جامعة الملك عبدالله
للعلوم والتقنية
King Abdullah University of
Science and Technology

أكاديمية كاوست
KAUST ACADEMY

From AI Foundations to Real-World Practices

King Abdullah University of Science and Technology (KAUST)
KAUST Academy

Table of Contents

Introduction

- Lifecycle of an AI product:
    - Idea: Identify business need / pain point.

Introduction

- Lifecycle of an AI product:
  - Idea: Identify business need / pain point.
  - Rapid Proof: Quickest viable demonstration (API / zero-shot model).

Introduction

- Lifecycle of an AI product:
  - Idea: Identify business need / pain point.
  - Rapid Proof: Quickest viable demonstration (API / zero-shot model).
  - Iteration: Collect real data, train model, validate metrics, optimize accuracy, speed and cost.

Introduction

- Lifecycle of an AI product:
  - Idea: Identify business need / pain point.
  - Rapid Proof: Quickest viable demonstration (API / zero-shot model).
  - Iteration: Collect real data, train model, validate metrics, optimize accuracy, speed and cost.
  - Production: model is mature enough? ⟶ Deploy at scale.

Introduction

- Lifecycle of an AI product:
    - Idea: Identify business need / pain point.
    - Rapid Proof: Quickest viable demonstration (API / zero-shot model).
    - Iteration: Collect real data, train model, validate metrics, optimize accuracy, speed and cost.
    - Production: model is mature enough? ⟶ Deploy at scale.
    - Maintenance: Adapt to data drift, optimize cost & performance.

Introduction

- Key Constraints for AI Products:
  - Data – availability, quality and labeling effort.

Introduction

- Key Constraints for AI Products:
  - Data – availability, quality and labeling effort.
  - Resources – training and inference resources.

Introduction

- Key Constraints for AI Products:
  - Data – availability, quality and labeling effort.
  - Resources – training and inference resources.
  - User Impact – accuracy, latency, reliability felt by end-users.

Introduction

- Key Constraints for AI Products:
  - Data – availability, quality and labeling effort.
  - Resources – training and inference resources.
  - User Impact – accuracy, latency, reliability felt by end-users.
  - Privacy & Governance Boundaries – closed source vs open source solutions.

Introduction

- Key Constraints for AI Products:
  - Data – availability, quality and labeling effort.
  - Resources – training and inference resources.
  - User Impact – accuracy, latency, reliability felt by end-users.
  - Privacy & Governance Boundaries – closed source vs open source solutions.
  - Budget – API spend, compute bills, labor & labeling costs.

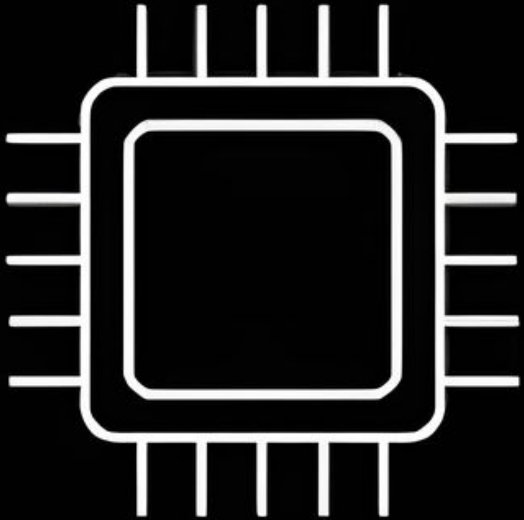
Introduction

- Key Constraints for AI Products:
  - Data – availability, quality and labeling effort.
  - Resources – training and inference resources.
  - User Impact – accuracy, latency, reliability felt by end-users.
  - Privacy & Governance Boundaries – closed source vs open source solutions.
  - Budget – API spend, compute bills, labor & labeling costs.

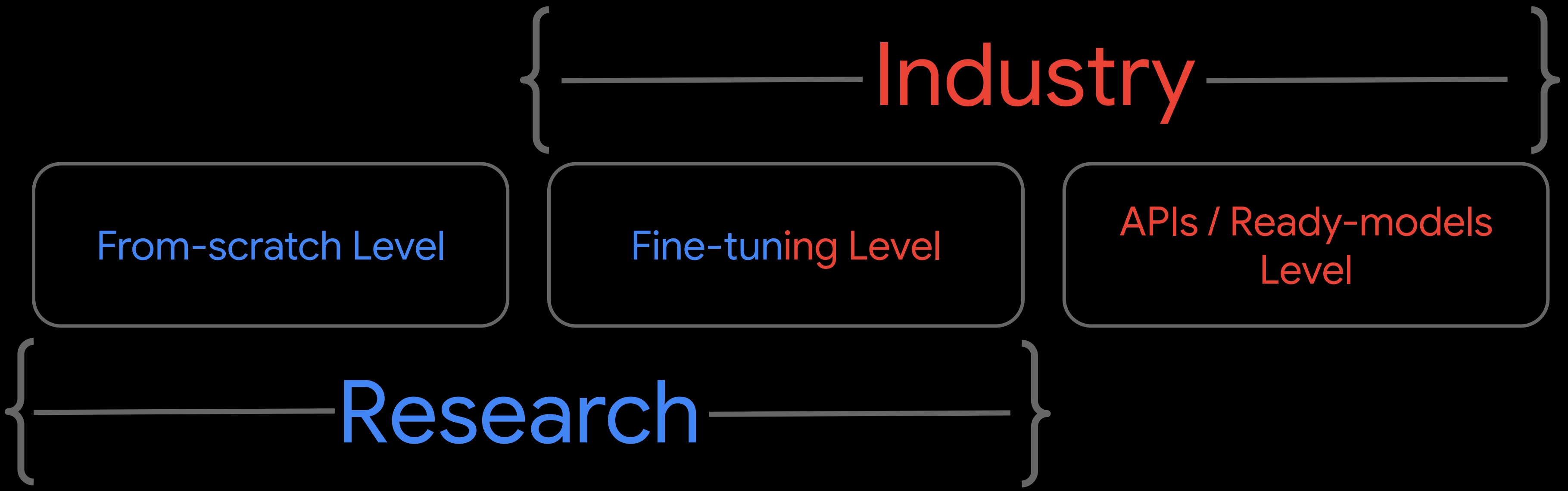
⇒ Every stage balances these constraints.

⇒ Start as simple as possible, then iterate toward the most efficient long-term solution.

Complexity & Resources Spectrum



Implementation Levels for AI Solutions



Complexity & Resources Spectrum

Level	What to do / use?	When to Choose
APIs / Ready-models Level	• Using closed-source APIs (OpenAI, Google Vertex,...) • Using zero-shot models (LLMs, SAM, Yolo-World,...) • Using publicly available models from Github / Hugging Face / ….	
Training / Fine-tuning Level		
From-scratch Level		

Complexity & Resources Spectrum

Level	What to do / use?	When to Choose
APIs / Ready-models Level	• Using closed-source APIs (OpenAI, Google Vertex,...) • Using zero-shot models (LLMs, SAM, Yolo-World,...) • Using publicly available models from Github / Hugging Face / ….	1- Need demo as soon as possible. 2- Zero-shot models are enough for your task. 3- little or no data. 4- No infrastructure. 5- Will not be used on private data.
Training / Fine-tuning Level		
From-scratch Level		

Complexity & Resources Spectrum

Level	What to do / use?	When to Choose
APIs / Ready-models Level	• Using closed-source APIs (OpenAI, Google Vertex,...) • Using zero-shot models (LLMs, SAM, Yolo-World,...) • Using publicly available models from Github / Hugging Face / ….	1- Need demo as soon as possible. 2- Zero-shot models are enough for your task. 3- little or no data. 4- No infrastructure. 5- Will not be used on private data.
Training / Fine-tuning Level	• Training & fine-tuning models on your data. • Distilling & quantizing ready-models.	
From-scratch Level		

Complexity & Resources Spectrum

Level	What to do / use?	When to Choose
APIs / Ready-models Level	• Using closed-source APIs (OpenAI, Google Vertex,...) • Using zero-shot models (LLMs, SAM, Yolo-World,...) • Using publicly available models from Github / Hugging Face / ….	1- Need demo as soon as possible. 2- Zero-shot models are enough for your task. 3- little or no data. 4- No infrastructure. 5- Will not be used on private data.
Training / Fine-tuning Level	• Training & fine-tuning models on your data. • Distilling & quantizing ready-models.	1- Idea proved to be feasible. 2- Zero-shot models are not enough for your task. 3- Have some data. 4- Can handle infrastructure. 5- Will use on private data. 6- Wants to decrease costs (it depends).
From-scratch Level		

Complexity & Resources Spectrum

Level	What to do / use?	When to Choose
APIs / Ready-models Level	• Using closed-source APIs (OpenAI, Google Vertex,...) • Using zero-shot models (LLMs, SAM, Yolo-World,...) • Using publicly available models from Github / Hugging Face / ….	1- Need demo as soon as possible. 2- Zero-shot models are enough for your task. 3- little or no data. 4- No infrastructure. 5- Will not be used on private data.
Training / Fine-tuning Level	• Training & fine-tuning models on your data. • Distilling & quantizing ready-models.	1- Idea proved to be feasible. 2- Zero-shot models are not enough for your task. 3- Have some data. 4- Can handle infrastructure. 5- Will use on private data. 6- Wants to decrease costs (it depends).
From-scratch Level	• Training from scratch / novel architectures	1- Research novelty. 2- Unavailable model.

Open-Source vs Closed-Source

- Prototype fast?

  ⇒ Whichever gets you a demo quicker, but mostly closed-source.

- Strict data privacy or customisation need?

  ⇒ Open-source and self-host.

- Need battle-tested uptime & rapid scaling & better performance?

  ⇒ Closed-source.

Scoping a New AI Project

Scoping a New AI Project

- You just joined a company and heard:

“We need an AI solution for this problem.”

- Which clarifying questions will you ask before writing a single line of code?

Scoping a New AI Project

- What Problem Are We Solving?

  ⇒ Immerse in the domain: Let stakeholders explain the issue in their own language; capture pain-points, constraints, success criteria.

  ⇒ Translate to AI language: Re-frame domain terms into ML tasks classification, detection, ranking, forecasting, etc
  (Kaggle experience helps a lot here).

Scoping a New AI Project

- Do We Even Need AI to solve this problem?

    ⇒ Stakeholder and customers wants to through AI on everything.

    ⇒ There are many problems in real world that don't need AI at all.

    ⇒ Are you sure you want to use AI to solve a problem?

    Or just to introduce some new fancy useless feature?

Scoping a New AI Project

- Can AI Realistically Solve It?

  ⇒ Stakeholders and customers don't know the limits of AI, thinking AI can solve any problem.

  ⇒ You should know the limits of these AI models and reflect them to stakeholders to handle their expectation.

Scoping a New AI Project

- Can AI Realistically Solve It?

⇒ Stakeholders and customers don’t know the limits of AI, thinking AI can solve any problem.

⇒ You should know the limits of these AI models and reflect them to stakeholders to handle their expectation.

But, how to know if a problem is solvable by AI or not?🤔

Feasibility Checks

1. Human Baseline: Can humans do this? how well?
2. Previous Work: Papers, benchmarks, existing APIs, Kaggle competitions, public repositories.
3. Data Reality: Do we have (or can we get) labelled examples?

Feasibility Checks

1. Human Baseline: Can humans do this? how well?
2. Previous Work: Papers, benchmarks, existing APIs, Kaggle competitions, public repositories.
3. Data Reality: Do we have (or can we get) labelled examples?

Let’s see some examples…

Feasibility Checks

- Problem: I want to read all car license plate numbers.

  ⇒ Human Baseline:

  ⇒ Pipeline:

Feasibility Checks

- Problem: I want to read all car license plate numbers.

  ⇒ Human Baseline: Very easy task, thus AI can do it as well.

  ⇒ Pipeline: Detection + OCR.

Feasibility Checks

- Problem: I want to predict the weight of an object based on its image.

  ⇒ Human Baseline:

  ⇒ Pipeline:

Feasibility Checks

- Problem: I want to predict the weight of an object based on its image.

  ⇒ Human Baseline: Not easy. Probably with medium accuracy.

  ⇒ Pipeline: Image regression on height, width and depth. Then some math to convert to weight.

Feasibility Checks

- Problem: I want to predict the price of a house based on its color.

  ⇒ Human Baseline:

  ⇒ Pipeline:

Feasibility Checks

- Problem: I want to predict the price of a house based on its color.

  ⇒ Human Baseline: impossible. There is no predictive signal in color.

Feasibility Checks

- Problem: I want to predict whether a user will like a specific phone.

  ⇒ Human Baseline:

  ⇒ Pipeline:

Feasibility Checks

- Problem: I want to predict whether a user will like a specific phone.

  ⇒ Human Baseline: Possible. I just need the user history.

  ⇒ Pipeline: Recommendation system task.

Feasibility Checks

- Problem: I want to build a model that translates cat meows into human language.

  ⇒ Human Baseline:

  ⇒ Pipeline:

Feasibility Checks

- Problem: I want to build a model that translates cat meows into human language.

  ⇒ Human Baseline: Impossible. No ground truth.

Feasibility Checks

- Problem: I want to forecast my company's sales 10 years into the future.

  ⇒ Human Baseline:

  ⇒ Pipeline:

Feasibility Checks

- Problem: I want to forecast my company's sales 10 years into the future.

  ⇒ Human Baseline: Extremely difficult with good accuracy. Long-term forecasting is unstable.

  ⇒ Pipeline: Time series forecasting.

Feasibility Checks

- Problem: I want to help blind people know if there's something near them.

  ⇒ Human Baseline:

  ⇒ Pipeline:

Feasibility Checks

- Problem: I want to help blind people know if there's something near them.

  ⇒ Human Baseline: Possible. But I see many object in the street, which one should i describe?

  ⇒ Pipeline: Detect → Classify (needs many classes or zero-shot capabilities) → Text-to-speech.

  ⇒ Challenges: real-time processing, model size, speech quality.

Feasibility Checks

- Problem: I have characteristics of all my store branches and want to understand what makes each one special.

  ⇒ Human Baseline:

  ⇒ Pipeline:

Feasibility Checks

- Problem: I have characteristics of all my store branches and want to understand what makes each one special.

  ⇒ Human Baseline: Possible. I can look into each store data and cluster them accordingly.

  ⇒ Pipeline: Clustering, or classification on store IDs, then analyze feature importance.

Feasibility Checks

- Problem: I want to predict if my tweet will go viral.

  ⇒ Human Baseline:

  ⇒ Pipeline:

Feasibility Checks

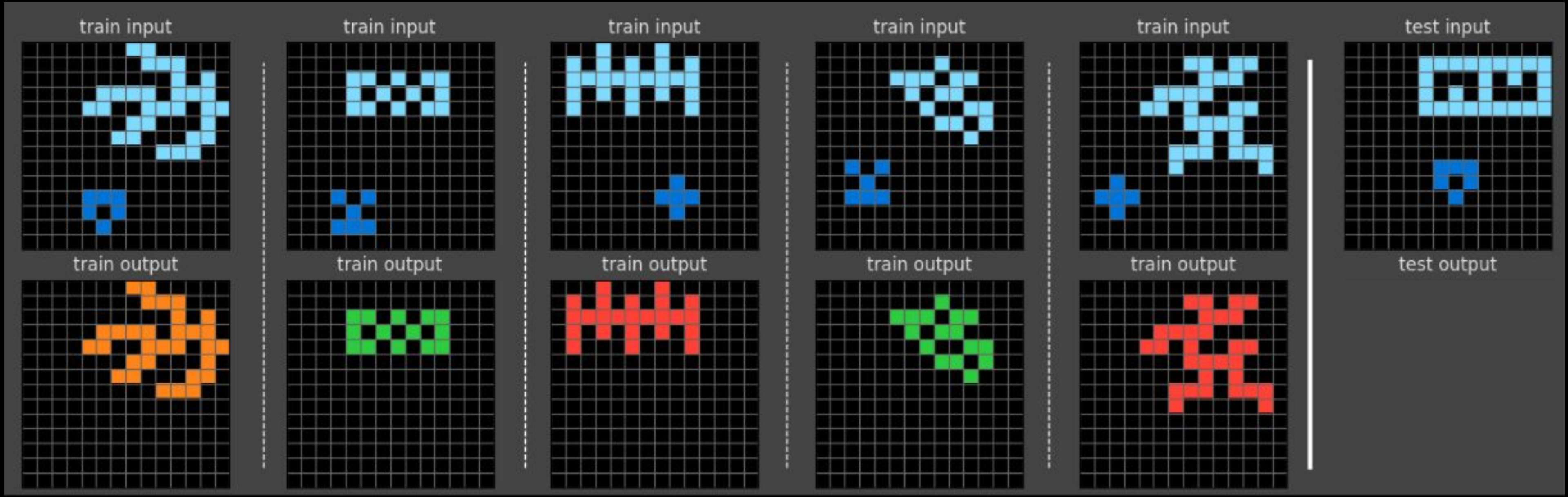
- Problem: I want to predict if my tweet will go viral.

  ⇒ Human Baseline: Maybe. Likely low accuracy.

  ⇒ Pipeline: Text classification (define what “viral” means first!)

Feasibility Checks

- Problem: I want AI to solve this puzzle:



Feasibility Checks

- Problem: I want AI to solve this puzzle:

  ⇒ Human Baseline: Very easy.

  ⇒ AI Solution: Extremely difficult!!!!!!

- This puzzle is part of a famous benchmark called ARC-AGI. All puzzles in this benchmark are easily solvable by humans, but so far, no AI has been able to solve them! link

Feasibility Checks

- Let’s get back to scoping AI problems…

Scoping a New AI Project

- Data?
    - What types of data do I need? And how much do I need to build a reliable system? ⇒ depends on the task and the availability of pretrained models.
    - Is it available? No? ⇒ Can i collect it? No? ⇒ Can I generate it? No? ⇒ Can I just use a zero-shot model to do the task?
    - Is the data quality good enough for a good model?
    - Unlabeled data? ⇒ Manual or Automatic Labeling.
    - Is the data private? No? ⇒ I can use closed-source.

Scoping a New AI Project

- What resources will be used for inference?
    - I can use GPU? ⇒ All types of solutions.
    - CPU only? ⇒ APIs + Non-DL solutions + small DL solutions (efficient and mobile architectures).
    - Edge Device? ⇒ APIs + Non-DL solutions + Extremely small DL solutions (should fit in small memory, e.g. < 64 MB).

Scoping a New AI Project

- Real-time or not?
  - Yes? ⇒ Hard problem.
    - You should decrease your models size as small as possible.
    - You should increase your inference resources as high as possible.
    - You should rely less on APIs, because of latency (it depends).
  - No? ⇒ :)

Scoping a New AI Project

- Any existing solutions?
    - Yes? ⇒ Understand it thoroughly. Consider as a baseline. Start from there.
    - No? ⇒ Search in the internet.

Scoping a New AI Project

- How to Think of Useful Features / Modeling ideas for this particular task?
    - Consider how a human would approach and solve the task.
    - Ask yourself: What information would I rely on to make an accurate decision in this context?
    - Translate that intuition into features.
    - Insights derived from human judgment often map directly to meaningful input variables.

Scoping a New AI Project

- How to evaluate the model?
    - Real data exist?
        - Yes? ⇒ evaluate on real data
        - No? Can I collect? No? Can I generate synthetic data that mimics my real world scenario?
    - <u>Fine-tuning / From-Scratch:</u> train on real / synthetic data, evaluate on real data.
    - <u>APIs:</u> no training, evaluate on real data.

Scoping a New AI Project

- What metric should I use?
    - Depends on the task (Regression, classification, retrieval,...).
    - Within each task, there are special metrics (e.g. MAE, RMSE, MAPE, SMAPE,...).
    - Each one serves a specific use case (e.g. I care about relative distance more than raw distance, MAE vs MAPE).
    - For more: link

Crafting AI Solution

- Ok, now we collected accurate information about the project. Let’s craft a solution…

Crafting AI Solution

1. Build the simplest (maybe expensive) possible solution as fast as possible end to end.

Crafting AI Solution

1. Build the simplest (maybe expensive) possible solution as fast as possible end to end.
2. Working perfectly? Start optimizing!
    a. Closed-source? ⇒ start replacing each closed-source component with an open-source one, and inspect performance difference.
    b. Resources intensive? Optimize the models sizes / input / preprocessing / GPUs-CPUs utilization.
    c. High latency? Calculate the system current latency. Consider as a baseline. Search for the bottlenecks, and keep finding lighter / faster versions.
    d. Low Accuracy? Consider closed-source / bigger models / better preprocessing / better problem formulation.

Crafting AI Solution

1. Build the simplest (maybe expensive) possible solution as fast as possible end to end.
2. Working perfectly? Start optimizing!
    a. Closed-source? ⇒ start replacing each closed-source component with an open-source one, and inspect performance difference.
    b. Resources intensive? Optimize the models sizes / input / preprocessing / GPUs-CPUs utilization.
    c. High latency? Calculate the system current latency. Consider as a baseline. Search for the bottlenecks, and keep finding lighter / faster versions.
    d. Low Accuracy? Consider closed-source / bigger models / better preprocessing / better problem formulation.
3. Perfect? ⇒ Deploy on small scale!
4. Perfect? ⇒ Deploy on a large scale, and monitor!

Example: Building an Arabic AI Teacher

1. Start simple — end-to-end!
   a. Use an LLM with basic Q&A ability.
   b. ✅ It works, but only with general knowledge.

Example: Building an Arabic AI Teacher

1. Start simple — end-to-end!
   a. Use an LLM with basic Q&A ability.
   b. ✅ It works, but only with general knowledge.
2. Add requirements → Iterate & Optimize:
   a. ❌ “I want it to use content from a specific book.” ⇒

Example: Building an Arabic AI Teacher

1. Start simple — end-to-end!
    a. Use an LLM with basic Q&A ability.
    b. ✅ It works, but only with general knowledge.
2. Add requirements ⟶ Iterate & Optimize:
    a. ❌ "I want it to use content from a specific book." ⇒ Add RAG
    b. ❌ "I want it to speak to the student." ⇒

Example: Building an Arabic AI Teacher

1. Start simple — end-to-end!
   a. Use an LLM with basic Q&A ability.
   b. ✅ It works, but only with general knowledge.
2. Add requirements ⟶ Iterate & Optimize:
   a. ❌ “I want it to use content from a specific book.” ⇒ Add RAG
   b. ❌ “I want it to speak to the student.” ⇒ Add Text-to-Speech (TTS).
   c. ❌ “I want it in real time.” ⇒

Example: Building an Arabic AI Teacher

1. Start simple — end-to-end!
   a. Use an LLM with basic Q&A ability.
   b. ✅ It works, but only with general knowledge.
2. Add requirements ⟶ Iterate & Optimize:
   a. ❌ “I want it to use content from a specific book.” ⇒ Add RAG
   b. ❌ “I want it to speak to the student.” ⇒ Add Text-to-Speech (TTS).
   c. ❌ “I want it in real time.” ⇒ Optimize latency and pipeline efficiency.
   d. ❌ “The tone is too harsh for young learners.” ⇒

Example: Building an Arabic AI Teacher

1. Start simple — end-to-end!
    a. Use an LLM with basic Q&A ability.
    b. ✅ It works, but only with general knowledge.
2. Add requirements ⟶ Iterate & Optimize:
    a. ❌ “I want it to use content from a specific book.” ⇒ Add RAG
    b. ❌ “I want it to speak to the student.” ⇒ Add Text-to-Speech (TTS).
    c. ❌ “I want it in real time.” ⇒ Optimize latency and pipeline efficiency.
    d. ❌ “The tone is too harsh for young learners.” ⇒ Fine-tune TTS or find a better one.
    e. ❌ “Someone hacked the prompt!” ⇒

Example: Building an Arabic AI Teacher

1. Start simple — end-to-end!
   a. Use an LLM with basic Q&A ability.
   b. ✅ It works, but only with general knowledge.
2. Add requirements → Iterate & Optimize:
   a. ❌ “I want it to use content from a specific book.” ⇒ Add RAG
   b. ❌ “I want it to speak to the student.” ⇒ Add Text-to-Speech (TTS).
   c. ❌ “I want it in real time.” ⇒ Optimize latency and pipeline efficiency.
   d. ❌ “The tone is too harsh for young learners.” ⇒ Fine-tune TTS or find a better one.
   e. ❌ “Someone hacked the prompt!” ⇒ Add prompt-guarding, filtering, and robustness checks.
3. When it’s stable:
   a. Deploy small-scale → Collect feedback → Iterate.
   b. Then deploy large-scale with monitoring, and keep improving.

Deployment Patterns

1. Ready-made API (OpenAI, AWS Rekognition, etc.)
    a. Model is hosted by the provider.
    b. No need to worry about scalability or infrastructure.
2. Self-hosted API (Flask / FastAPI in Docker)
    a. Custom model, hosted on your own server and exposed as an API.
    b. Full control — you must handle infrastructure, uptime, and scaling.
3. Edge / On-device (ONNX, TFLite, CoreML):
    a. Custom model runs directly on the target device.
    b. Zero network latency & strong privacy.
    c. Common in IoT and mobile deployments.

Note: Learn about cloud tools (e.g. AWS, GCP, etc.) — they're essential in real-world deployments.

Thanks for Attending!

Prepared by: Mohamed Eltayeb