the accuracy of 91.49%, which is EfficientNetV1 with an augmentations level of 10%. On the other hand, deeper networks such as ResNets gained higher accuracies as the level of augmen-

Table 1. The performances of the proposed methods when they were trained using different strategies.

Models	Training from scratch			Fine-tuning last layer			Fine-tuning whole network		
	Params (M)	Training (ms/it)	Acc (%)	Params (M)	Training (ms/it)	Acc (%)	Params (M)	Training (ms/it)	Acc (%)
MobileNetV1	3.280	284	82.93	0.051	**41**	81.43	3.280	284	89.49
MobileNetV2	**2.322**	317	84.37	0.064	46	78.67	**2.322**	317	87.55
MobileNetV3	4.291	**282**	86.49	0.064	43	84.80	4.291	**282**	89.93
EfficientNetV1	4.113	336	88.24	**0.063**	98	85.12	4.113	336	90.56
EfficientNetV2	5.922	321	**89.06**	0.071	86	**86.64**	5.922	321	**90.87**
ResNet50V1	23.663	294	73.80	0.102	80	75.30	23.663	294	85.87
ResNet50V2	23.667	297	75.55	0.102	90	71.92	23.667	297	82.80

tation increased. This is because deeper networks have a larger number of parameters compared to the amount of available data. This typically leads to over-fitting, and a higher level of regularization can overall lead to better performance. However, their performances were still far behind those of EfficientNetV1. The ResNet50V1 only achieved an accuracy of 85.87%.

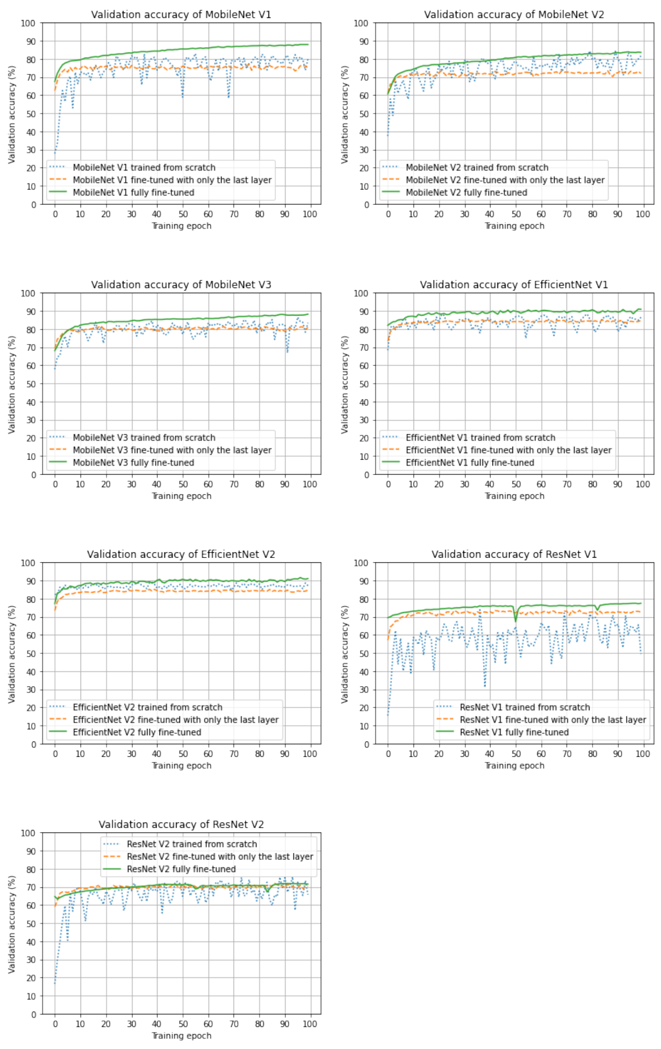

Fig. 6. The validation accuracies of different models when they were trained from scratch, partially fine-tuned, and fully fine-tuned.

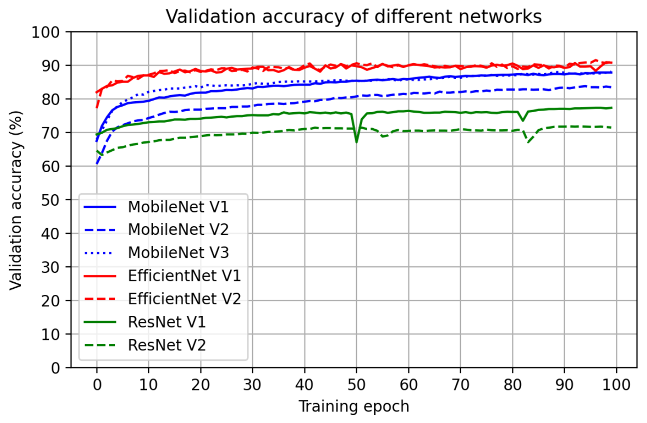

Fig. 7. The comparison of validation accuracies when models were fully fine-tuned.

Models	Accuracy (%)			
	Aug 0%	Aug 5%	Aug 10%	Aug 15%
MobileNetV1	89.49	89.99	**90.12**	89.93
MobileNetV2	87.55	**88.93**	87.93	88.24
MobileNetV3	89.93	**90.43**	90.18	90.12
EfficientNetV1	90.56	90.12	**91.49**	91.43
EfficientNetV2	90.87	90.74	**90.99**	90.06
ResNet50V1	85.85	85.62	85.30	**85.87**
ResNet50V2	82.80	84.12	83.99	**81.18**

Table 2. The accuracies of the models when they were trained with various levels of image augmentations.

5.2.3. Accuracy vs run times

The performance of the models was additionally compared between accuracies and run times. During testing, an image was cropped and resized to 224 × 224. The run times were measured by using both a CPU and a GPU. The CPU used for testing was the Intel(R) Xeon(R) @ 2.00GHz. For the GPU, an NVIDIA Tesla P100 with 16 GB of RAM was used for measuring the inference speed. The comparison is shown in Table 3.

As shown in the table, the GPU inference times of the MobileNets were around 80 ms and the accuracies were maximum at 90.43% for the MobileNetV3. For EfficientNets, the GPU inference time is around 95 ms while reaching an accuracy of 91.49%. Whereas the GPU prediction times of the ResNets were around 120 ms and only reached just 85.87% for the ResNet50V1. The CPU inference times demonstrate similar trends between MobileNets, EfficientNets, and ResNets with around 700 ms, 1100 ms, and 3000 ms, respectively. The experimental results confirmed that the architectures of the EfficientNets are more suitable for the problem of Thai food image classification compared to MobileNets and ResNets in terms of optimal training speed, inference speed, and accuracy.

Models	FLOPs (G)	Inference (ms)		Acc (%)
		CPU	GPU	
MobileNetV1	0.57	740	**74**	90.12
MobileNetV2	0.34	**635**	79	88.93
MobileNetV3	**0.24**	650	81	90.43
EfficientNetV1	0.40	1205	103	**91.49**
EfficientNetV2	0.71	1097	91	90.99
ResNet50V1	3.92	2993	116	85.87
ResNet50V2	4.21	3115	128	84.62

Table 3. The comparison of different models in terms of inference times and accuracies.

5.3. Comparison with State-of-the-Art Models

Here, the proposed methods are compared with recent models from [1, 34, 35]. The performances are measured in terms of a number of model parameters and image classification accuracy. The results are shown in Table 4. From the table, we can observe that the previous state-of-the-art model was the NU-ResNet from [35] with an accuracy of 83.07%. Considering our proposed models, we can notice that Transfer Learning allows the models to perform better. For example, the accuracy of the ResNet50V1 trained from scratch reported in [35] at 72.88% can be boosted significantly with fine-tuning to reach 85.87% which is already better than the NU-ResNet. Our best-performing model is EfficientNetV1 which can achieve noticeably higher accuracy than previous methods, achieving a new state-of-the-art result of 91.49%.

Regarding the number of parameters, we can notice that the models' size is not necessarily proportional to the accuracies of the models. Previous works [1, 34, 35] demonstrated that while the size of the models was still relatively low, increasing the model size seemed to improve the accuracies as shown in NU-InNet 1.0, NU-InNet 1.1, and NU-ResNet. We can also see a similar trend with our MobileNetV1, MobileNetV2, MobileNetV3, EfficientNetV1, and EfficientNetV2. On the other hand, extremely large models such as ResNet50V1 and ResNet50V2 tended to over-fit the small dataset and performed worst even with higher levels of image augmentations. From Table 4, we can see that the EfficientNetV1 can achieve 91.49% accuracy using 4.11 million parameters while the previous stat-of-the-art NU-ResNet reached 83.07% while using 1.48 million parameters.

Models	Params (M)	Acc (%)
NU-InNet 1.0 [1]	0.88	69.80
NU-InNet 1.1 [34]	0.94	80.34
NU-ResNet [35]	1.48	83.07
ResNet50V1 [35]	23.66	72.88
Our MobileNetV1	3.28	90.12
Our MobileNetV2	3.32	88.93
Our MobileNetV3	4.29	90.43
Our EfficientNetV1	4.11	**91.49**
Our EfficientNetV2	5.92	90.99
Our ResNet50V1	23.66	85.87
Our ResNet50V2	23.66	84.62

Table 4. Comparison with previous state-of-the-art models in terms of model parameters and accuracies.

5.4. Qualitative Results

In this section, we qualitatively demonstrate the result of the EfficientNetV1 prediction using Gradient-weighted Class Activation Mapping (Grad-CAM) [50], which localizes parts of the image that contribute the most to the model's classification. Throughout the experiments, the output heat maps of the Grad-CAM were overlaid on top of the input images whose transparency values were set to 0.8. The areas that contributed more toward the prediction were highlighted in red while parts of the images that contributed less were colored blue. The outputs in the figures are best viewed in color.

5.4.1. Correctly classified Images

In order to understand how our network made predictions, we first investigate the results from Grad-CAM on correctly classified images. We intentionally selected ten images with the highest probabilities and ten images with the least probabilities. Their results are demonstrated in Fig. 8 and Fig. 9, respectively.

In Fig. 8, we can notice that all the images were classified with high probabilities reaching nearly 100%. Additionally, the focuses of the model on these images seem sensible as they tended to localize to the features of the images that are specific to the food label. For example, considering the first and the ninth images, which depict "Gaeng Jued" (Clear soup with egg tofu and minced pork), the EfficientNetV1 consistently focused on the egg tofu locations. A similar pattern occurs in the dish "Phat Kaphrao" (Stir-fried pork and basil) shown in the seventh and the eighth columns. Both images' classifications seemed to be mainly influenced by the pork and basil parts.

In Fig. 9, all of the images were correctly classified with a probability as small as 50%. Although the probabilities in this figure are relatively low, the output of the Grad-CAM still tended to localize well specifically to the food label. Particularly, the model paid attention to the bitter melon in the Bitter-melon soup of the first image, the crab's craw in "Boo Pad Pong Gali" (Stir-friend Crab curry) of the second image, and the egg part of the "Kai Jeow Moo Saap" (Omelette with minced pork) in the third image. Again, in the ninth image, the egg tofu of the "Gaeng Jued" (Clear soup with egg tofu and minced pork) was correctly focused.

5.4.2. Misclassified images

We also studied the behavior of our model when it misclassified food images. We purposely selected ten images that were incorrectly recognized with the highest probabilities and show their Grad-CAM output in Fig. 10. We can notice from the figure that the model could misclassify Thai food images with a high probability of almost 100%. Nevertheless, five out of ten images also had their correct label as the top five predictions, and the correct labels are colored in green. Some of the images' predictions seem to be completely misunderstood by the model. For example, the prawn in the first and the fourth images were mistaken for "Khao Niew Ma Muang" (Mango with sticky rice). For those whose correct labels were among the top five predictions, the results from the Grad-CAM tended to localize and paid attention to parts of the images that could be ambiguous between labels that are semantically and visually similar. In particular, two close dishes such as "Kor Mhoo Yang" (Grilled pork's neck) and "Num Tok Mhoo" (Spicy grilled pork) were misjudged by the model in the second and the third images.

Additionally, we studied the output of the Grad-CAM on ten misclassified images that had the lowest probabilities as shown in Fig. 11. In some of the images, the model seemed to already focus on the core feature part of the images but misinterpreted them. For instance, the crab shell in the "Boo Pad Pong Gali" (Crab curry) dish was the main focus of the model, but it was misjudged as "Tom Kha Gai" (Thai chicken coconut soup). Similarly, in the second image, the model localized the egg part of the "Son in law eggs" (Fried eggs with tamarind sauce) but classified the image as "Gai Yang" (Grilled chicken). This confirms the difficulty of the food image classification problem.

6. Conclusion

In this paper, we studied extensively the effects of network size, training strategies, and the level of image augmentation on the accuracies of Thai food image classification for the THFOOD-50 dataset. Our experiments confirmed that utilizing Transfer Learning by fine-tuning the whole network on EfficientNetV1 at the image augmentation level of 10% can greatly push the limit of the accuracy up to 91.49%, about 10% more than the previous state-of-the-art model, while maintaining the testing time with a GPU and a CPU to 103 ms and 1205 ms, respectively. Our qualitative results from Grad-CAM also confirmed the localization reasoning of the proposed training method. For future works, a study on how the network can utilize co-occurrence information of the ingredients may improve the performance of the networks further as we can notice from the output of the Grad-CAM method that the proposed model only tends to look for a specific key feature in the images in order to make the predictions.

References

[1] C. Termritthikun, P. Muneesawang, and S. Kanprachar, "Nu-innet: Thai food image recognition using convolutional neural networks on smartphone," *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)*, vol. 9, pp. 2289–8131, 08 2017.

[2] P. Mookdarsanit and L. Mookdarsanit, "Name and recipe estimation of thai-desserts beyond image tagging," *Kasem Bundit Engineering Journal*, vol. 8, pp. 193–203, 2018.

[3] N. Hnoohom and S. Yuenyong, "Thai fast food image classification using deep learning," in *2018 International ECTI northern section conference on electrical, electronics, computer and telecommunications engineering (ECTI-NCON)*. IEEE, 2018, pp. 116–119.

[4] U. Tiankaew, P. Chunpongthong, and V. Mettanant, "A food photography app with image recognition for thai food," in *2018 Seventh ICT International Student Project Conference (ICT-ISPC)*. IEEE, 2018, pp. 1–6.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proceedings of the 25th International Conference on Neural Information Process-*

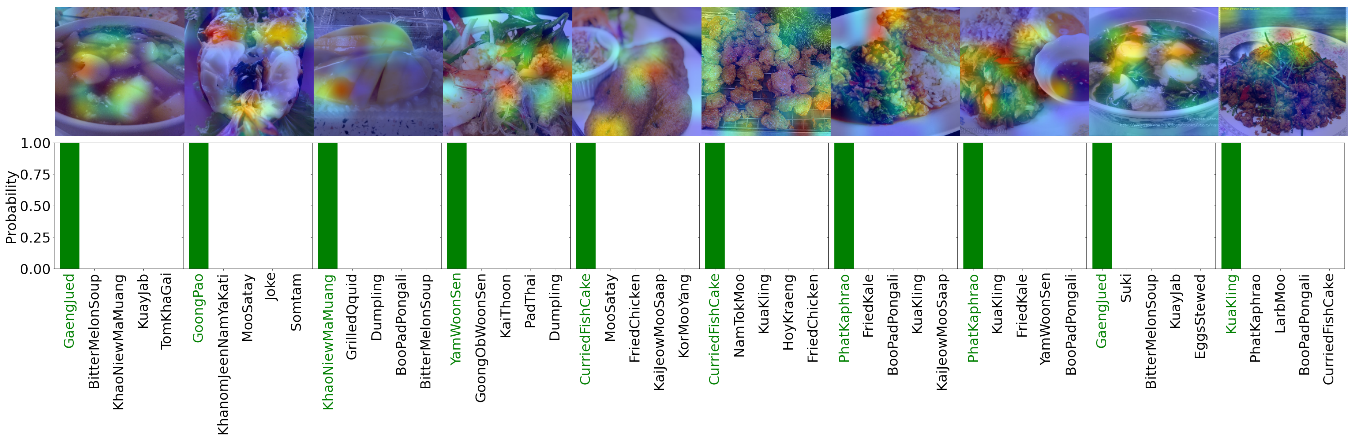

Fig. 8. The outputs of the Grad-CAM method are reported for the top ten images that were correctly classified by our EfficientNetV1 with the highest probabilities.

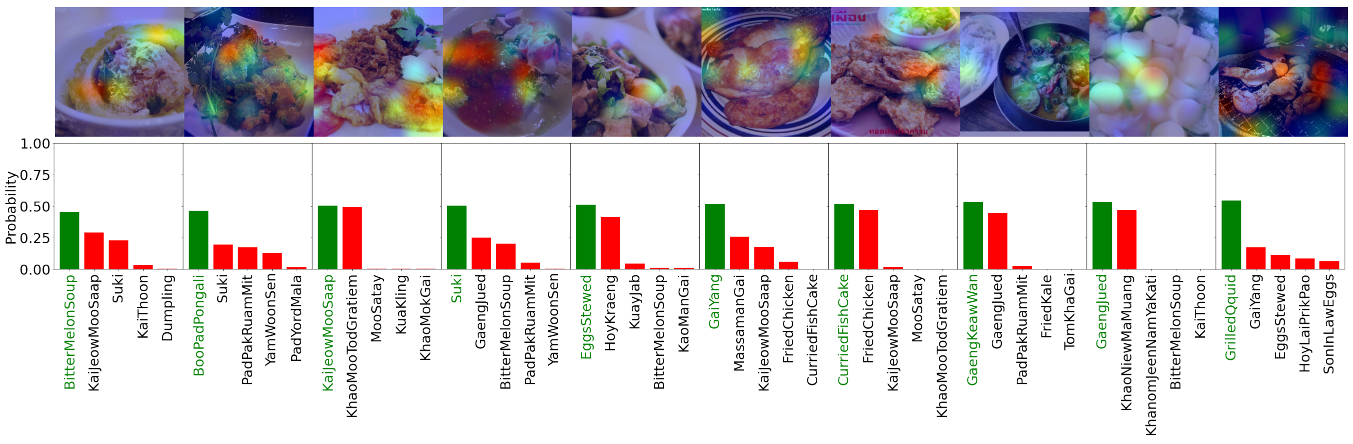

Fig. 9. The outputs of the Grad-CAM method are reported for the last ten images that were correctly classified by our EfficientNetV1 with the least probabilities.

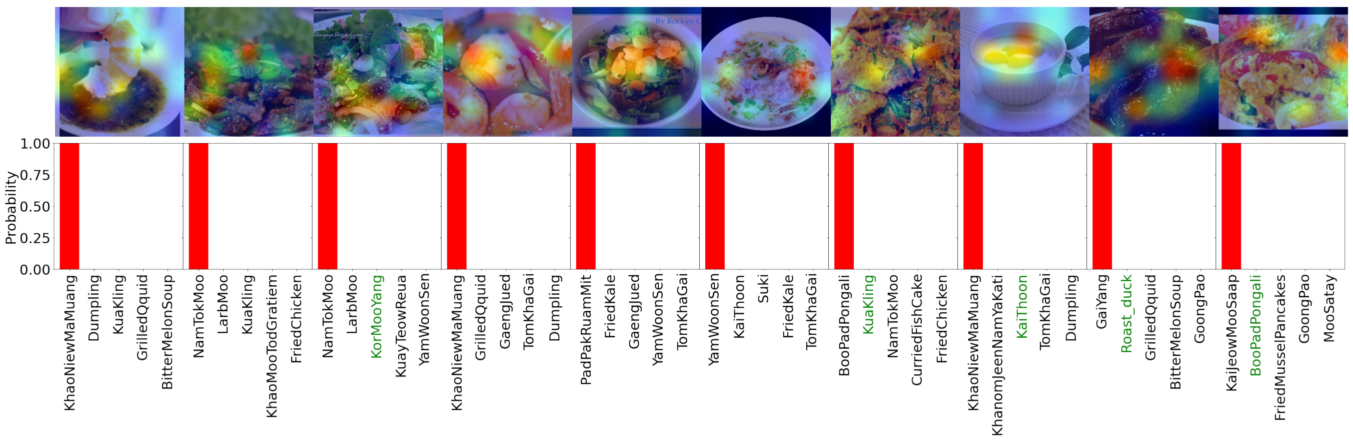

Fig. 10. The outputs of the Grad-CAM method are reported for the top ten images that were incorrectly classified by our EfficientNetV1 with the highest probabilities.

DOI:10.4186/ej.2022.26.10.57

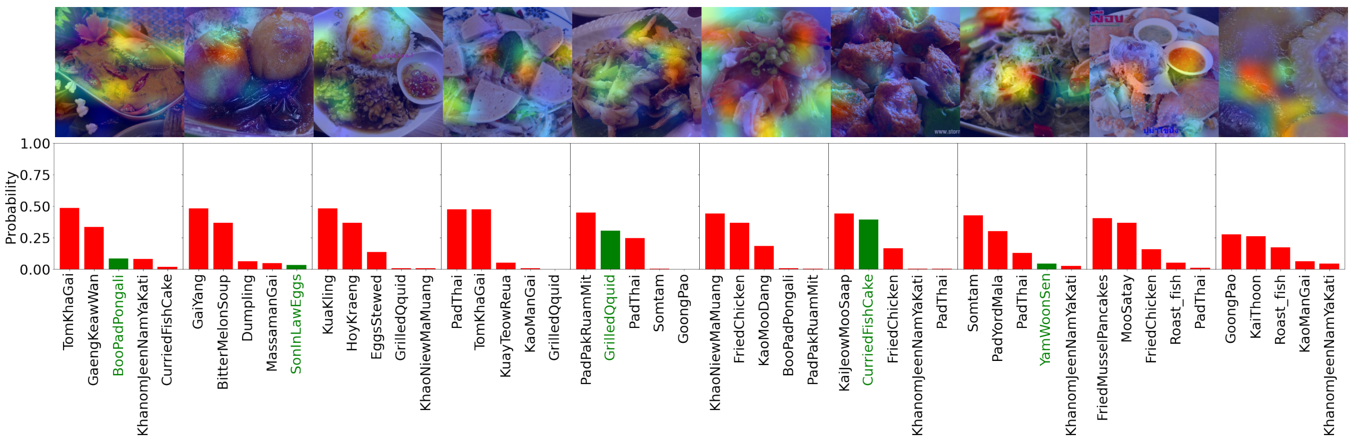

Fig. 11. The outputs of the Grad-CAM method are reported for the top ten images that were incorrectly classified by our EfficientNetV1 with the least probabilities.

ing Systems - Volume 1, ser. NIPS'12. Red Hook, NY, USA: Curran Associates Inc., 2012, p. 1097–1105.

[6] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations*, 2015.

[7] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1–9.

[8] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[9] C. Papageorgiou and T. Poggio, "A trainable system for object detection," *International journal of computer vision*, vol. 38, no. 1, pp. 15–33, 2000.

[10] Z. Zou, Z. Shi, Y. Guo, and J. Ye, "Object detection in 20 years: A survey," *arXiv preprint arXiv:1905.05055*, 2019.

[11] J. Redmon, S. Divvala, R. Girshick, and 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.

[12] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *European Conference on Computer Vision*. Springer, 2020, pp. 213–229.

[13] L. Basyal, S. Kaushal, and G. Singh, "Voice recognition robot with real time surveillance and automation," *International Journal of Creative Research Thoughts*, vol. 6, no. 1, pp. 2320–2882, 2018.

[14] L. Eljawad, R. Aljamaeen, M. Alsmadi, I. Almarashdeh, H. Abouelmagd, S. Alsmadi, F. Haddad, R. Alkhasawneh, M. Alazzam *et al.*, "Arabic voice recognition using fuzzy logic and neural network," *International Journal of Applied Engineering Research (IJAER)*, pp. 651–662, 2019.

[15] Y. Xiwen, "Design of voice recognition acoustic compression system based on neural network," *Wireless Personal Communications*, pp. 1–19, 2021.

[16] G. K. Berdibaeva, O. N. Bodin, V. V. Kozlov, D. I. Nefed'ev, K. A. Ozhikenov, and Y. A. Pizhonkov, "Pre-processing voice signals for voice recognition systems," in *2017 18th International Conference of Young Specialists on Micro/Nanotechnologies and Electron Devices (EDM)*. IEEE, 2017, pp. 242–245.

[17] H. Jelodar, Y. Wang, R. Orji, and S. Huang, "Deep sentiment classification and topic discovery on novel coronavirus or covid-19 online discussions: Nlp using lstm recurrent neural network approach," *IEEE Journal of Biomedical and Health Informatics*, vol. 24, no. 10, pp. 2733–2742, 2020.

[18] F. N. Iandola, A. E. Shaw, R. Krishna, and K. W. Keutzer, "Squeezebert: What can computer vision teach nlp about efficient neural networks?" *arXiv preprint arXiv:2006.11316*, 2020.

[19] A. Kumar, J. M. Chatterjee, and V. G. Díaz, "A novel hybrid approach of svm combined with nlp and probabilistic neural network for email phishing," *International Journal of Electrical and Computer Engineering*, vol. 10, no. 1, p. 486, 2020.

[20] K. Yanai and 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)*, 2015, pp. 1–6.

DOI:10.4186/ej.2022.26.10.57

[21] L. Bossard, M. Guillaumin, and L. Van Gool, "Food-101 – mining discriminative components with random forests," in *European Conference on Computer Vision*, 2014.

[22] P. Kaur, K. Sikka, W. Wang, S. J. Belongie, and A. Divakaran, "Foodx-251: A dataset for fine-grained food classification," *CoRR*, vol. abs/1907.06167, 2019. [Online]. Available: http://arxiv.org/abs/1907.06167

[23] W. Min, L. Liu, Z. Wang, Z. Luo, X. Wei, X. Wei, and S. Jiang, "Isia food-500: A dataset for large-scale food recognition via stacked global-local attention network," in *Proceedings of the 28th ACM International Conference on Multimedia*, 2020.

[24] M. Jalal, K. Wang, S. Jefferson, Y. Zheng, E. O. Nsoesie, and M. Betke, "Scraping social media photos posted in kenya and elsewhere to detect and analyze food types," in *Proceedings of the 5th International Workshop on Multimedia Assisted Dietary Management*, 2019, pp. 50–59.

[25] X. Chen, H. Zhou, and L. Diao, "Chinesefoodnet: A large-scale image dataset for chinese food recognition," *CoRR*, vol. abs/1705.02743, 2017. [Online]. Available: http://arxiv.org/abs/1705.02743

[26] D. Sahoo, W. Hao, S. Ke, W. Xiongwei, H. Le, P. Achananuparp, E.-P. Lim, and S. C. H. Hoi, "Foodai: Food image recognition via deep learning for smart food logging," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 2260–2268. [Online]. Available: https://doi.org/10.1145/3292500.3330734

[27] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[28] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[29] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4510–4520.

[30] A. Howard, M. Sandler, B. Chen, W. Wang, L.-C. Chen, M. Tan, G. Chu, V. Vasudevan, Y. Zhu, R. Pang, H. Adam, and Q. Le, "Searching for mobilenetv3," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 1314–1324.

[31] M. Tan and Q. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. PMLR, 09–15 Jun 2019, pp. 6105–6114. [Online]. Available: https://proceedings.mlr.press/v97/tan19a.html

[32] M. Tan and Q. V. Le, "Efficientnetv2: Smaller models and faster training," 2021. [Online]. Available: https://arxiv.org/abs/2104.00298

[33] K. He, X. Zhang, S. Ren, and J. Sun, "Identity Mappings in Deep Residual Networks," in *Computer Vision – ECCV 2016*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Cham: Springer International Publishing, 2016, pp. 630–645.

[34] C. Termritthikun and S. Kanprachar, "Accuracy improvement of thai food image recognition using deep convolutional neural networks," in *2017 international electrical engineering congress (IEECON)*. IEEE, 2017, pp. 1–4.

[35] ——, "Nu-resnet: Deep residual networks for thai food image recognition," *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)*, vol. 10, pp. 29–33, 2018.

[36] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ser. ICML'10. Madison, WI, USA: Omnipress, 2010, p. 807–814.

[37] M. Tan, B. Chen, R. Pang, V. Vasudevan, and Q. V. Le, "Mnasnet: Platform-aware neural architecture search for mobile," *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2815–2823, 2019.

[38] T.-J. Yang, A. Howard, B. Chen, X. Zhang, A. Go, M. Sandler, V. Sze, and H. Adam, "Netadapt: Platform-aware neural network adaptation for mobile applications," in *The European Conference on Computer Vision (ECCV)*, September 2018.

[39] T. Ridnik, E. B. Baruch, A. Noy, and L. Zelnik-Manor, "Imagenet-21k pretraining for the masses," *CoRR*, vol. abs/2104.10972, 2021. [Online]. Available: https://arxiv.org/abs/2104.10972

[40] J. Puigcerver, C. Riquelme, B. Mustafa, C. Renggli, A. S. Pinto, S. Gelly, D. Keysers, and N. Houlsby, "Scalable transfer learning with expert models," *arXiv preprint arXiv:2009.13239*, 2020.

DOI:10.4186/ej.2022.26.10.57

[41] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8697–8710.

[42] A. Kolesnikov, L. Beyer, X. Zhai, J. Puigcerver, J. Yung, S. Gelly, and N. Houlsby, "Big Transfer (BiT): General Visual Representation Learning," in *Computer Vision – ECCV 2020*, A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, Eds. Cham: Springer International Publishing, 2020, pp. 491–507.

[43] G. Van Rossum and F. L. Drake, *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009.

[44] F. Chollet *et al.*, "Keras," https://github.com/fchollet/keras, 2015.

[45] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, no. 1, pp. 1–48, 2019.

[46] A. Mikołajczyk and M. Grochowski, "Data augmentation for improving deep learning in image classification problem," in *2018 international interdisciplinary PhD workshop (IIPhDW)*. IEEE, 2018, pp. 117–122.

[47] J. Wang, L. Perez *et al.*, "The effectiveness of data augmentation in image classification using deep learning," *Convolutional Neural Networks Vis. Recognit*, vol. 11, pp. 1–8, 2017.

[48] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo, "Cutmix: Regularization strategy to train strong classifiers with localizable features," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 6023–6032.

[49] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: http://arxiv.org/abs/1412.6980

[50] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 618–626.

DOI:10.4186/ej.2022.26.10.57

Sirawit Ittisoponpaisan received a BSc degree in Bio and Brain Engineering from the Korea Advanced Institute of Science and Technology (KAIST) in 2013. Later, he received the Royal Thai Government Scholarship covering full tuition fees for his postgraduate studies. He received an MSc degree in Bioinformatics and Theoretical Systems Biology in 2014 and a Ph.D. degree in Life Sciences Research in 2019 from Imperial College London. His Ph.D. research projects have gained the attention of many researchers and have been cited over 100 times, making him one of the three national winners of the award for an outstanding Ph.D. thesis, given by the National Research Council of Thailand (NRCT) in 2020. Currently, he is working as a lecturer and researcher at Prince of Songkla University, Thailand. His work covers mainly bioinformatics software development and data analysis.

Chayanat Kaipan was born in Songkla, Thailand in 1995. He is in his fourth year of studying computer science at Nakhon Si Thammarat Rajabhat University. His research interests include web development, machine learning, and deep learning. Mr.Kaipan received a silver medal at the first Science Symposium at the department of science and technology, Nakhon Si Thammarat Rajabhat University.

Somporn Ruang-on was born in Nakhon Si Thammarat, Thailand in 1976. He received his B.S. degree in computer science from Phetchaburi Rajabhat University in 1994. In 2003, he finished his M.S. in information technology from Sripatum University. He received a Ph.D. degree in quality information technology from Phetchaburi Rajabhat University in 2013. Since 2006, he has been an Assistant Professor at the Computer Science Department, Nakhon Si Thammarat Rajabhat University. Dr. Ruang-on research interests include software engineering, system development, and data science.

Rattayagon Thaiphan received her B.S. and M.S. degrees in mathematics and computer science from Prince of Songkla University in 1993 and 2000, respectively. She is now an Assistant Professor at the Computer Science Department, at Nakhon Si Thammarat Rajabhat University. Her research interests include programming and web development. Mrs. Thaiphan received the Best Student Paper Award at the 6th National Science and Technology Conference (NSCIC2021).

Kritaphat Songsri-in was born in Phuket, Thailand in 1991. He finished an MEng and a Ph.D. in computing from Imperial College London in 2011 and 2020, respectively. He has been a lecturer in the department of computer science at Nakhon Si Thammarat Rajabhat University since 2020. His research interests include machine learning, deep learning, and computer vision. He has published in and is a reviewer for multiple international conferences and journals such as IEEE Transactions on Image Processing and IEEE Transactions on Information Forensics & Security. Dr. Songsri-in was a recipient of the Royal Thai Government Scholarship covering his undergraduate and postgraduate degrees in 2010. He received the Best Student Paper Awards at the IEEE 13th International Conference for Automatic Face and Gesture Recognition (FG2018) and the 6th National Science and Technology Conference (NSCIC2021). In 2021, his Ph.D. thesis received an award from the National Research Council of Thailand (NRCT).