# ECE 6143 Machine Learning Project Report

Yihua Yang

yy5028@nyu.edu

Dec.12, 2023

# Models

## Previous Model

Optimizing Metrics: Based on the evaluation result, the previous model has an overall accuracy of 0.7622 and a loss of 0.7037. Specifically, the model did not do well (accuracy below 0.8) in classifying: dairy products (accuracy: 0.41), seafood (accuracy: 0.62), bread (accuracy: 0.63), dessert (accuracy: 0.64), Egg (accuracy: 0.71), and Meat (accuracy: 0.72).

Operational Metrics: The typical response time of this model is 1.17 seconds with a transaction rate of 7.21 when it is deployed as a single replica. The disk space it needs is 9.27 MB. When the model is loaded for inference, the memory consumption is 843 MB.

## Model 1 (Base Model: VGG16)

The base model contains 14714688 parameters, and when creating the augmented dataset, the following transformations are applied:

- Rescale: Each pixel value is scaled down by a factor of 255 to make all pixel values in the range between 0 and 1.
- Rotation: Each image will be randomly chosen to rotate by a degree in the range between 0 and 1.
- Zoom: Random zooming with a zoom range of 10% will be applied to the training images.
- Width Shift: The images are randomly translated horizontally by 10% of the image width.
- Height Shift: The images are randomly translated vertically by 10% of the image height.
- Shear: The images are sheared by 10%.
- Horizontal Flip: Images are randomly flipped horizontally.

- Fill Mode: When doing the transformations, some new pixels will probably be created, and all the gaps will be filled with the value of the nearest pixel value.

The epoch number set in the training phase is 200 with an early stopping function. Therefore, the actual epochs it needs in the training phase is usually below 100. The optimizer function applied here is Adam, and the batch size is 32. A decay learning rate function is used during the training stage. Specifically, the initial learning rate is 0.001, with a decay rate of 0.1 and decay steps of 2000. As a result, due to the above implement, the validation accuracy is 0.6519.

In the fine-tuning phase, the last 5 layers are unfreezed. Compared to the training phase, the epoch number now is 50 with an early stopping function, which can potentially decrease the number of actual epochs, and the initial learning rate is decreased from 0.001 to 0.00001 but with the same decay rate and decay steps in the decay learning rate function for fine tuning. For other parameters that are not mentioned, they are identical to the training step. At the end, the final accuracy on the evaluation set is 0.8491.

To increase the accuracy of the model prediction, some steps such as data augmentation and hyperparameter tuning would not affect the response time. However, for some steps like fine-tuning more layers of a pre-trained model can increase the computational cost during the inference process, although these steps can lead to higher accuracy.

Optimizing Metrics: Based on the evaluation result, the first model has an overall accuracy of 0.8491 and a loss of 0.5321. Specifically, the model did not do well in classifying dairy products (accuracy: 0.62), Bread (accuracy: 0.72), and Egg (accuracy: 0.78). When classifying other kinds of food, its accuracy is above 0.8, which can be considered as an acceptable result.

Operational Metrics: The typical response time of this model is 1.08 seconds with a transaction rate of 7.71 when it is deployed as a single replica. The disk space it needs is 110 MB. When the model is loaded for inference, the memory consumption is 973 MB.

**Model 2 (Base Model: MobileNetV2)**

The base model contains 2257984 parameters in total, and when creating the augmented dataset, the following transformations are applied (as same as model 1):

- Rescale: Each pixel value is scaled down by a factor of 255 to make all pixel values in the range between 0 and 1.
- Rotation: Each image will be randomly chosen to rotate by a degree in the range between 0 and 1.
- Zoom: Random zooming with a zoom range of 10% will be applied to the training images.
- Width Shift: The images are randomly translated horizontally by 10% of the image width.
- Height Shift: The images are randomly translated vertically by 10% of the image height.
- Shear: The images are sheared by 10%.
- Horizontal Flip: Images are randomly flipped horizontally.
- Fill Mode: When doing the transformations, some new pixels will probably be created, and all the gaps will be filled with the value of the nearest pixel value.

The optimizer, learning rate, and batch size are the same as the model 1. Due to the better performance in training phase, although it uses the identical early stopping function and the epoch number is the same, it typically ends before epoch 70.

Since the layer structure of model 2 is different than the model 1, all layers after the first 149 layers are unfreezed. Other than that, the epochs it uses for fine-tuning, the optimizer, the learning rate, and the batch size are identical to what model 1 uses in this stage. Finally, the accuracy of this model in classifying the evaluation data is 0.8312.

Optimizing Metrics: Based on the evaluation result, the first model has an overall accuracy of 0.8312 and a loss of 0.5205. Specifically, the model did not do well in classifying: Dairy Products (accuracy: 0.60), Dessert (accuracy: 0.74), Egg (accuracy: 0.75), Bread (accuracy: 0.78), and Fried Food (accuracy: 0.79). When classifying

other kinds of food, its accuracy is above 0.8, which can be considered as an acceptable result.

Operational Metrics: The typical response time of this model is 1.11 seconds with a transaction rate of 7.45 when it is deployed as a single replica. The disk space it needs is 63.1 MB. When the model is loaded for inference, the memory consumption is 1018 MB.

In conclusion, based on the performance discussed above, the previous model is Pareto dominated by both model 1 and model 2 in the performance of bad cases (accuracy below 0.8) and overall accuracy. Considering the operational metrics, all the three models are Pareto efficient. Specifically, the previous model has the lowest disk space usage, and the model 1 has the best response time and the best transaction rate, while the model 2 is somewhere between model 1 and model 0 in all specs. Since the memory usage in a single test may be easily affected by other factors, the memory consumption information in the operational metrics part is for a simple reference only. Therefore, considering optimizing metrics and operational metrics, choosing model 1 and choosing model 2 are both rational because both choices are Pareto efficient. To further decide which one is better, it will depend on what to the specific application and business needs. For example, if the disk space is the first criteria to consider, then the model 2 should be chosen; if the overall accuracy is what to be considered first, the model 1 should be chosen.

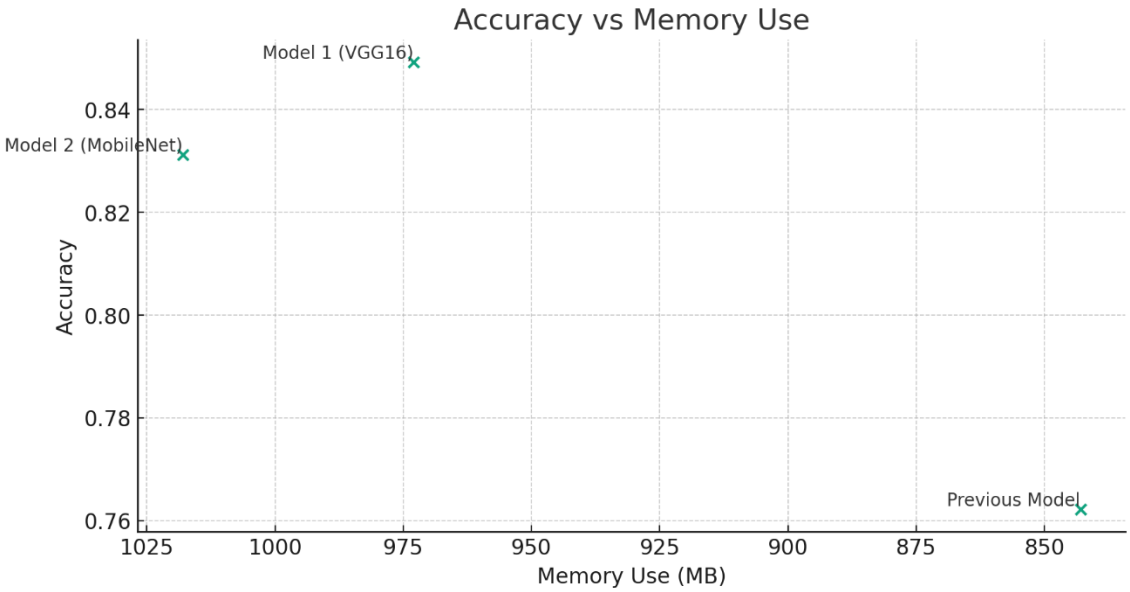
**Model Summary**

For a better understanding of all three models, a summary table and some plots are listed below:

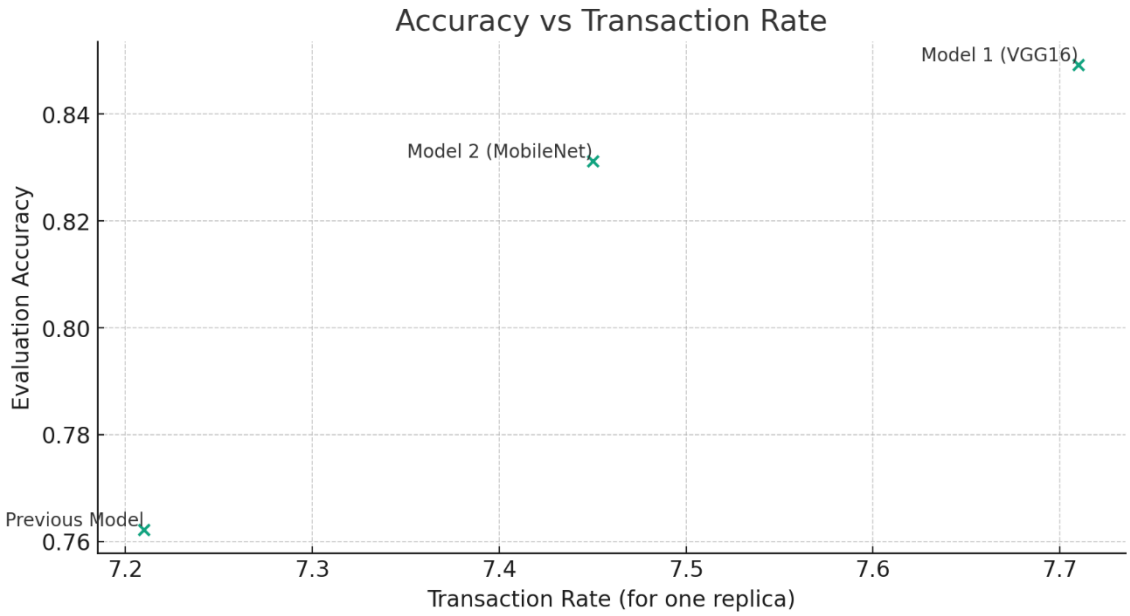
| | *Previous Model* | *Model 1 (VGG16)* | *Model 2 (MobileNet)* |
|---|---|---|---|
| *Overall Accuracy* | 0.7622 | 0.8492 | 0.8312 |
| *Disk Space* | 9.27 MB | 110 MB | 63.1 MB |

| | | | |
|---|---|---|---|
| *Response Time* | 1.17 | 1.08 | 1.11 |
| *Bad Classification* | in 6 classes | in 3 classes | in 5 classes |
| *Memory Use* | 843 MB | 973 MB | 1018 MB |

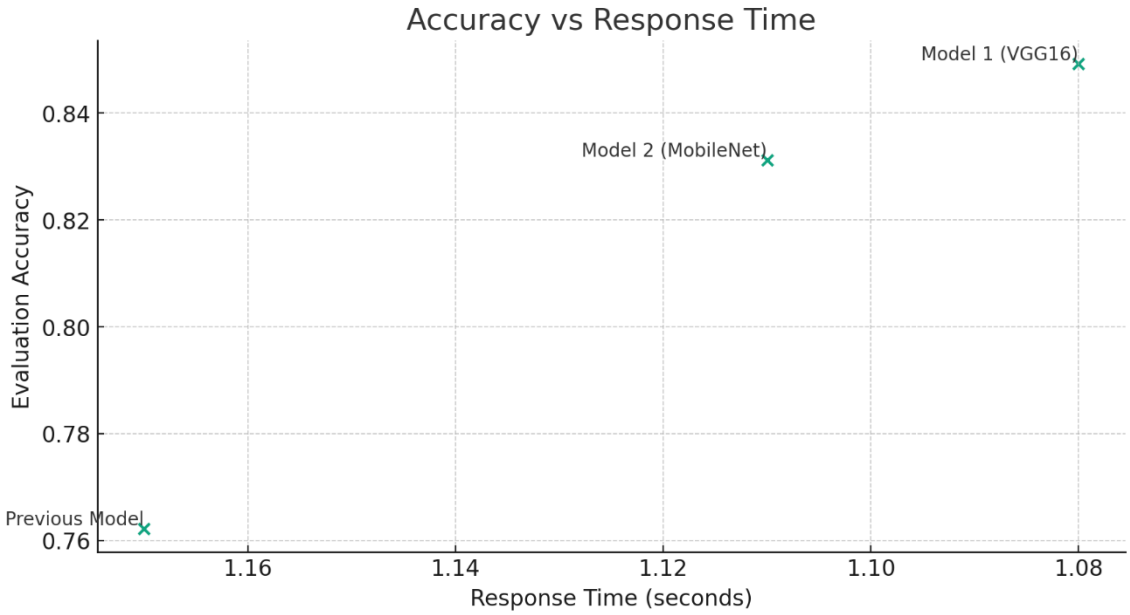
Plot of Accuracy vs. Memory Use:




Plot of Accuracy vs. Transaction Rate:



Plot of Accuracy vs. Response Time:




## Deployment

There are two deployment configurations compared in this section: the given one, *deployment_hpa.yaml*, and another one is *deployment_hpa+.yaml*. The latter configuration increases the replica limit from 5 to 10, enhancing performance under high-load conditions, and minimum memory is changed from 2 GB to 1 GB, with all other parameters remaining unchanged. To be specific, when the replica limit was set to 5 in the original deployment file, there were some failed cases during the high workload test. Thus, now in the deployment_hpa+.yaml file, the limit of replica is increased to 10, such that it is sufficient to handle the high workload without crash and improve the performance in response time and transaction rate without wasting the resource that much. If the maximum of the replica is set to 9, it may crash during high workload, based on the result of one instance of the evaluation. Since the memory usage for a single replica is around 1 GB, it would be a good idea to use 1 GB as minimum memory to avoid redundancy. The details of different models using different deployment files are summarized and listed below:

*Using deployment_hpa.yaml*

| | CPU Usage (Cores) | Memory Usage (KB) | Response Time (Sec) | Transaction Rate |
|---|---|---|---|---|
| *Previous Model* | 2.379 | 5041017 | 0.37 | 13.28 |
| *Model 1* | 2.425 | 6425222 | 0.46 | 13.82 |
| *Model 2* | 2.361 | 4841846 | 0.48 | 12.87 |

*Using deployment_hpa+.yaml*

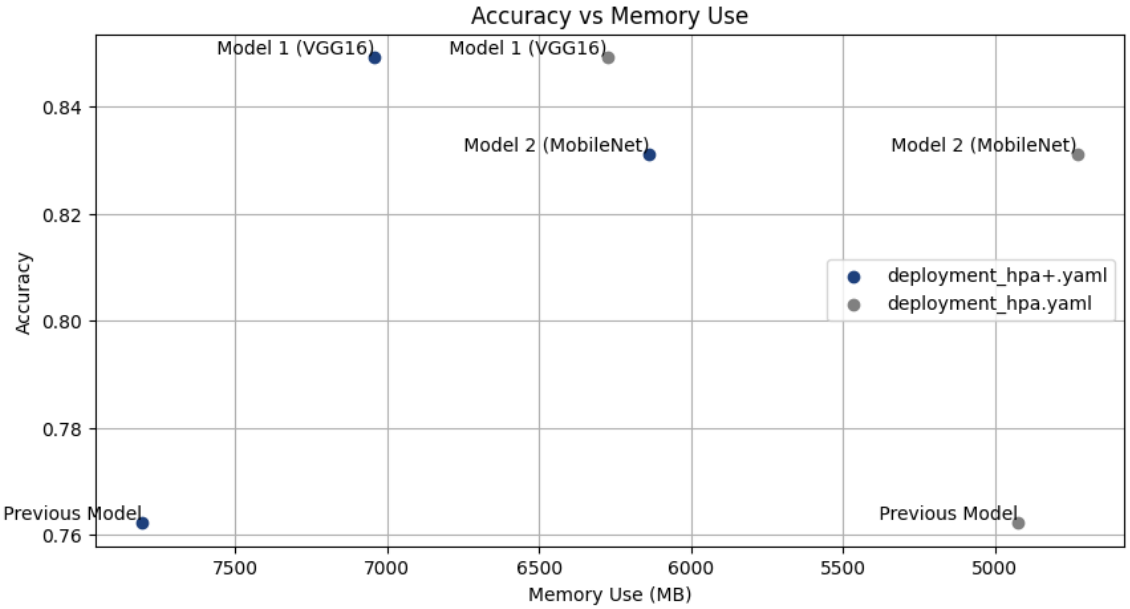
| | CPU Usage (Cores) | Memory Usage (KB) | Response Time (Sec) | Transaction Rate |
|---|---|---|---|---|
| *Previous Model* | 2.504 | 7994066 | 0.42 | 14.14 |
| *Model 1* | 2.499 | 7210982 | 0.43 | 14.15 |
| *Model 2* | 2.512 | 6284285 | 0.42 | 14.22 |

Based on the tables above, it is evident that the implementation of *deployment_hpa+.yaml* provides a better performance for all models in response time and transaction rate, especially when encountering peak load scenarios, although the resource usage of *deployment_hpa+yaml* is slightly higher than its competitor. Hence, it will be better to deploy *deployment_hpa+.yaml* to handle more workload.

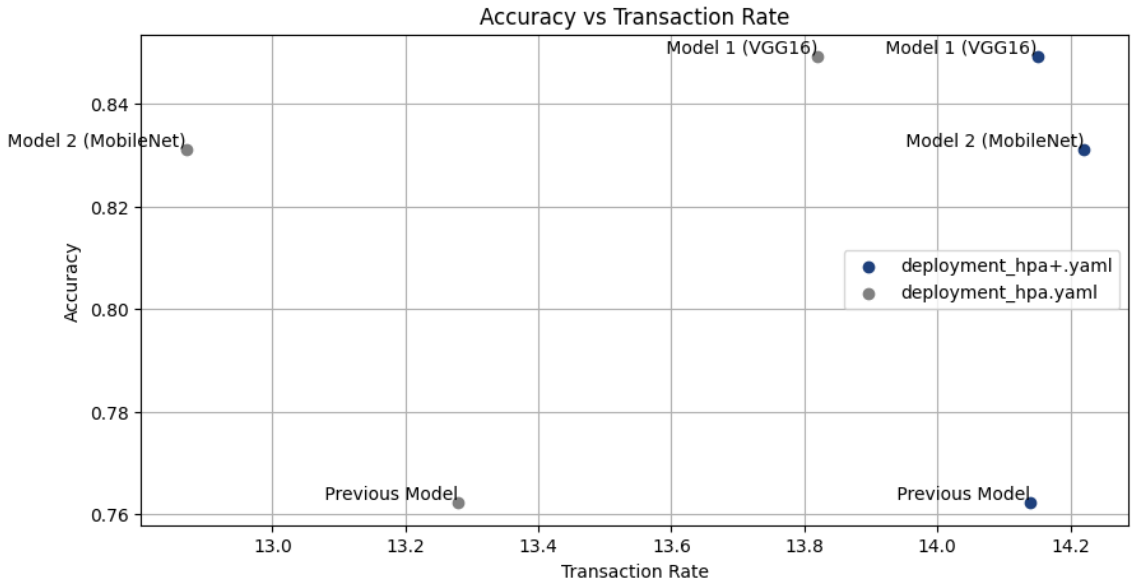
Among the three models, Model 2, which uses MobileNetV2 as the base model, used the least memory compared to other models while keeping the performance of response time and transaction rate similar to the others. Thus, from the perspective of memory utilization, Model 2 should be the first choice.

**Evaluation**

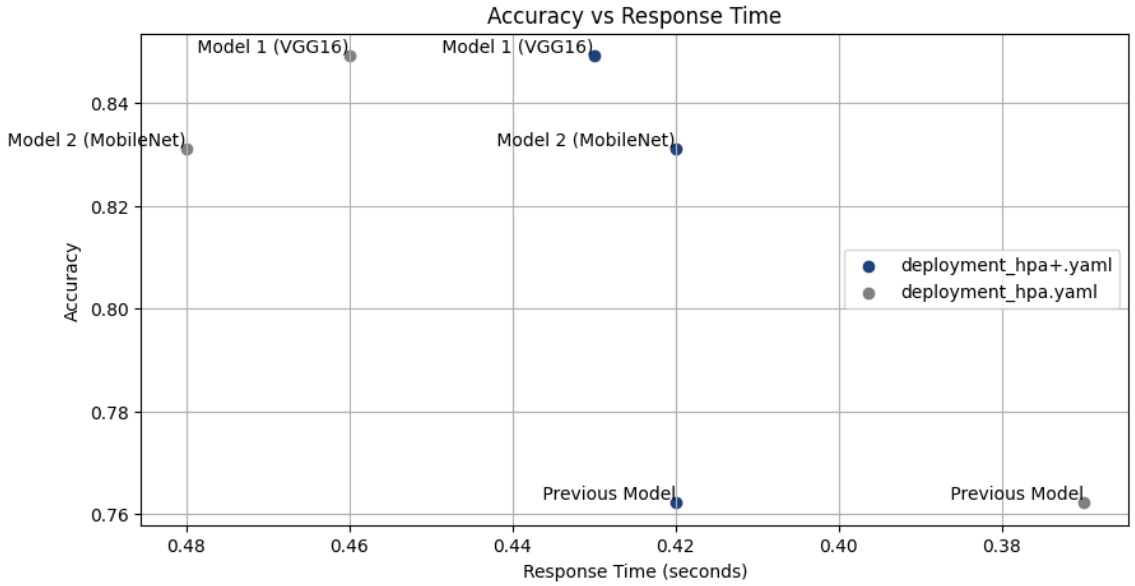
Based on the accuracy of each model and the performance of the model when deployed using deployment_hpa+.yaml, the plots of different combinations are given as the following:




In the plot of Accuracy vs. Memory Use, it is obvious that the Model 1 should be chosen in the case of considering accuracy first. Otherwise, the Model 2 shows a good performance in memory usage while keeping a decent accuracy compared to the previous model. Besides, using *deployment_hpa.yaml* demonstrates enhanced memory conservation across the models.




In the plot of Accuracy vs. Transaction Rate, on one hand, like the previous graph, Model 1 is the one that has the best accuracy. On the other hand, Model 2 provides the best performance in transaction rate. Additionally, all models that deployed with *deployment_hpa+.yaml* have a better transaction rate.




In the plot of Accuracy vs. Response Time, although the old model shows an extraordinary performance in response time when using *deployment_hpa.yaml*, due to its

lowest accuracy, it should not be considered as a good choice. Similarly, Model 1 is always the best choice for accuracy. The Model 2 has a good balance between accuracy and response time. Ignoring the special case of the previous model, in general, deployment_hpa+.yaml offers a better response time for most models.

To summarize, these combinations are preferred in the following cases:

- Best Accuracy and Decent Memory Use: Model 1 + *deployment_hpa.yaml*
- Best Accuracy and Decent Transaction Rate: Model 1 + *deployment_hpa+.yaml*
- Best Accuracy and Decent Response Time: Model 1 + *deployment_hpa+.yaml*
- Best Memory Use and Decent Accuracy: Model 2 + *deployment_hpa.yaml*
- Best Transaction Rate and Decent Accuracy: Model 2 + *deployment_hpa+.yaml*
- Best Response Time: Previous Model + deployment_hpa.yaml
- Decent Response Time and Decent Accuracy: Model 2 + deployment_hpa+.yaml

**Appendix A: Saved Models**

- Model 1: this model is trained in the notebook *training model1.ipynb*, and the link to the model: https://drive.google.com/file/d/1f9aRhFS255P4zrLwLOVD0wAFR5A3Fakm/view?usp=sharing

- Model 2: this model is trained in the notebook *training model2.ipynb*, and the link to the model: https://drive.google.com/file/d/1F6HQ_wqaj2PDEbHNIi2PXFo9XR4_SJHn/view?usp=sharing

- The previous model is evaluated in the notebook *evaluating_old_model.ipynb*.

**Appendix B: Deployment Files**

- Deployment_hpa+.yaml:

```
- apiVersion: autoscaling/v1
- kind: HorizontalPodAutoscaler
- metadata:
-   name: ml-app-hpa
- spec:
-   maxReplicas: 10
-   minReplicas: 1
-   scaleTargetRef:
-     apiVersion: apps/v1
-     kind: Deployment
-     name: ml-app-hpa
-   targetCPUUtilizationPercentage: 40
- ---
- 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: 3
-           resources:
-             limits:
-               cpu: "2"
-               memory: "4Gi"
-             requests:
-               cpu: "1"
-               memory: "1Gi"
-
```