Machine Learning (ECE-GY 6143) Final Project Report
Deployment Track

Models

Previous Model

Observations

From the confusion matrix of the **previous model** it is seen that certain classes are posing challenges for accurate classification. Notably, the **'Dairy Product'** category exhibits suboptimal performance, with a diagonal value of **0.41**, indicating that this class is often misclassified. For instance, dairy items like **milk** are sometimes incorrectly identified as **'Dessert'**. This misclassification could stem from shared characteristics in the model's feature space between dairy products and desserts, such as color and texture commonalities in images.

Additionally, the **'Dessert'** and **'Fried Food**' categories demonstrate moderate performance, with respective diagonal values of **0.64** and **0.71**. The model tends to confuse **'Desserts'** with **'Eggs'**, which could be attributed to similar visual features like shape and color in items such as cakes, which were classified as eggs. The **'Fried Food'** category faces misidentification issues, particularly with **'Meat'**, where items like **French fries** are wrongly labeled as **meat**. This suggests a possible overlap in the textural or shape features that the model uses to distinguish these categories.

Model Metrics

The **precision** metric for the previous model is approximately **76.53%**, which indicates that when the model predicts a particular class, it is correct about 76.53% of the time. The **recall** is similarly positioned at around **76.22%**, revealing that the model successfully identifies 76.22% of all relevant instances within the classes. This close proximity between precision and recall suggests a balanced classification performance, where neither false positives nor false negatives significantly outweigh the other. The **F1-score** is at **75.64%,** providing a single metric that encapsulates the model's accuracy in terms of both precision and recall. This value points to a moderately high level of performance but also highlights room for improvement, particularly in reducing both false positives and false negatives. The overall accuracy of the model is **76.22%**, which implies that the model correctly classifies 76.22% of all the cases. While this accuracy is relatively high, the F1-score indicates that the balance between precision and recall could be improved.

Best Model

Describe your model

1. **What base model did you use? How many parameters does this base model have?**
    a. The base model I used was **MobileNetV2**.
    b. The total number of parameters in the MobileNetV2 base model is **2,257,984**. Of these, **2,223,872** are trainable parameters, and **34,112** are non-trainable parameters.
2. **Describe the specific transformations you used to create an augmented data set.**
    a. To augment the dataset, the following transformations were applied:
        i. Rescaled pixel values by 1/255 to bring them into the range [0, 1].
        ii. Applied a rotation range of 20 degrees for random rotation.
        iii. Applied a zoom range of 0.2 for moderate zooming in and out.
        iv. Shifted the image width and height by 20% for both width and height.
        v. Sheared the image by 0.2.
        vi. Applied horizontal flips to the images.
        vii. Utilized the "nearest" fill mode to fill in new pixels that may be created by rotations or width/height shifts.
3. **For how many epochs did you train your classification head, and with what optimizer, learning rate, and batch size? What was the validation accuracy at the end of this training stage?**
    a. The classification head was trained for **20** epochs
    b. **Optimizer:** Adam
    c. **Learning rate:** 0.0005
    d. **Batch size:** 32.
    e. **Accuracy at the end of this training stage:** 84.08%
4. **Which layers did you un-freeze for fine-tuning, how many epochs did you fine-tune for, and what optimizer, learning rate, and batch size did you use for fine-tuning? What was the validation accuracy at the end of this training stage?**
    a. The last 5 layers of the base model were unfrozen and made trainable. The fine-tuning was carried out for an additional 20 epochs.
    b. **Optimizer:** Adam
    c. **Learning rate:** 0.0001
    d. **Batch size:** 64.
    e. **Accuracy at the end of this training stage:** 86.59%
5. **What was your final accuracy on the evaluation set?**
    a. **Final Accuracy:** 87.87%
6. **Did your model correctly predict the class of your custom test image?**
    a. Yes

The choice of **MobileNetV2** as the base model was influenced by its superior performance over other tested models like **VGG16, VGG19, ResNet50,** and **EfficientNetB0**, which failed to achieve comparable accuracy levels. This compact yet powerful model offered good accuracy, with over **2.2 million trainable parameters**, making it ideal for the food classification problem. The training commenced with a focus on feature learning, using **20 epochs** to adjust the classification head with a carefully selected learning rate and batch size, ensuring a solid foundational knowledge without overfitting. A further period of fine-tuning of the model's deeper layers for an additional 20 epochs enhanced validation accuracy incrementally. These steps were critical in achieving an impressive final evaluation accuracy of **87.87%** and demonstrated the model's robust generalization capabilities as it correctly classified a custom test image.

To improve the model's accuracy and control overfitting, a series of fully connected layers were introduced. Each dense layer, featuring **1024, 512,** and **256** neurons, was accompanied by **batch normalization** and a **50% dropout** rate. Batch normalization helped in stabilizing the learning process by normalizing the input layers, allowing for faster convergence and acting as a regularizer to some degree. Dropout was employed as an additional regularization technique to encourage the model to develop redundant pathways, thus improving robustness and reducing the likelihood of reliance on any single pattern or feature. **L1 and L2 regularization** were also incorporated within these layers to penalize the complexity of the model, fostering a preference for simpler, more generalizable patterns in the weights. This strategic layering and regularization were decisive in refining the model's accuracy and ensuring that it maintained high performance when exposed to new and varied data inputs.

Observations

The confusion matrix for my model indicates a substantial improvement in classification accuracy across the 11 food categories. Notably, the **'Dairy Product'** category has seen an increase in its classification accuracy, with the diagonal value now at **0.74**, up from the **previous 0.41**. This demonstrates a significant enhancement in the model's ability to distinguish dairy products, likely due to improved feature extraction or more representative training data. Additionally, the **'Dessert'** category has improved from **0.64 to 0.78**, and **'Fried Food'** from **0.71 to 0.82**, indicating that the model has become better at discerning between visually similar food items, such as distinguishing cakes from eggs and French fries from meat, which were points of confusion in the earlier model.

Model Metrics

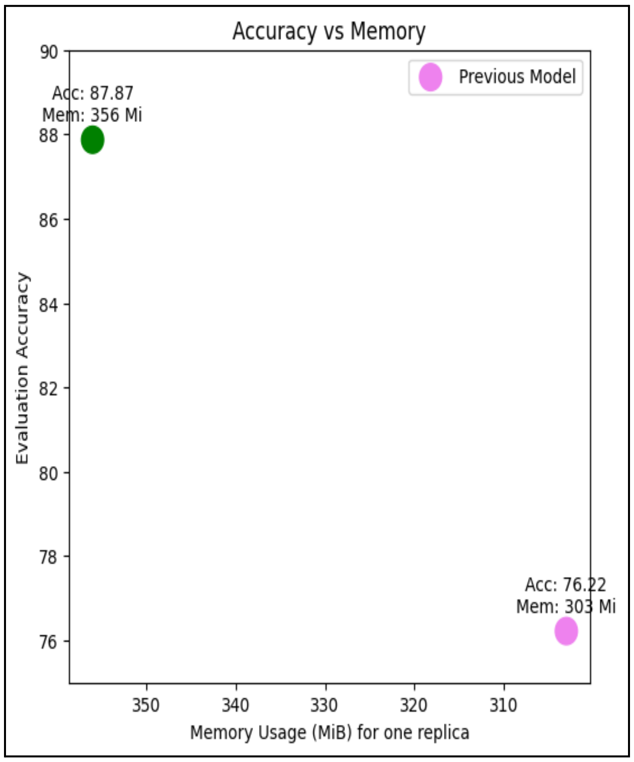
The **precision** of my model stands at **88.35%,** and the **recall** is at **87.87%,** both of which are approximately 11% higher than the previous model's metrics. The increased precision suggests that the model's predictions are more reliable, with fewer false positives. The enhanced recall indicates that the model is now better at identifying all relevant instances of each class, resulting in fewer false negatives. The **F1 Score**, which considers both precision and recall, is at **87.90%**, showing a significant improvement from the earlier **75.64%.** This increase is a strong indication that the model has achieved a better balance in terms of precision and recall, suggesting fewer instances where the model is confused between classes.

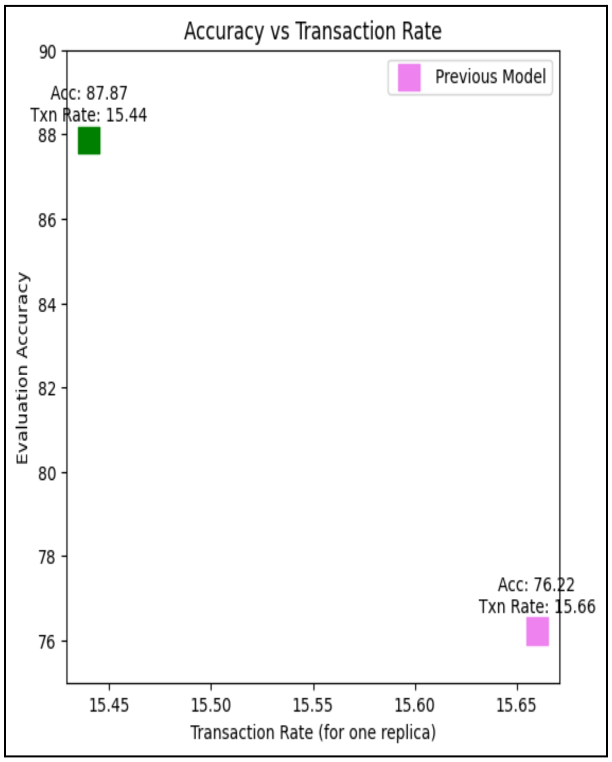
Comparatively, the new model's **overall accuracy** has escalated to **87.87%**, an increase of more than 11% from the previous accuracy of 76.22%. This leap in performance is a testament to the effectiveness of the model's improvements. The higher accuracy implies that the model is more adept at correctly classifying a broader range of food items. This enhancement can be attributed to a more refined model architecture. The improvements across precision, recall, and F1-score in conjunction with the accuracy underscore the new model's superior ability to generalize across diverse food categories, making it significantly more robust and reliable.

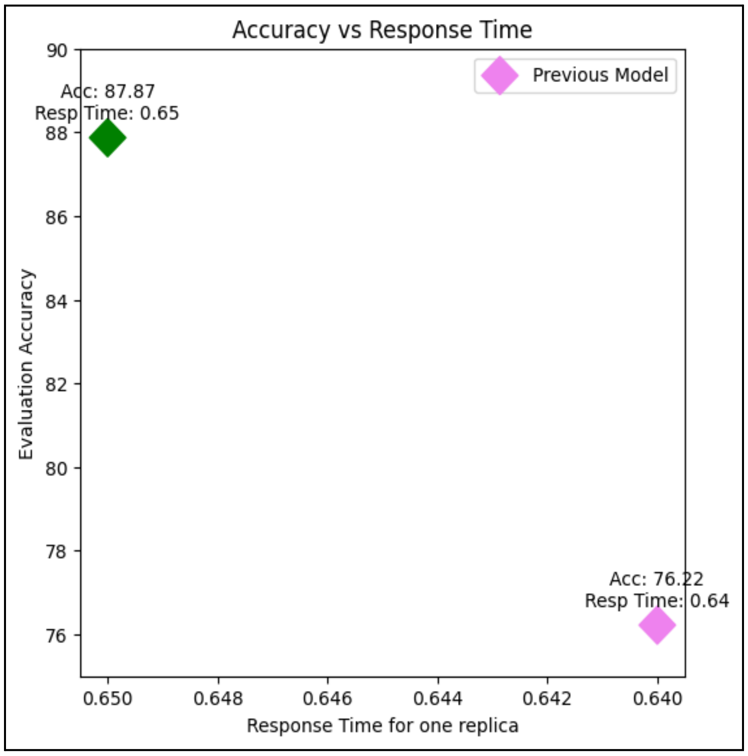
Model Summary

Metrics	Previous Model	Best Model
Base Model	MobileNetV2	MobileNetV2
Final Model's Summary		
Total no. of parameters	2272075	4240971
Trainable parameters	14091	1544128
Non-trainable parameters	2257984	34,112
Model Metrics		
Evaluation Accuracy	76.23%	87.87%
Precision	76.53%	88.35%
Recall	76.22%	87.87%
F1 Score	75.64%	87.90%
Operational Metrics - One Replica		
Memory	303 Mi	351 Mi
Response Time	0.64	0.65
Transaction Rate	15.66	15.44

One Replica Plots:







Operational Metrics

Previous Model

The previous model has moderate resource requirements and responsiveness in a production environment. With a single model replica deployed, it achieves a transaction rate of **15.66** transactions per second, handling inference requests efficiently. Its response time is **0.64** seconds, which is generally fast but may require optimization for real-time systems. The model's memory usage is **303 Mi**, making it lightweight and cost-effective to host.

Best Model

The new model, while more complex, demonstrates operational metrics that are comparable to the older version, with a slight increase in memory usage from **303 Mi to 356 Mi**, indicating that the additional complexity of the fully connected layers and the larger architecture has a modest impact on resource requirements. The transaction rate is nearly consistent, with a marginal decrease from **15.66 to 15.44** transactions per second, suggesting that the enhanced model maintains its inference throughput effectively. Response time remains virtually unchanged, with a negligible increase from **0.64 to 0.65** seconds.

Deployments

Strategy 1: Slightly Reduced Memory Usage with Slightly Higher Response Time Explanation

The modifications made to the deployment_hpa.yaml file have resulted in a small change in memory usage. The key changes in the configuration revolve around the resources allocated to the containers and the autoscaling parameters.

Initially, the resource **limits for memory** in the deployment were set at **"4Gi"**, with **requests** at **"2Gi"**. In the modified configuration file, the memory limit was increased to **"6Gi"**, while the memory request remained at **"2Gi"**. This increase in the memory limit allows each pod more breathing space to handle spikes in usage, which is crucial during periods of high load.

The HPA settings were adjusted to target a **higher CPU utilization percentage** before scaling, increased from **40% to 80%**. This means that scaling out occurs less frequently but at higher loads, which can lead to more consistent and efficient usage of resources across fewer pods. By running fewer pods on average, the overall memory requested across all pods is slightly reduced. However, this strategy led to **increased response times and reduced transaction rates**.

Strategy 2: Significantly Reduced Memory Usage with Greater Response Time Explanation

The adjustments made to the deployment_hpa.yaml file significantly affected memory usage and response time, focusing on resource allocation and autoscaling behavior.

The **memory request** was increased substantially from **2Gi to 4.5Gi**, while the **memory limit** was raised from **4Gi to 5Gi**. This closer alignment of memory requests to the actual average usage likely led to more efficient resource allocation and reduced memory contention, as Kubernetes could schedule pods on nodes with sufficient memory more effectively. This resulted in a drastic decrease in memory usage.

The key change in the HPA settings was lowering the **targetCPUUtilizationPercentage** from **40% to 30%,** promoting more aggressive scaling. This means the application scales out quicker, potentially reducing the memory load per pod. However, this could lead to pods handling higher loads relative to their capacity before scaling, impacting response times. The **average response time increased** significantly from **0.113 seconds to 1.078 seconds**.

Deployment Summary

Strategy 1: Slightly Reduced Memory Usage with Slightly Higher Response Time

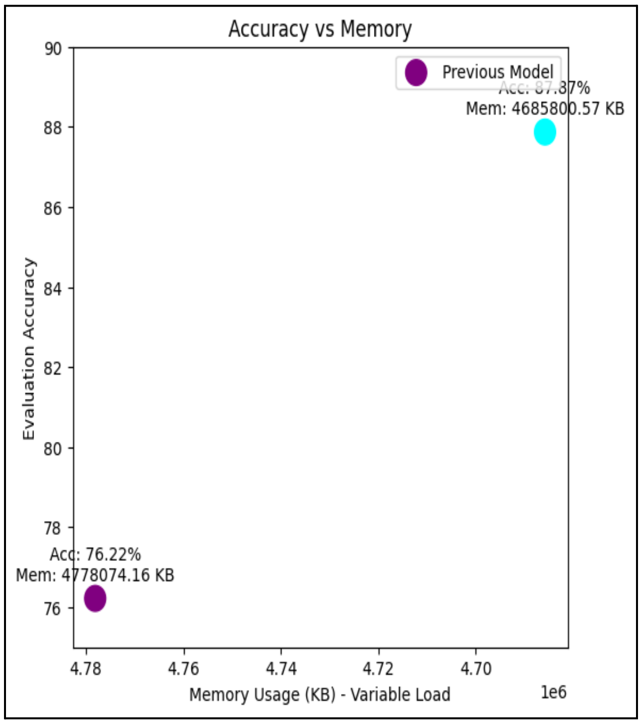
	Previous Model	Best Model
Metrics		
Memory Requested	6885550.05	6752102.10
Memory Used	4778074.16	4685800.57
Response Time	0.11	0.21
Transaction Rate	15.27	13.56
Configuration Changes Made		
targetCPUUtilizationPercentage	40	80
limits: cpu	2	3
limits: memory	4Gi	6Gi

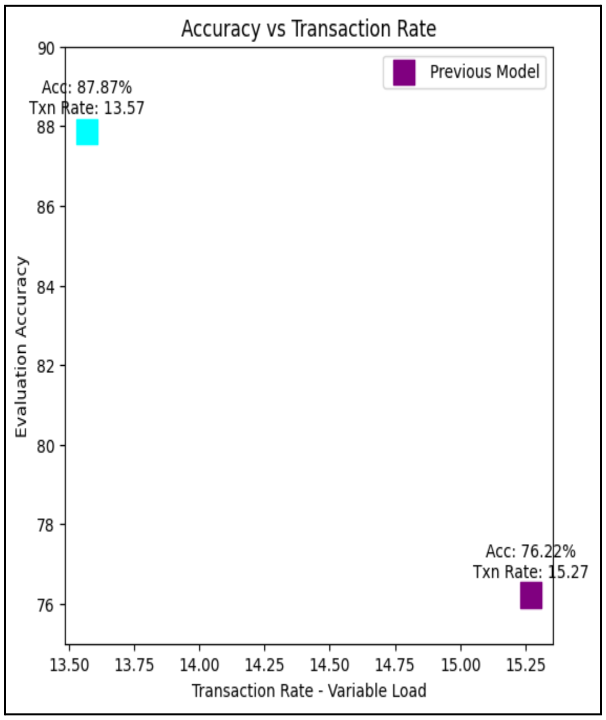
Strategy 2: Significantly Reduced Memory Usage with Greater Response Time

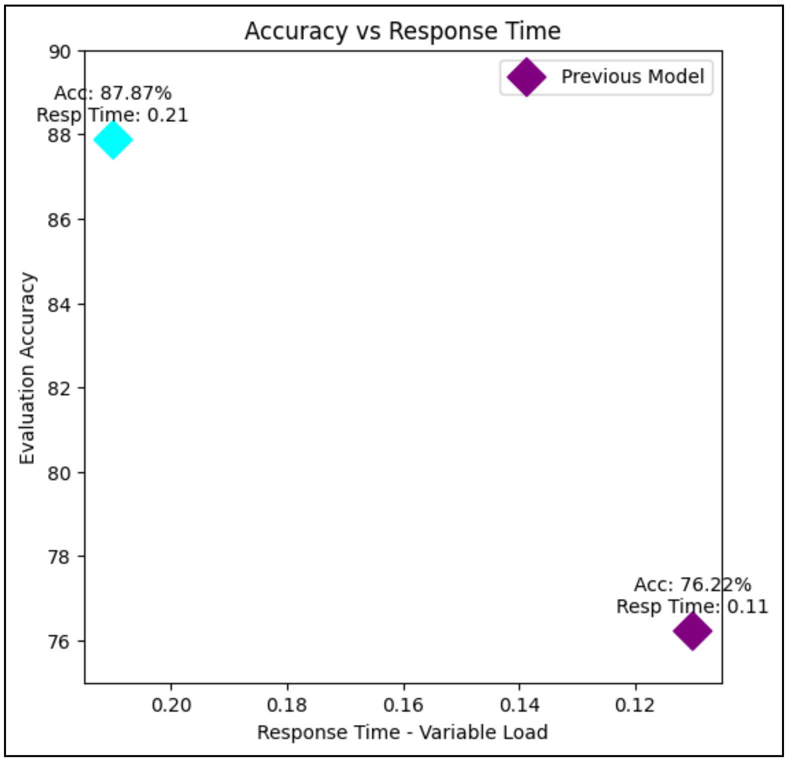
	Previous Model	Best Model
Metrics		
Memory Requested	6885550.05	4718592.0
Memory Used	4778074.16	459522.02
Response Time	0.11	1.08
Transaction Rate	15.27	6.04
Configuration Changes Made		
targetCPUUtilizationPercentage	40	30
successThreshold	3	1
limits: memory	4Gi	5Gi
requests: memory	2Gi	4.5Gi

Evaluation

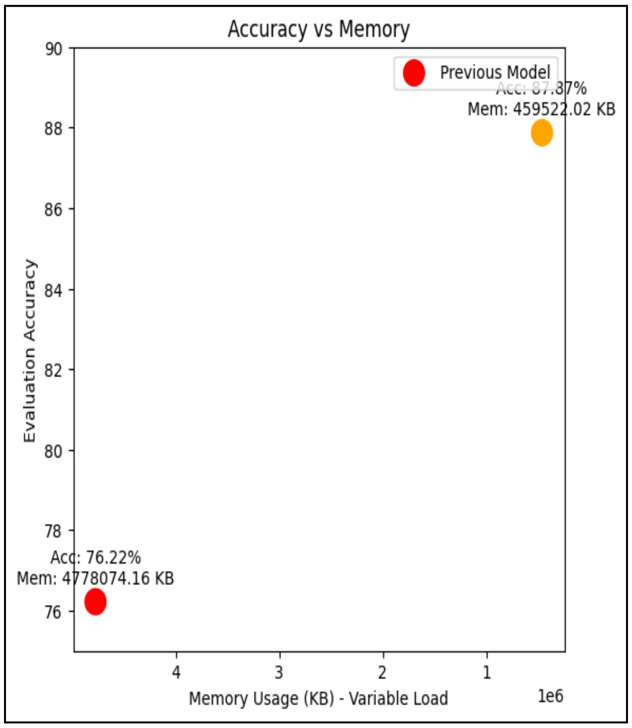
Strategy 1: Slightly Reduced Memory Usage with Slightly Higher Response Time

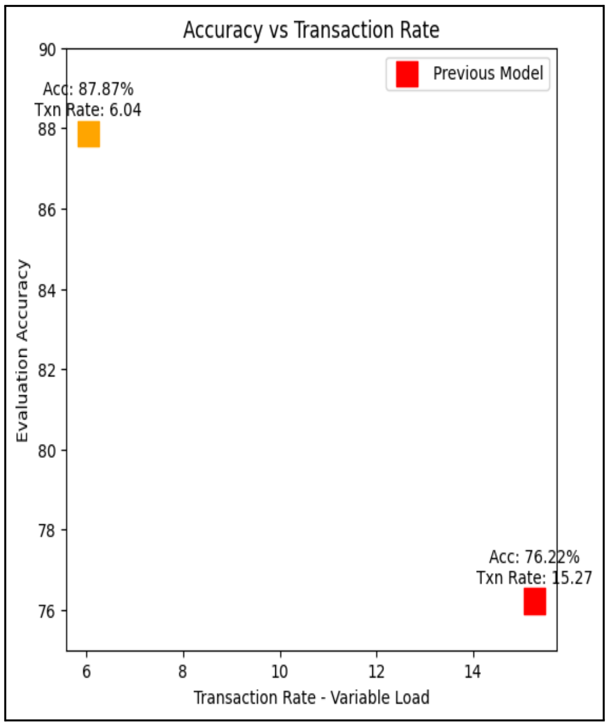


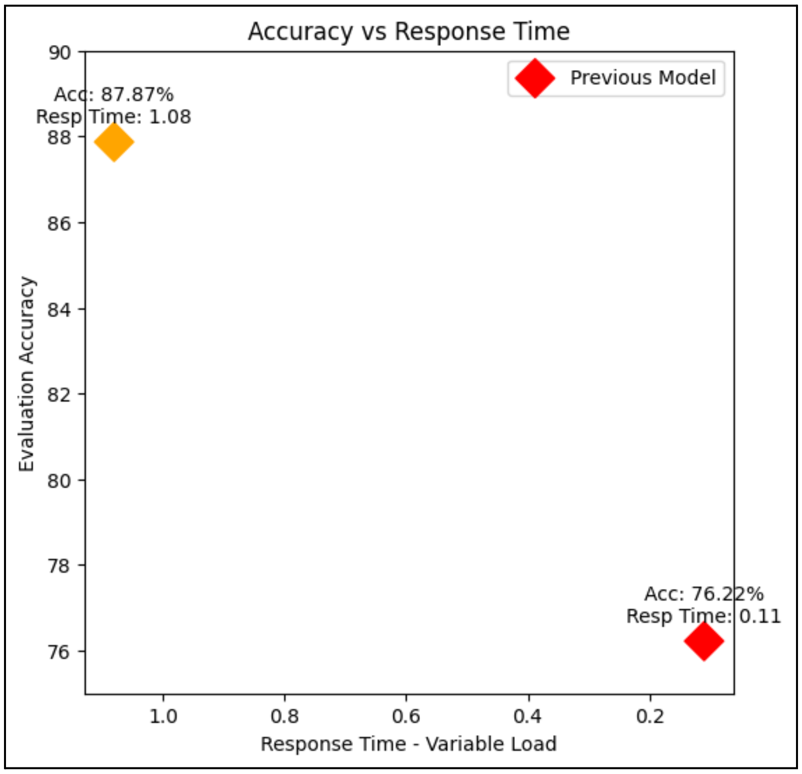




Strategy 2: Significantly Reduced Memory Usage with Greater Response Time







Explanation

In **Strategy 1**, the Kubernetes setup is fine-tuned to **slightly decrease memory consumption** while allowing for a **minor increase in response time**. The memory limit for each pod is raised from 4Gi to 6Gi, but the memory request remains at 2Gi. This adjustment means each pod has more leeway to manage peak load times effectively. Also, the autoscaler settings are modified to let pods reach 80% CPU usage before scaling up. This results in fewer pods running overall, aiding in modest memory savings. However, because pods now handle a higher load before scaling occurs, response times are slightly longer. This strategy is best suited for scenarios where **memory savings are a priority**, **but response time should not be compromised much**.

Strategy 2 targets a **drastic reduction in memory usage** but with a **noticeable impact on response time**. The memory request per pod is significantly increased from 2Gi to 4.5Gi, with a limit set to 5Gi. Aligning the memory request more closely with actual usage dramatically improves memory efficiency. Additionally, the autoscaler is configured at just 30% CPU usage. While this ensures a more distributed workload among pods, it also means each pod faces a higher load before additional pods are added, leading to longer response times, increasing from around 0.113 seconds to about 1.078 seconds. This strategy is ideal for environments where **reducing memory usage is crucial, and slower response times can be tolerated**.

Appendix A: Saved models

Previous Model

Google Drive Link - https://drive.google.com/uc?id=1pe_bkPxDOtyi6ewJ7qq2LqIcHoEbqwyw

- This is linked to the notebook named: **previous-model-sg7885.ipynb**
- This is the model left by the previous employee.

Best Model

Google Drive Link -
https://drive.google.com/uc?id=1DJXgykCECDBpwQ_vd8UUNEYDfIUFWEA1

- This is linked to the notebook named: **best-model-sg7885.ipynb**
- This is my best model.

Appendix B: Deployment files

Yaml file 1

- Linked to **Strategy 1: Slightly Reduced Memory Usage with Slightly Higher Response Time**

deployment.yaml

```
apiVersion: autoscaling/v1
kind: HorizontalPodAutoscaler
metadata:
 name: ml-app-hpa
spec:
 maxReplicas: 5
 minReplicas: 1
 scaleTargetRef:
   apiVersion: apps/v1
   kind: Deployment
   name: ml-app-hpa
 targetCPUUtilizationPercentage: 80
---
apiVersion: v1
kind: Service
metadata:
 name: ml-service-lb
spec:
 selector:
   app: ml-app-hpa
```

```yaml
 ports:
 - protocol: "TCP"
   port: 6000
   targetPort: 5000
   nodePort: 32000
 type: LoadBalancer
---


apiVersion: apps/v1
kind: Deployment
metadata:
 name: ml-app-hpa
spec:
 selector:
   matchLabels:
     app: ml-app-hpa
 replicas: 1
 template:
   metadata:
     labels:
       app: ml-app-hpa
   spec:
     containers:
     - name: ml-app
       image: node-0:5000/ml-app:0.0.1
       imagePullPolicy: Always
       ports:
       - containerPort: 5000
       readinessProbe:
         httpGet:
           path: /test
           port: 5000
         periodSeconds: 5
         initialDelaySeconds: 5
         successThreshold: 3
       resources:
         limits:
           cpu: "3"
           memory: "6Gi"
         requests:
           cpu: "1"
```

```
          memory: "2Gi"
```

Yaml file 2

- Linked to **Strategy 2: Significantly Reduced Memory Usage with Greater Response Time**

deployment.yaml

```
apiVersion: autoscaling/v1
kind: HorizontalPodAutoscaler
metadata:
 name: ml-app-hpa
spec:
 maxReplicas: 5
 minReplicas: 1
 scaleTargetRef:
   apiVersion: apps/v1
   kind: Deployment
   name: ml-app-hpa
 targetCPUUtilizationPercentage: 30
---
apiVersion: v1
kind: Service
metadata:
 name: ml-service-lb
spec:
 selector:
   app: ml-app-hpa
 ports:
 - protocol: "TCP"
   port: 6000
   targetPort: 5000
   nodePort: 32000
 type: LoadBalancer
---

apiVersion: apps/v1
kind: Deployment
metadata:
 name: ml-app-hpa
spec:
 selector:
   matchLabels:
     app: ml-app-hpa
```

```
replicas: 1
template:
  metadata:
    labels:
      app: ml-app-hpa
  spec:
    containers:
    - name: ml-app
      image: node-0:5000/ml-app:0.0.1
      imagePullPolicy: Always
      ports:
      - containerPort: 5000
      readinessProbe:
        httpGet:
          path: /test
          port: 5000
        periodSeconds: 5
        initialDelaySeconds: 5
        successThreshold: 1
      resources:
        limits:
          cpu: "2"
          memory: "5Gi"
        requests:
          cpu: "1"
          memory: "4.5Gi"
```