PROJECT REPORT

CNN MODEL DEPLOYMENT ON AWS S3

THARUN KUMAR SRINIVASAN

Introduction

Convolutional Neural Networks (CNNs) are widely used in image classification tasks. This project involves training a CNN model, saving it, and deploying it using AWS S3 for storage and retrieval. The objective is to enable remote access to the trained model for inference and further development.

Project Objectives

- Train a CNN model for image classification.
- Store the trained model securely in **AWS S3**.
- Retrieve and use the model for image classification.
- Ensure efficient management of AWS resources to avoid unnecessary costs.

System Requirements

1. AWS Requirements

To use AWS for this project, the following services and configurations are required:

- **AWS Account** (Free-tier available)
- **AWS S3** (Amazon Simple Storage Service) for storing the trained model.
- **AWS IAM User & Access Keys** for authentication.
- **AWS SageMaker** (Optional, if deploying an endpoint).
- **AWS EC2** (Optional, if training the model on AWS instead of Google Colab).

2. Python Requirements (Google Colab)

Ensure the following Python libraries are installed in Google Colab:

```
!pip install boto3
!pip install tensorflow
```

- **boto3** - AWS SDK for Python to interact with S3.
- **tensorflow** - For training and loading the CNN model.
- **tarfile** - For compressing and decompressing the model files.

Steps Involved

1. Training the CNN Model

- Implement a CNN model using TensorFlow/Keras.
- Train the model on an image dataset.
- Save the trained model as `cnn_model.h5`.

2. Compressing and Uploading the Model to AWS S3

```
import tarfile
import boto3

# Define filenames
model_dir = "cnn_model_saved"
tar_filename = "cnn_model_saved.tar.gz"

# Compress the model directory
with tarfile.open(tar_filename, "w:gz") as tar:
    tar.add(model_dir, arcname=".")

# Upload to S3
s3 = boto3.client("s3")
s3.upload_file(tar_filename, "tharun-cnn-bucket-aws", tar_filename)

print(f"✅ Model uploaded as {tar_filename} to S3 bucket!")
```

3. Downloading and Extracting the Model from AWS S3

```
import boto3
import tarfile

# AWS S3 Configurations
s3 = boto3.client("s3")
bucket_name = "tharun-cnn-bucket-aws"
tar_filename = "cnn_model_saved.tar.gz"

# Download the model
s3.download_file(bucket_name, tar_filename, tar_filename)

# Extract the model
with tarfile.open(tar_filename, "r:gz") as tar:
    tar.extractall("cnn_model_saved")
print("✅ Model downloaded and extracted successfully!")
```

4. Using the Model for Classification

```
from tensorflow.keras.models import load_model
import numpy as np
from tensorflow.keras.preprocessing import image

# Load the model
model = load_model("cnn_model_saved/cnn_model.h5")

# Load and preprocess an image for classification
def classify_image(image_path):
    img = image.load_img(image_path, target_size=(224, 224))
    img_array = image.img_to_array(img)
    img_array = np.expand_dims(img_array, axis=0)
    img_array /= 255.0

    prediction = model.predict(img_array)
    print("Prediction:", prediction)

# Example usage
classify_image("sample_image.jpg")
```

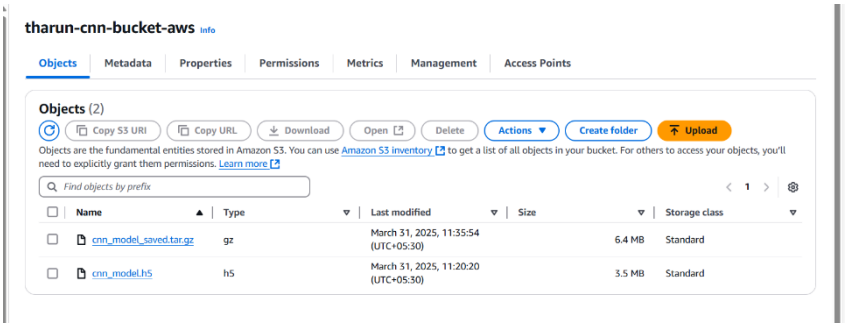
AWS Domains Used

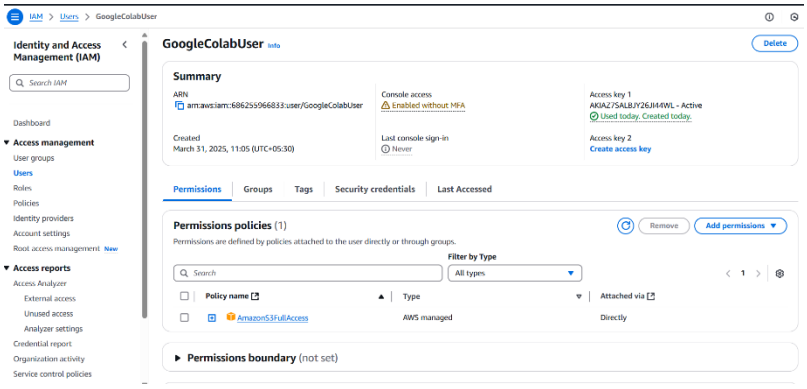
- **Amazon S3**: Used for storing and retrieving the model.
- **AWS IAM**: Used for authentication (Access Keys required).
- **AWS SageMaker** : For deploying an endpoint.
- **AWS EC2** : For training if Google Colab is not used.

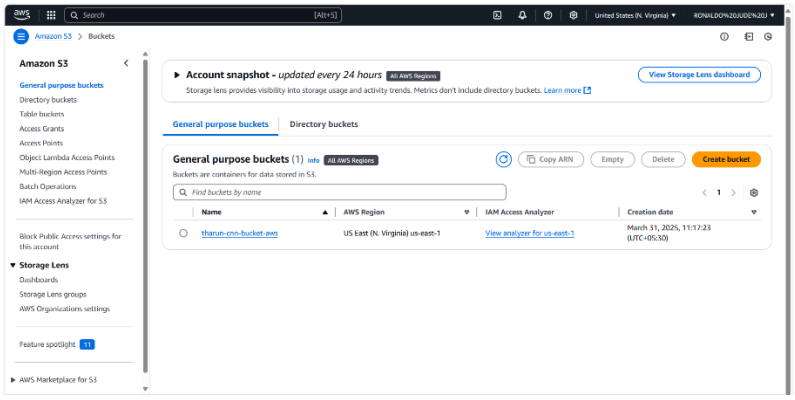
Screenshots

(Add the following screenshots to the report)

1. AWS S3 Bucket with Uploaded Model (`cnn_model_saved.tar.gz`)



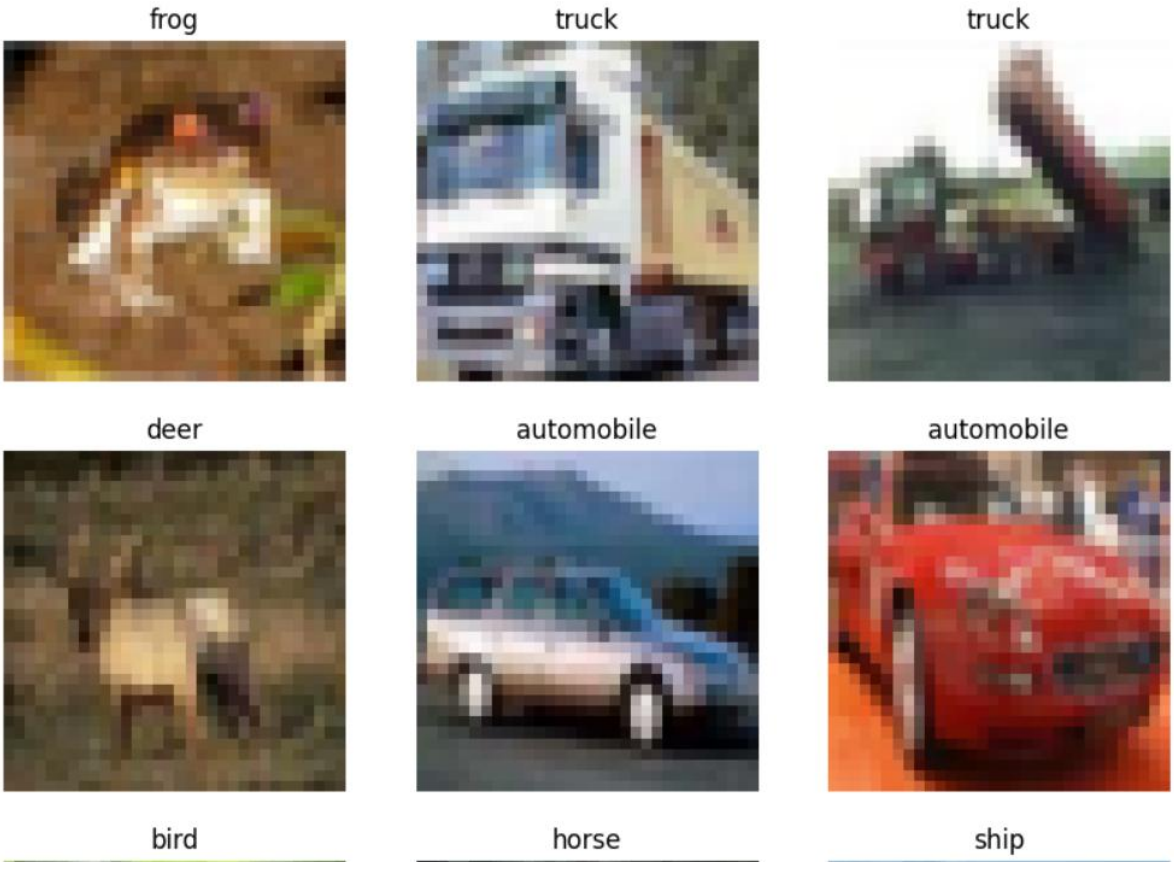




2. Google Colab Execution Output (Uploading and Downloading Steps)

```
Epoch 1/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 66s 82ms/step - accuracy: 0.3789 - loss: 1.7205 - val_accuracy: 0.5457 - val_loss: 1.2868
Epoch 2/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 80s 80ms/step - accuracy: 0.5932 - loss: 1.1646 - val_accuracy: 0.6195 - val_loss: 1.0996
Epoch 3/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 84s 83ms/step - accuracy: 0.6558 - loss: 0.9909 - val_accuracy: 0.6578 - val_loss: 0.9886
Epoch 4/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 77s 77ms/step - accuracy: 0.6931 - loss: 0.8798 - val_accuracy: 0.6843 - val_loss: 0.9110
Epoch 5/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 83s 78ms/step - accuracy: 0.7175 - loss: 0.8143 - val_accuracy: 0.6982 - val_loss: 0.8815
Epoch 6/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 62s 79ms/step - accuracy: 0.7406 - loss: 0.7393 - val_accuracy: 0.6791 - val_loss: 0.9446
Epoch 7/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 83s 80ms/step - accuracy: 0.7607 - loss: 0.6894 - val_accuracy: 0.6826 - val_loss: 0.9464
Epoch 8/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 79s 77ms/step - accuracy: 0.7839 - loss: 0.6184 - val_accuracy: 0.7069 - val_loss: 0.8582
Epoch 9/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 61s 78ms/step - accuracy: 0.8003 - loss: 0.5727 - val_accuracy: 0.6972 - val_loss: 0.9204
Epoch 10/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 83s 80ms/step - accuracy: 0.8195 - loss: 0.5269 - val_accuracy: 0.7073 - val_loss: 0.9198
```

3. Model Classification Output Example



Conclusion

This project successfully demonstrates how to train, compress, upload, and retrieve a CNN model using **Google Colab and AWS S3**. By following best practices, we ensure cost efficiency and scalability for real-world applications.