ML System Design

Design Instagram Feed ranking system

→ Gather requirements (functional

→ Posts from non-connected people. Focus on suggested posts

→ Business Focus: improve engagement
improve DAU & sessions
on engagement for the viewer

→ we would work on individual level metric

→ Design objective func to align to business motive: DAU/sessions

→ Viewing content, engaging, liking, etc.
Objective is correlated to business metric. Won't directly optimize DAU since its a gross metric. We want a metric at a individual level such that it contributes to DAU

→ ML model that improves engagement
view, likes, comment

Non functional requirements

→ Scalable
→ Available

→ Tooling: Debug, monitoring, MLOps
warnings, alerts

→ Analytics
↓
popular posts

→ Monetize (not in scope)

Estimations

→ DAU: 500Million
→

Feed ranking system

→ Involves 3 phases
→ candidate generation
→ Ranking
→ Post-processing stage
(Fairness, diversity, Freshness)
Mostly focus on making sure business constraints. NOT generating new candidates

Data / Features:

→ viewer (aggregated & delayed features)
↓ engagement over time window
↘ like count 14 days back

→ post → Historical engagement on this post
→ post creator's features
→ embedding for text, image, video
↓
can be collected from pretrained models

→ viewer interaction history with post/content

→ view interaction history network.

Labels:

→ interaction, view, like
→ can be binary: 1 is positive interaction
→ Difficult gather 0 label
(threshold up until which positive value

(x, a, r, p)
x - vector
a - action recommended
r - reward (1/0)
p - probability

ANN: Approximate nearest neighbours
compute embeddings for viewers & posts.

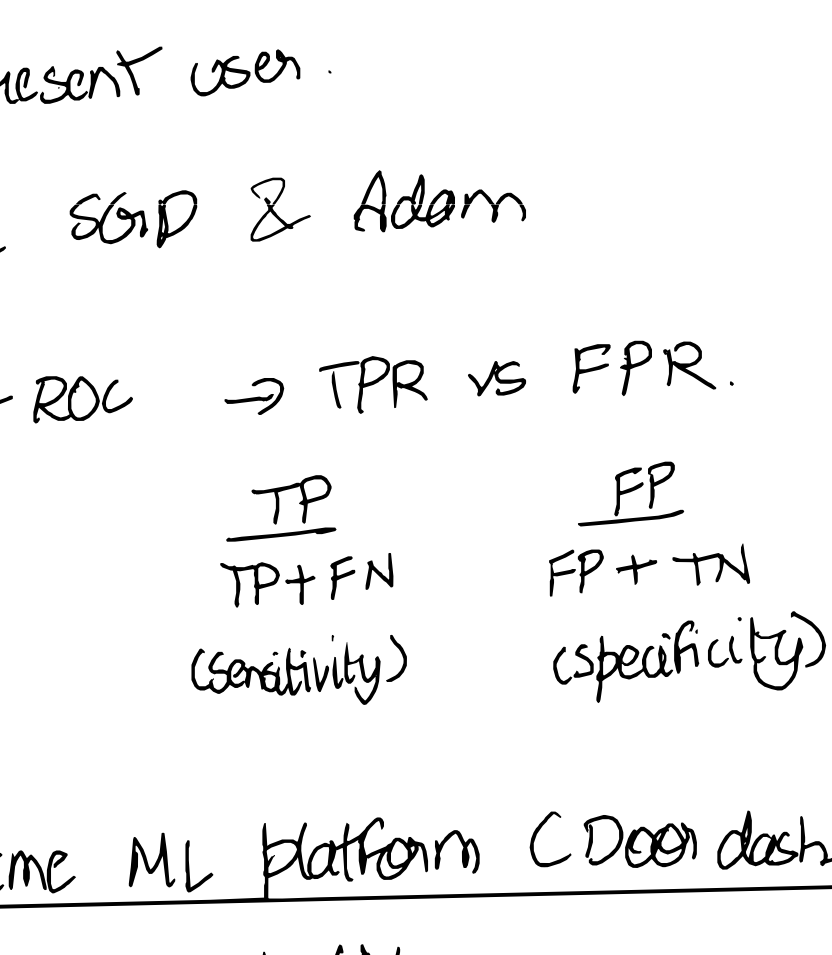
1000 posts for the current viewers
Used ANN to identify candidates

How to learn embeddings?

→ collaborative filtering
Factorization
$A = UV^T$
$A: m \times n$
$U: m \times d$
$V: d \times n$

d - smaller dimen
Approximate similarity between user & view vector

→ Two tower technique



Use BCE loss to train the model.
Learns better & better weights to represent user.

Use SGD & Adam

AUC-ROC → TPR vs FPR.

$\frac{TP}{TP+FN}$ (sensitivity) $\frac{FP}{FP+TN}$ (specificity)

Realtime ML platform (Doordash)

3 sided marketplace
Merchant, Consumer, Dasher

Create order → checkout → Dispatch → Deliver
↓
Rec personalized & search promotion.

Checkout → Rec similar dishes
Fraud

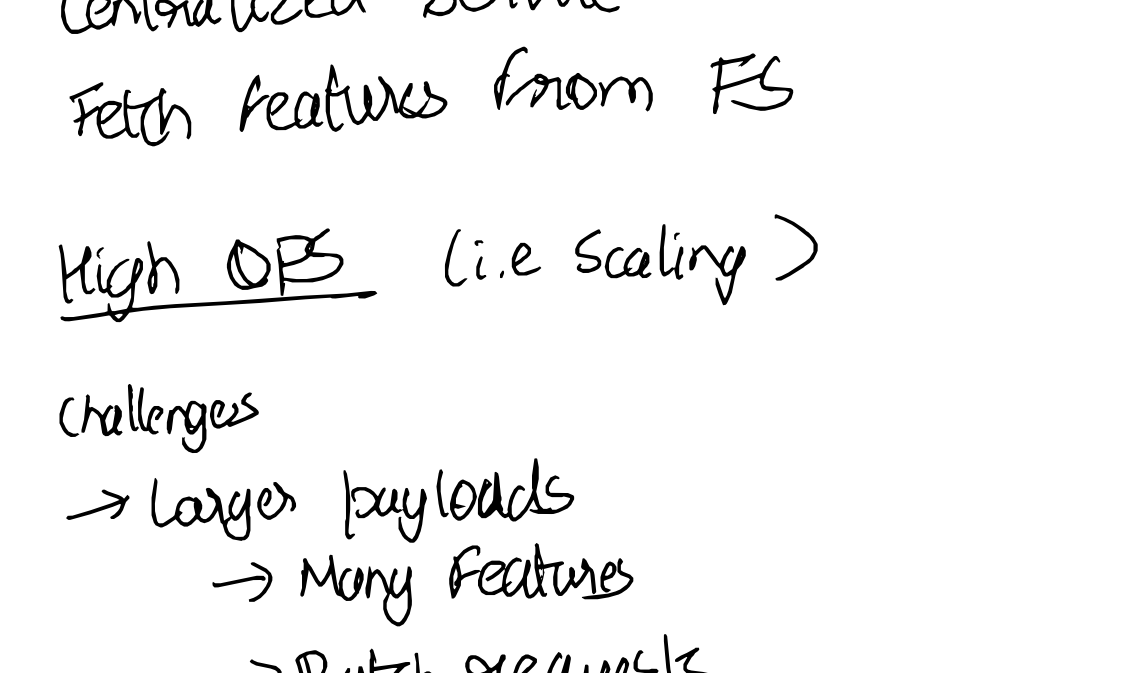
Dispatch → Assign to Dasher
Optimization problem

Delivering → Money for dashers
Mobilize / training them
Match dashers
Multi order pickup.

Central ML platform

Feature engineering
Model training
Model management
Model serving
Model insights
Observability & Monitoring

Feature engineering



Training

Script → Git → MLFlow → Store → Serve

Pred

Centralized service
Fetch features from FS

High QPS (i.e Scaling)

Challenges
→ Larger payloads
→ Many features
→ Batch requests
→ Heavier computations
→ Prod + Experiment traffic
→ Real-time auditing

More models → More features → More expt

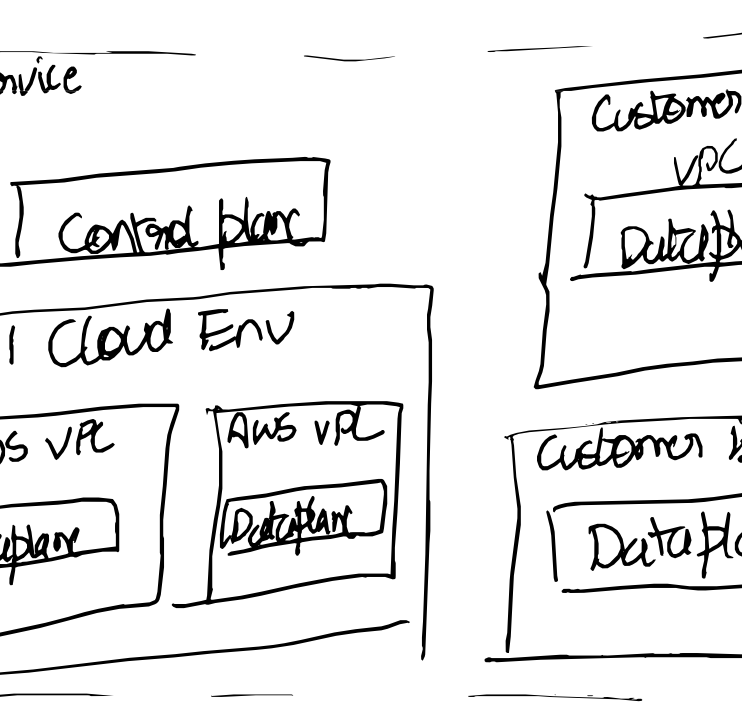
LLM Kubernetes - Predibase

LLM - Trained on lots of data with compute

Blockers for productionizing LLM

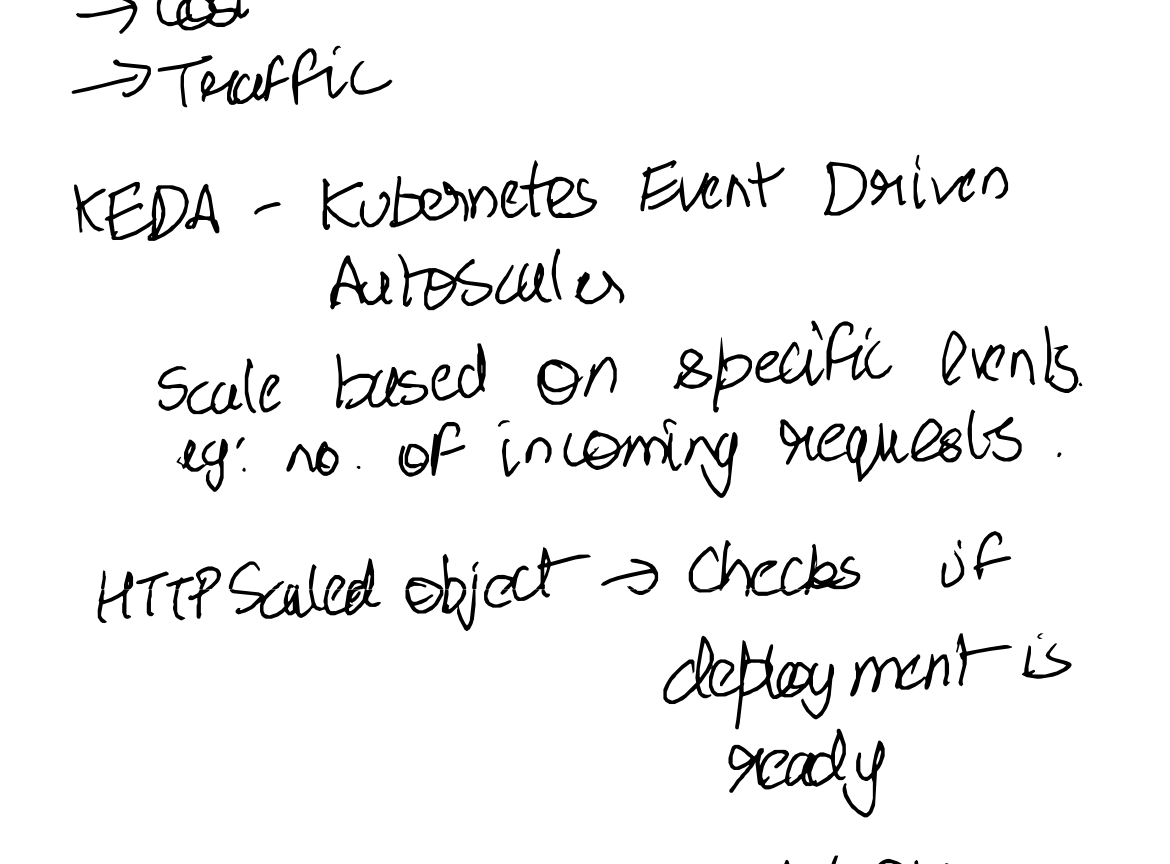
→ Tools are complex
→ Fine-tuning is reliable
→ Model serving is expensive

Serving stack



Multi-Cluster Service Mesh

→ Deploy directly is Customer VPC
→ Deploy dataplane where compute is cheap
→ Deploy new dataplane per VPC customer, not full stack



Istio allows configuring auth policies

Serverless inference

→ Cost
→ Traffic

KEDA - Kubernetes Event Driven Autoscaler
Scale based on specific events
eg: no. of incoming requests.

HTTPScaled object → Checks if deployment is ready

Scale from 0 is VERY SLOW
~8 min to download container
~15 min to download weights.

Cold start opt

→ Containers are large
→ LLM weights are large

Container mirror - Spegel

→ similar to P2P networking
→ Runs on each node
→ When downloading container, it'll go to Spegel instead of going to container registry
→ Can download specific layers

Weight download optimizations

→ Unoptimized
download → .bin to .safetensors
↓
send to customer VPC

Slow! Takes around 20 min

→ Optimized
→ Add init container
Share volume between container & main server
→ Private cache & data planes blob storage
→ slow initially (when cache empty), but faster after that
→ Can copy fast from S3 using the S3 CRT library.
Use multi-threaded downloads which is faster!!
→ Total time down to 5 min!!

Application

→ LoRA adapter finetuning
→ LoRAX server to dynamically load adapters