Intro to Machine Learning

Lab: Deploying Machine learning systems

Yashwanth Alapati, ya2351

SLIDE 1: Table of Model training choices

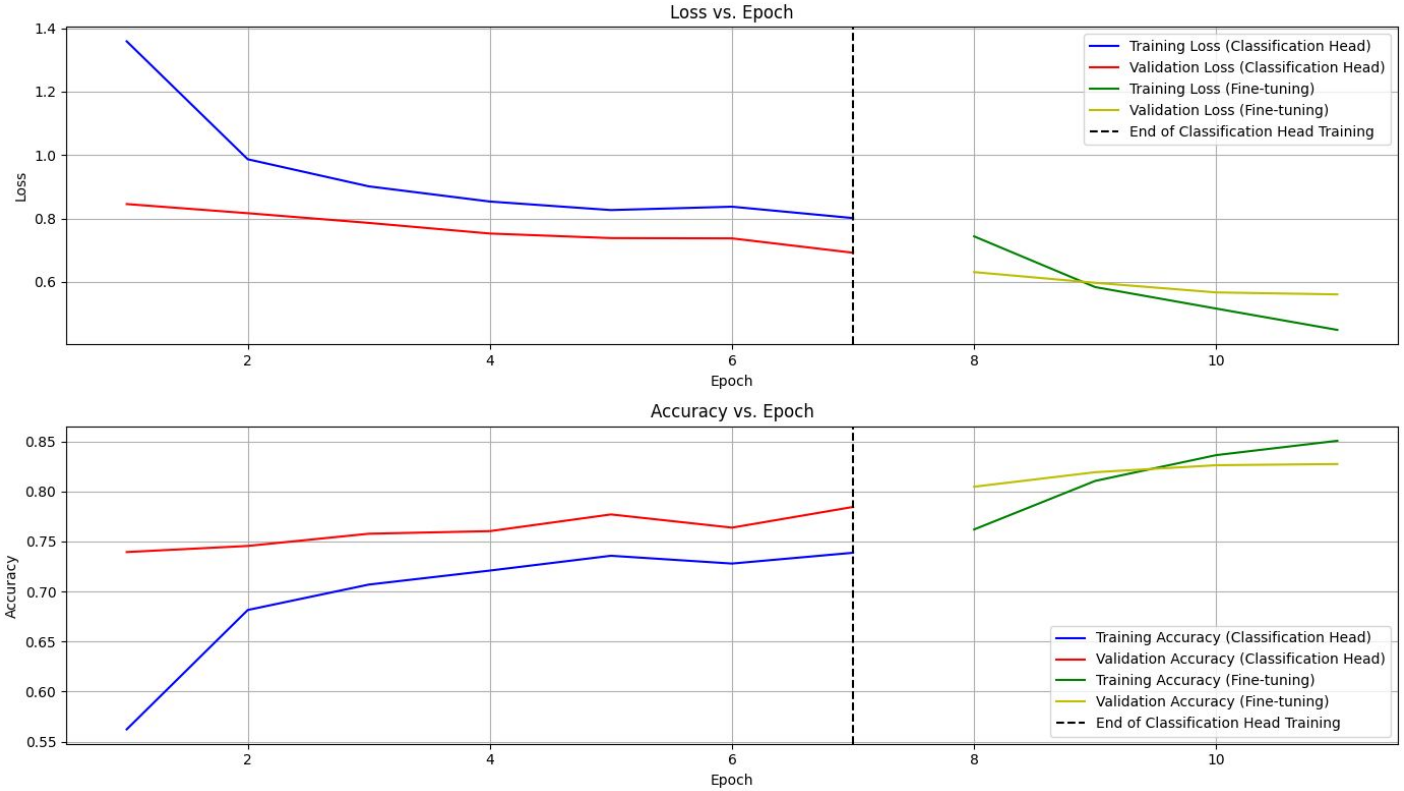
INFO	OPTIMIZING ACCURACY	OPTIMIZING LATENCY
Transform in Data Augmentation	Rescale,rotation,zoom,width_shift, height_shift, shear, horizontal_flip, fill_mode	Rescale,rotation,zoom,width_shift, height_shift, shear, horizontal_flip, fill_mode
Base model	ResNet50V2 size=98,top-1=76,CPU inference time=45.6	MobileNet size=16,top-1=70.4,CPU inference time=22.6
epochs,optimizer,lr,	7 epochs Adam @ 0.01	7 epochs Adam @ 0.01
Number of layers unfrozen	10	10
epochs,optimizer,lr(after fine tuning)	4 epochs Adam @0.0001	4 epochs Adam @0.0001
Final accuracy(eval_set)	0.8500	0.8419

Explanation

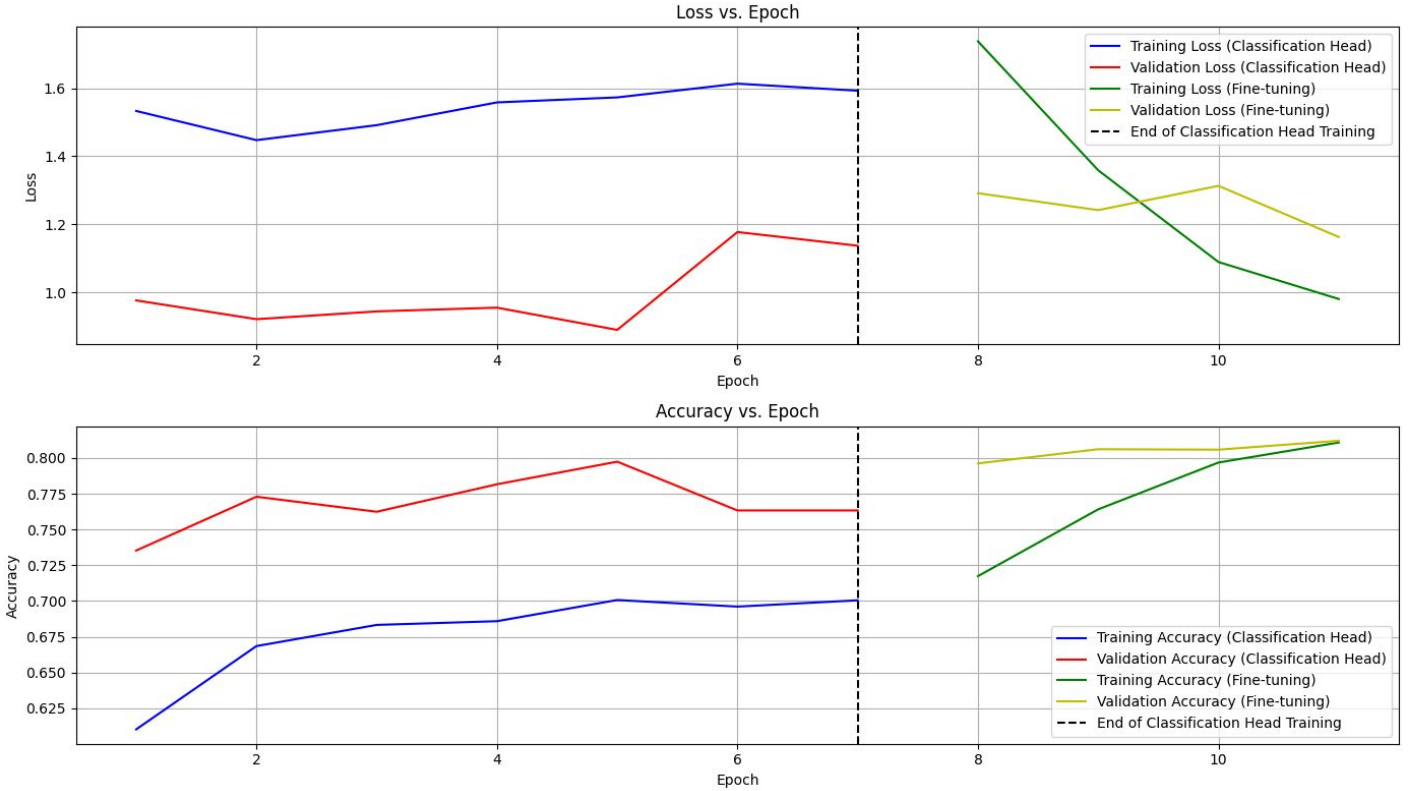
- I have chosen ResNet50V2 as the accuracy model for it has good top-1 accuracy value.
- I have chosen MobileNet as the latency model for it has very quick inference time and it is lightweight.
- I have not made different configuration choices for the accuracy model and the latency model.
- Since, we are focused on comparing how two different models perform in latency focused vs accuracy focused settings, I think it is right to have the training settings similar to get better comparisons
- Surprisingly, the latency model gave as good accuracy as the accuracy model, however it did not classify correctly the image I have uploaded.

NOTE: I had initially made two separate graphs for classification head and fine tuning. Later I realized I'm supposed to make only one total graph. Due to time constraint, I manually plotted the training graphs using the output values of epoch during training.

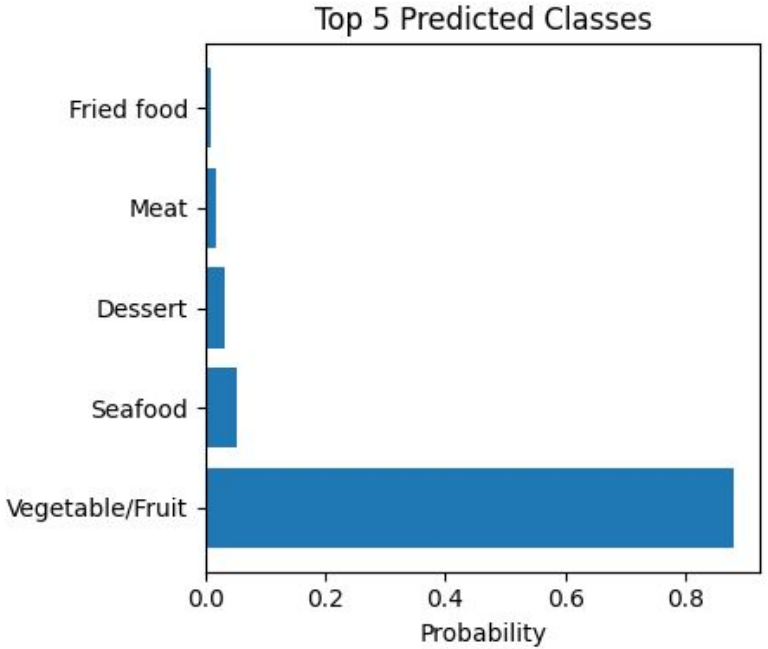
Training(accuracy model)



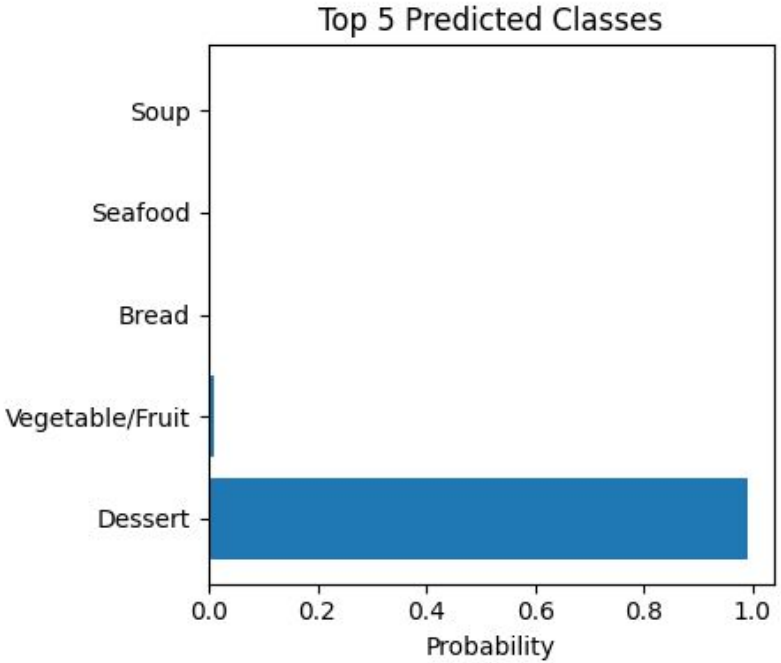
Training(latency model)



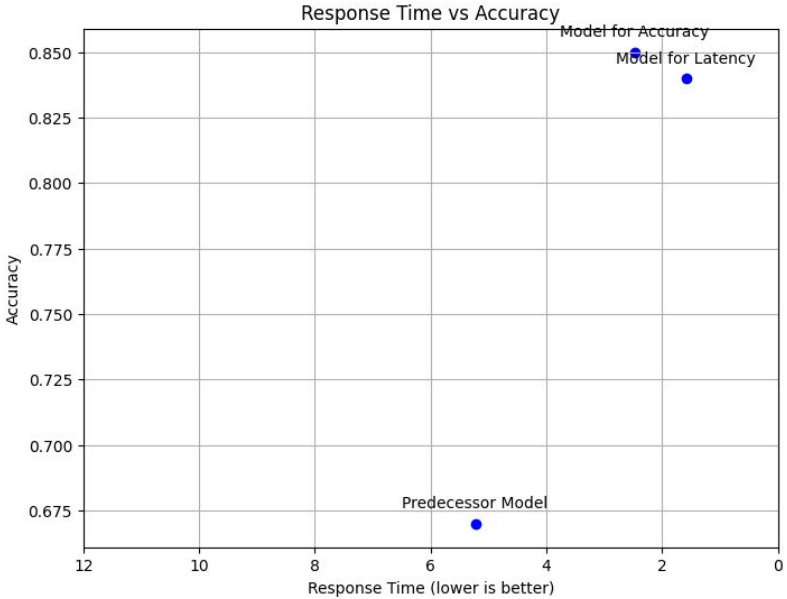
Accuracy model(own image)

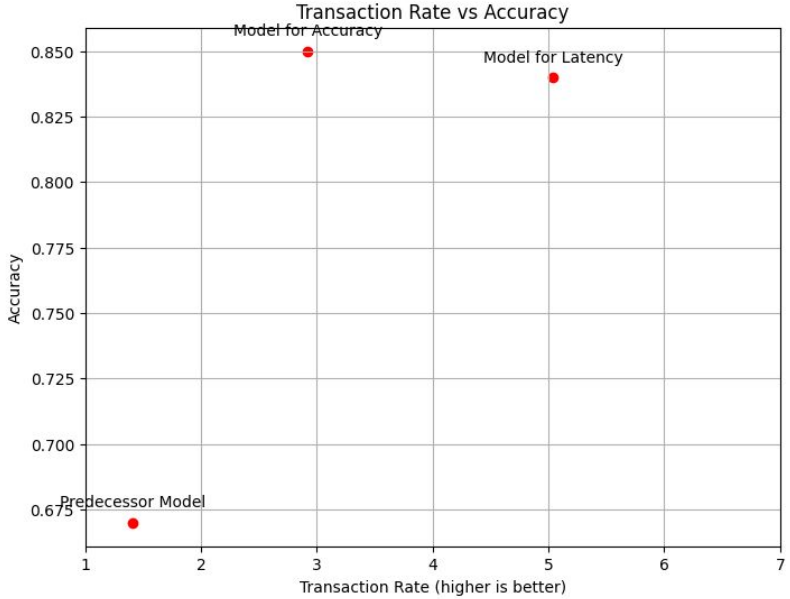


Latency model(own image)



SLIDE 2: Performance of models when deployed as a single pod deployment

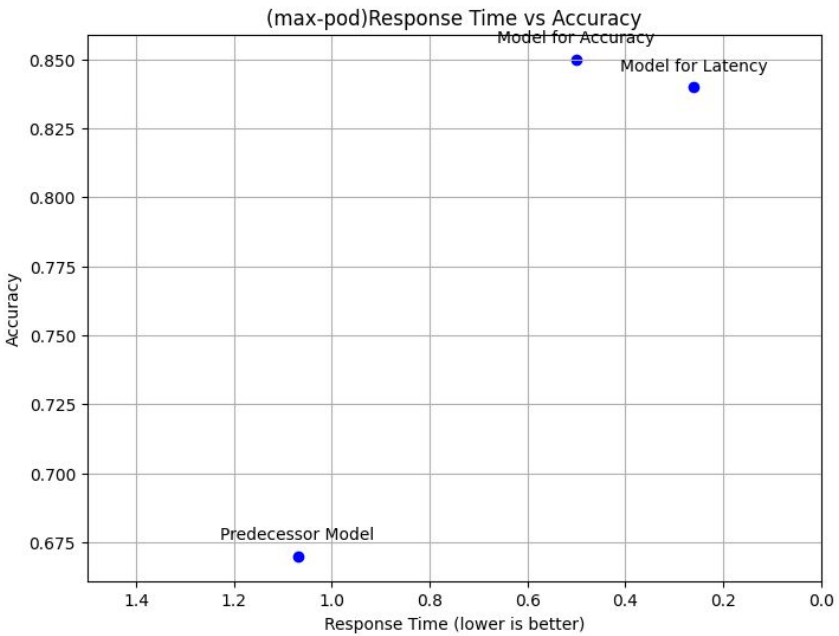


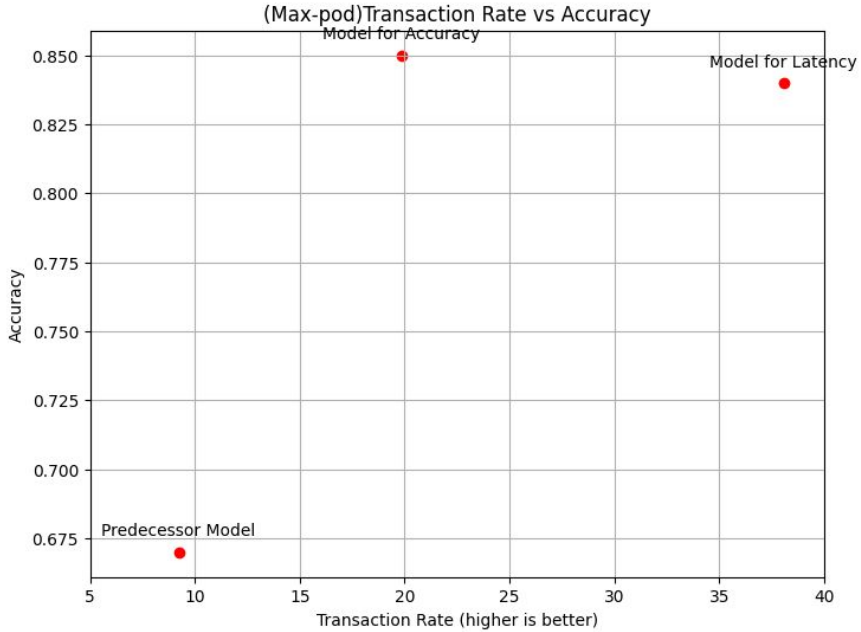


Explanation

- The availability has not been 100% for single pod deployment as expected.
- As far as response time is concerned we expected the latency model to have the best value as it is lightweight.Followed by accuracy model and predecessor model.The values turned out to be exactly as expected.
- As far as transaction rate is concerned, we expected the same. Latency model to be the best, followed by accuracy model and predecessor model.

SLIDE 3: Performance of models when deployed as a Max-size deployment





Explanation

- The availability has been 100% for Max pod deployment as expected.This is reflected in the siege results.
- As far as response time is concerned we expected the latency model to have the best value as it is lightweight.Followed by accuracy model and predecessor model.The values turned out to be exactly as expected.
- As far as transaction rate is concerned, we expected the same. Latency model to be the best, followed by accuracy model and predecessor model.

SLIDE4: Table showing replicas and resource request configurations for “max-size” deployment

	Previous	Accuracy	Latency
No.of replicas	9	9	9
CPU resource requests	1.5	1	1
Memory resource requests	2Gi	1.5Gi	1.5Gi
CPU resource limits	1.5	1.5	1.5
Memory resource limits	2Gi	2Gi	2Gi

Explanation

- As instructed in the deploy.md I had first run with 20 replicas and standard configurations.
- However, when I started playing around, I was getting better hits with the configurations mentioned in the previous slide table.
- I decided to keep the same configurations for both the accuracy and latency model, as we are focused on the performance and analysis of these model. It seems fair to provide same configuration.
- I had also tried with different configuration for latency model, but it didn’t have as many hits as the mentioned configurations did.

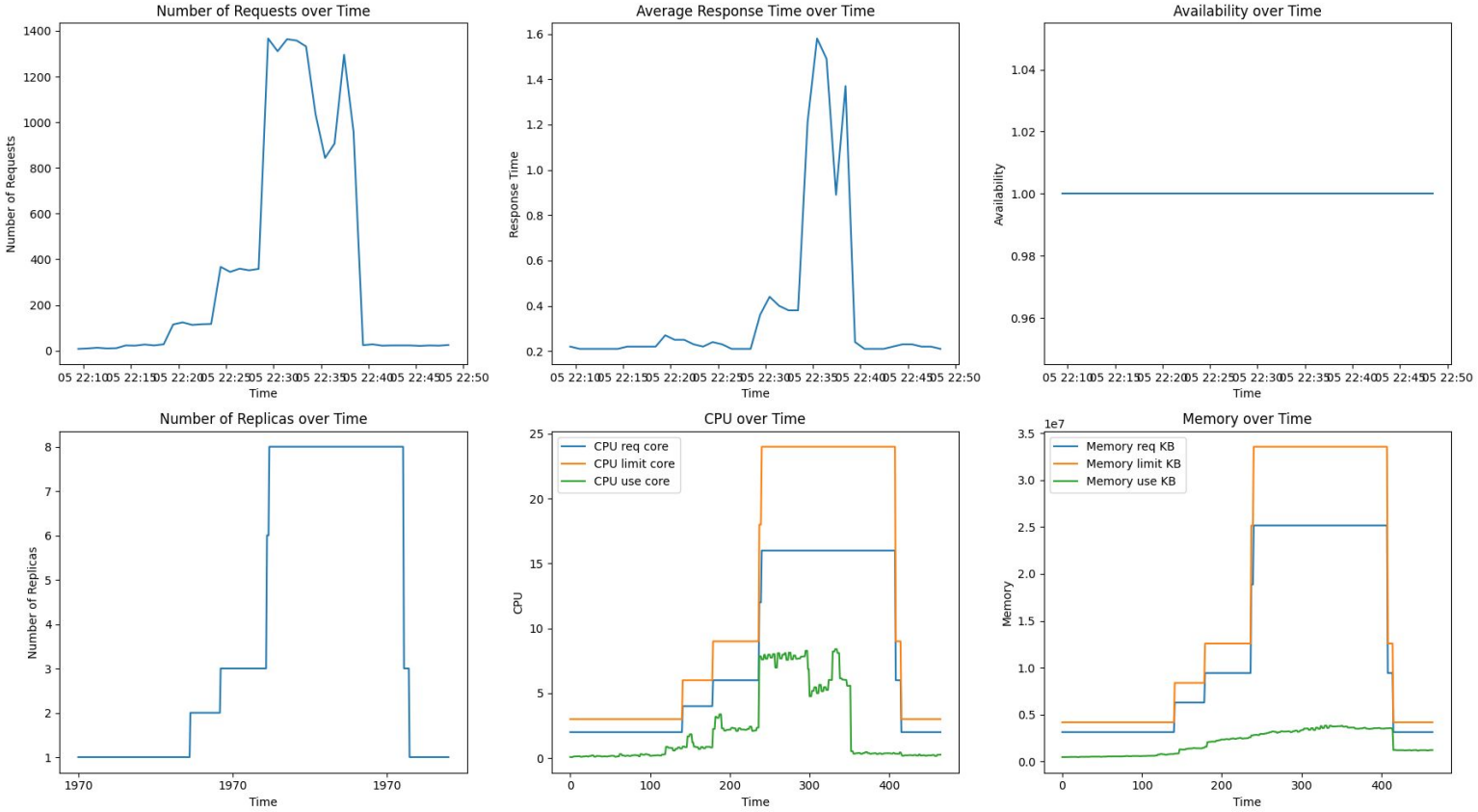
SLIDE 5: Table showing horizontal scaling configurations

	Accuracy	Latency
Min replicas	1	1
Max replicas	6	8
Target cpu utilization	60	40
CPU resource requests	1	2
Memory resource requests	1.5Gi	3Gi
CPU resource limits	1.5	3
Memory resource limits	2Gi	4Gi

Explanation

- I started playing around with the configurations.
- I had first tried the latency model with lower replica number than the accuracy model. This seemed like a fair approach as latency model wouldn’t need much resources/replicas. However, the graphs have been very vague and difficult to understand.
- I chose to go for higher replicas and the graphs from following slides were output and these are easy to understand.
- I played around with the configurations a lot, I got similar plots as well. But I’m putting out the best configurations I had observed: with 40 cpu utilization for latency model and 60 cpu utilization for accuracy model.You'd expect the latency model to have lower utilization as it is light weight.

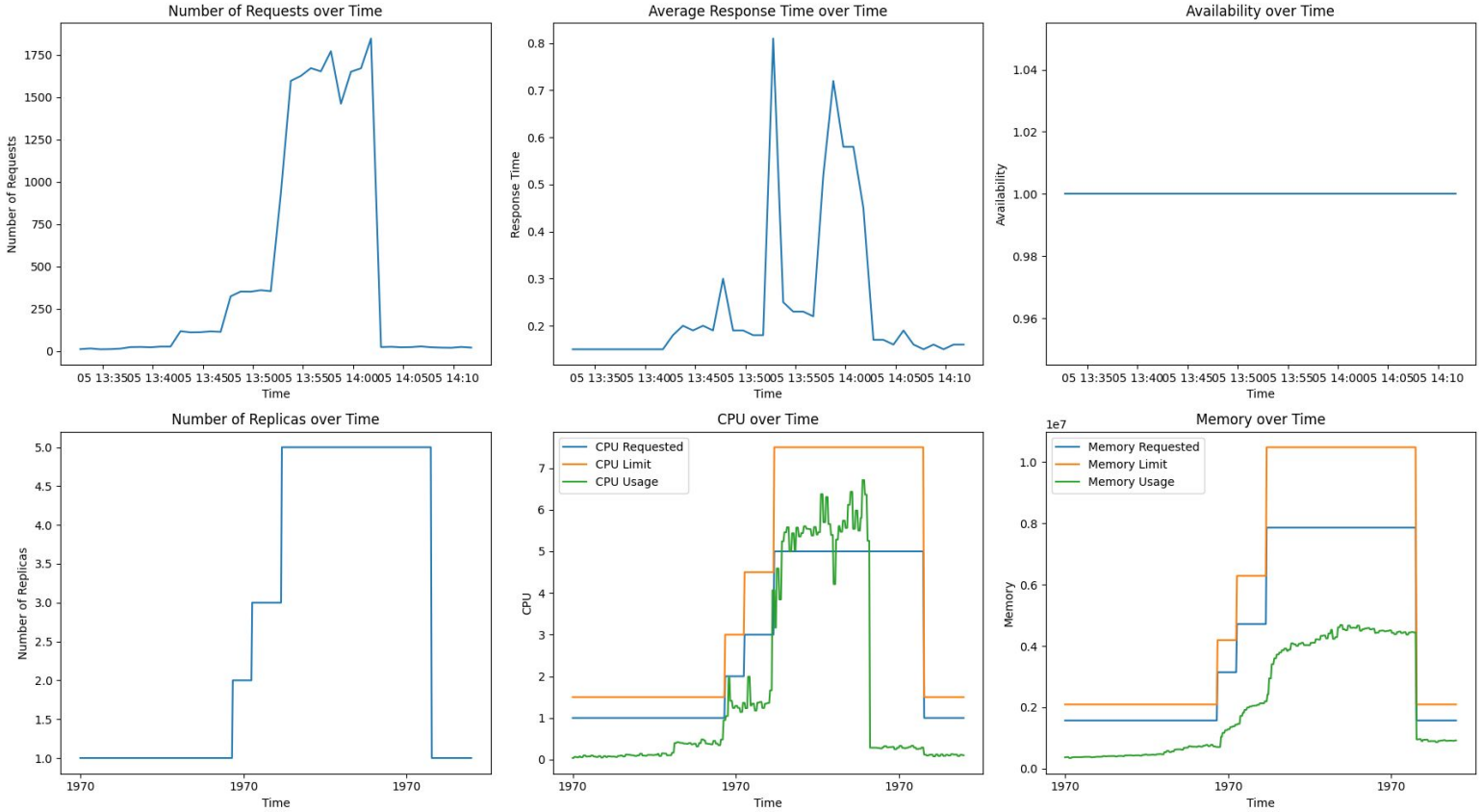
SLIDE 6: Visualize the deployment for accuracy model over time



Explanation

- **Availability**:Despite the high load periods, the availability over time remains 100% suggesting a resilient service that can handle variable load.
- **Replicas:** The number of replicas adjusts flexibly in response to changing load.It scales up during high traffic periods and scales down during low traffic periods.
- **Response time:**When requests are high, the response time is longer, which is obvious.
- **Cpu and memory use:** The CPU usage plots show minor over bound. Possibly due to extra usage than expected.The memory usage plots show that the actual resource usage (mem_use_KB) follow the requested resources (mem_req_KB), indicating efficient resource management.The resource limits (cpu_lim_core and mem_lim_KB) are set higher than the actual usage, allowing for flexibility.

SLIDE 7: visualize the deployment for latency model over time



Explanation

- **Availability**:Despite the high load periods, the availability over time remains 100% suggesting a resilient service that can handle variable load.
- **Replicas:** The number of replicas adjusts flexibly in response to changing load.It scales up during high traffic periods and scales down during low traffic periods.
- **Response time:**When requests are high, the response time is longer, which is obvious.
- **Cpu and memory use:** The CPU and memory usage plots show that the actual resource usage (cpu_use_core and mem_use_KB) follow the requested resources (cpu_req_core and mem_req_KB), indicating efficient resource management.The resource limits (cpu_lim_core and mem_lim_KB) are set higher than the actual usage, allowing for flexibility.

SUMMARY

- Predecessor had opted for VGG16 which achieved an accuracy of 0.67
- I had chosen ResNet50V2 for accuracy model that achieved an accuracy of 0.85
- I had chosen MobileNet for latency model that achieved an accuracy of 0.84
- While the accuracy model performed well on own test image, the latency model wasn't very great.
- On single pod and max-pod the latency model had better response time and transaction rate as expected due to its lightweightedness.
- The horizontal scaling gave results as expected, however slight over shooting of cpu usage for latency model than requested.