Final Report

Student Name: Yinhao Liu

Net ID: yl11221

Class: ECEGY6143, Section INET-Fall 2023

Professor: Dr. Fraida Fund

1. Models

In the model section, I will compare and evaluate the optimization metrics and operational metrics for the previous model, VGG16, as well as two commonly used models in image recognition: ResNet101V2 and MobileNetV2. In all the comparisons below, the loss function used is 'sparse_categorical_crossentropy', which is commonly employed as a loss function for handling multi-classification problems.。

a) Previous Model:

Optimization Metrics Evaluation:

Accuracy on the evaluation set: 0.7622

Identification of food categories with poor performance:

1. Bread 2. Dairy product 3. Dessert 4. Seafood

Operational Metrics Evaluation:

Typical response time: 0.63s

Transaction rate: 15.77

Disk space usage: 9.27 MB

Memory usage when the model is loaded for inference: 311.5MB

1.2 Submitted Models

ResNet101V2:

Base Model and Parameters:

The following are the parameter counts for ResNet101V2:

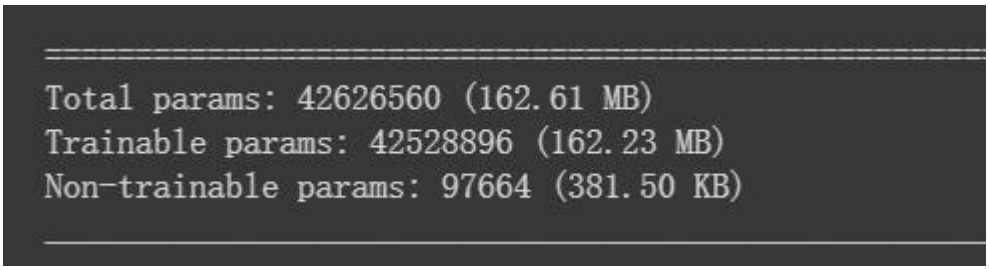


Fig. 1 Params number of ResNet101V2

Data Augmentation Transformations:

Scaling: Scaling the pixel values of the image to a range between 0 and 1

Rotation: Randomly rotating the image within a given range of degrees.

Zooming: Randomly zooming into the image.

Translation: Randomly moving the image in horizontal and vertical directions.

Shearing: Applying random shear transformations.

Horizontal Flipping: Randomly flipping the image horizontally.

Training of Classification Head:

Optimizer: Adam

Loss Function: Sparse Categorical Crossentropy

Metrics: Accuracy

Number of Epochs: 20

Batch Size: 32

Validation Accuracy at the End of This Stage: 0.8249

Fine-Tuning:

Layers Unfrozen for Fine-Tuning: All layers in the base model except the last five were frozen.

Number of Epochs for Fine-Tuning: 5

Optimizer for Fine-Tuning: Adam with a learning rate of 1e-5.

Batch Size: 32

Validation Accuracy at the End of This Stage: 0.8598

Final Accuracy on Evaluation Set： Accuracy: 0.8473, Loss: 0.4586.

```
model.evaluate(evaluation_gen)

105/105 [==============================] - 20s 188ms/step - loss: 0.4586 - accuracy: 0.8473
[0.4586409628391266, 0.8473259806632996]
```

Fig. 2 Accuracy & Loss of ResNet101V2

Accuracy Improvement Steps:

Steps That Do Not Affect Response Time:

1. Data Augmentation
2. Early Stopping
3. Sparse Categorical Crossentropy Loss

Steps That Could Affect Response Time:

1. Unfreezing Layers for Fine-Tuning
2. Adam Optimizer with Reduced Learning Rate for Fine-Tuning

Rationale for Decisions:

1. **Early Stopping:** This technique ensures the model doesn't learn noise or irrelevant patterns, which could degrade its performance on unseen data. It does not affect inference time as it's only a training phase adjustment.
2. **Sparse Categorical Crossentropy Loss:** This loss function efficiently calculates the difference between the predicted probabilities and the actual class, directly impacting the model's ability to learn accurate classifications.
3. **Adam Optimizer with Reduced Learning Rate for Fine-Tuning:** A reduced learning rate is used to prevent the model from diverging during the fine-tuning process. The Adam optimizer is chosen for its efficiency in handling sparse gradients and adaptive learning rate capabilities, but changes in its parameters can affect convergence time during inference.

MobileNetV2:

The following are the parameter counts for MobileNetV2:

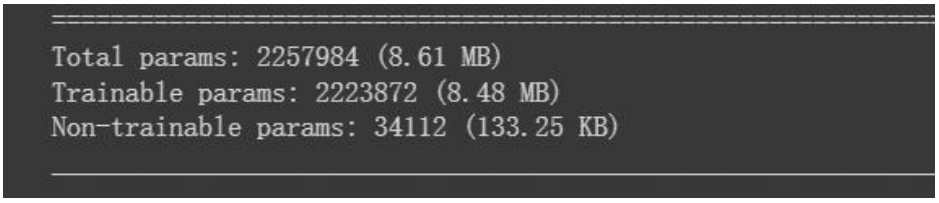


Fig. 3 Params number of MobileNetV2

Data Augmentation Transformations:

Scaling: Scaling the pixel values of the image to a range between 0 and 1

Rotation: Randomly rotating the image within a given range of degrees.

Zooming: Randomly zooming into the image.

Translation: Randomly moving the image in horizontal and vertical directions.

Shearing: Applying random shear transformations.

Horizontal Flipping: Randomly flipping the image horizontally.

Training of Classification Head:

Optimizer: Adam

Loss Function: Sparse Categorical Crossentropy

Metrics: Accuracy

Number of Epochs: 20

Batch Size: 32

Validation Accuracy at the End of This Stage: 0.8037

Fine-Tuning:

Layers Unfrozen for Fine-Tuning: All layers in the base model except the last five were frozen.

Number of Epochs for Fine-Tuning: 5

Optimizer for Fine-Tuning: Adam with a learning rate of 1e-5.

Batch Size: 32

Validation Accuracy at the End of This Stage: 0.8151

Final Accuracy on Evaluation Set: Accuracy: 0.8378, Loss: 0.5118.

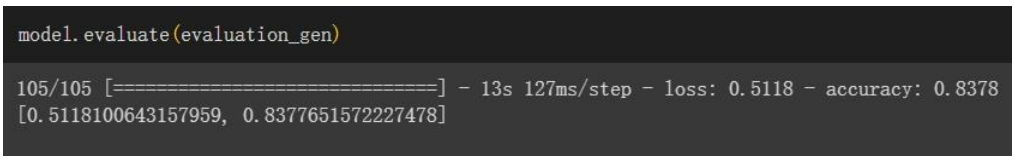


Fig. 4 Accuracy & Loss of MobileNetV2

Accuracy Improvement Steps:

Steps That Do Not Affect Response Time:

1. Data Augmentation
2. Early Stopping
3. Sparse Categorical Crossentropy Loss

Steps That Could Affect Response Time:

1. Unfreezing Layers for Fine-Tuning
2. Adam Optimizer with Reduced Learning Rate for Fine-Tuning

Rationale for Decisions:

1. **Early Stopping:** This technique ensures the model doesn't learn noise

or irrelevant patterns, which could degrade its performance on unseen data. It does not affect inference time as it's only a training phase adjustment.

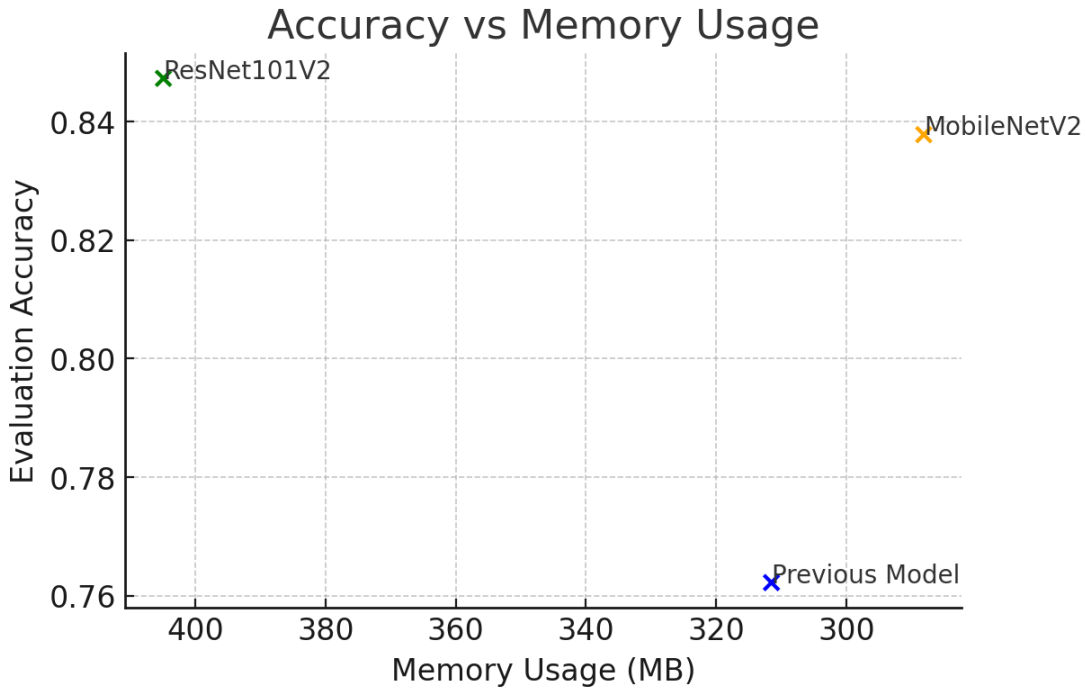
2. **Sparse Categorical Crossentropy Loss:** This loss function efficiently calculates the difference between the predicted probabilities and the actual class, directly impacting the model's ability to learn accurate classifications.
3. **Adam Optimizer with Reduced Learning Rate for Fine-Tuning:** A reduced learning rate is used to prevent the model from diverging during the fine-tuning process. The Adam optimizer is chosen for its efficiency in handling sparse gradients and adaptive learning rate capabilities, but changes in its parameters can affect convergence time during inference.

Model Summary:

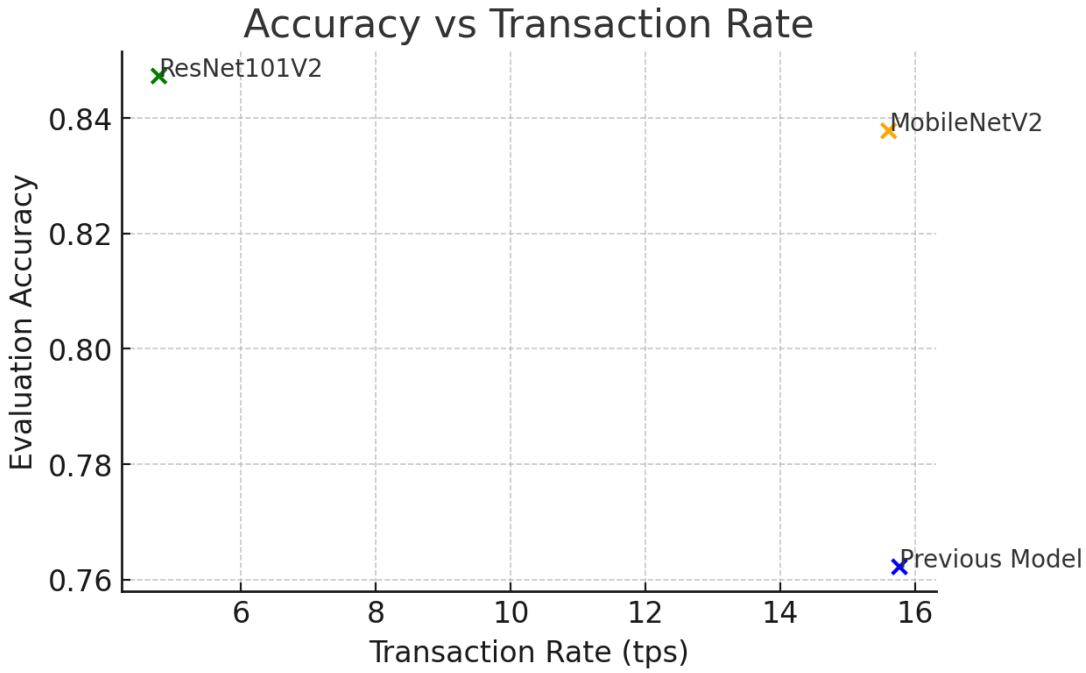
1. **Summary Table**

Metric	Previous Model	MobileNetV2	ResNet101V2
Transactions	465	461	97
Availability (%)	100.00	100.00	8.58
Elapsed Time (s)	29.49	29.54	20.29
Response Time (s)	0.63	0.63	1.67
Transaction Rate (tps)	15.77	15.61	4.78
Concurrency	9.92	9.86	7.98
Successful Transactions	465	461	97
Failed Transactions	0	0	1033
Longest Transaction (s)	1.82	2.00	2.68
Shortest Transaction (s)	0.25	0.39	0.54
Disk space usage (MB)	9.27	12.4 MB	172 MB
Memory usage	311.5MB	288.1MB	404.9MB
Evaluation Accuracy	0.7622	0.8378	0.8473

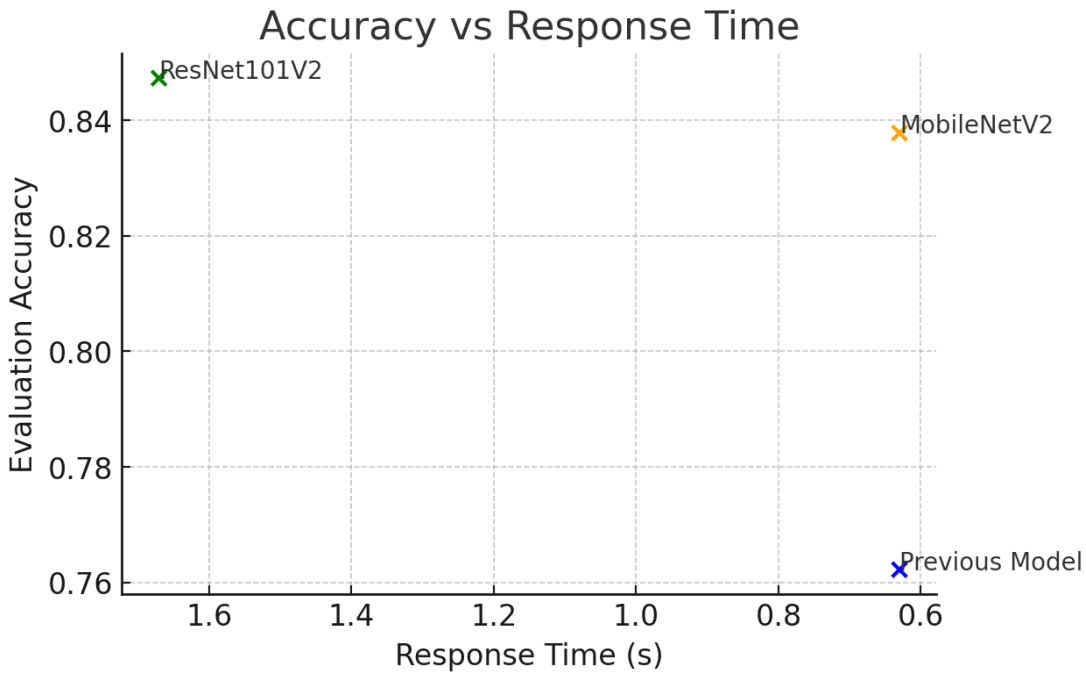
Accuracy vs Memory:



Accuracy vs Transaction rate:



Accuracy vs Response time:



Discussion:

1. **Memory Usage**: The plots indicate that MobileNetV2 has the lowest memory footprint among the three models, followed by the Previous Model, and ResNet101V2 consumes the most memory. Despite this, MobileNetV2 does not compromise on accuracy, achieving the second-highest accuracy score, which makes it a compelling choice for scenarios where memory is a constraint.
2. **Transaction Rate**: All three models show a high transaction rate, but the Previous Model and MobileNetV2 have a slightly higher transaction rate than ResNet101V2. This suggests that they are more suitable for applications requiring rapid processing of data or high throughput.
3. **Response Time**: The Previous Model and MobileNetV2 share the same low response time, which contributes to a better user experience for real-time applications. Although ResNet101V2 has a higher accuracy, its longer response time could be a limiting factor for latency-critical applications.
4. **Accuracy**: ResNet101V2 tops the accuracy metric, albeit by a small margin over MobileNetV2. The Previous Model, while still above the 75% threshold, lags the other two. This indicates a trade-off situation where ResNet101V2 might be preferred in cases where accuracy is paramount, and the slightly longer response time is acceptable.
5. **Failed Transactions**: It's noteworthy that ResNet101V2 had a significant number of failed transactions, which could be indicative of stability or compatibility issues in the deployment environment.

2. Deployments

2.1 Deploy service using container orchestration (Kubernetes)

Based on the Model Summary from the first section, in this deployment

mode, both the previous model and MobileNetV2 demonstrate excellent performance, with an Availability of 100%. However, the Availability of ResNet101V2 is only 8.58%. This indicates that ResNet101V2 is not suitable for this deployment mode. Therefore, I will employ two different deployment modes for ResNet101V2.

2.2 Deploy service with load balancing

Metric	Value
transactions	542.00
availability	100.00
elapsed_time	29.66
data_transferred	0.02
response_time	0.54
transaction_rate	18.27
concurrency	9.91
successful_transactions	542.00
failed_transactions	0.00
longest_transaction	1.46
shortest_transaction	0.16
Memory usage	642.2 (for each pod)

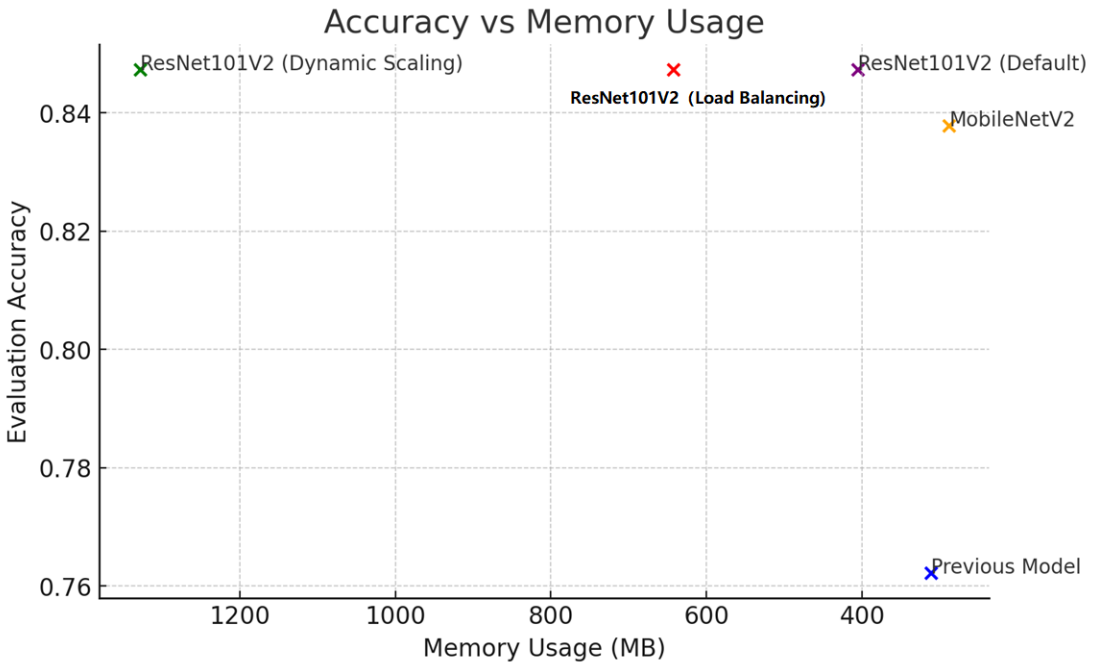
Test Results with load balancing

2.3 Deploy service with dynamic scaling

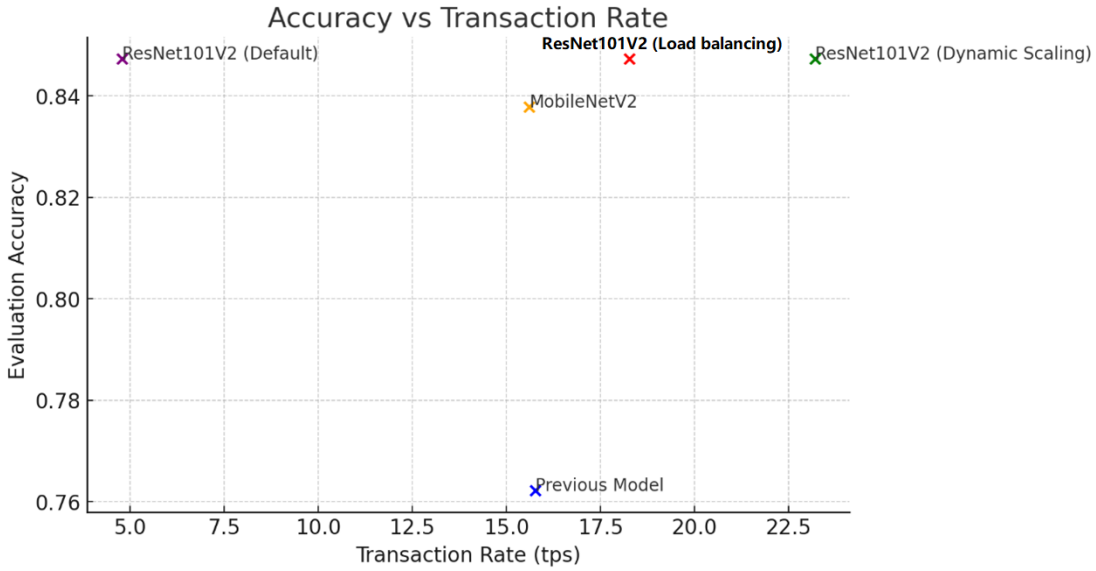
Metric	Value
Transactions	8346
Availability (%)	100.00
Elapsed Time (s)	359.80
Data Transferred	0.25
Response Time (s)	0.43
Transaction Rate (tps)	23.20
Throughput	0.00
Concurrency	9.99
Successful Transactions	8346
Longest Transaction (s)	1.49
Shortest Transaction (s)	0.16
Memory usage	1328.8 (for each pod)

Model Summary (previous vs ResNet101V2(with load balancing & dynamic scaling):

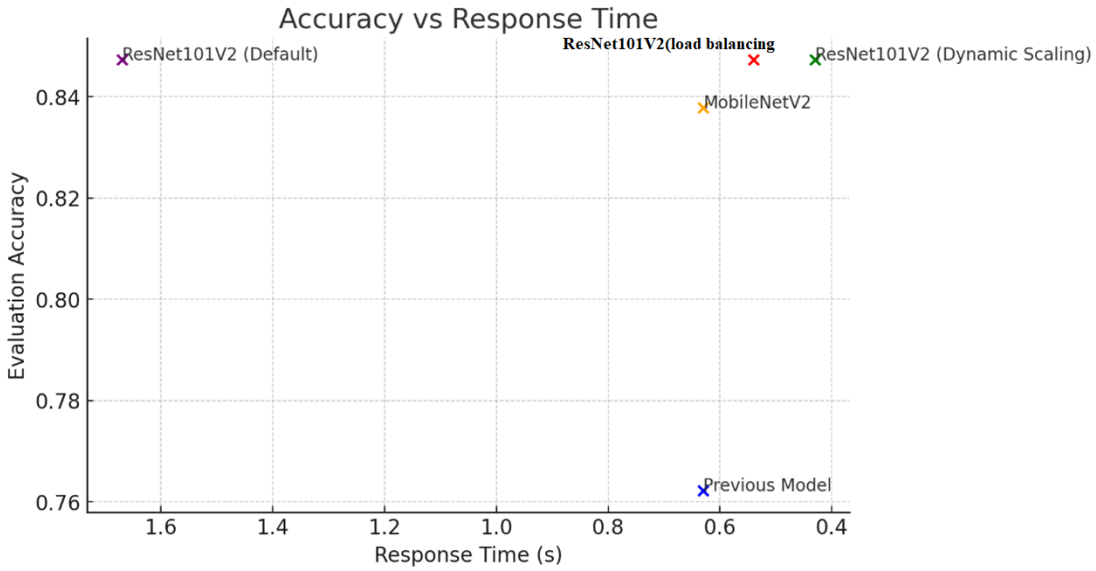
Accuracy vs Memory:



Accuracy vs Transaction rate:



Accuracy vs Response time:



Analysis and Discussion:

1. **Previous Model:**
   - **Pros:** This model has moderate memory usage and reasonable response times, making it a balanced option for general use cases.
   - **Cons:** It doesn't have the highest transaction rate or the best accuracy.
   - **Best Use Case:** It's a good all-rounder and can be considered for cost-effective solutions where top-tier accuracy and response times are not critical.
2. **MobileNetV2:**
   - **Pros:** It offers high accuracy with the least memory usage among all models, suggesting it is the most resource efficient.
   - **Cons:** The transaction rate is not as high as the dynamic scaling option of ResNet101V2.
   - **Best Use Case:** Ideal for mobile or edge computing scenarios where resources are limited but high accuracy is still required.
3. **ResNet101V2 (Dynamic Scaling):**
   - **Pros:** Exhibits the highest transaction rate and high accuracy, indicating robust performance under heavy loads.
   - **Cons:** It has the highest memory usage, which could be costly.
   - **Best Use Case:** Suitable for high-demand situations where maximum performance and accuracy are critical, and resources are not a limiting factor.
4. **ResNet101V2 (Load Balancing):**
   - **Pros:** Provides a balance between performance and resource usage, with good accuracy and lower memory usage than the dynamic scaling deployment.
   - **Cons:** The response time and transaction rate are not as good as the dynamic scaling option.
   - **Best Use Case:** Good for services where load varies and cost-effectiveness is important without significantly compromising on

performance.

5. **ResNet101V2 (Default):**
    - **Pros:** Offers high accuracy, which is consistent across different deployment strategies.
    - **Cons:** Has a lower transaction rate compared to dynamic scaling and uses more memory than the load-balancing deployment, combined with a significantly higher number of failed transactions, suggesting potential reliability issues.
    - **Best Use Case:** Might be reserved for use cases where the model's specific strengths are crucial, and the higher resource consumption and potential stability issues can be managed.

Summary:

- If the priority is to minimize costs and conserve resources, MobileNetV2 might be the best choice.
- For services that require high throughput and can accommodate higher costs, ResNet101V2 (Dynamic Scaling) would be appropriate.
- When there is a need for a balance between performance and cost, ResNet101V2 (Load Balancing) might be the optimal path.
- If the application can handle variability in performance and potential stability issues for the sake of high accuracy, then ResNet101V2 (Default) could be considered.

Appendix A: Saved models

1. **Saved model, corresponding to training notebook ResNet101V2:** https://drive.google.com/file/d/1g4DS-hf2qsunsBer3dQ0rzUIb7aZcRFV/view?usp=sharing
2. **Saved model_2, corresponding to training notebook MobileNetV2:** https://drive.google.com/file/d/1AG8fw3QDqK-8JBBDjrmRH6RwcTMl3FyG/view?usp=sharing

Appendix B: Deployment files

1. **Previous model, MobileNetV2 model, ResNet101V2 model:** https://drive.google.com/file/d/1DMYTd7_xvv5R8iXa6aTDYzYLa9layiKQ/view?usp=sharing
2. **ResNet101V2(load balancing)** model: https://drive.google.com/file/d/1BaNCIw-7jkRm6EIbL0xx0s9Utq_RGN_x/view?usp=sharing
3. **ResNet101V2**(Dynamic Scaling) **model:** https://drive.google.com/file/d/1MfspWnzYsmi8q9leAi1zaBAtcxcRygEM/view?usp=sharing