Fine-Grained Thai Food Image Classification

Thean Cheat Lim
Khoury College of Computer Sciences
Northeastern University
Boston, USA
lim.the@northeastern.edu

Wenlin Fang
Khoury College of Computer Sciences
Northeastern University
Boston, USA
fang.wenl@northeastern.edu

***Abstract*—This project aims to compare the performance of four different models for fine-grained Thai food image classification. We used a dataset of 15,770 images of 50 Thai dishes and trained three pre-trained models (DenseNet, ResNet, and SwinV2) and one custom CNN model based on VGG16. We experimented with freezing and unfreezing the pre-trained model weights and evaluated the models. We also used Grad-CAM to visualize the attention maps of the models and analyze their predictions.The goal of this project is to compare how well four different models can classify images of Thai food into 50 categories. The models are DenseNet, ResNet, SwinV2, and a custom CNN model that we built based on VGG16. We used a large dataset of 15,770 images of 50 Thai dishes and fine-tuned the pre-trained models for our task. We also trained the custom CNN model from scratch and evaluated its performance. We experimented with different settings of freezing and unfreezing the pre-trained model weights and reported the results. We also used Grad-CAM to visualize the attention maps of the models and analyze how they make predictions. We found that SwinV2 performed the best, achieving 92.92% test accuracy, while the custom CNN model performed the worst, achieving only 59% test accuracy. We also observed that each model learned and used slightly different features to distinguish between Thai dishes.**

***Index Terms*—DenseNet, ResNet, SwinV2, classification**

I. Introduction

Food image classification is a challenging and important task in computer vision that involves recognizing and labeling different types of food in images. It has gained significant attention in recent years due to its potential applications in various fields such as food recommendation, cultural education, and tourism. Fine-grained food classification, in particular, has become increasingly important as it involves distinguishing between visually similar dishes that differ in minor details, such as ingredients, cooking methods, or garnishes.

In this paper, we focus on fine-grained Thai food classification, which poses a unique challenge due to the diversity and complexity of Thai cuisine. We aim to compare the performance of four different vision models including DenseNet, ResNet, SwinV2, and our custom CNN model which is based on VGG16, and visualize their activation maps using Grad-CAM. We describe our methodology, experiments, results, and insights gained from the project.

II. Related Works

A. DenseNet

Among the various CNN architectures, DenseNet [1] has shown promising results in image classification tasks. DenseNet is a deep CNN architecture proposed by Huang et al. (2018) that addresses the vanishing gradient problem and improves the feature reuse in CNNs. DenseNet introduces a novel connectivity pattern, where each layer is connected to all subsequent layers in a feedforward fashion. This architecture allows for direct feature reuse and reduces the number of parameters, leading to better performance with fewer computational resources. As such, DenseNet achieves high accuracy and generalization performance on various image classification benchmarks.

B. VGG16

VGG16, introduced by Simonyan and Zisserman [2], is a deep convolutional neural network architecture widely used for image classification tasks. It consists of 13 convolutional layers followed by 3 fully connected layers, and it achieved state-of-the-art results on the ImageNet dataset at the time of its release.

One of the main advantages of VGG16 is its simplicity and uniformity. The architecture uses only 3×3 convolutional filters, which have been shown to be effective in capturing local patterns and details in images. Additionally, the uniform structure of the network makes it easy to implement and fine-tune for various tasks.

C. ResNet

ResNet was first proposed by He et al. [3] in 2016, with the aim of addressing the problem of vanishing gradients in deep neural networks. The key idea behind ResNet is the use of residual connections, which enable the network to learn residual functions instead of the traditional feed-forward functions. This approach has been shown to alleviate the problem of vanishing gradients, which can hinder the training of very deep neural networks.

In the paper, we can see that the ResNet models achieved better performance in image classification tasks compared to other deep networks such as VGG, which was one of the reasons why we wanted to use ResNet50 for our Thai food classification task.

D. SwinV2

Ze Liu et al. introduced the Swin Transformer [4], a transformer-based architecture for computer vision problems. It employs shifted windows in a hierarchical transformer

design to lower the computational cost of self-attention. The Swin Transformer has demonstrated cutting-edge performance on various computer vision benchmarks, including ImageNet.

Building on the success of the original Swin Transformer, the authors of the SwinV2 [5] offered numerous new strategies to overcome challenges in big vision model training and application. These strategies include cross-layer parameter sharing for training stability, feature map alignment across scales, and dynamic patch partitioning to close resolution gaps between pre-training and fine-tuning.

E. Grad-CAM

Grad-CAM (Gradient-weighted Class Activation Mapping) [6] was developed as a method for creating class activation maps for deep neural networks, with the goal of highlighting the portions of an image that are most essential for the network's prediction. The method takes the weighted sum of the gradients with respect to the class score from the gradient information flowing into the last convolutional layer of a convolutional neural network (CNN) to build a class activation map. This generates a coarse localization map that indicates the critical portions of the input image for network prediction.

Grad-CAM was applied to a variety of CNN architectures and applications and the findings demonstrated that Grad-CAM can not only provide interpretable explanations for the network's predictions, but it can also aid in the visualization of the network's internal representation and the identification of its weak places.

III. Methods

A. Dataset

The `THFOOD-50` [7], Fine-Grained Thai Food Image Classification Datasets, was used in this project. `THFOOD-50` contains 15,770 images of 50 famous Thai dishes. Some example data are shown in Figure 1.

We split the dataset into train, test, and validation datasets and store them on Huggingface Datasets repository (`thean/THFOOD-50`).

B. Data Preprocessing

To preprocess the dataset, we first calculated the mean and standard deviation of the entire dataset, which were used to normalize the images. The images were resized to 170 by 150 using the torchvision transforms and converted to tensors. Due to the large size of the dataset, we used PyTorch's `DataLoader` to calculate means and standard deviations for each batch, and then took the means of these values to get the final values for the whole dataset.

For the training data transformation and data augmentation, we applied `RandomResizedCrop` and `RandomHorizontalFlip` to generate 128 X 128 random crops from the training images on the fly. The images were then normalized using the previously calculated means and standard deviations.

For the validation and test data, we resized the images and applied a center crop to obtain 128 X 128 image boxes. We also normalized these images using the previously calculated means and standard deviations.

Fig. 1. Example of dishes from the `THFOOD-50` dataset.

C. Models

We fine-tuned three pre-trained models for our specific task, leveraging their ability to learn highly discriminative features from images. To perform Transfer Learning, we replaced the final classification layer to match the number of classes (i.e., 50 classes of Thai dishes) in our dataset and fine-tuned the network. This allows the network to learn specific features for our task while still benefiting from the pre-trained weights that capture generic features from large-scale datasets.

We tried two variations of Transfer Learning: (1) we froze pretrained model parameters and only trained the last layer, (2) we trained the entire models.

The pretrained models used were the DenseNet166 and ResNet50 models from `torchvision` and the `microsoft/swinv2-tiny-patch4-window8-256` checkpoint from HuggingFace.

For the custom model, we decided to use the well-established VGG16 architecture as a foundation. However, instead of using all 16 layers, we utilized a network consisting of 7 convolutional layers, 2 fully connected layers, and a final output layer with 50 neurons - one per class. We opted for a smaller model to explore how it would perform with fewer layers and gain a better understanding of these deep neural network models.

The first layer takes an input image of size 128 x 128 (3 RGB channels) and applies a convolution operation with a kernel size of 3x3, resulting in 64 output channels. Batch normalization and ReLU activation function are applied after the convolution layer.

Max pooling is performed after the second and fifth convolution layers to reduce the spatial dimensions of the output feature maps.

The fully connected layers have 4096 neurons each, with dropout regularization applied to avoid overfitting. Finally, the

TABLE I
MODEL PERFORMANCE

	Train Acc. %	Validation Acc. %	Test Acc. %
Custom Model	62.76	57.19	59.14
ResNet50*	92.19	79.69	80.50
DenseNet161*	75.27	76.25	73.06
SwinV2*	95.57	93.44	92.92

* The entire model parameters were trained/not frozen.

output layer has 50 neurons with a linear activation function, representing the 50 possible Thai food classes.

The forward method takes an input tensor and passes it through the layers in order, applying the necessary transformations along the way. The output is the class scores for each of the 50 classes. The input images to this model must be of size 128 by 128.

D. Grad-CAM

To visualize how our models are making the predictions, we used Grad-CAM to visualize what the models are paying attention to.

Based on the documentation of Pytorch Grad-CAM [8], for the ResNet50 model, we visualized the last residual block in the fourth layer of the network. For the Dense161 model, we visualized the last dense block in the network. For our custom model, we visualized the convolution layer in layer 7 of the model. Lastly, we visualized the activation map of the last LayerNorm layer of the SwinV2 model.

IV. EXPERIMENTS AND RESULTS

A. Custom Model

We developed a custom deep neural network for classifying Thai food based on the VGG16 architecture. The network consists of seven layers as described in the Models section.

During training, we observed that the initial accuracy of the model was only 2%. However, after five days of training, the accuracy increased to more than 50% within the first three days. The highest accuracy achieved was 59%, which suggests that further training did not significantly improve the model's performance.

B. Transfer Learning - Training only the Final Layer

We froze all model parameters except for the final layer and trained the models. The models could not perform better than 60% accuracy on the validation set. Notably, this variation of Transfer Learning trained faster, but the validation accuracy stagnated very quickly.

C. Transfer Learning - Training the Entire Model

After we had observed that the accuracy on the validation set had become stuck after freezing the model parameters, we decided to try unfreezing the parameters and continuing to train the model. This helped to significantly increase the model's accuracy. For example, the best accuracy that ResNet50 had achieved so far was 80.5% after we had unfrozen the model parameters. Refer to Table I for performance comparisons.

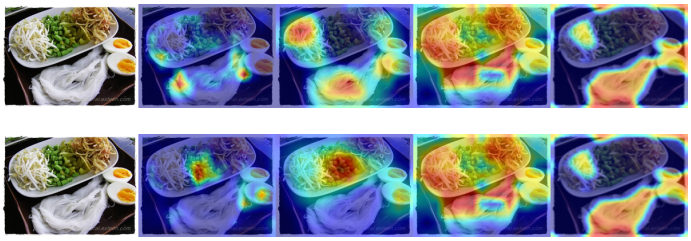


Fig. 2. Activation map for KhanomJeenNamYaKati (top) vs. the highest scoring category (bottom). The leftmost image is the input image, followed by activation maps from ResNet50, DenseNet161, SwinV2, and the custom model.

D. Grad-CAM Activation Maps

Visualization of models attentions during prediction were done. Interestingly, we found that each model learned and used slightly different discriminative features (high-intensity zones) to make predictions. In addition, we also compared the activation maps between the target class (ground truth) vs. the highest scoring category (prediction) – this tells how and where a model pays attention "incorrectly". For example, we can tell that ResNet and DenseNet focuses on the "incorrect" parts of the input image and made the incorrect predictions from Figure 2. Please proceed to Appendix for additional activation map visualizations.

V. DISCUSSION

Our experiments allowed us to observe and compare the performance of the four models on the fine-grained Thai food image classification task. The results revealed that the pre-trained models, namely DenseNet, ResNet, and SwinV2, had a clear advantage over the custom CNN model based on VGG16. This can be attributed to the fact that pre-trained models leverage knowledge from previous training on large-scale datasets, which enables them to extract more informative features from the images.

In terms of the differences between the pre-trained models, we found that SwinV2 outperformed both DenseNet and ResNet, suggesting that the transformer-based architecture is more effective in handling fine-grained image classification tasks. The superior performance of SwinV2 can be linked to its hierarchical design and shifted window operation, which result in more efficient computation and improved attention capabilities.

The DenseNet161 and ResNet50 models also performed well in the Thai food classification task, with ResNet achieving slightly better results. The difference in the performance could be due to many factors such as different training time, and the different architecture of the two models. ResNet50 may train faster than DenseNet161 because the dense concatenation in DenseNet causes a problem of requiring high GPU memory and more training time [9]. ResNet50 uses skip connections to avoid the vanishing gradient problem, which can make it easier for the model to learn representations of Thai food images. But DenseNet161 uses dense connections to encourage feature

reuse, which can be beneficial in other types of tasks but may not be as effective for Thai food image classification.

While the ResNet50 and DenseNet161 models achieved high test accuracy (>70%), we noted that their performance did not improve despite additional training. This presents a challenge for the future project, and we are keen to explore strategies to improve model performance when it plateaus.

Additionally, we found that training only the final layer of pretrained models did not yield satisfactory results. This fits our expectations as none of the models were trained to extract fine-grained discriminative features between Thai dishes. Therefore, training the entire network was necessary to achieve the desired classification accuracy.

Regarding the custom CNN model based on VGG16, its performance was lower than the pre-trained models. The custom model experiment highlights the importance of the number of layers in a neural network and the impact of increasing the number of training epochs. While our custom model did not perform as well as the pre-trained VGG16 model from PyTorch, we anticipate the latter would demonstrate significantly better performance given its larger capacity. Our findings suggest that decreasing the number of layers in a model may lead to decreased performance. Overall, our custom model represents a valuable learning experience for building and training a deep neural network from scratch.

Lastly, to analyze the performance of the models, we also visualized their attention maps using Grad-CAM. These visualizations provided valuable insights into the decision-making processes of the models, helping us to understand which regions of the images the models focused on during classification. This information could further refine the models and improve their performance. More, this informative and enjoyable approach enhanced our understanding of the inner workings of deep neural networks.

VI. Summary

We presented a comparative study of four different models for fine-grained Thai food image classification: DenseNet, ResNet, SwinV2, and a custom CNN model based on VGG16 architecture. We used a large-scale, high-quality dataset of 15,770 images of 50 Thai dishes. We experimented with different settings of freezing and unfreezing the pre-trained model weights and reported the results. We showed that SwinV2 outperformed the other models by a large margin, achieving 92.92% test accuracy. We also demonstrated that the custom model was insufficient to capture the complexity of Thai food, achieving only 59% test accuracy. Furthermore, we applied Grad-CAM to visualize the attention maps of the models and interpret their predictions. We found that each model learned and used slightly different discriminative features to make predictions, and some models failed to focus on the relevant parts of the input image and made incorrect predictions.

References

[1] G. Huang, Z. Liu, L. van der Maaten and K. Q. Weinberger, "Densely Connected Convolutional Networks," arXiv preprint arXiv:1608.06993, 2018.

[2] G. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," in Proceedings of the International Conference on Learning Representations (ICLR), 2015.

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

[4] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin and B. Guo, "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows," in Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), 2021, pp. 3980-3989.

[5] Z. Liu, H. Hu, Y. Lin, Z. Yao, Z. Xie, Y. Wei, J. Ning, Y. Cao, Z. Zhang, L. Dong, F. Wei, and B. Guo, "Swin Transformer V2: Scaling Up Capacity and Resolution," in Proceedings of the IEEE/CVF International Conference on Computer Vision and Pattern Recognition (CVPR), 2022, pp. 7223-7232.

[6] L. Selvaraju, A. Das, R. Vedantam, M. Cogswell, D. Parikh, and D. Batra, "Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization," in Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 618–626.

[7] 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, no. 2-6, pp. 63-67, 2017.

[8] J. Gildenblat et al., "PyTorch library for CAM methods," GitHub, 2021. [Online]. Available: https://github.com/jacobgil/pytorch-grad-cam.

[9] J. Zhang, K. Huang, J. Zhang, and Q. Zhu, "ResNet or DenseNet? Introducing Dense Shortcuts to ResNet," in Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), 2021, pp. 2307-2316.

[10] S. Saha, "Building an Image Classification Model with PyTorch from Scratch," Medium, 2021. [Online]. Available: https://medium.com/bitgrit-data-science-publication/building-an-image-classification-model-with-pytorch-from-scratch-f10452073212

[11] Y. Lecun, "Convolutional Network: How to choose output channels, number, stride, and padding," Stack Exchange, 2019. [Online]. Available: https://stats.stackexchange.com/questions/380996/convolutional-network-how-to-choose-output-channels-number-stride-and-padding/381032

[12] Pluralsight, "Image Classification with PyTorch," Pluralsight, 2021. [Online]. Available: https://www.pluralsight.com/guides/image-classification-with-pytorch

[13] Microsoft, "Train a machine learning model with PyTorch on Windows," Microsoft Learn, 2021. [Online]. Available: https://learn.microsoft.com/en-us/windows/ai/windows-ml/tutorials/pytorch-train-model

[14] A. Khan, "Implementing VGG from scratch using PyTorch," Paperspace, 2018. [Online]. Available: https://blog.paperspace.com/vgg-from-scratch-pytorch/

[15] R. Singh, "Understanding Dimensions in PyTorch," Towards Data Science, 2020. [Online]. Available: https://towardsdatascience.com/understanding-dimensions-in-pytorch-6edf9972d3be

[16] V. Saini, "Saving and Loading Your Model to Resume Training in PyTorch," Medium, 2020. [Online]. Available: https://medium.com/analytics-vidhya/saving-and-loading-your-model-to-resume-training-in-pytorch-cb687352fa61

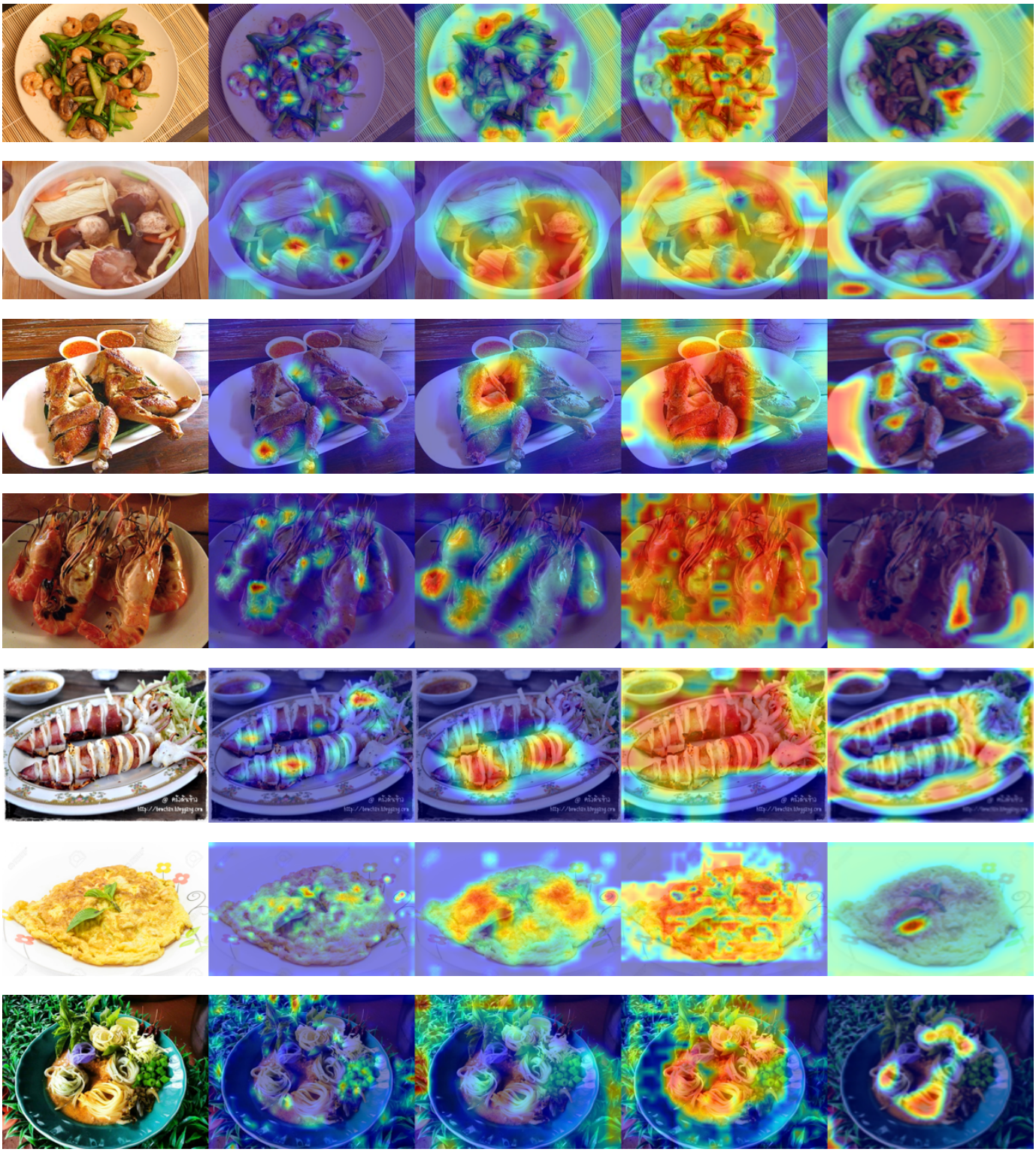
[17] S. Saha, "A Comprehensive Guide to Image Augmentation using PyTorch," Towards Data Science, 2021. [Online]. Available: https://towardsdatascience.com/a-comprehensive-guide-to-image-augmentation-using-pytorch-fb162f2444be

[18] S. Mallick, "Cropping an Image using OpenCV," LearnOpenCV, 2021. [Online]. Available: https://learnopencv.com/cropping-an-image-using-opencv/

[19] Hugging Face, "Fine-tuning a model on an image classification task," GitHub, 2021. [Online]. Available: https://github.com/huggingface/notebooks/blob/main/examples/image_classification.ipynb

[20] Hugging Face, "Image Classification," Hugging Face Docs, 2021. [Online]. Available: https://huggingface.co/docs/transformers/tasks/image_classification

APPENDIX



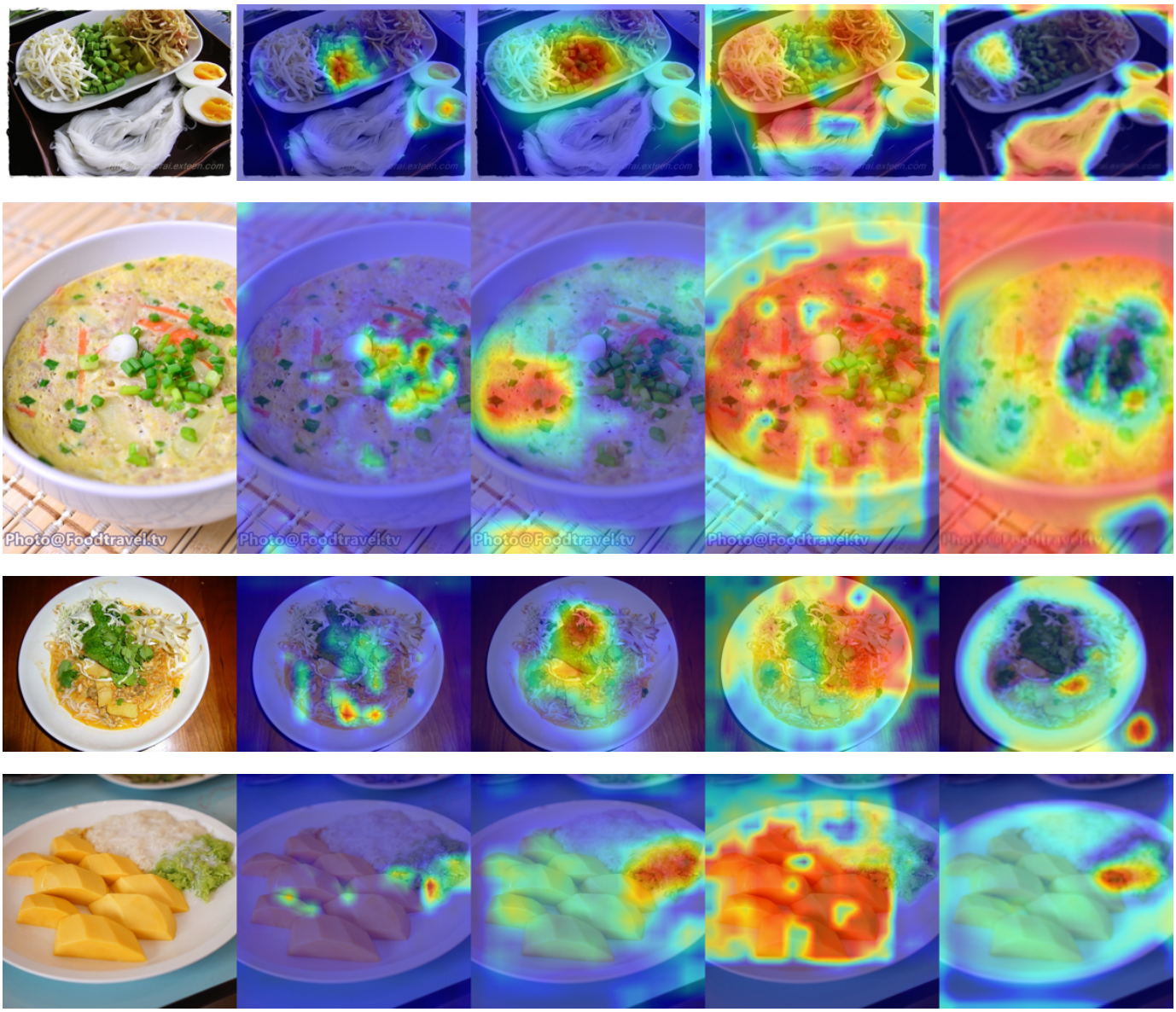
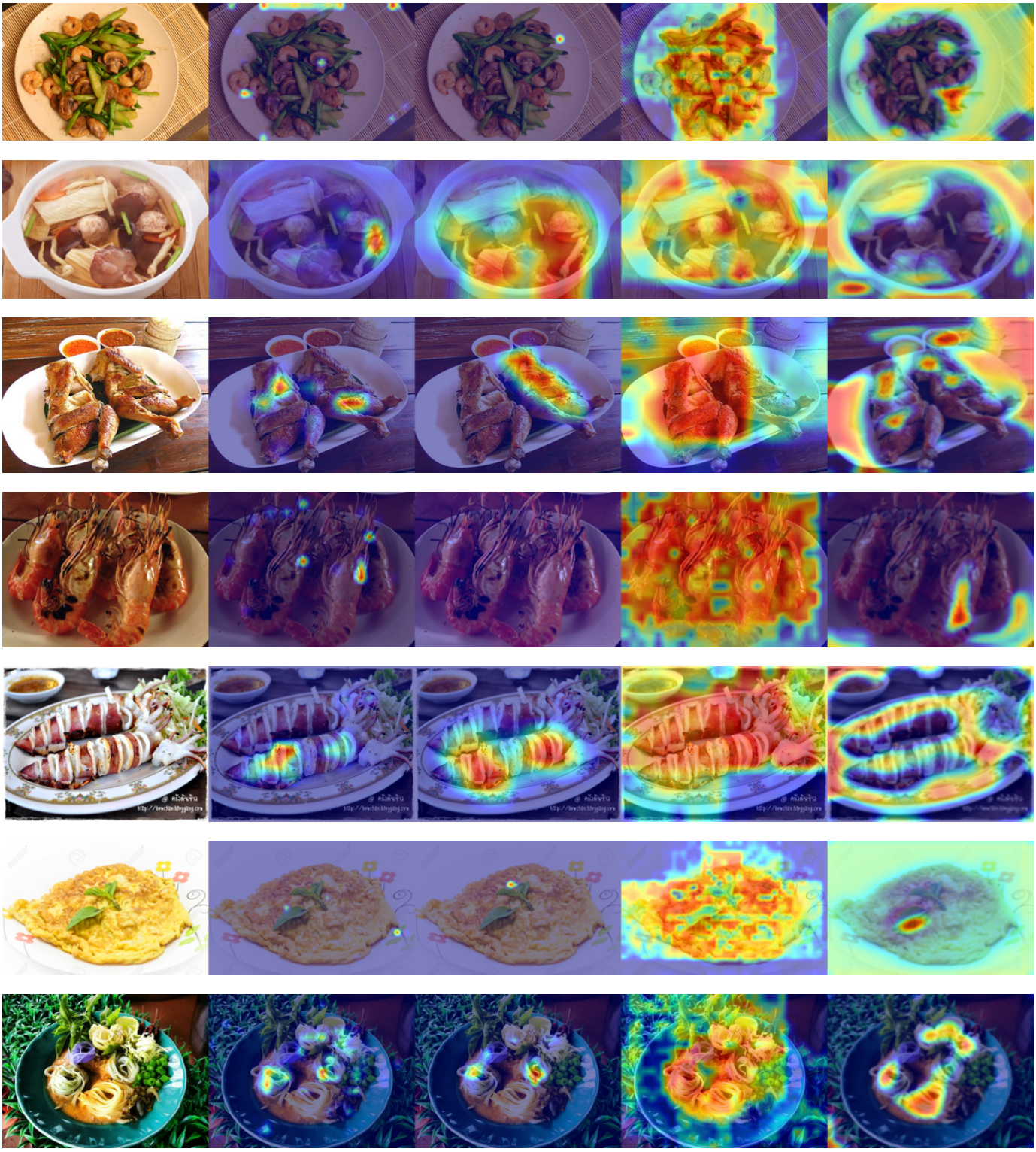


Fig. 3. Activation map of the highest scoring category. The leftmost image is the input image, followed by activation maps from ResNet50, DenseNet161, SwinV2, and the custom model.



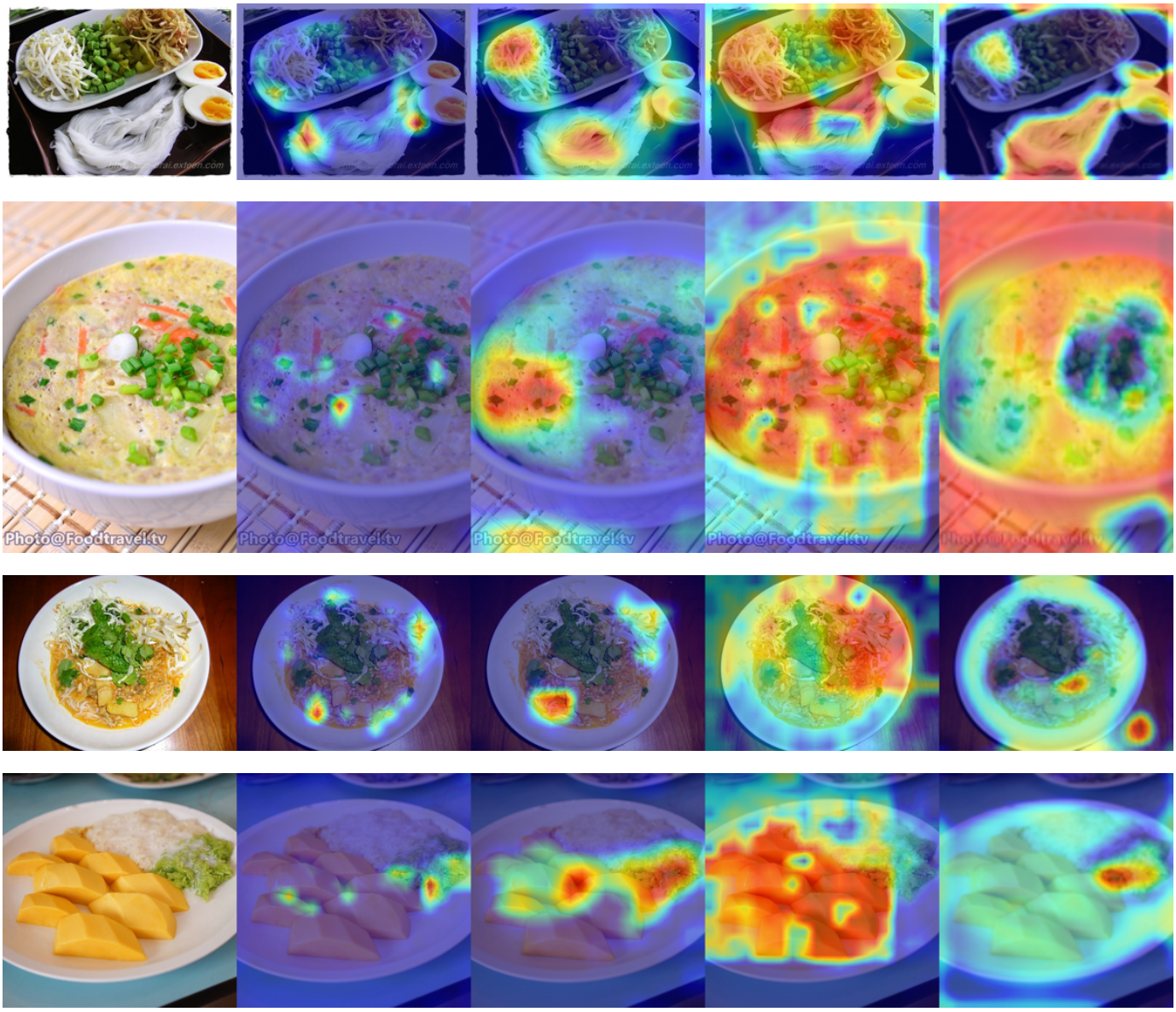


Fig. 4. Activation map of the target category. The leftmost image is the input image, followed by activation maps from ResNet50, DenseNet161, SwinV2, and the custom model.