ENSEMBLING DEEP LEARNING MODELS FOR MULTI-LABEL FOOD IMAGE RECOGNITION

Orfeas Gkonis
Nikolaos Laoutaris
Vasileios Kesopoulos

International Hellenic University
MSc in Data Science

ABSTRACT

Image classification is a complex process that may be affected by many factors. This report examines current practices, problems, and prospects of food image classification. Non-parametric classifiers such as neural networks have become increasingly popular in image classification problems, that said we implemented many different pretrained models with different architectures in-order to solve a multi label classification problem through a competition in Kaggle. Finally, we have also summarized the main analysis and discussion in this article, as well as introducing some of the current trends.

Index Terms—Multi-label classification, Deep learning, Food recognition, Image classification, Vision Transformers, Model ensembling, Transfer learning, Neural networks

1. DATA AND PROBLEM DESCRIPTION

Multi-label image classification is the task of predicting a set of labels corresponding to objects, attributes or other entities present in an image. Our data consists of 41.000 images, where 40.000 of them are for training and 1.000 for testing. Our task is to build/fine tune a Neural Network to detect all the labels present in each of the test images.

Each image in the dataset might contain multiple food items, making this a challenging classification task. The labels span 498 unique categories, ranging from common ingredients to specific dishes. Class imbalance is an issue, as some foods appear frequently while others are rare. With over 40,000 high-resolution images, training deep models is computationally expensive and experimenting freely with many architectures or hyperparameters is not feasible. This required us to be selective and strategic with our choices.

The data was given to us through a Kaggle competition that started in March 2025 and ended in June 2025. There was a public test set which was 51% of the test set (1000 images), but the winner was determined by the best private score in the private set which was 49% of the test set.

2. DISCRIPTION OF MODELS USED IN EACH SUBMISSION

2.1. Pretrained Architectures

To tackle the complexity of multi-label food classification, we employed a variety of pretrained convolutional and transformer-based models as our base learners. These models differ in architectural design, receptive field behavior and feature representation strategies, offering a diverse backbone.

Below is a high-level overview of the main architectures used in our experiments:

- Swin Transformer (V1, V2): Vision transformers that process images in non-overlapping windows with shifted window attention, enabling hierarchical feature learning.
- ResNet (18, 34, 50): Residual networks based on stacked convolutional layers with identity shortcut connections, enabling stable training of deeper models. The deeper versions use bottleneck blocks with 1×1 convolutions.
- Inception (V2, V3): Utilizes parallel convolutional paths (Inception modules) with different kernel sizes to capture multi-scale features efficiently, along with factorized convolutions to reduce computation.
- ConvNeXt: A convolutional architecture inspired by modern transformer design choices, such as large kernels, layer normalization and GELU activations.
- EfficientNet: Scales depth, width and resolution uniformly using a compound scaling method for optimal performance with fewer parameters.
- Vision Transformer (ViT): Its architecture comprises of several key stages: Image patching and embedding, positional encoding, transformer encoder, classification head (MLP).

2.2 Modifications

Building on the pretrained architectures described above, we experimented with several standard and advanced techniques to tailor them to our task.

Custom Classification Head
Each model's original classification layer was replaced with a new fully connected head, sized appropriately for the 498 food categories. For some experiments we tried using even deeper final classifier architectures, sometimes including Dropout and ReLU layers along with the fully connected ones.

Layer Freezing and Unfreezing
We experimented with three strategies:
· Partial unfreezing from the start: keeping early layers frozen to preserve low-level pretrained features while fine-tuning deeper ones.
· Gradual unfreezing ("warm start"): beginning with only the classifier unfrozen and progressively unfreezing higher layers.
· Full unfreezing: retraining all layers of the model for a specified number of epochs.

Data Augmentation
To improve generalization, we experimented with applying image augmentations during training. These include:
· Color Jitter: Randomly adjusts brightness, contrast and saturation to simulate varying lighting conditions.
· Horizontal Flip: Horizontally flips the image to augment left/right symmetry.
· Vertical Flip: Vertically flips the image.
· Gaussian Blur: Applies light blurring to simulate low-focus or noisy conditions.
· Random Resized Crop: Zooms into a random crop of the image while preserving aspect ratio and image dimensions.
· Random rotation: Randomly rotates the image by a certain degree range, which helps recognize the existing objects in the image regardless of their orientations.
· Random Perspective: Distorts the perspective of input images by shifting their corner points, simulating a 3D-like transform.
· Random Affine: Applied a combination of geometric transformations to an image, simulating how it might appear from different viewpoints or distortions.
· Random Auto contrast: Randomly adjust the contrast of an image or images by a random factor.
· TrivialAugment: Applies a single transformation, randomly selected from a small pool of operations, with fixed intensity.
· AutoAugment: Uses a learned policy that applies a sequence of transformations selected from a large set, with varying intensities.
· RandAugment: Randomly applies a set of transformations in a sequential manner with a specified magnitude (reduced search space and computational cost compared to AutoAugment).

To further improve inference, we tried using Test-Time Augmentation (TTA), applying several augmentations to each test image and aggregating the model's predictions across these augmented views. If a prediction is consistent across multiple transformed versions of the same image, it is more likely to be true.

Loss Function
We used binary cross-entropy with logits to handle the multi-label nature of the task and to avoid the need for manual activation during training.

Optimizer
We used the Adam and AdamW optimizers with differential learning rates across layers. Earlier backbone layers were often assigned a lower learning rate than the deeper ones.

Learning Rate Scheduling
Three learning rate schedulers were explored:
· Cosine Annealing: gradually reduces the learning rate following a cosine decay curve.
· Step: reduces the learning rate by a fixed factor every n epochs.
· Reduce On Plateau: monitors a chosen validation metric (e.g., micro F1) and reduces the learning rate when it stops improving significantly.

Early Stopping
To prevent overfitting and reduce unnecessary training time, we incorporated early stopping based on metrics such as the validation F1-score. Training was halted if no improvement was observed after a fixed number of consecutive epochs (patience). This ensured that models did not over-train once performance plateaued, and helped select the best-performing checkpoint for each configuration.

Gradient Clipping
To prevent exploding gradients and stabilize training, we tried applying gradient clipping, a technique that limits the magnitude of gradients by scaling them down if they exceed a specified threshold.

Weight decay
Weight decay, often referred to as L2 regularization, although it is not the same, is a common regularization technique for neural networks. It helps the neural networks to learn smoother, simpler functions which most of the time generalize better compared to spiky, noisy ones. It has been observed that lower weight decay values help models prevent underfitting while higher values prevent models from overfitting. Weight decay played an important role to prevent over/underfitting of our best model's submission (ViT).

3. COMPARATIVE EXPERIMENTS AND RESULTS

3.1 Single Model Performance

The following table summarizes some of our key experiments with different backbones, training durations and fine-tuning strategies.

Model backbone	Epochs	Val. F1	Public score	Unfreeze	Learning Rate	Scheduler
Swin V1	10	0.5267	0.50809	-	10^{-3}	StepLR(5, 0.5)
Swin V1	6	0.5619	0.54353	Phase 4	Head: 10^{-3}, Phase 4: 10^{-4}	StepLR(5, 0.5)
Swin V1	6	0.5762	0.56130	Phases 3, 4	Head: 10^{-3}, Phase 4: 10^{-4}, Phase 3: 10^{-5}	StepLR(5, 0.5)
Swin V1	33	0.6014	0.54300	Full	Head: $5*10^{-4}$, Phase 2: $2*10^{-5}$	CosineAnnealing (30, 10^{-7})
Swin V2	7	0.5704	0.53080	Phase 4	Head: 10^{-3}, Phase 4: 10^{-4}	StepLR(5, 0.5)
Swin V2	8	0.5943	0.53577	Phases 3, 4	Head: 10^{-3}, Phase 4: 10^{-4}, Phase 3: 10^{-5}	ReduceOnPlateau (1%, 0.5)
Resnet 50	17	0.50110	0.47359	Phase 4	10^{-3}	StepLR(5, 0.5)
Resnet 50	17	0.5113	0.49680	Phases 3, 4	10^{-3}	StepLR(5, 0.5)
Resnet 50	16	0.5141	0.47687	Phases 3, 4	10^{-3}	StepLR(5, 0.5)
Resnet 34	9	0.34934	0.32397	-	10^{-3}	-
ConvNeXt	13	0.4729	-	Phase 7	Head: 10^{-3}, Phase 7: 10^{-4}	StepLR(5, 0.5)
Inception V3	6	0.4147	-	Phases 6e, 7	Head: 10^{-3}, Phase 7: 10^{-4}, Phase 6e: 10^{-5}	StepLR(5, 0.5)
ViT	21	0.5821	0.56545	Till 4th block	Head: 10^{-3}, Bottom Blocks: $5 \cdot 10^{-5}$ Mid blocks: 10^{-5}, Top Blocks: $5 \cdot 10^{-6}$	StepLR
ViT	24	0.599	0.5589	Full unfreeze	Head: 10^{-3}, Bottom Blocks: $5 \cdot 10^{-5}$ Mid blocks: 10^{-5}, Top Blocks: $5 \cdot 10^{-6}$	StepLR
ViT	13	0.56	0.54756	Till 6th block	Head: 10^{-3}, 11th Block:10^{-4}, Rest: $5 \cdot 10^{-5}$	StepLR

We observed that partial unfreezing, particularly targeting deeper layers, consistently improved performance. In contrast, models like ResNet50 showed only moderate gains from deeper heads or broader unfreezing, suggesting diminishing returns. Swin emerged as our top performer, achieving the best validation and submission scores when fine-tuned selectively. The poor performance of shallow models like ResNet34 highlights the importance of feature depth in this complex classification task. Some models, such as ConvNeXt and Inception, underperformed possibly due to architectural mismatch or suboptimal hyperparameters, and were not included in the ensembles. We also experimented on full-unfreezing the model as a two-stage training strategy on a SwinV1 architecture, where initially, only the classification head was trained for just 3 epochs achieving a steady f1 score of ~0.20 and then unfreezing all the layers of the model. Even though unfreezing the entire model is computationally intensive and more prone to catastrophic forgetting, we observed from this specific experiment a significant increase in f1 score of ~0.50 in the following epoch, reaching ~60 after completion (33 total epochs).

3.2 Ensemble Comparisons

The next table presents results from various ensemble configurations. Note that we chose to showcase only the portion that gave us the most useful insights, out of the great number of ensembles we put to the test.

Ensemble models	Conflict rate	Submission score
SwinV1 (x5), SwinV2 (x2), ViT (x3)	0.56 %	0.59204
SwinV1 (x6), SwinV2 (x2)	0.51 %	0.58837
SwinV1 (x3), SwinV2 (x1)	0.36 %	0.57422
SwinV1 (x3)	0.27 %	0.56351
Resnet50 (x3), SwinV1 (x3)	0.53 %	0.56529
Resnet50 (x2), SwinV1 (x3)	0.47 %	0.56872
Resnet50 (x3), SwinV1 (x2)	0.50 %	0.54172
Resnet50 (x1), SwinV1 (x3)	0.39 %	0.56482
Resnet50 (x2), SwinV1 (x2)	0.65 %	0.53500
Resnet50 (x3)	0.06 %	0.51681

We found that ensembles dominated by Swin models achieved the best results overall. Adding ViTs to the mix provided a slight performance boost over Swin-only ensembles. In contrast, ensembles with a heavier presence of ResNet50 models performed worse, even when their internal conflict rates were low. Notably, the pure ResNet50 ensemble had the lowest conflict rate but also the lowest score, highlighting that high agreement between weak models does not guarantee strong performance. These findings support the idea that architectural diversity within an ensemble is more beneficial than mere prediction consistency.

3.3 Final Results

Best Individual Model Configuration

Our strongest single model was a fine-tuned Vision Transformer, obtained via the timm library (vit_base_mci_224.apple_mclip) with pre-trained various datasets crawled from the web or publicly available datasets such as YFCC100M. The input image resolution was standardized to 224×224. Based on EDA, we applied dataset-specific normalization. The set of augmentations we applied during training consisted of Color Jitter, Random Horizontal Flip, Random Vertical Flip, Gaussian Blur and Random Resized Crop, Random Perspective, Random Affine, Random Autocontrast.

We unfroze the model to fine-tune all of the blocks except the first three blocks, alongside the custom classification head, which was a simple fully connected layer. We used a layer-wise learning rate and weight decay schedule: $(10^{-3}, 10^{-4})$ for the head, $(5 \cdot 10^{-5}, 5 \cdot 10^{-5})$ for 11th block,$(10^{-5}, 10^{-5})$ for the 10th to the 6th block and $(5 \cdot 10^{-6}, 5 \cdot 10^{-6})$ for the 5th and 4th block. After achieving a 0.5821 validation high score using these settings, the model was trained for 8 epochs.

After post-processing, we selected the optimal classification threshold of 0.5. This model achieved a satisfying 0.56454 public leaderboard score.

Best Ensemble Configuration

Our best-performing ensemble combined ten models: five SwinV1, two SwinV2, and three ViT models. This diverse combination outperformed all others we tested, achieving a submission score of 0.59204, the highest among our configurations. While it ultimately earned us second place in the competition, it marked our most effective strategy.

- SwinV1 (×5): These models were fine-tuned with varying depths (e.g., unfreezing stage 4 vs. stages 3–4) and training strategies. Two of them incorporated advanced augmentations like TrivialAugment and AutoAugment.
- SwinV2 (×2): Both used larger input resolutions (256×256) and were fine-tuned on deeper layers.
- ViT (×3): These were fine-tuned using consistent training setups and resolution of 224×224, with relatively high thresholds to reduce over-prediction.

All models were pretrained and used normalization consistent with our EDA. We used weighted voting to aggregate predictions across the ensemble.

Interestingly, no fallback mechanism was applied post-ensemble. While individual models used fallbacks (e.g., selecting the most confident label when no thresholded label passed), we found that allowing some zero-label predictions at the ensemble level improved overall performance.

This final ensemble had a conflict rate of 0.56%, meaning 2,780 out of 498,000 label decisions differed across models. This small amount of disagreement reflects the value of architectural diversity in ensembling, where varied perspectives lead to a more robust overall prediction.

4. CONCLUSIONS

This project provided valuable hands-on experience in applying deep learning to a challenging multi-label image classification task. Beyond raw performance, it offered key insights into model development and evaluation.

A central early realization was the importance of transfer learning. With limited data and a large label space, using pre-trained models was essential while training from scratch was never viable. Adapting strong feature extractors proved to be the only practical path to competitive results.

We also found data augmentation crucial for generalization. Even modest transformations improved robustness and reduced overfitting.

Optimization played a major role too. Warm-start strategies and fine-tuning learning rates per layer with partially frozen backbones outperformed global adjustments.

Post-submission exploratory analysis revealed that some models occasionally failed to assign any labels. This led to fallback prediction mechanisms, reinforcing the value of error analysis and iterative refinement.

Ensembling eventually emerged as a powerful method. Individually mediocre models still contributed positively if their errors were diverse. This architectural and training variety enabled ensembles to outperform any single model, highlighting the value of diversity in model design.

In hindsight, relying solely on the competition's test set was a bad decision. A better approach would have been to split the original training data into distinct train, validation, and test subsets, enabling more flexibility for post-hoc experimentation. This became apparent as our ensemble grew, incorporating models trained on the entire training set. Without a separate held-out set, we lacked a reliable way to evaluate individual model contributions (such as through a Leave-One-Out approach) or to make informed decisions about which models to retain in the ensemble.

We also faced broader ML challenges. Performance gains eventually plateaued, likely due to dataset limitations. More labeled data or improved curation would be needed for further improvement. Hardware constraints shaped many of our decisions, limiting model size and training duration and reminding us that real-world ML is often defined more by practicality than theory.

Throughout this journey, we learned the importance of disciplined experimentation. Meticulous documentation of architectures, hyperparameters, thresholds and results, was vital in navigating the complexity of our setup. Without it, drawing conclusions or tracking progress was impossible.

Finally, interpretability remained a challenge. Even our best models behaved as black boxes, highlighting that high performance doesn't guarantee understanding, a gap that remains central in modern deep learning.

Taken together, these insights highlight the multi-faced nature of machine learning: a blend of science, engineering and craft, where success depends on both algorithmic choices and practical realities of implementation.

12. REFERENCES

[1] S. Rokhva and B. Teimourpour, “Accurate & real-time food classification through the synergistic integration of EfficientNetB7, CBAM, transfer learning, and data augmentation,” Food and Humanity, vol. 4, p. 100492, May 2025, doi: 10.1016/J.FOOHUM.2024.100492.

[2] G. Deng, D. Wu, and W. Chen, “Attention Guided Food Recognition via Multi-Stage Local Feature Fusion,” Computers, Materials and Continua, vol. 80, no. 2, p. 1985, Aug. 2024, doi: 10.32604/CMC.2024.052174.

[3] L. Jha, S. Miller, and J. Wu, “Comparative Analysis of VGG16 and ResNet-50 Architectures for Food Image Classification: A Deep Dive Into Performance and Practical Implications,” ISMSIT 2024 - 8th International Symposium on Multidisciplinary Studies and Innovative Technologies, Proceedings, 2024, doi: 10.1109/ISMSIT63511.2024.10757277.

[4] X. Gao, Z. Xiao, and Z. Deng, “High accuracy food image classification via vision transformer with data augmentation and feature augmentation,” J Food Eng, vol. 365, p. 111833, Mar. 2024, doi: 10.1016/J.JFOODENG.2023.111833.

[5] H. Bae and E. Baek, “Food image classification based on the Swin Transformer model,” in 2024 Joint 13th International Conference on Soft Computing and Intelligent Systems and 25th International Symposium on Advanced Intelligent Systems (SCIS&ISIS), 2024, pp. 1–3. doi: 10.1109/SCISISIS61014.2024.10759949.

[6] S. E. Sreedharan, G. N. Sundar, and D. Narmadha, “NutriFoodNet: A High-Accuracy Convolutional Neural Network for Automated Food Image Recognition and Nutrient Estimation,” Traitement du Signal, vol. 41, no. 4, pp. 1953–1965, Aug. 2024, doi: 10.18280/ts.410425.