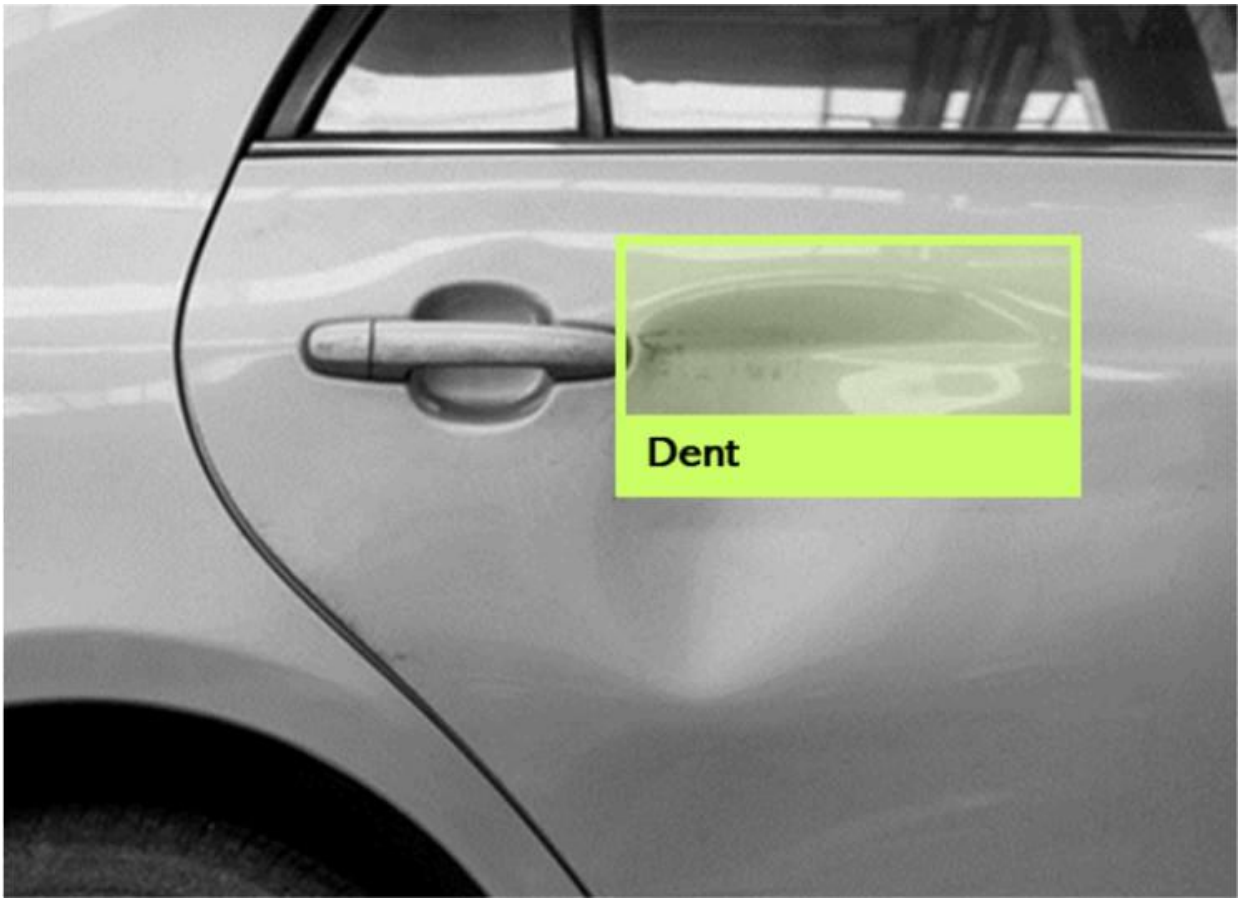
Carsdepo Damage Detection Case study

MLOps_CV_Case_Study

ML C-064 – Subrat Kumar Biswal

GitHub: https://github.com/subratb252/MLOps_Cv_Case_Study



Contents

Q1. System design: Describe the KPI that the business should track.

Answer: Below are the KPI's that business should track.

1. **Damage Detection Accuracy:**
    - **Precision:** The proportion of correctly identified damage instances (scratches/dents) out of all instances identified as damage.

$$\text{Precision} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Positives (FP)}}$$

    - **Recall:** The proportion of correctly identified damage instances out of all actual damage instances present in the images.

$$\text{Recall} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Negatives (FN)}}$$

    - **F1-Score:** The harmonic mean of precision and recall, providing a balanced measure of accuracy.
    - **Mean Average Precision (mAP):** A widely used metric for object detection, measuring the average precision at different recall levels.

$$\text{mAP} = \frac{1}{N} \sum_{i=1}^{N} \text{AP}_i$$

        - AP_i: Average Precision for class *i*
        - N: Number of classes (2 here — scratches and dents)
        - Typically computed at different IoU thresholds like 0.5, 0.75, etc. (e.g., mAP@0.5)

2. **Model Inference Speed:**

- **Latency:** The time taken to process an image and return damage detection results. This is crucial for real-time or near-real-time applications.
- **Throughput:** The number of images processed per unit of time.

$$\text{Throughput} = \frac{\text{Total number of images processed}}{\text{Total time taken (seconds)}}$$

  - Often expressed as images per second (img/sec).

3. **Customer Satisfaction:**
   - **Reduction in pricing disputes:** Measure the decrease in complaints or disputes related to inaccurate car valuations.

$$\text{Dispute Reduction \%} = \frac{\text{Old Disputes} - \text{New Disputes}}{\text{Old Disputes}} \times 100$$

   - **Increase in transaction speed:** Track the reduction in the time taken to complete car transactions.

$$\text{Speed Improvement \%} = \frac{\text{Old Transaction Time} - \text{New Transaction Time}}{\text{Old Transaction Time}} \times 100$$

4. **Operational Efficiency:**
   - **Reduction in manual inspection costs:** Quantify the savings from replacing manual labor with automated inspection.

$$\text{Cost Reduction \%} = \frac{\text{Old Cost} - \text{New Cost}}{\text{Old Cost}} \times 100$$

- **Percentage of automated vs. manual inspections:** Track the proportion of inspections handled by the ML system.

$$\text{Automation } \% = \frac{\text{Images Inspected by ML}}{\text{Total Inspections}} \times 100$$

---

5. **Data Quality and Coverage:**
    - **Percentage of labelled images:** Track the amount of images that have been labelled, and improve that number.

$$\text{Labelled Image } \% = \frac{\text{Number of Labelled Images}}{\text{Total Images}} \times 100$$

    - **Distribution of damage types:** Ensure the model is performing well across all types of damage.
    - **Data drift detection:** Track any changes in the image data distribution over time.

Q2. System design: Your company has decided to build an MLOps system. What advantages would you get from building an MLOps system rather than a simple model?

Answer:

1. Automated Data Pipelines for Image Ingestion and Preprocessing	
Simple Model	**MLOps**

1. Automated Data Pipelines for Image Ingestion and Preprocessing	
Would require manual image processing, leading to bottlenecks and inconsistencies.	Automates the ingestion of millions of car images, resizing, normalization, and augmentation. This ensures consistent data quality and reduces manual effort, which is critical for handling the large volume of images.

2. Scalable Model Training and Experiment Tracking	
Simple Model	**MLOps**
Training would be manual and time-consuming, hindering the ability to experiment and optimize the model effectively.	Enables automated training of the YOLOv8 model on cloud GPUs, with experiment tracking to optimize hyperparameters and improve accuracy. This allows for rapid iteration and model improvement, which is essential to meet the 1-2% tolerance level.

3. Automated Model Deployment and Real-Time Inference	
Simple Model	**MLOps**

3. Automated Model Deployment and Real-Time Inference	
Deployment would be manual and complex, leading to delays and potential errors.	Automates the deployment of the trained model to cloud servers and mobile devices, ensuring real-time damage detection. This is crucial for providing instant feedback to users and streamlining the car valuation process.

4. Continuous Model Monitoring and Maintenance	
Simple Model	**MLOps**
Would lack robust monitoring capabilities, making it difficult to detect and address performance issues.	Provides continuous monitoring of model performance (precision, recall, MAP), latency, and data drift. This enables Carsdepo.com to detect and address performance degradation proactively, ensuring the model remains accurate and reliable.

5. Automated Retraining with New Labelled Data	
Simple Model	**MLOps**

5. Automated Retraining with New Labelled Data	
Retraining would be manual and time-consuming, hindering the model's ability to adapt to new data.	Automates the retraining of the model when new labelled data becomes available, ensuring the model stays up-to-date and accurate. This is essential for handling the evolving nature of car damage and improving model performance over time.

6. Version Control and Reproducibility	
Simple Model	**MLOps**
Would lack version control, making it difficult to track changes and reproduce results.	Tracks model versions, code, and data, ensuring reproducibility and enabling easy rollback if needed. This is crucial for maintaining model integrity and complying with regulatory requirements.

7. Improved Collaboration and Efficiency	
Simple Model	**MLOps**

7. Improved Collaboration and Efficiency	
Would lead to communication gaps and inefficiencies due to manual processes and lack of collaboration tools.	Facilitates collaboration between data scientists, ML engineers, and operations teams with standardized workflows and tools. This improves efficiency and reduces the time taken to develop and deploy the damage detection system.

Q3. System design: You must create an ML system that has the features of a complete production stack, from experiment tracking to automated model deployment and monitoring.

For this problem, create an ML system design (diagram)

Answer:

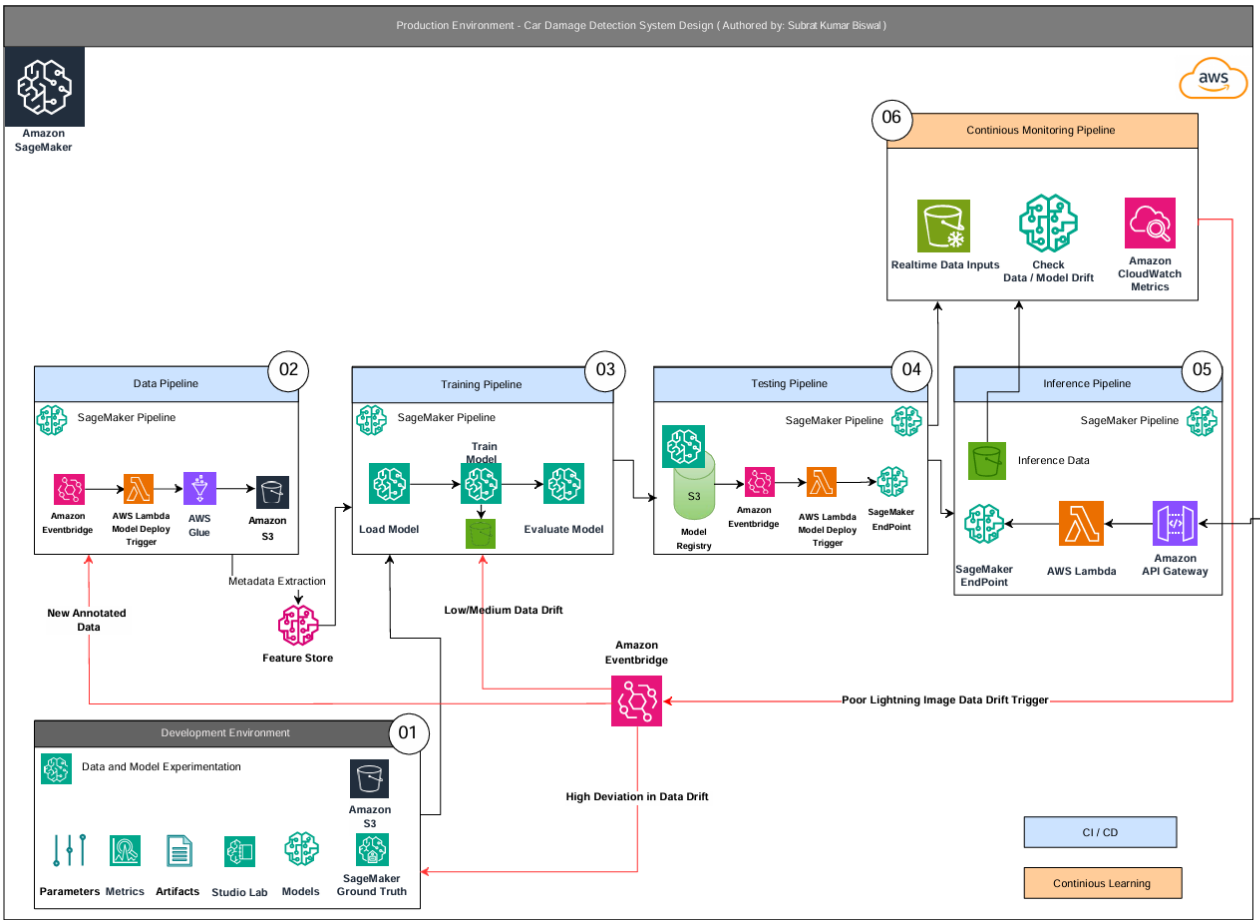


Diagram in PDF - CarDepo_Object_Detection.drawio.pdf

https://github.com/subratb252/MLOps_Cv_Case_Study/blob/main/MLOps_Cv_Case_Study.drawio.pdf

Q4. System design: After creating the architecture, please specify your reason for choosing the specific tools you chose for the use case.

Overall the choice of tools are cloud based managed services specially **Amazon SageMaker.**

Why Amazon SageMaker Instead of Other MLOps Tools?

1. **Fully Managed –** Amazon SageMaker is a powerful, fully managed machine learning (ML) service that simplifies the entire ML lifecycle, from data preparation and model building to training, deployment, and monitoring, allowing users to focus on innovation rather than infrastructure management
2. **End-to-End ML Lifecycle –** Handles everything from preprocessing to deployment.
3. **Built-in Experiment Tracking & Monitoring –** Eliminates the need for MLflow, Prometheus, or TensorBoard.
4. **Seamless Integration with AWS Services –** Works smoothly with S3, Lambda, CloudWatch, and API Gateway.
5. **Scalability –** Can handle millions of car images effortlessly.

Below detailed tables present reasons for choosing each tool and the benefit of that by each stages of end to end ML System.

Component / Stage	Tools Chosen	Reason
Development Environment - Experiment Tracking		
Data Storage	Amazon S3	Stores raw car images (`/raw/`) and labels, Supported and works natively with AWS Service.
Image Classification and Labelling Service	Amazon SageMaker Ground Truth	Human or Semi Automated Labelling as we know there are some images which are not labeled. This helps to label Image data with Bounding Box.
Data exploration / EDA	Amazon SageMaker Studio	Explore image quality, metadata, class distribution
Experiment Tracking	SageMaker Experiments	Tracks Models, Hyperparams, Metrics across multiple trials.
Initial Model Training	SageMaker Training Jobs	Trains and Evaluate Yolo, Faster RCNN
Metric Visualization	SageMaker Debugger	Debug Performance and Training Issues
Data Pipeline		
Event based Data processing Trigger	EventBridge, AWS Lambda	Event-Driven function that triggers preprocessing of new

		Images when uploaded into S3.
Image Preprocessing and Augmentation	Amazon SageMaker Processing Jobs	It provides scalable compute resources to preprocess Images using **OpenCV** and **TensorFlow**
Metadata extraction & integration	AWS Glue / Feature Store	Managed ETL (Extract, Transform, Load) service that integrates structured metadata with image files.
Storage for Images and metadata	Amazon SageMaker Projects and Pipeline Managed service, It uses S3 for storage	To Store processes images back into storage. Scalable, cost-effective, and secure storage for large datasets (millions of images). It would integrate well with SageMaker stages further.
Training Pipeline Details		
Experiment Tracking	Amazon SageMaker Studio - Experiments and Trials	Automatically tracks different training runs, hyperparameters, and performance metrics. Allows comparison of different object detection models (**YOLOv8, Faster R-CNN, etc.**).

Model Training	Amazon SageMaker Training Jobs	Provides **GPU-accelerated training** without managing infrastructure. Enables Conditional logic (e.g re-run training if if performance is below threshold)
Testing Pipeline		
Model Versioning and Registry for Production	Amazon SageMaker - Model Registry, S3 Storage for Models	Stores and manages different versions of models. Ensures the latest and best model is always deployed.
Model Deployment and Trigger for Stage and Production Deployment	Amazon Eventbridge	**Triggers deployment** to a staging or production endpoint **Notifies teams** about model updates (via email/Slack/etc.) **Logs the event** for audit and traceability
Model Deployment and Trigger for Stage and Production Deployment	AWS Lambda	Lambda functions are triggered by EventBridge when a new model version is registered or approved in the registry. **Deploying the model** to

		a SageMaker endpoint **Running validation tests** before promotion to production
Hosting Model to Endpoint for Staging	Amazon SageMaker EndPoint	This helps to host model to the endpoints and help users to test and consume.
Inference Pipeline		
Hosting Model to Endpoint for Production	Amazon SageMaker Studio - Endpoints	Fully managed API for serving real-time predictions at scale. Supports **multi-model endpoints** if multiple versions of the model are required.
External Users Consuming Production Model	Amazon API Gateway, Lamba, EventBridge	When external users try to consume Production via Web App, Amazon Lambda gets triggered via API Gateway and Run the Production Model.
Inference Data Storage	Amazon S3	Inference data gets stored into S3 and Passed on to the Monitoring Pipeline further.
Data and Model Monitoring		
Detection Drift and Performance Issues	Amazon SageMaker Model Monitor	Continuously monitors **data distribution shifts**

		and alerts when drift is detected. Ensures the model remains accurate over time.
Logging and Alerting	Amazon CloudWatch	Collects logs, metrics, and sends alerts when issues occur. Allows monitoring of **inference latency, errors, and resource utilization**.
Triggering Retraining on New Image Data	Amazon EventBridge	Detects when new labeled images are available in the Original Directory of S3 and triggers retraining. Also based on Drift % deviation it helps to navigate to a particular stage.

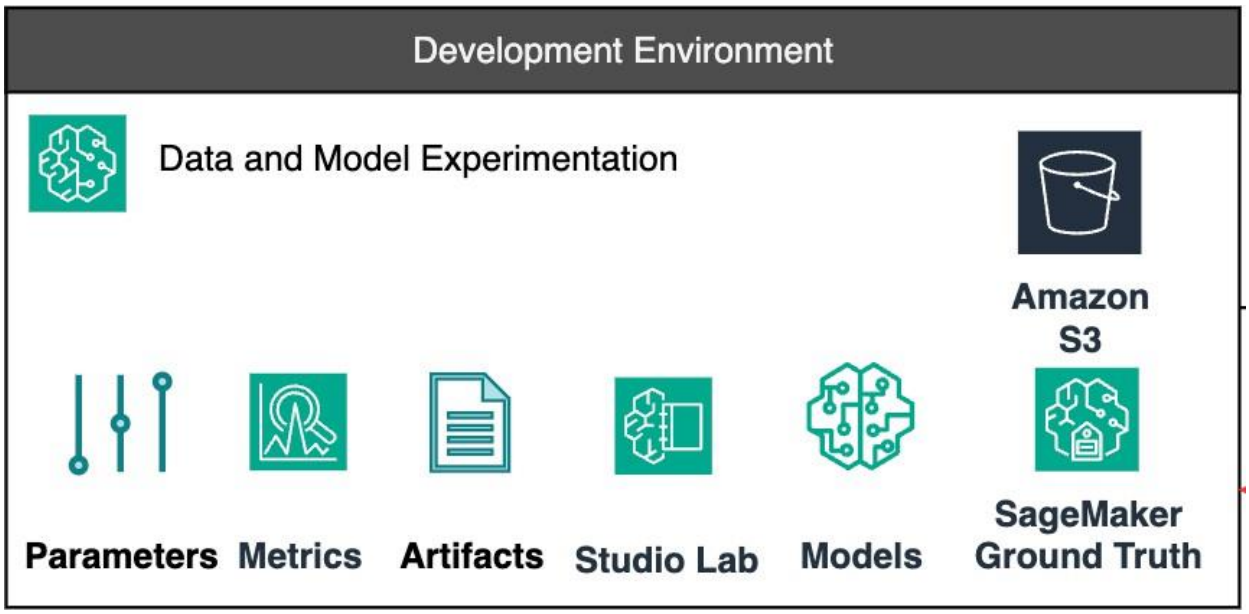
Q5. Workflow of the solution:

Answer:

We have selected an Amazon Managed Service solution to address the given requirement. We can understand the solution of workflow in below steps.

Step 1: Data and Model Experimentation

Purpose: We would be developing an object detection model manually by trial and error methods. We would try testing different hyperparameters and analysing data sets.



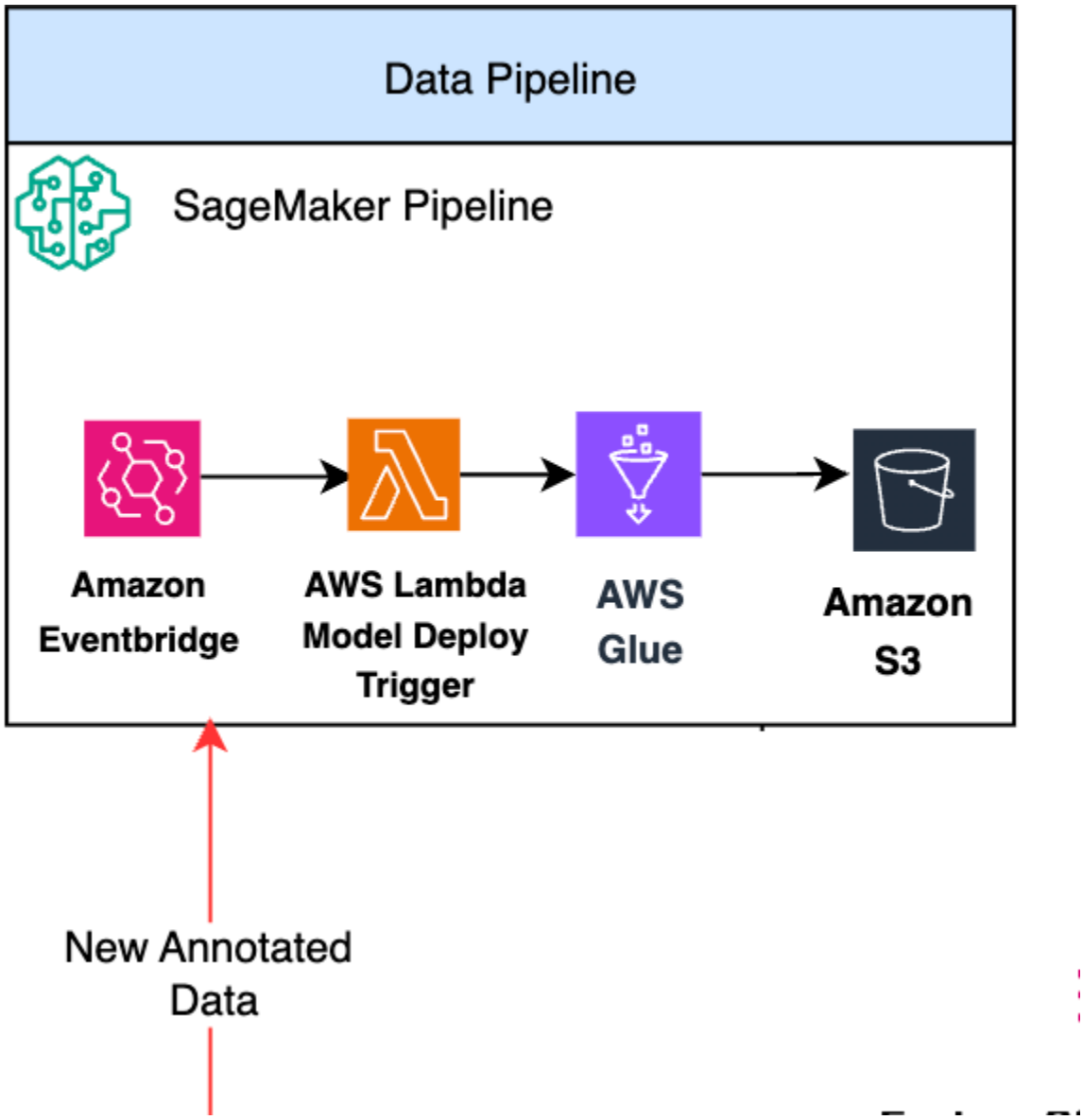
Component/Stage	**Tools**	**How it connects / Why Used**
Data Storage	Amazon S3	Stores raw car images (`/raw/`) and labels
Labelling Service	Amazon SageMaker Ground Truth	Human or Semi Automated Labelling as we know there are some images which are not labelled. This helps to label Image data with Bounding Box.

Data exploration / EDA	Amazon SageMaker Studio	Explore image quality, metadata, class distribution
Experiment Tracking	SageMaker Experiments	Tracks Models, Hyperparams, Metrics across multiple trials.
Initial Model Training	SageMaker Training Jobs	Trains and Evaluate Yolo, Faster RCNN
Metric Visualization	SageMaker Debugger	Debug Performance and Training Issues

Connection Summary:

1. Image data goes through SageMaker Ground Truth and Labelling applies to data (scratches and Dent)
2. Training runs are initiated from SageMaker Studio and Experiments are tracked.
3. Model Artifacts are stored in S3 which will be later used in Registry.
4. EDA and Metric Visualization are techniques used for exploring Image quality and class distribution.

Step 2: Automation of Data Pipeline



Component/Stage	Tools	How it connects / Why Used
Trigger for New Data	Amazon EventBridge	Detects new file upload

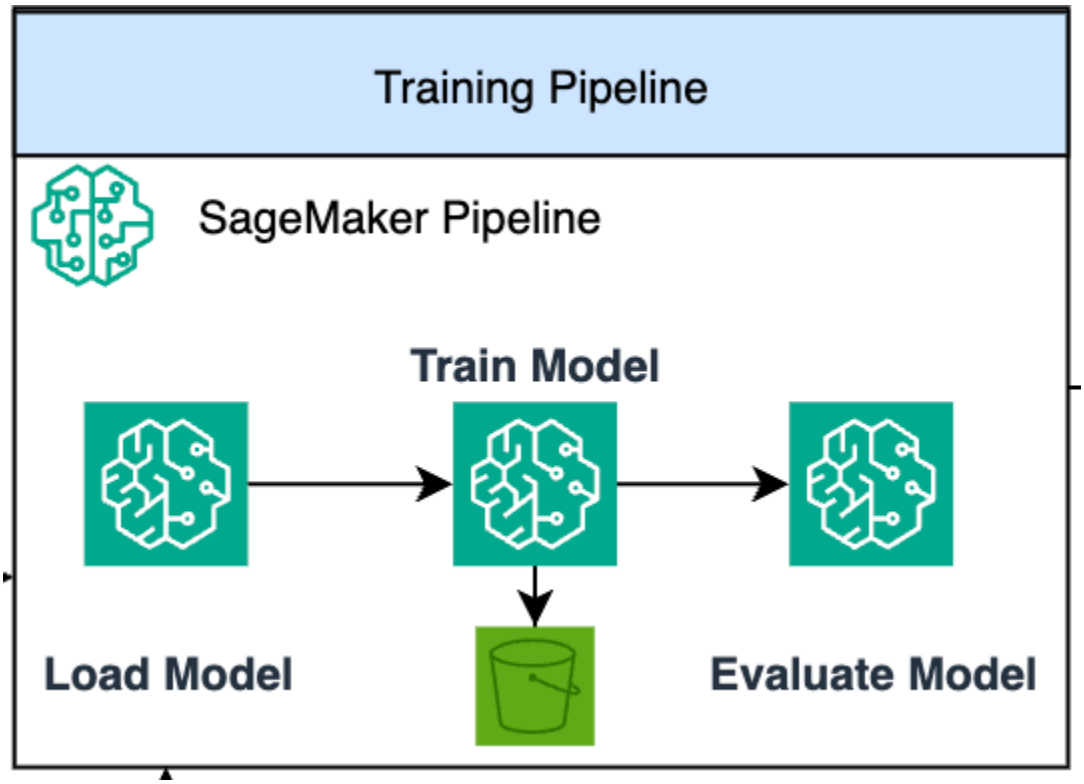
		in S3 (raw/)
Processing Trigger	AWS Lambda	Starts data processing on new Image upload
Data Processing	AWS Glue	Resizes, Augmentation on Image Data
Processed Data Storage	Amazon S3	Cleaned/Augmented data goes into this dir for training

Connection Summary:

1. Eventbridge detects image upload and triggers lambda
2. Lambda launches SageMaker Processing Job for Image Augmentation and Resizes
3. Output gets stored back in Amazon S3 (e.g S3/processed/ directory)

Step 3: Automation of Training Pipeline

Purpose: Automatically train and register a new model when data is updated or triggered manually.



Component/Stage	**Tools**	**How it connects / Why Used**
Training Orchestrator	Amazon SageMaker Pipeline	Coordinating pipeline steps
Loading Model	SageMaker Load Model	Load the Model from the Development environment or Use SageMaker Existing Model. E.g YOLOv8
Model Training	SageMaker Training Jobs	Train Object Detection Model
Model Evaluation	Script in Pipeline Step	Compute Precision, Recall, F1, mAP and analyze calculated metrics.

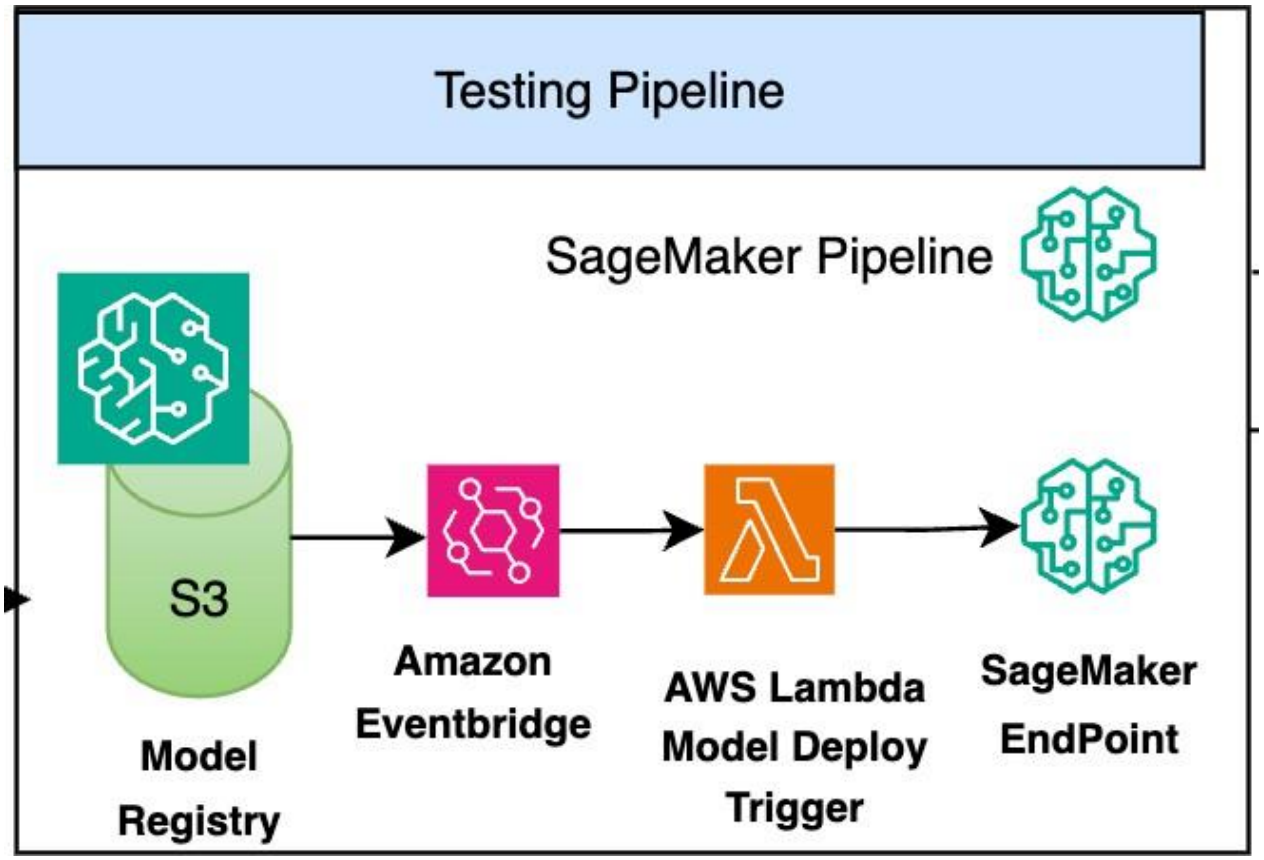
		Generating evaluation reports and visualizations. Evaluating the impact of different thresholds on the model's performance.
Output artifact storage	Amazon S3	Stores Model, Logs and Metrics (s3/processed)

Connection Summary:

1. Loading the best model as identified during experimentation, SageMaker also provides pre pre-existing Image based Object detection model.
2. Model Training using Amazon SageMaker Training Jobs. AWS Glue can be used for the same.
3. Model Evaluation is applied on Trained model. And metrics can be evaluated like Precision, Recall and F1 Score via Script and output results usage.
4. Output would be stored back to Amazon s3.

Step 4: Automation of Testing Pipeline (Staging Environment Validation and Deployment of Best Model to Production)

Purpose: This is the stage where the Model selected from previous stage gets deployed to Staging environment.



Component/Stage	Tools	How it connects / Why Used
Model Registry	Amazon SageMaker Model Registry	Registers model version with metadata
Model Metadata stored in S3	Amazon S3	Model Registry metadata stored in S3

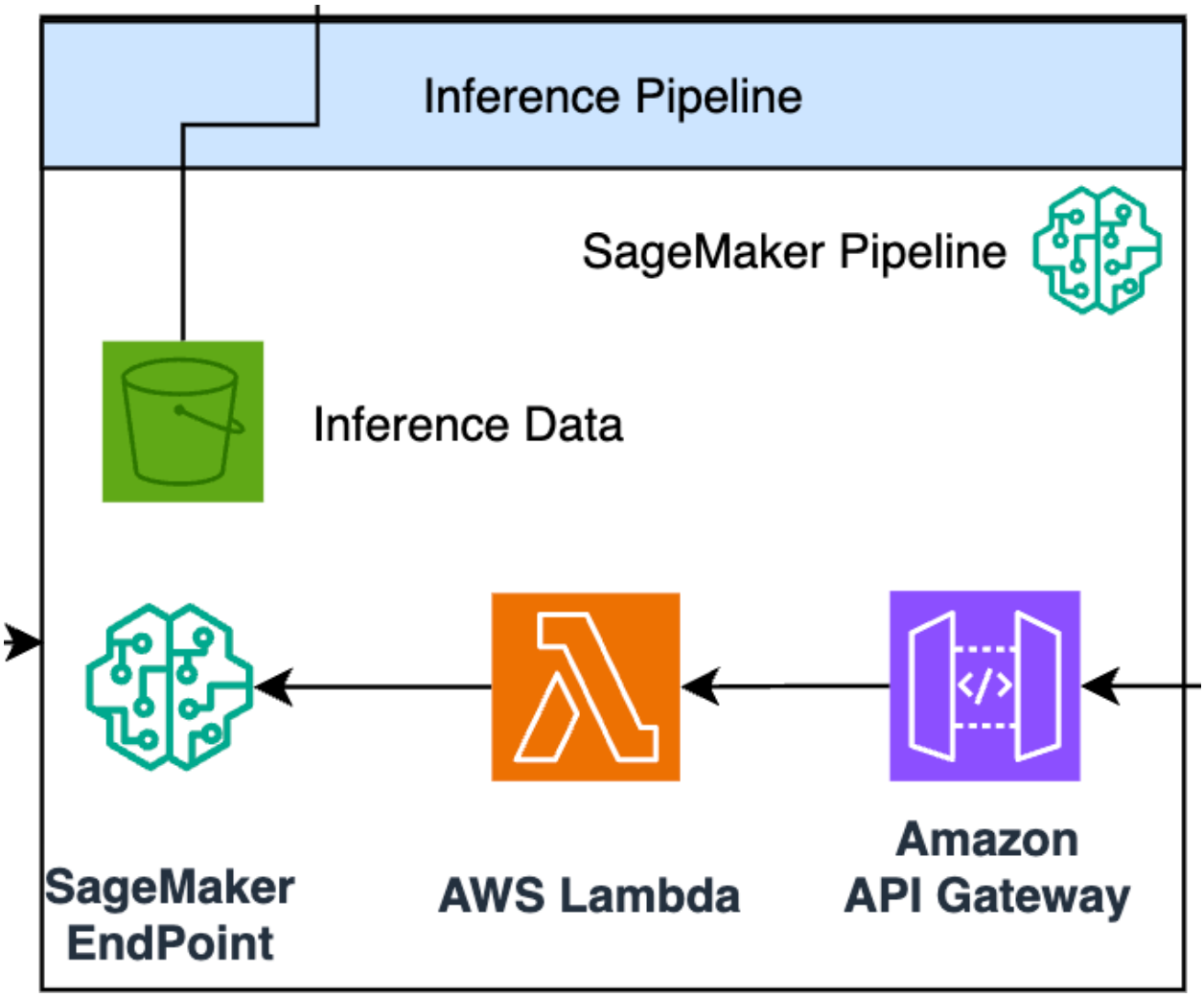
Model Deployment Trigger	Amazon Eventbridge	Manually or Automated via evaluation marking best model for the Deployment to Staging, Same after validation Trigger for Production Deployment
Deployment Launcher	AWS Lambda	Deploys approved model to endpoint
Staging / Production Inference Hosting	Amazon SageMaker Endpoint	Tester can test realtime prediction on this endpoint

Connection Summary:

1. Best Model from Model Registry is approved to get deployed into the Staging environment.
2. Event Bridge gets triggered and gives AWS lambda a signal for deployment
3. SageMaker Endpoint for Staging Environment host the model for inferences
4. After the Staging inferences we would navigate to the next Inferences for Production pipeline.

Step 5: Automation of Inference Pipeline

Purpose: Automatically deploy the latest approved model and expose it via an endpoint for inference.



Component/Stage	**Tools**	**How it connects / Why Used**
Model approval trigger	Manual in SageMaker Studio	Making model that performed well in Staging for the Deployment in Production
Event Trigger	EventBridge	Detects new model version

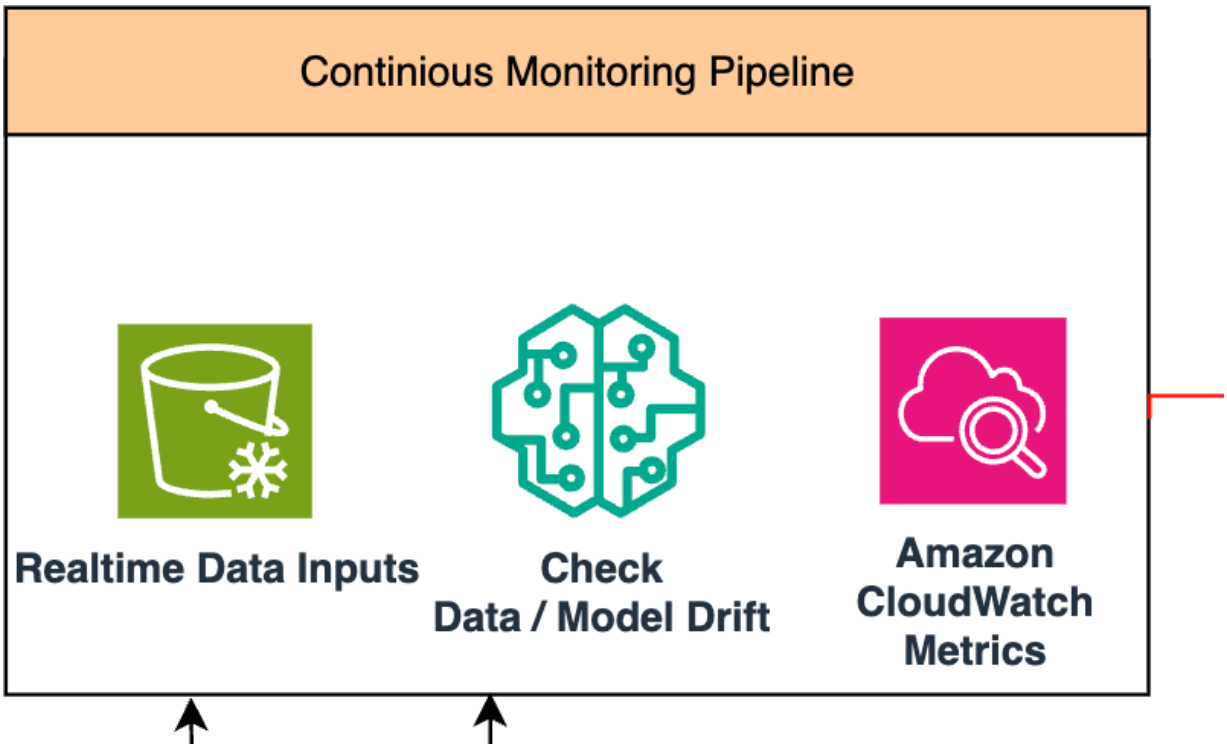
Deployment Launcher	AWS Lambda	Deploys Approved model to end point of Production
Inference Hosting	Amazon SageMaker Endpoint	Real-time Damage Detection inferences prediction
API Layer	API Gateway	Make endpoint consumable for Web or Mobile Apps.

Connection Summary:

1. Approval in Model Registry triggers EventBridge
2. Lambda deploys the model to the endpoint
3. API Gateway exposes endpoints to external consumers.

Step 6: Continuous Monitoring Pipeline

Purpose: Monitor for data drift, model performance, and trigger alerts/retraining if needed.



Component/Stage	**Tools**	**How it connects / Why Used**
Model approval trigger	Manual in SageMaker Studio	Making model that performed well in Staging for the Deployment in Production
Drift and Data Quality Check	SageMaker Model Monitor	Monitors input/output drift, feature statistics
Logs and Alerts	Amazon CloudWatch	Logs PRedictions, Trigger alerts
Event handler	EventBridge + AWS Lambda	On drift detection, triggers retraining
Drift Deviation and	EventBridge Custom	Depending on the

Retrigger	Code	deviation, redirect the stages.

Connection Summary:

1. Monitor Inferences data vs. Training Data
2. If Data drift detected (e.g Lightning Issues)
    a. Alert Via CloudWatch
    b. EventBridge Triggers Lambda
    c. Lambda further starts
        i. Retraining if Drift is <5%
    d. If Drift is more than beyond threshold 8%, It means major drift is there and hence we should start with development environment experiment tracking.
3. Model Monitor can also check for missing input features, low confidence, etc.

Special Scenarios

Scenario 1: Data Drift Detected (e.g., poor lighting)

Triggering from Pipeline: SageMaker Model Monitor

Action Sequence per System Design:

- Model Monitor sends alert → EventBridge → Lambda Triggers
- Lambda starts retraining via **SageMaker Training Pipelines**
- New model trained and re-evaluated
- If passed approved criteria → automatically deployed

Scenario 2: New Annotated Data Available

Trigger: New labeled images uploaded to S3

Action:

- EventBridge in **Data Pipeline** detects upload
- Triggers Lambda → SageMaker Processing
- Data is preprocessed → Training Pipeline triggered
- Model is trained and evaluated →Model Registered in Model Registry