Report

Over two months, I dove deep into the world of data science, focusing on the exciting fields of deep learning model development, optimization, and deployment. My main goal was to achieve high performance on Kaggle competitions, which led me to push the boundaries of what I knew, especially using TensorFlow and AWS SageMaker's flexible features.

At the heart of my work was creating a model that could accurately classify images. I used advanced techniques like convolutional neural networks and tried out different complex designs, including VGG16(fig 1), ResNet50 , ResNet101(fig 2),ResNet152 all equipped with ImageNet weights. These models were great because they use something called skip connections, which help the model learn from huge amounts of data without getting overwhelmed.

Adjusting hyperparameters was a major learning point for me. This task was tough but really opened my eyes to how tiny tweaks could make or break a model's accuracy. I played around with settings like how fast the model learns (learning rates), how much information it ignores (dropout percentages), and how it avoids being too specific to the training data (regularization).

It was a balancing act, and getting it right was key to making the models work well.
Dealing with the high computational needs of these models was another hurdle. AWS SageMaker was a lifesaver here, giving me the chance to use different kinds of processing power (like CPU and GPU instances) as needed. I started by using both CPU and GPU power but quickly realized that the cost and power of moderate GPUs didn't always match the task at hand. So, I learned to be smarter with resources, saving the heavy GPU power for only the training phase and using CPUs for the lighter work.

I also explored using TPUs (Tensor Processing Units), which are incredibly efficient for large-scale machine learning projects. Although my project didn't heavily rely on TPUs, just learning about their potential was an eye-opener for me.

When it came to making my models ready for the real world, AWS SageMaker played a big role again. Moving from developing models to launching them online involved choosing the right setup, setting up endpoints, and handling the technical details of deployment. This experience showed me how cloud services like AWS S3 simplify managing data and can help keep costs down while ensuring everything runs smoothly.

Looking back, I see how much I've grown not only in my technical abilities but also in how I approach problems. I made plenty of mistakes along the way, like underestimating the importance of carefully choosing hyperparameters or not initially understanding the best ways to manage computational resources. Each mistake was a lesson, teaching me to be persistent, adaptable, and always eager to learn more. In sharing this journey with you, I want to highlight not only the technical skills I've gained but also my passion for tackling tough challenges and my dedication to bringing creative and effective solutions to your team.

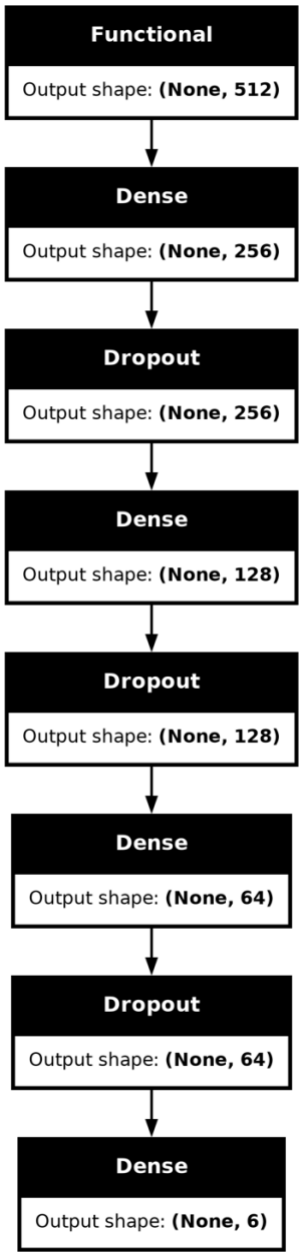

fig 1

Total params: 14,714,688

Challenges/Mistakes	Solutions
Hyperparameter Optimization	Adjusting the parameters for how quickly the model learns and how long it learns for was tricky. At first, the settings weren't quite right, so the model didn't perform well. I had to change

Challenges/Mistakes	Solutions
	things to find the best balance between accuracy and how long it takes to compute(fig 2).
Data Augmentation Techniques	Using techniques to make the model better at understanding different types of data was tough because it needed a lot of compute power. I had to use the techniques carefully to make sure they helped without using too many resources.
GPU Utilization in SageMaker	I started off using a pretty strong instance for all the work, which cost more than I needed to spend. Later, I switched to using different computers for different jobs, which helped me save money and do things faster.
AWS S3 and SageMaker Integration	I had some problems because my data storage and my notebook server instance were in different places, which made things more expensive. When I put them in the same place, it was easier and cheaper
Serverless Inference Deployment	Putting the model into action was challenging in the fact that weights file wasn't getting read properly. I had to set up everything very carefully multiple times in the end to set up the endpoint and made sure it worked well. Going thorough a lot of aws documentation and github got me through this final hurdle(fig 3).
Instance Selection for Training vs. Preprocessing	I didn't realize at first that different jobs needed different kinds of compute power. Figuring that out helped me to use resources better and save money. I used one kind of compute resoruce for regular work and a different kind for the hard and computationally expensive stuff(fig 4).
Data Transfer Costs	I didn't realize how much it would cost to move our data between different places. This taught me that we need to be careful about how we move data around, especially when we're using computers far away from each other(fig 5a, 5b, 5c).

Challenges/Mistakes	Solutions
	fig 5a shows the mistake i made, by creating the notebook server in a different region.
Learning Curve with SageMaker	SageMaker had a lot of choices, which made it hard to figure out what to do at first. I spoke to other people who use it and read the aws documentation which helped me understand how to use it better and get the most out of it.

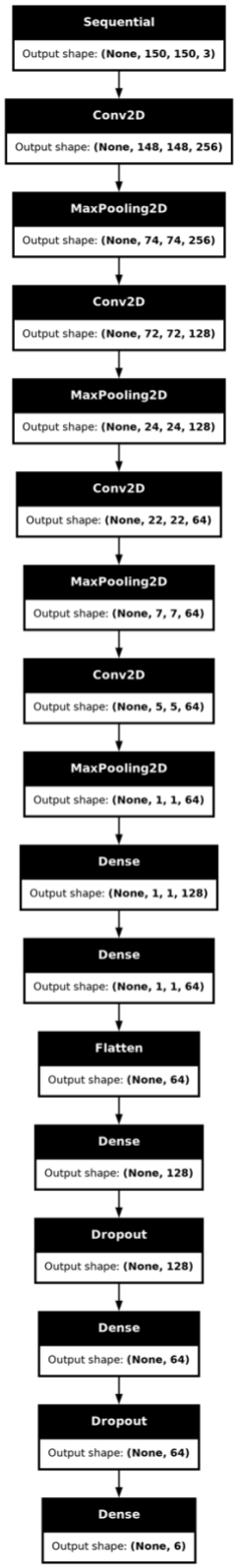
Sequential
Output shape: (None, 150, 150, 3)
Conv2D
Output shape: (None, 148, 148, 256)
MaxPooling2D
Output shape: (None, 74, 74, 256)
Conv2D
Output shape: (None, 72, 72, 128)
MaxPooling2D
Output shape: (None, 24, 24, 128)
Conv2D
Output shape: (None, 22, 22, 64)
MaxPooling2D
Output shape: (None, 7, 7, 64)
Conv2D
Output shape: (None, 5, 5, 64)
MaxPooling2D
Output shape: (None, 1, 1, 64)
Dense
Output shape: (None, 1, 1, 128)
Dense
Output shape: (None, 1, 1, 64)
Flatten
Output shape: (None, 64)
Dense
Output shape: (None, 128)
Dropout
Output shape: (None, 128)
Dense
Output shape: (None, 64)
Dropout
Output shape: (None, 64)
Dense
Output shape: (None, 6)

fig 2

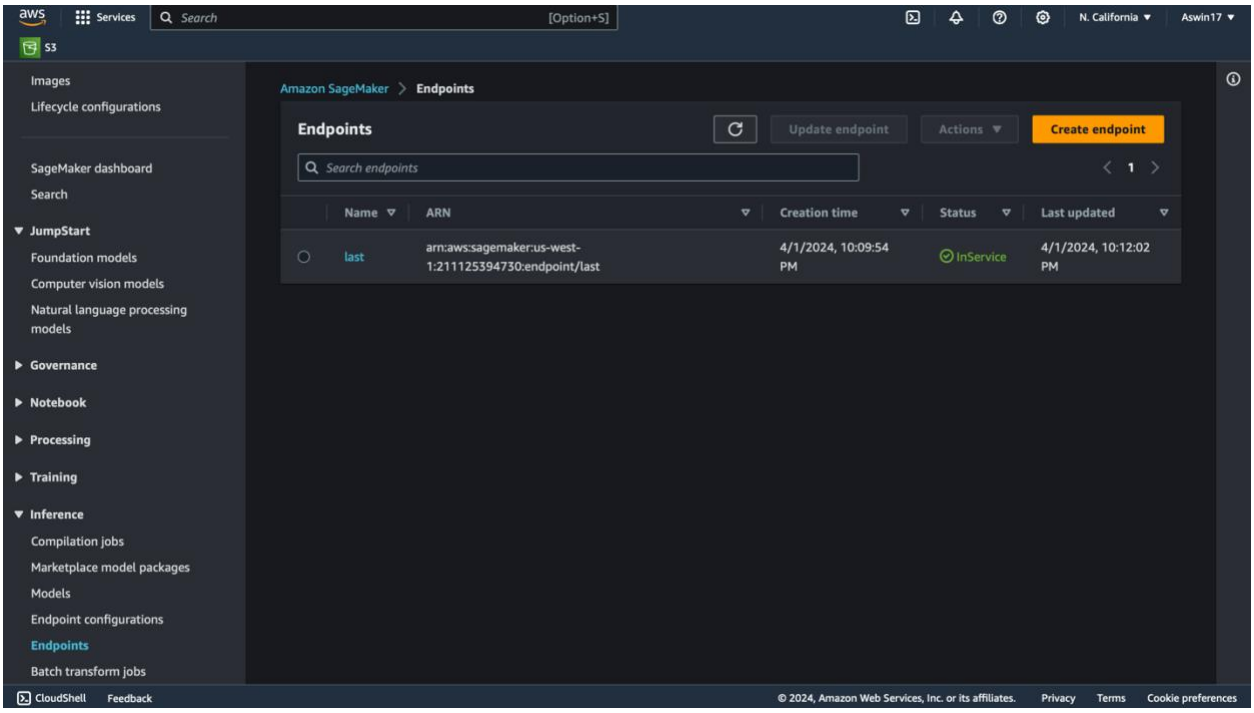
aws
Services
Search
[Option+S]
N. California
Aswin17
S3
Images
Lifecycle configurations
SageMaker dashboard
Search
JumpStart
Foundation models
Computer vision models
Natural language processing models
Governance
Notebook
Processing
Training
Inference
Compilation jobs
Marketplace model packages
Models
Endpoint configurations
Endpoints
Batch transform jobs
Amazon SageMaker
Endpoints
Endpoints
Update endpoint
Actions
Create endpoint
Search endpoints
1
Name
ARN
Creation time
Status
Last updated
last
arn:aws:sagemaker:us-west-1:211125394730:endpoint/last
4/1/2024, 10:09:54 PM
InService
4/1/2024, 10:12:02 PM
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

fig 3

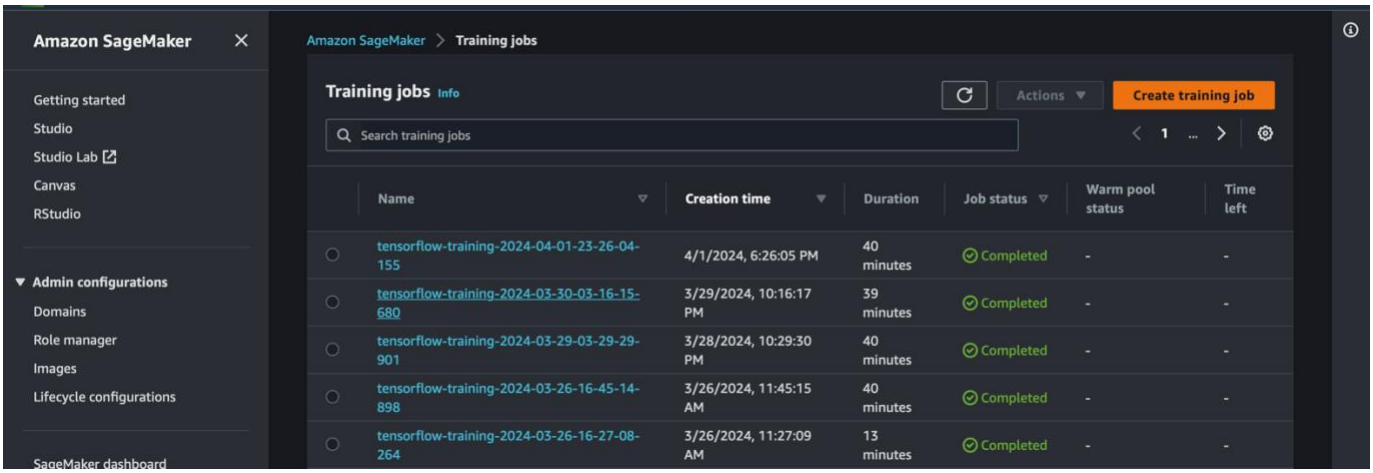
Amazon SageMaker
Getting started
Studio
Studio Lab
Canvas
RStudio
Admin configurations
Domains
Role manager
Images
Lifecycle configurations
SageMaker dashboard
Amazon SageMaker > Training jobs
Training jobs Info
Actions
Create training job
Search training jobs
Name
Creation time
Duration
Job status
Warm pool status
Time left
tensorflow-training-2024-04-01-23-26-04-155
4/1/2024, 6:26:05 PM
40 minutes
Completed
tensorflow-training-2024-03-30-03-16-15-680
3/29/2024, 10:16:17 PM
39 minutes
Completed
tensorflow-training-2024-03-29-03-29-29-901
3/28/2024, 10:29:30 PM
40 minutes
Completed
tensorflow-training-2024-03-26-16-45-14-898
3/26/2024, 11:45:15 AM
40 minutes
Completed
tensorflow-training-2024-03-26-16-27-08-264
3/26/2024, 11:27:09 AM
13 minutes
Completed

fig 4

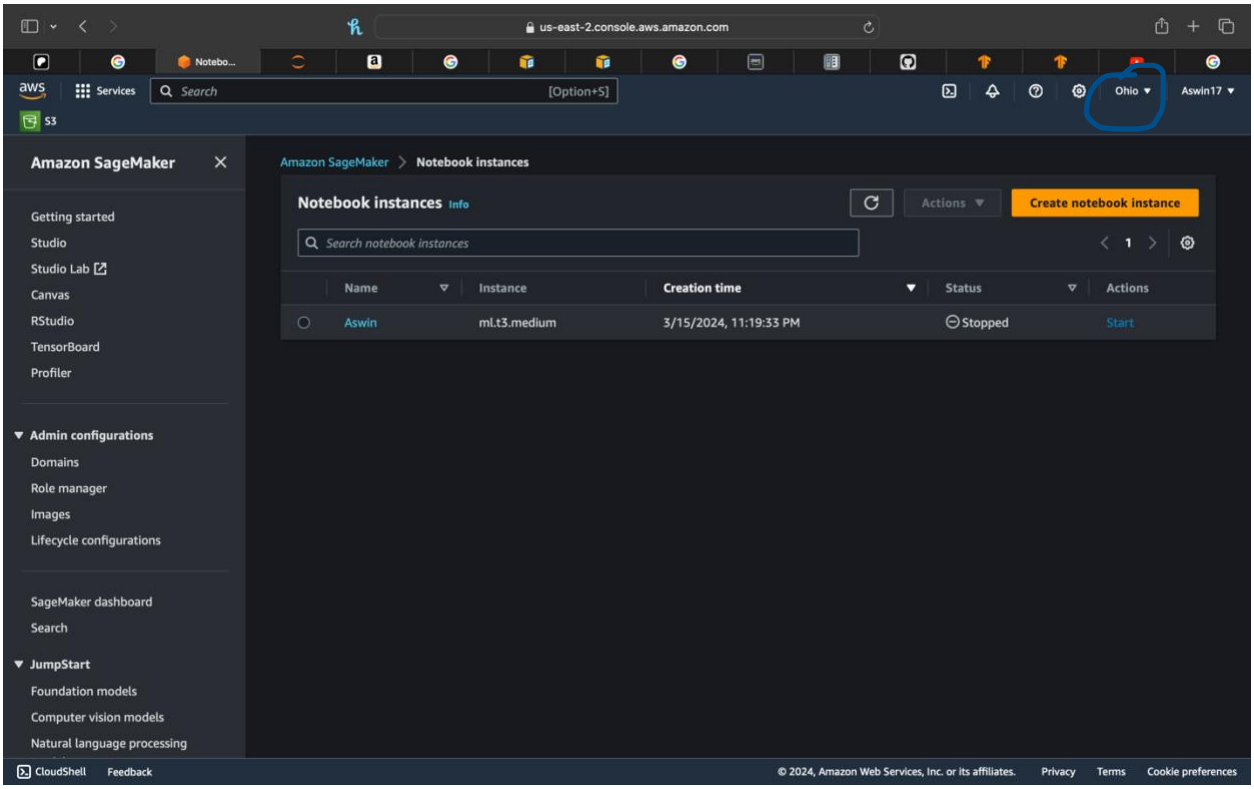
us-east-2.console.aws.amazon.com
Services
Search
[Option+S]
Ohio
Aswin17
S3
Amazon SageMaker
Getting started
Studio
Studio Lab
Canvas
RStudio
TensorBoard
Profiler
Admin configurations
Domains
Role manager
Images
Lifecycle configurations
SageMaker dashboard
Search
JumpStart
Foundation models
Computer vision models
Natural language processing
Amazon SageMaker > Notebook instances
Notebook instances Info
Actions
Create notebook instance
Search notebook instances
Name
Instance
Creation time
Status
Actions
Aswin
ml.t3.medium
3/15/2024, 11:19:33 PM
Stopped
Start
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

fig 5a

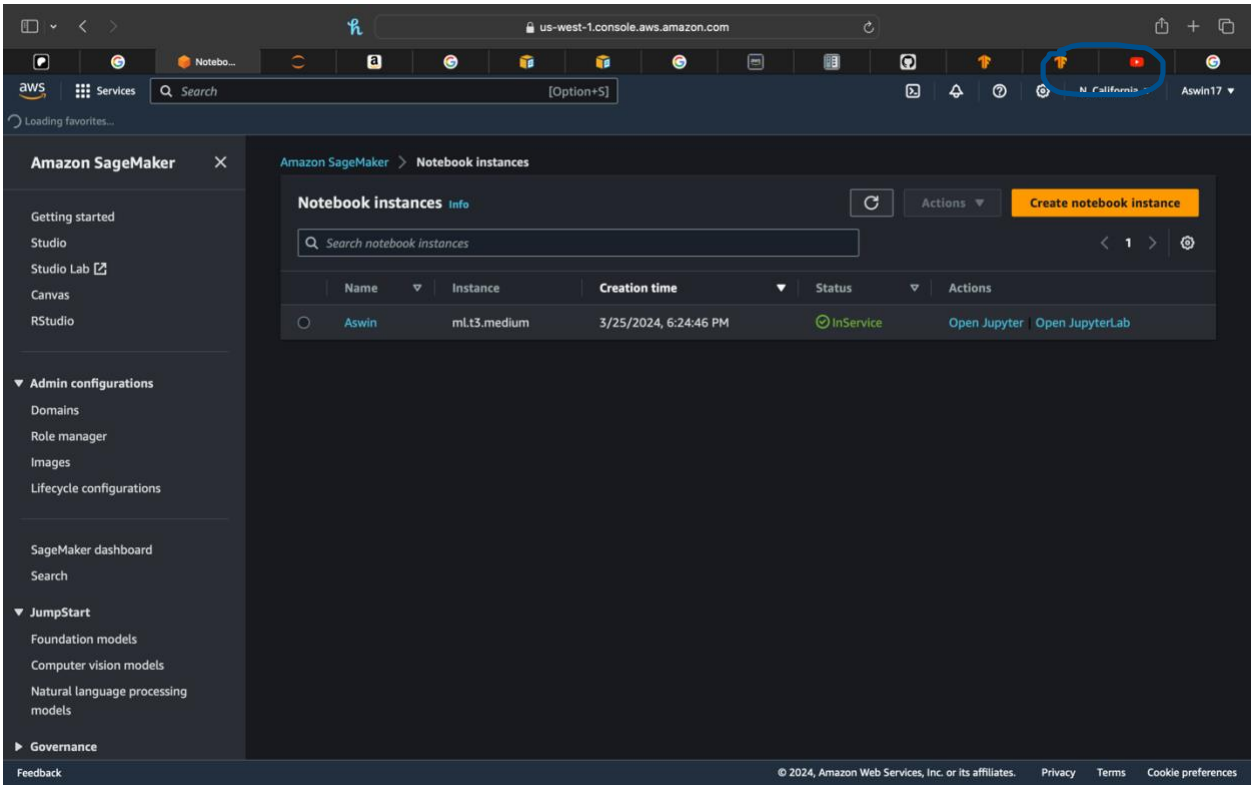
us-west-1.console.aws.amazon.com
Notebo...
aws
Services
Search
[Option+S]
Aswin17
Loading favorites...
Amazon SageMaker
Getting started
Studio
Studio Lab
Canvas
RStudio
Admin configurations
Domains
Role manager
Images
Lifecycle configurations
SageMaker dashboard
Search
JumpStart
Foundation models
Computer vision models
Natural language processing models
Governance
Amazon SageMaker > Notebook instances
Notebook instances Info
Actions
Create notebook instance
Search notebook instances
1
Name
Instance
Creation time
Status
Actions
Aswin
ml.t3.medium
3/25/2024, 6:24:46 PM
InService
Open Jupyter
Open JupyterLab
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

fig 5b

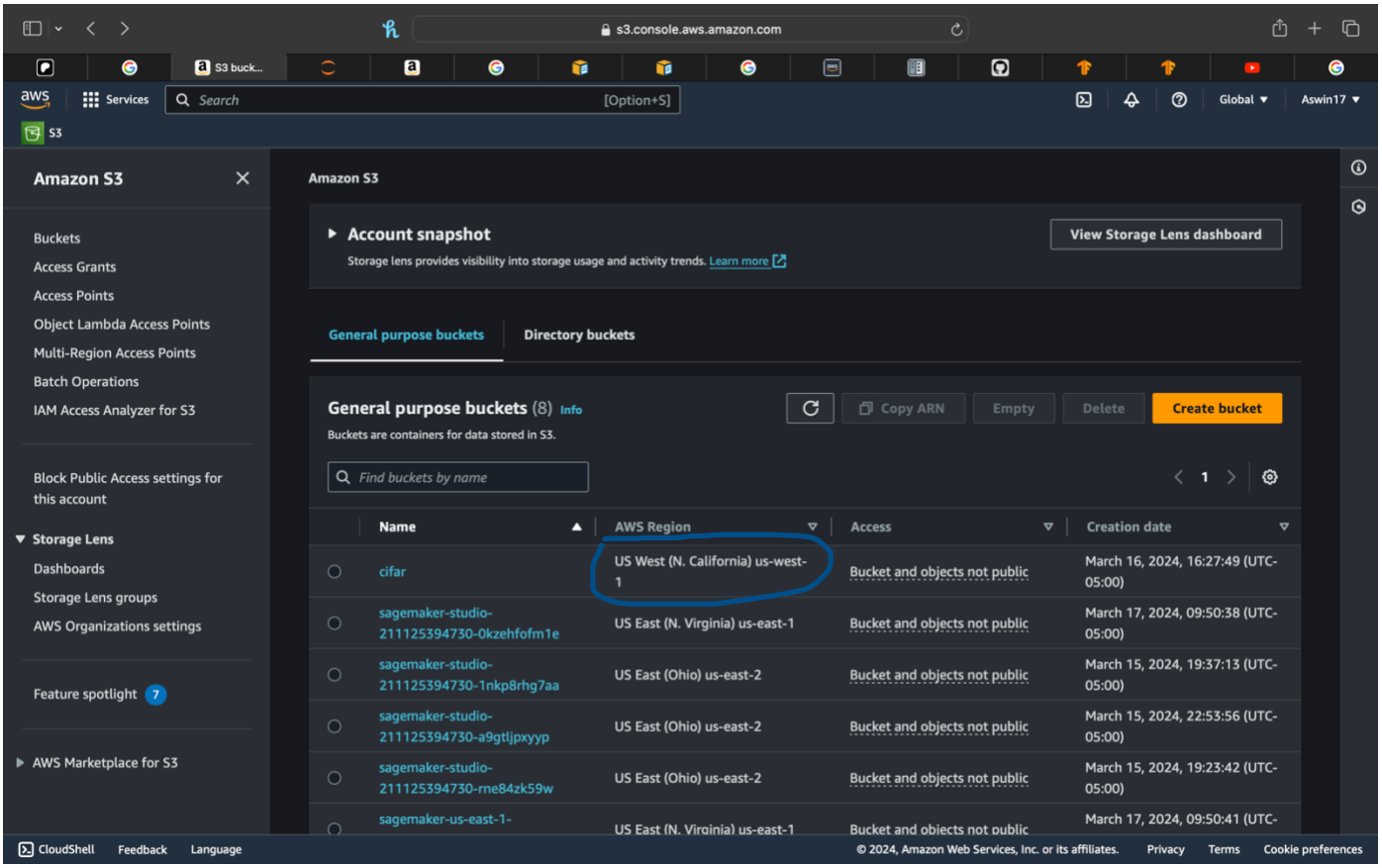

fig 5c

Objective:

I aimed to create a model that could accurately identify what's in pictures—like distinguishing mountains, streets, buildings, glaciers, forests, and seas. I used Convolutional Neural Networks (CNNs) and a technique called transfer learning to make the program learn better and make more accurate guesses. Then deploy the same thing in aws sagemaker utilizing all the end to end pipelines provided by them.

How I Used AWS SageMaker:

AWS SageMaker, a powerful tool from Amazon, was incredibly useful. It helped speed up the learning process made it easy to get it ready for real-world use, making the whole process faster and smoother. I used to s3 to aquire data , preprocess it , train the model and hyperparameter tune it and then deploy the model in a serveless inference environment wherein a single input image is sent and the output is the result.

Data:

The total training images are around 14,230 which are divided into 80% training and 20% as test

The test images are around 3000. The distribution of the classes among the datasets are given(fig 6, fig 7)).

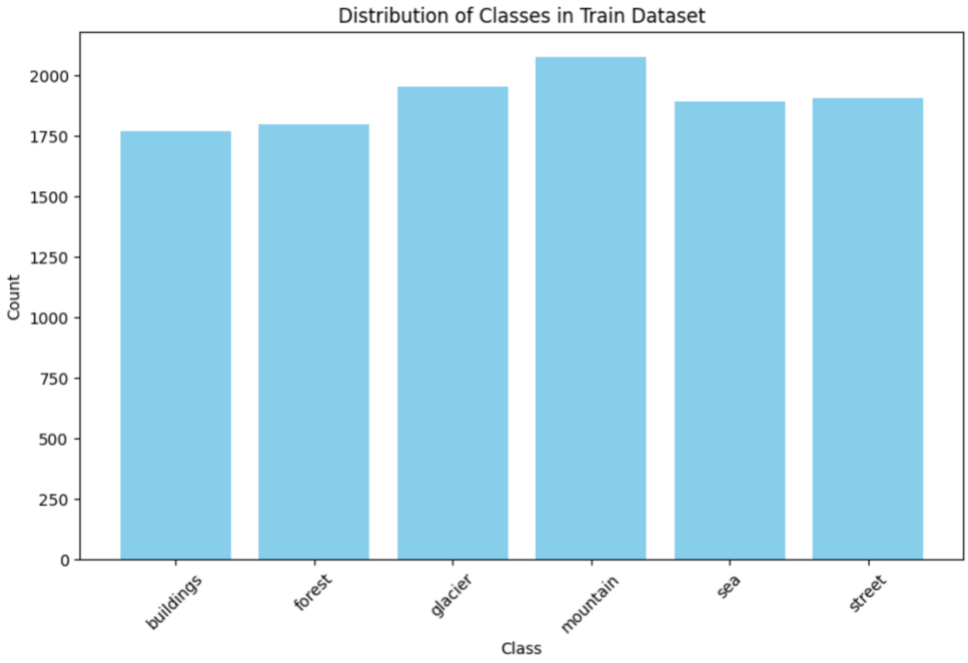

fig 6

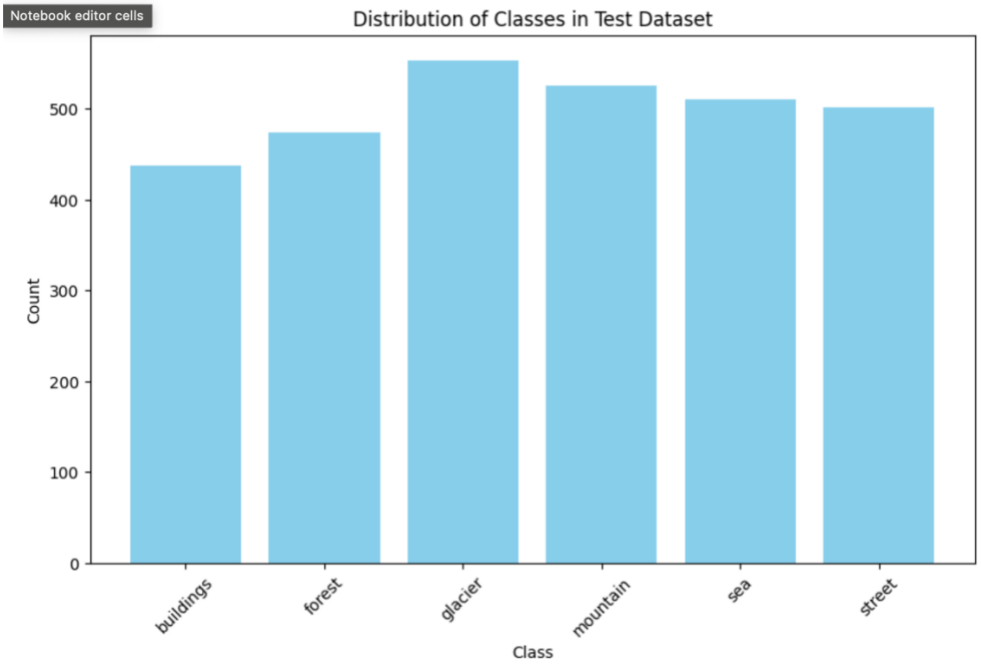

fig 7

Tools I Chose:

For this project, I worked with several important tools. TensorFlow helped me build the models and optimize them. AWS SageMaker was essential for making everything run efficiently. Python was the programming language I used to write my code.

Training :

My approach to model accuracy in the beginning was to use base CNN and go from there to hyper parameter tune the model, with many variable parameters, after that it involved integrating ImageNet weights into architectures such as ResNet50, ResNet101, Inception Network, and VGG16. This strategy harnessed their advanced feature extraction capabilities, significantly enhancing our model's accuracy.

Results:

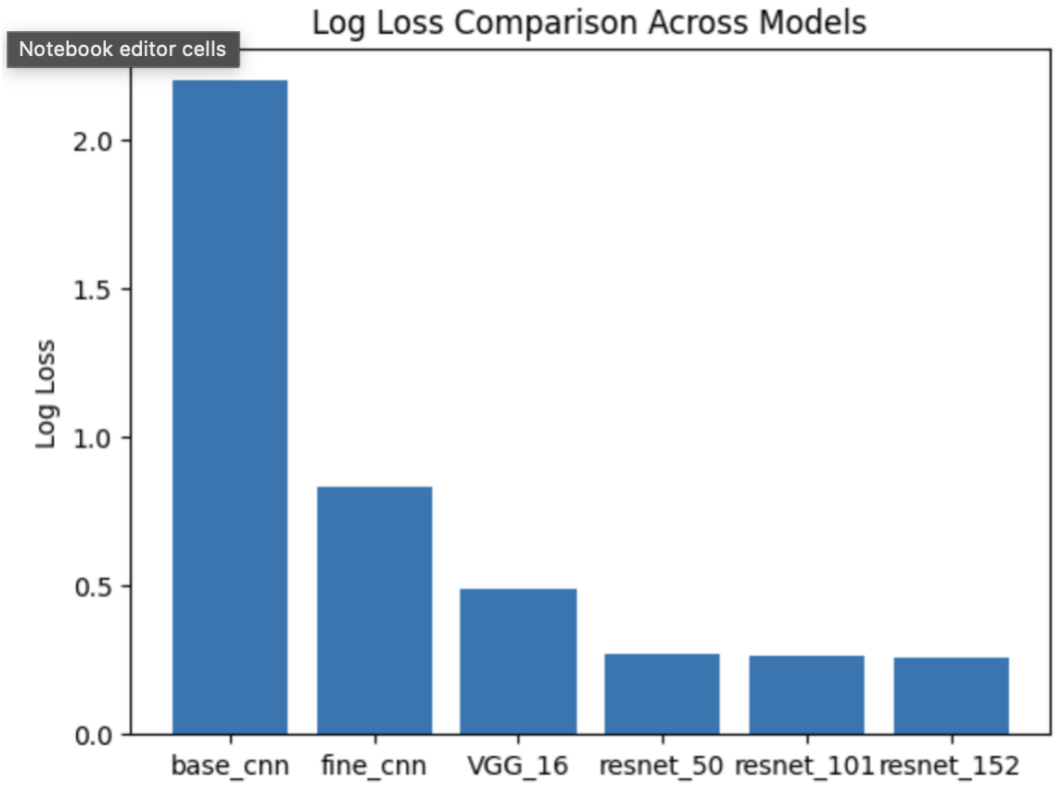

fig 8

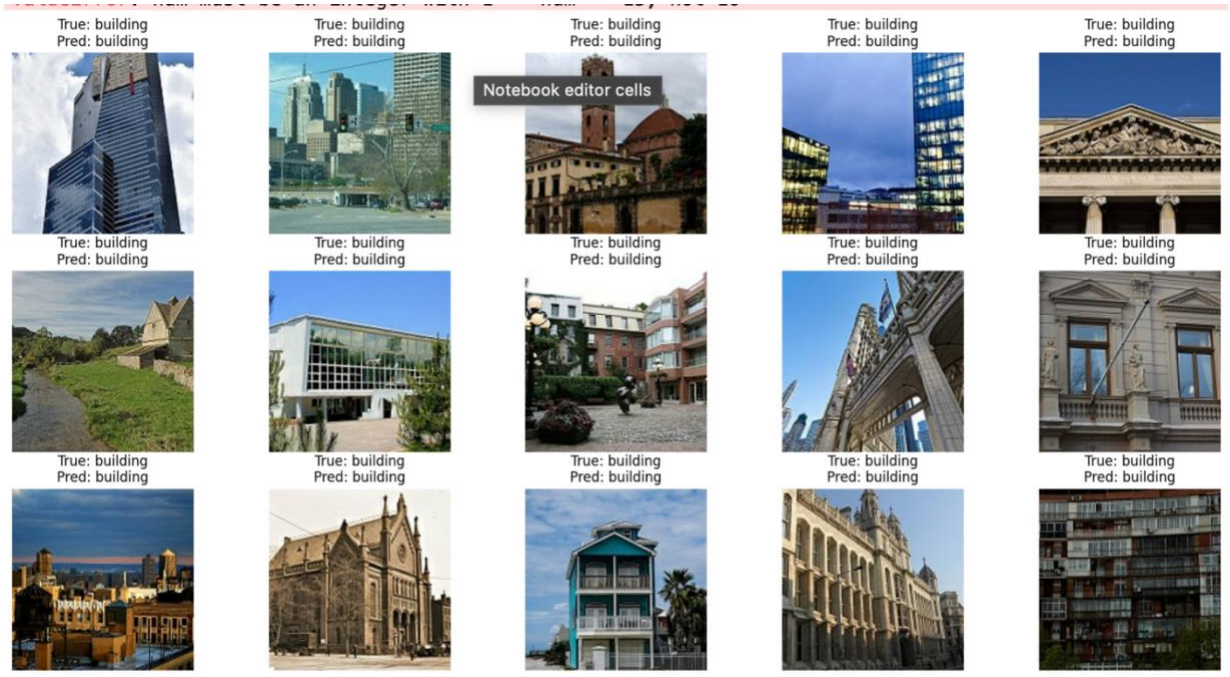
True: building
Pred: building
Notebook editor cells

fig 9


1/1 11s 11s/step
1/1 0s 36ms/step
1/1 0s 31ms/step
1/1 0s 30ms/step
1/1 0s 29ms/step
1/1 0s 32ms/step
True: building
Pred: building
True: forest
Pred: forest
True: glacier
Pred: glacier
True: mountain
Pred: mountain
True: sea
Pred: sea
True: street
Pred: street
Notebook editor cells

fig 10

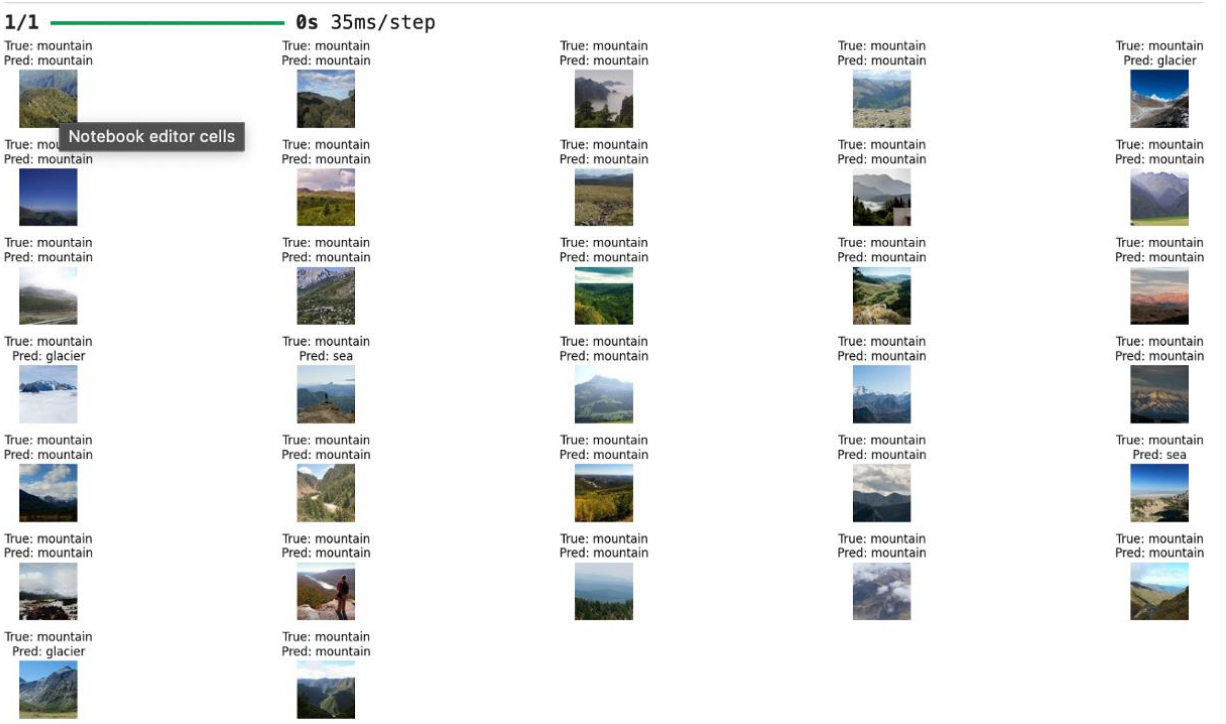

fig 11

The first simple model I made (base_cnn) wasn't very good at guessing, as it made quite a few mistakes. When I made some changes to this model (fine_cnn) and tuned , it got a lot better, and the number of mistakes it made went down by half.

Then, I tried some complex models, like VGG_16, which were already trained a bit with lots of images. This one was even better at guessing the right answers and made fewer mistakes. But the best ones were the ResNet programs—ResNet_50, ResNet_101, and ResNet_152. Each of these was better than the last, with ResNet_152 making the fewest mistakes of all.

This tells me that the more complex ResNet programs are really good for this kind of job. They've learned from lots of images, so they're pretty smart at figuring out new ones they haven't seen before.

Future Work:

Moving forward, there's plenty more to do. Since ResNet_152 did the best job, I'll keep working with it to see if it can get even better. There might be ways to teach and train it even more so that it can understand images faster or more accurately.

I've also put one of these models on AWS SageMaker. That means it can now work on its own without needing a lot of help from me. I'll also keep an eye on how it's doing to make sure it keeps getting better at predicitng right. If it starts making more mistakes, I'll tweak it to get it back on track. Plus, I'm going to explore new tools and techniques that might come out.

Lastly, I'll make sure that the model doesn't just do well in practice, but that it's also cost-effective to use.

Key Learnings and Outcomes:

- Not underestimating the impact of hyperparameters
- The importance of computational resource management
- Understanding the nuances of model training and deployment
- Understanding how Gpu’s and Tpu’s improve performance in terms of speed, and learnt working in a multi Gpu cluster environment.
- Understanding AWS sagemaker end to end pipeline.
- Learning about complex architectures such as resnets, vgg, and inception networks