# (Data Engineering essay)

## Introduction

Using deep learning (image recognition algorithms) to enhance the bags shipping/recognition operation at airports.

## Limitations/ bottleneck

1. storage of edges.
2. internet connection speed of edges.
3. detection algorithm execution time.

## Assumptions

1. for confidentiality, the airport is not sharing passengers’ information with our system.
2. existing integration with airport data to track airplane that is loading bags now from belt, and airplane that is offloading bags.
3. no more than 1 bag per second at each edge.

## System Explanation:

There are 2 main operations running side by side in the existing system.

1. operation of feature extraction at origin side and sharing it with edge at destination side.
2. image capturing, uploading to cloud, archiving and training.

And we can categorize data into two categories:

1. small data of image features vector that should be transferred between edges, but it’s critical to be transferred during flight time, and all of it.
2. Big size data of bags images with medium to high resolution for training, and not all of it required.

And according to data readiness state, we will need

1. hot storage for active learning function, and transmission of features vectors.
2. Cold storage for images used by training model, or recent images.
3. Archival with data that is older than retention threshold.

## Design

As in the attached architecture design and ER  
ER diagram is for infrastructure database on cloud, which made flexible enough to points 3 to 5

also scalability is not an issue at all with Azure cloud platform as we can scale up/ scale down as required.

### *System Components:*

**Origin Edge/ Dest. Edge** : a computer machine with processing power, connected to cameras and network card. It has enough storage.  
The storage can be categorized to:

1- system storage and algorithms.

2- storage for features in any suitable data structure like json.

3- storage for captured images for a period of time (not required for Dest. Edge).

Origin Edge is capable of extracting features from images in a specific time (t1)

Dest. Edge is capable of classify images based on passed features vector in time (t2)

**Prioritizer**: in case of network congestion, it prioritize uploading the features vectors before screenshots. Time of upload is (t\_up)

**Monitoring** : watches heartbeats from all edges, and gives alert if edge didn’t send any data for more than predefined period.

**Training (ML machine)**: powerful machine that runs deep learning algorithms, train it with input images and exports the trained module weights to edges.

**Dispatching** : this unit is responsible for dispatching data (features vectors) coming from origin edges and dispatch them to destination edges.

This can be Kafka cluster, where each origin edge is producing feature vectors to dest. topic.  
And Dest edge is subscribing to the origin topics.

(t3)

## Scenarios

### *Happy Scenario*

0. Origin Edge has been initialized with detection algorithm.

1. Cameras capture bag on belt
2. algorithm runs and generates features vector
3. features vector is inserted into kafka destination topic on the cloud
4. images uploaded to blob storage on Azure cloud
5. Images are forwarded to GCP for the purpose of training algorithms
6. destination consumer consumes from the above topic and forwards features to destination edge before the flight reaches destination.

t1+t\_up+t3 < flight\_duration

1. When bags reaches the destination, destination edge recognize them by using the received features vectors.
2. If any missing bad or unrecognized bag detected, it fires alarm

### *Bad Scenarios*

1- network down for time less than flight duration but still t1+t\_up+t3 < flight\_duration

things can still work if we didn’t upload images and saved the whole bandwidth to transfer the features vectors.

So same steps like happy scenario, but we won’t execute steps (4,5)

captured images will stay on edge storage till we clear it, or upload it later

2- network down for longer than flight duration t1+t\_up+t3 > flight\_duration

we will execute same as happy scenario but we won’t execute steps (4,5)  
also we won’t be able to determine bags as no features vectors received.  
So we can store the images of bags at destination, and when we receive features vectors, we can rerun step 7 and detect missing bags, but this will be outdated.

3- network down for time less than flight duration but still t1+t\_up+t3 < flight\_duration

and also disk storage is almost full.

Steps as in bad scenario 1 , in addition we clear space according to preset rules.

## Reporting / Dashboard

1. live edges (from last heartbeat), alarm if any edge is down.
2. Storage used/free for each edge
3. rush hours/ free hours for flights
4. Number of bags uploaded by origin edges. (aggregate over destinations)
5. Number of bags detected by destination edge. + or -
6. Alert for missing bags.
7. Cameras quality
8. average confidence

## Future Work

1. Splitting Kafka topics to destinations & hours, to make sure earlier flights are consumed first.
2. Using different cloud zones to reduce upload time for data and images.
3. mechanism to interfere manually in case of incorrect bag received.
4. Integration and manual intervention to remove bags if passenger for any reason (like security) is prevented from traveling.
5. algorithms can be downloaded by edges.
6. controlling camera resolution based on contingency detected while uploading
7. measuring compression time vs. Network upload time. And pick the suitable operation.
8. recognition center is applying multiple algorithms and communicate the algorithm ID with highest accuracy with detection edges to use.