

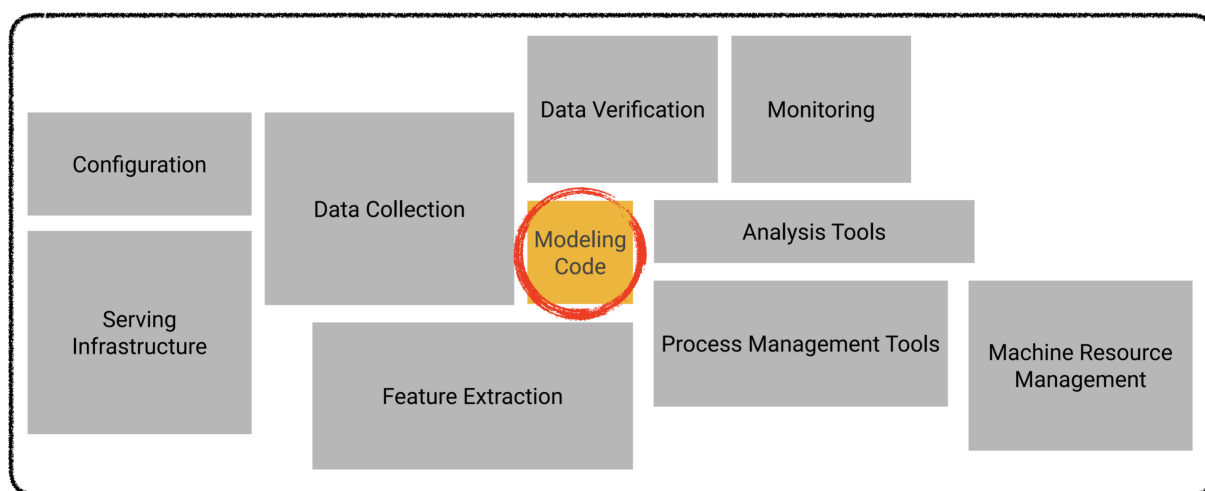
# Machine Learning System Design

---

1. ML System Design Flow
2. ML System Design Sample Questions
3. ML System Design Topics
4. ML at Big Tech companies

Designing ML systems for production

Deploying deep learning models in production can be challenging, and it is beyond training models with good performance. Several distinct components need to be designed and developed in order to deploy a production level deep learning system.



*Robert Crowe at O'Reilly's AI conference 2019, TFX workshop,  
<https://conferences.oreilly.com/artificial-intelligence/ai-ca-2019>*

Approaching an ML system design problem follows a similar logical flow to the generic software system design. ( For more insight on general system design interview you can e.g. check out [Grokking the System Design Interview](#) and [System design primer](#) ). However, there are certain components in the design of an ML based system that needs to be addressed and need special attention, as you will see below in ML System Design Flow.

## ML System Design Interview

- In an ML system design interview you are exposed to open ended questions with no single correct answer.
- The goal of ML system design interview is evaluate your your ability to zoom out and design a production-level ML system that can be deployed as a service within a company's ML infrastructure.

## 1. ML System Design Flow

---

In order to design a solid ML system for real world applications, it is important to follow a design flow. I recommend using the following **9-step ML System Design Flow** to design SW system solutions for ML-relevant business problems both at work and during interviews:

### The 9-step ML System Design Flow:

Step 1	<a href="#">Problem Formulation</a>
Step 2	<a href="#">Metrics (Offline and Online)</a>
Step 3	<a href="#">Architectural Components (MVP Logic)</a>
Step 4	<a href="#">Data Collection and Preparation</a>
Step 5	<a href="#">Feature Engineering</a>
Step 6	<a href="#">Model Development and Offline Evaluation</a>
Step 7	<a href="#">Prediction Service</a>
Step 8	<a href="#">Online Testing and Deployment</a>
Step 9	<a href="#">Scaling, Monitoring, and Updates</a>

Note: Remember when using this design flow during an interview to be flexible. According to the needs of the interview or the interests of the interviewer, you may skip some of these components or spend more time for a deep dive in one or two components.

## 1. Problem Formulation

- Clarifying questions
- Use case(s) and business goal
- Requirements
  - Scope (features needed), scale, and personalization
  - Performance: prediction latency, scale of prediction
  - Constraints
  - Data: sources and availability
- Assumptions
- Translate an abstract problem into an ML problem
  - ML objective,
  - ML I/O,
  - ML category (e.g. binary classification, multi-classification, unsupervised learning, etc)
- Do we need ML to solve this problem?
  - Trade off between impact and cost
    - Costs: Data collection, data annotation, compute

- if Yes, we choose an ML system to design. If No, follow a general system design flow.
- Note: in an ML system design interview we can assume we need ML.

## 2. Metrics (Offline and Online)

- Offline metrics (e.g. classification, relevance metrics)
  - Classification metrics
    - Precision, Recall, F1, ROC AUC, P/R AUC, mAP, log-loss, etc
    - Imbalanced data
  - Retrieval and ranking metrics
    - Precision@k, Recall@k (do not consider ranking quality)
    - mAP, MRR, nDCG
  - Regression metrics: MSE, MAE,
  - Problem specific metrics
    - Language: BLEU, BLEURT, GLUE, ROUGE, etc
    - ads: CPE, etc
  - Latency
  - Computational cost (in particular for on-device)
- Online metrics
  - CTR
  - Task/session success/failure rate,
  - Task/session total (e.g. watch) times,
  - Engagement rate (like rate, comment rate)
  - Conversion rate
  - Revenue lift
  - Reciprocal rank of first click, etc,
  - Counter metrics: direct negative feedback (hide, report)
- Trade-offs b/w metrics

## 3. Architectural Components (MVP Logic)

- High level architecture and main components
  - Non-ML components:
    - user, app server, DBs, KGs, etc and their interactions
  - ML components:
    - Modeling modules (e.g. candidate generator, ranker, ect)
    - Train data generator
  - ...
- Modular architecture design
  - Model 1 architecture (e.g. candidate generation)
  - Model 2 architecture (e.g. ranker, filter)
  - ...

## 4. Data Collection and Preparation

- Data needs
  - target variable
  - big actors in signals (e.g. users, items, etc)

- type (e.g. image, text, video, etc) and volume
- Data Sources
  - availability and cost
  - implicit (logging), explicit (e.g. user survey)
- Data storage
- ML Data types
  - structured
    - numerical
    - categorical (ordinal, nominal),
  - unstructured(e.g. image, text, video, audio)
- Labelling (for supervised)
  - Labeling methods
    - Natural labels (extracted from data e.g. clicks, likes, purchase, etc)
      - Missing negative labels (not clicking is not a negative label):
        - Negative sampling
    - Explicit user feedback
    - Human annotation (super costly, slow, privacy issues)
  - Handling lack of labels
  - Programmatic labeling methods (noisy, pros: cost, privacy, adaptive)
    - Semi-supervised methods (from an initial smaller set of labels e.g. perturbation based)
    - Weak supervision (encode heuristics e.g. keywords, regex, db, output of other ML models)
  - Transfer learning:
    - pre-train on cheap large data (e.g. GPT-3),
    - zero-shot or fine-tune for downstream task
  - Active learning
  - Labeling cost and trade-offs
- Data augmentation
- Data Generation Pipeline
  - Data collection/ingestion (offline, online)
  - Feature generation (next)
  - Feature transform
  - Label generation
  - Joiner

## 5. Feature Engineering

- Choosing features
  - Define big actors (e.g. user, item, document, query, ad, context),
  - Define actor specific features (current, historic)
    - Example user features: user profile, user history, user interests
    - Example text features: n-grams (uni,bi), intent, topic, frequency, length, embeddings
  - Define cross features (e.g. user-item, or query-document features)
    - Example query-document features: tf-idf
    - Example user-item features: user-video watch history, user search history, user-ad interactions(view, like)
  - Privacy constraints

- Feature representation
  - One hot encoding
  - Embeddings
    - e.g. for text, image, graphs, users (how), stores, etc
    - how to generate/learn?
    - pre-compute and store
  - Encoding categorical features (one hot, ordinal, count, etc)
  - Positional embeddings
  - Scaling/Normalization (for numerical features)
- Preprocessing features
  - Needed for unstructured data
    - Text: Tokenize (Normalize, pre-tokenize, tokenizer model (ch/word/subword level), post-process (add special tokens))
    - Images: Resize, normalize
    - Video: Decode frames, sample, resize, scale and normalize
- Missing Values
- Feature importance
- Featurizer (raw data -> features)
- Static (from feature store) vs dynamic (computed online) features

## 6. Model Development and Offline Evaluation

- Model selection (MVP)
  - Heuristics -> simple model -> more complex model -> ensemble of models
    - Pros and cons, and decision
    - Note: Always start as simple as possible (KISS) and iterate over
  - Typical modeling choices:
    - Logistic Regression
    - Linear regression
    - Decision tree variants
      - GBDT (XGBoost) and RF
    - SVM
    - Neural networks
      - FeedForward
      - CNN
      - RNN
      - Transformers
  - Decision Factors
    - Complexity of the task
    - Data: Type of data (structured, unstructured), amount of data, complexity of data
    - Training speed
    - Inference requirements: compute, latency, memory
    - Continual learning
    - Interpretability
  - Popular NN architectures

- Dataset
  - Sampling
    - Non-probabilistic sampling
    - Probabilistic sampling methods
      - random, stratified, reservoir, importance sampling
  - Data splits (train, dev, test)
    - Portions
    - Splitting time-correlated data (split by time)
      - seasonality, trend
    - Data leakage:
      - scale after split,
      - use only train split for stats, scaling, and missing vals
  - Class Imbalance
    - Resampling
    - weighted loss fcn
    - combining classes
- Model training
  - Loss functions
    - MSE, Binary/Categorical CE, MAE, Huber loss, Hinge loss, Contrastive loss, etc
  - Optimizers
    - SGD, AdaGrad, RMSProp, Adam, etc
  - Model training
    - Training from scratch or fine-tune
  - Model validation
  - Debugging
  - Offline vs online training
- Model offline evaluation
- Hyper parameter tuning
  - Grid search
- Iterate over MVP model
  - Model Selection
  - Data augmentation
  - Model update frequency
- Model calibration

## 7. Prediction Service

- Data processing and verification

- Web app and serving system
- Prediction service
- Batch vs Online prediction
  - Batch: periodic, pre-computed and stored, retrieved as needed - high throughput
  - Online: predict as request arrives - low latency
  - Hybrid: e.g. Netflix: batch for titles, online for rows
- Nearest Neighbor Service
  - Approximate NN
    - Tree based, LSH, Clustering based
- ML on the Edge (on-device AI)
  - Network connection/latency, privacy, cheap
  - Memory, compute power, energy constraints
  - Model Compression
    - Quantization
    - Pruning
    - Knowledge distillation
    - Factorization

## 8. Online Testing and Model Deployment

- A/B Experiments
  - How to A/B test?
    - what portion of users?
    - control and test groups
    - null hypothesis
- Bandits
- Shadow deployment
- Canary release

## 9. Scaling, Monitoring, and Updates

- Scaling for increased demand (same as in distributed systems)
  - Scaling general SW system (distributed servers, load balancer, sharding, replication, caching, etc)
    - Train data / KB partitioning
  - Scaling ML system
    - Distributed ML
      - Data parallelism (for training)
      - Model parallelism (for training, inference)
        - Asynchronous SGD
        - Synchronous SGD
    - Distributed training
      - Data parallel DT, RPC based DT
    - Scaling data collection
      - MT for 1000 languages
      - NLLB
    - Monitoring, failure tolerance, updating (below)

- Auto ML (soft: HP tuning, hard: arch search (NAS))
- Monitoring:
  - Logging
    - Features, predictions, metrics, events
  - Monitoring metrics
    - SW system metrics
    - ML metrics (accuracy related, predictions, features)
      - Online and offline metric dashboards
  - Monitoring data distribution shifts
    - Types: Covariate, label and concept shifts
    - Detection (stats, hypothesis testing)
    - Correction
- System failures
  - SW system failure
    - dependency, deployment, hardware, downtime
  - ML system failure
    - data distribution difference (test vs online)
    - feedback loops
    - edge cases (e.g. invalid/junk input)
    - data distribution changes
  - Alarms
    - failures (data pipeline, training, deployment), low metrics, etc
- Updates: Continual training
  - Model updates
    - train from scratch or a base model
    - how often? daily, weekly, monthly, etc
  - Auto update models
  - Active learning
  - Human in the loop ML

### Other topics:

- Extensions:
  - Iterations over the base design to add a new functional feature
- Bias in training data
  - Bias introduced by human labeling
- Freshness, Diversity
- Privacy and security

## 2. ML System Design Sample Questions

---

### Design a:

- Recommendation System
  - Video recommendation (Netflix, Youtube)
  - Friend/follower recommendation (Facebook, Twitter)



- People you may know (LinkedIn)
  - Replacement product recommendation (Instacart)
  - Rental recommendation (Airbnb)
  - Place recommendation
- Newsfeed system (ranking)
- Search system (retrieval, ranking)
  - Semantic Search system (retrieval, ranking)
  - Document search, Image search, Video search
  - Query: Text, Image, Video
- Ads serving system (retrieval, ranking)
- Named entity linking system (tagging, resolution)
- Harmful/spam content detection system
- Autocompletion / Typeahead suggestion system
- Ride matching system
- Language identification system
- Chatbot system
- Question answering system
- Proximity service / Yelp
- Food delivery time approximation
- Self-driving car (Perception, Prediction, Planning)
- Sentiment analysis system
- Healthcare diagnosis system
- Fraud detection system

### 3. ML System Design Topics

---

I observed there are certain sets of topics that are frequently brought up or can be used as part of the logic of the system. Here are some of the important ones:

#### Recommendation Systems

- Candidate generation
  - Collaborative Filtering (CF)

- User based, item based
- Matrix factorization
- Two-tower approach
- Content based filtering
- Ranking
- Learning to rank (LTR)
  - point-wise (simplest), pairwise, list-wise

## Search and Ranking (Ads, newsfeed, etc)

- Search systems
  - Query search (keyword search, [semantic search](#))
  - Visual search
  - Video search
  - Two stage model
    - document selection
    - document ranking
- Ranking
  - Newsfeed ranking system
  - Ads ranking system
  - Ranking as classification
  - Multi-stage ranking + blender + filter

## NLP

- Feature engineering
  - Preprocessing (tokenization)
- Text Embeddings
  - Word2Vec, GloVe, Elmo, BERT
- NLP Tasks:
  - Text classification
    - Sentiment analysis
    - Topic modeling
  - Sequence tagging
    - Named entity recognition
    - Part of speech tagging
      - POS HMM
      - Viterbi algorithm, beam search
  - Text generation
    - Language modeling
      - N-grams vs deep learning models (trade-offs)

- Decoding
  - Sequence 2 Sequence models
    - Machine Translation
      - Seq2seq models, NMT, Transformers
  - Question Answering
  - [Adv] Dialog and chatbots
    - [CMU lecture on chatbots](#)
    - [CMU lecture on spoken dialogue systems](#)
- Speech Recognition Systems
  - Feature extraction, MFCCs
  - Acoustic modeling
    - HMMs for AM
    - CTC algorithm (advanced)

## Computer Vision

- Image classification
  - VGG, ResNET
- [Object detection](#)
  - Two stage models (R-CNN, Fast R-CNN, Faster R-CNN)
  - One stage models (YOLO, SSD)
  - [Vision Transformer \(ViT\)](#)
  - NMS algorithm
- Object Tracking

## Graph problems

- People you may know

# 4. ML at big tech companies

---

Once you learn about the basics, I highly recommend checking out different companies blogs on ML systems. You can refer to some of those resources in the [ML at Companies](#) section.

## More resources

---

- For more insight on different components above you can check out the following resources):
  - [Full Stack Deep Learning course](#)
  - [Production Level Deep Learning](#)
  - [Machine Learning Systems Design](#)
  - [Stanford course on ML system design](#)