Multimodal NLP and Agentic AI

Naeemullah Khan
naeemullah.khan@kaust.edu.sa

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

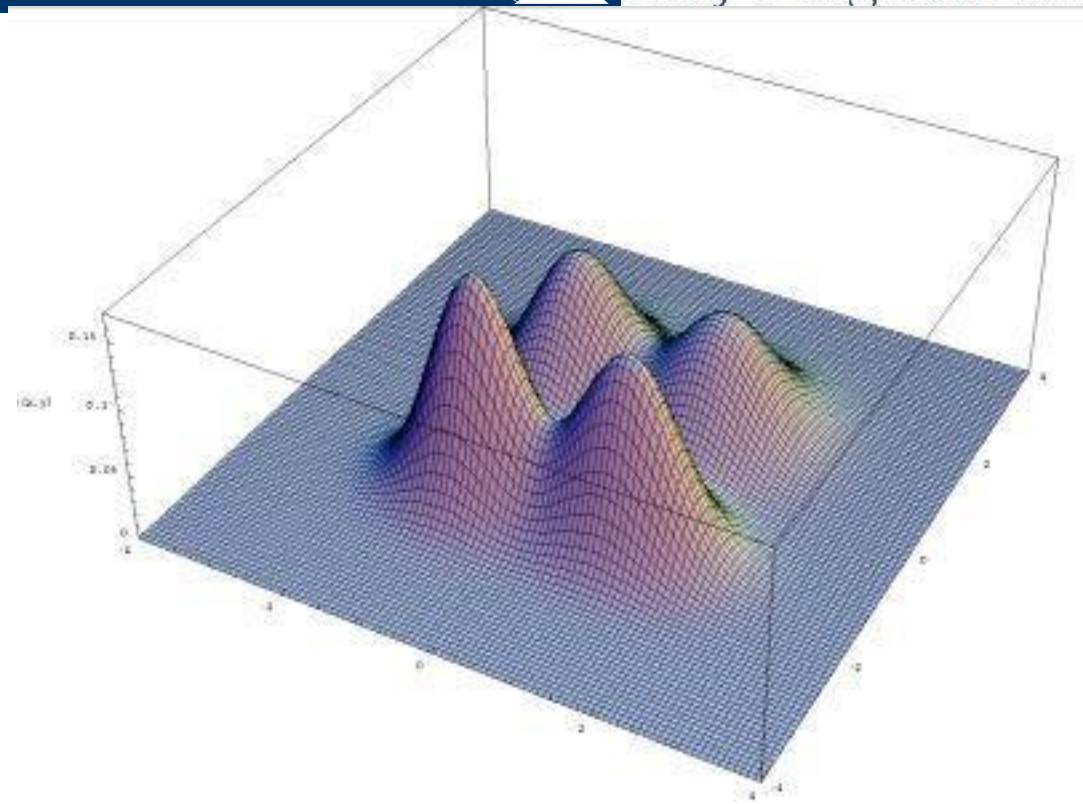
LMH
Lady Margaret Hall

July 4, 2025

Motivation

- Traditional NLP models are **unimodal** (text-only) and **reactive**.
- Modern AI agents need to perceive, reason, and act in complex environments using **multiple modalities** (e.g., vision, text, speech).
- **Multimodal systems** and **agentic AI** aim to build goal-directed, autonomous, and continuously learning agents.
- Rise of agent frameworks (Auto-GPT, BabyAGI) and vision-language models enables new research frontiers in decision-making, robotics, and assistive AI.

What is multimodality?



multimodal adjective

mul·ti·mod·al ˌməl-tē-ˈmō-dᵊl -ˌtī-

: having or involving several modes, modalities, or maxima

multimodal distributions

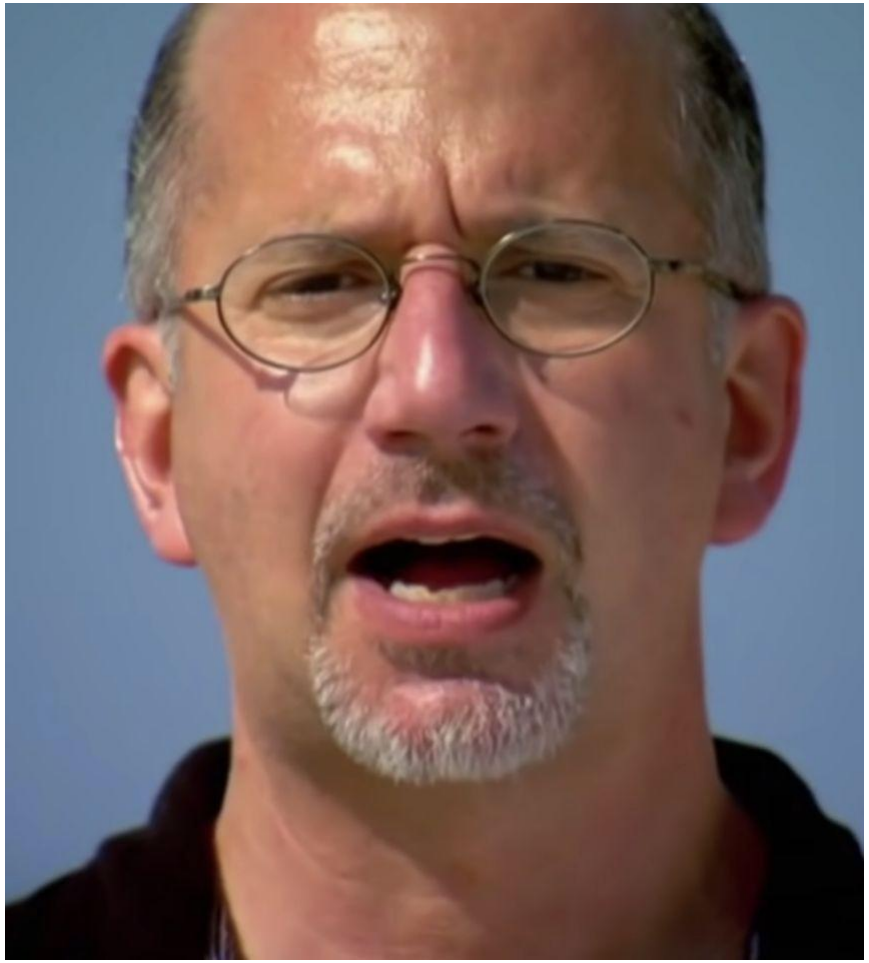
multimodal therapy

In our case, focusing on NLP: text + one or more other *modality* (images, speech, audio, olfaction, others). We'll mostly focus on images as the other modality.

Multimodal brains

McGurk effect (McGurk & MacDonald, 1976)

https://www.youtube.com/watch?v=2k8fHR9jKVM



Learning Outcomes

أكاديمية كاوست
KAUST ACADEMY

LMH
Lady Margaret Hall

- Explain core vision-language models (e.g., CLIP, VisualBERT, FLAVA).
- Compare multimodal architectures and their integration mechanisms.
- Understand the principles of agentic AI and agentic loops.
- Differentiate between reactive and proactive agent behavior.
- Describe continual learning challenges in agentic systems.
- Critically evaluate frameworks like Auto-GPT and BabyAGI.
- Reflect on safety, control, and ethical issues in agentic AI.

Vision-Language Models

CLIP: Contrastive Language-Image Pretraining

- Developed by OpenAI (Radford et al., 2021)
- Trained on 400M (image, text) pairs
- Learns a joint embedding space using contrastive loss
- Image and text encoders are trained to match correct pairs

Use cases:

- Zero-shot classification
- Semantic similarity
- Prompt-based image querying

CLIP: Contrastive Language-Image Pretraining

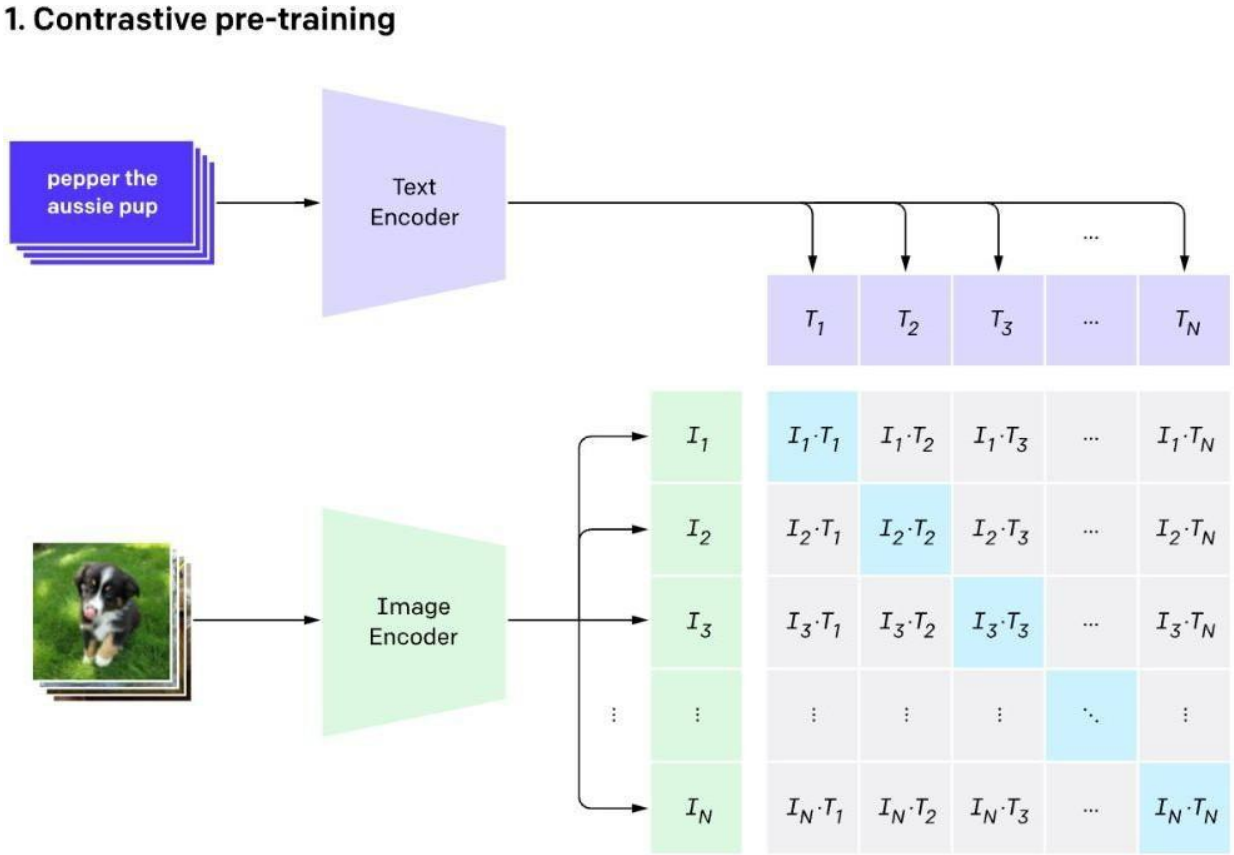
Architecture:

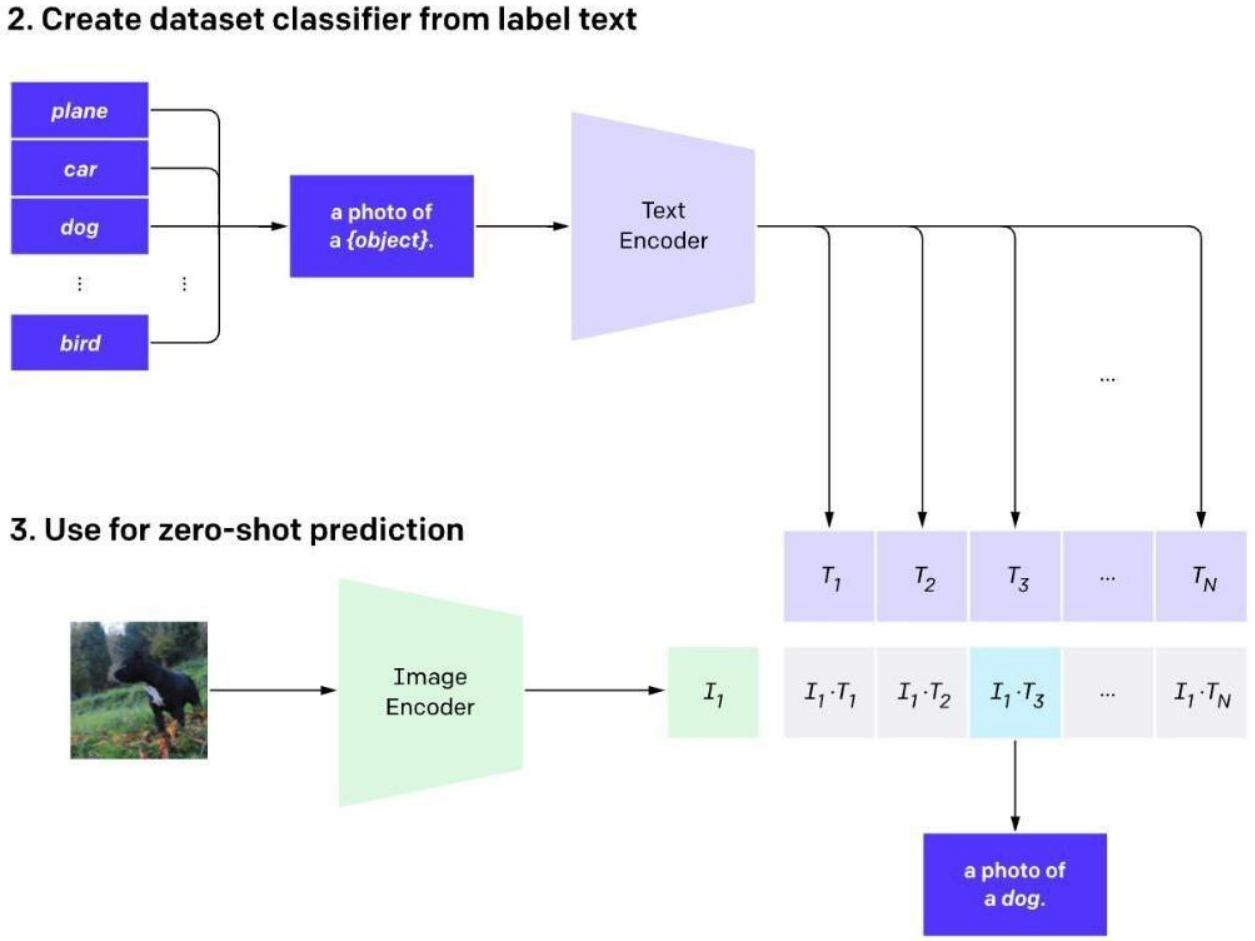
- ResNet / ViT for images
- Transformer for text
- Cosine similarity loss

Paper: CLIP: Learning Transferable Visual Models From Natural Language Supervision https://arxiv.org/abs/2103.00020

CLIP (Radford et al. 2021)

Exact same contrastive loss as earlier, but.. Transformers and *web data*!

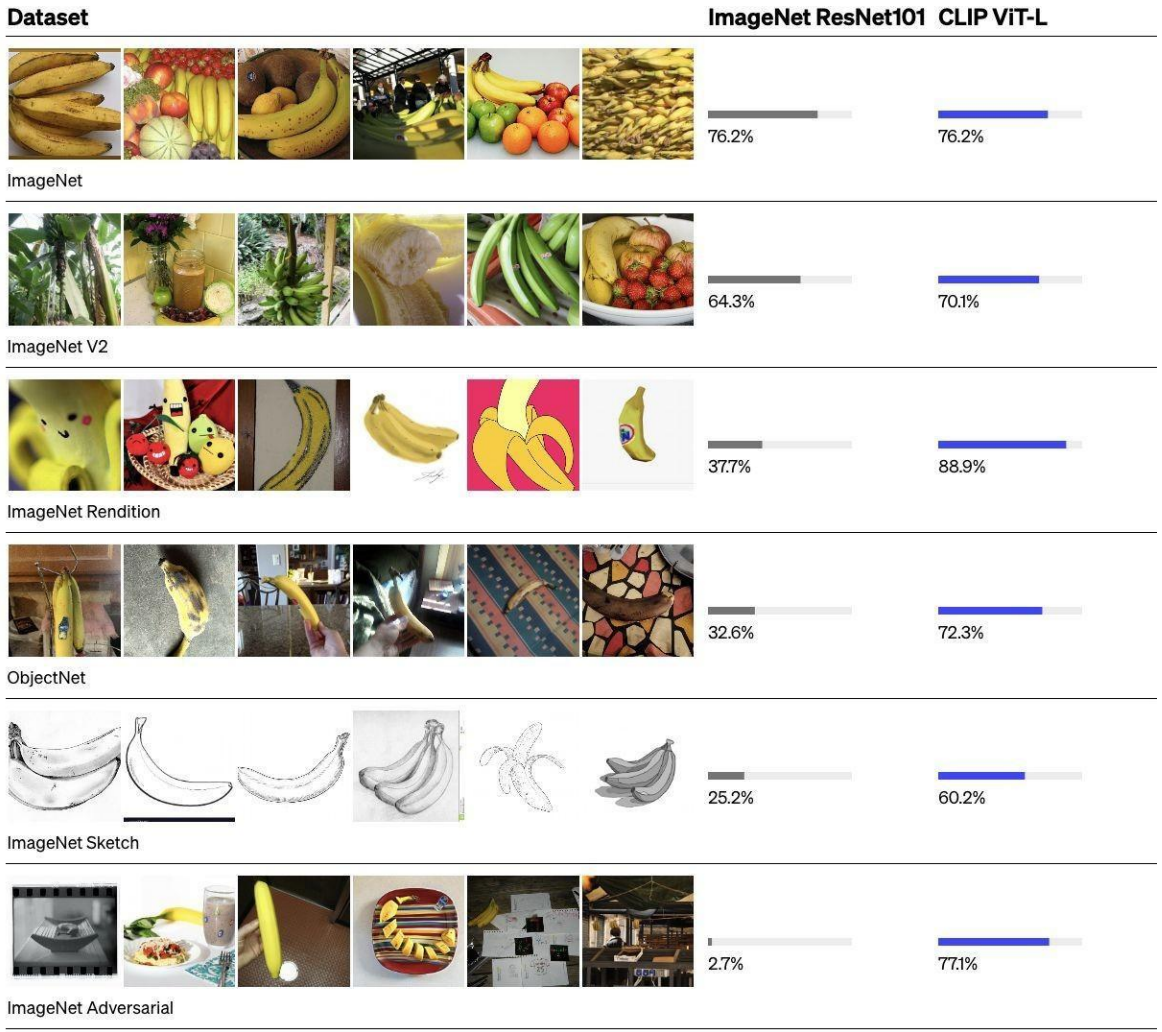




CLIP Robustness

IMHO one of the best papers ever written in our field: extremely thorough, worth a close read.

Generalizes MUCH better →



VisualBERT

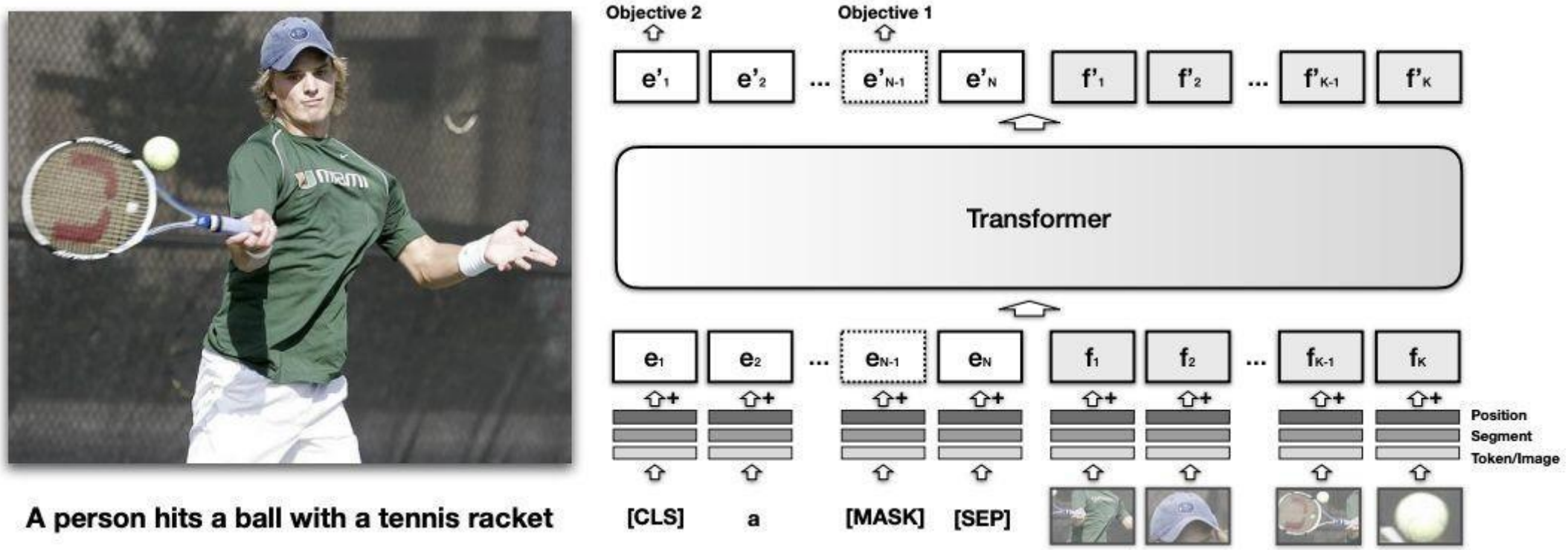
- Joint Transformer-based model: processes both image regions and text.
- **Early fusion:** image embeddings are added as input tokens to the Transformer.
- **Pretraining objectives:**
  - Masked Language Modeling (MLM)
  - Next Sentence Prediction (NSP)
  - Image-Text Alignment

Applications:

- Visual Question Answering (VQA)
- Image captioning
- Visual commonsense reasoning

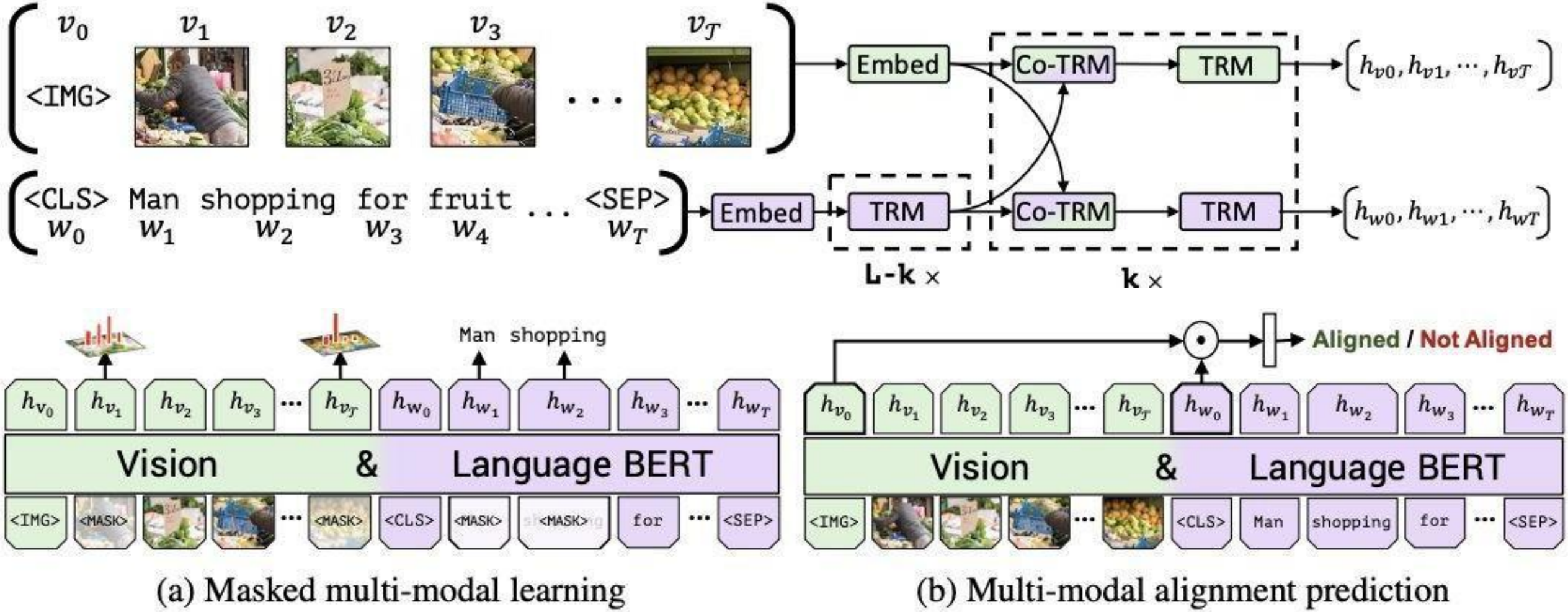
Paper: VisualBERT: A Simple and Performant Baseline for Vision and Language

Visual BERTs: VisualBERT



VisualBERT Li et al. 2019

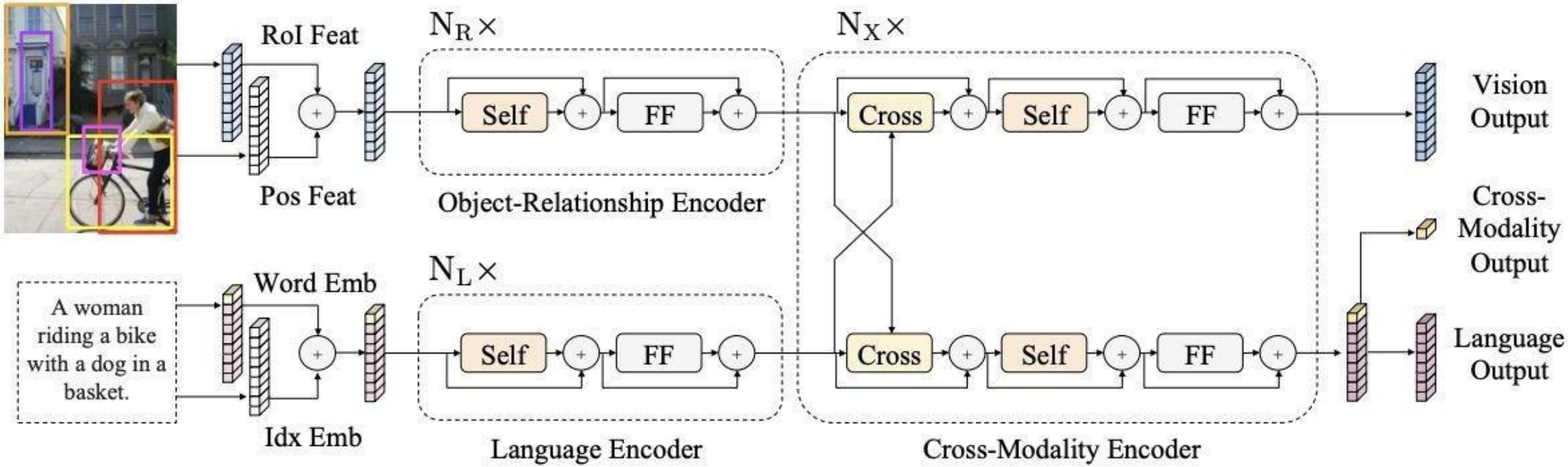
Visual BERTs: ViLBERT



ViLBERT Lu et al. 2019

Visual BERTs: LXMERT

Learning Cross-Modality Encoder Representations from Transformers



LXMERT Tan & Bansal 2019

Visual BERTs: Supervised Multimodal Bitransformers

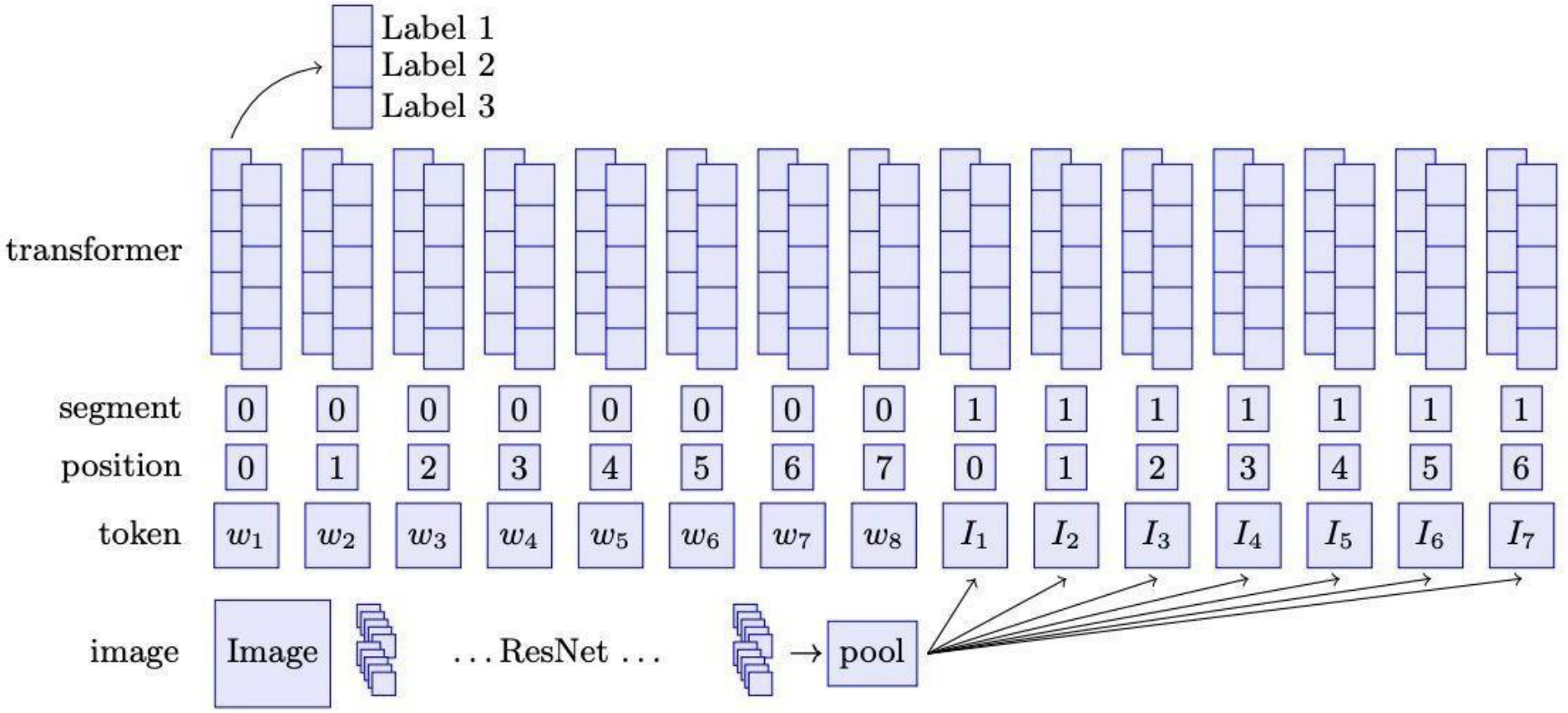
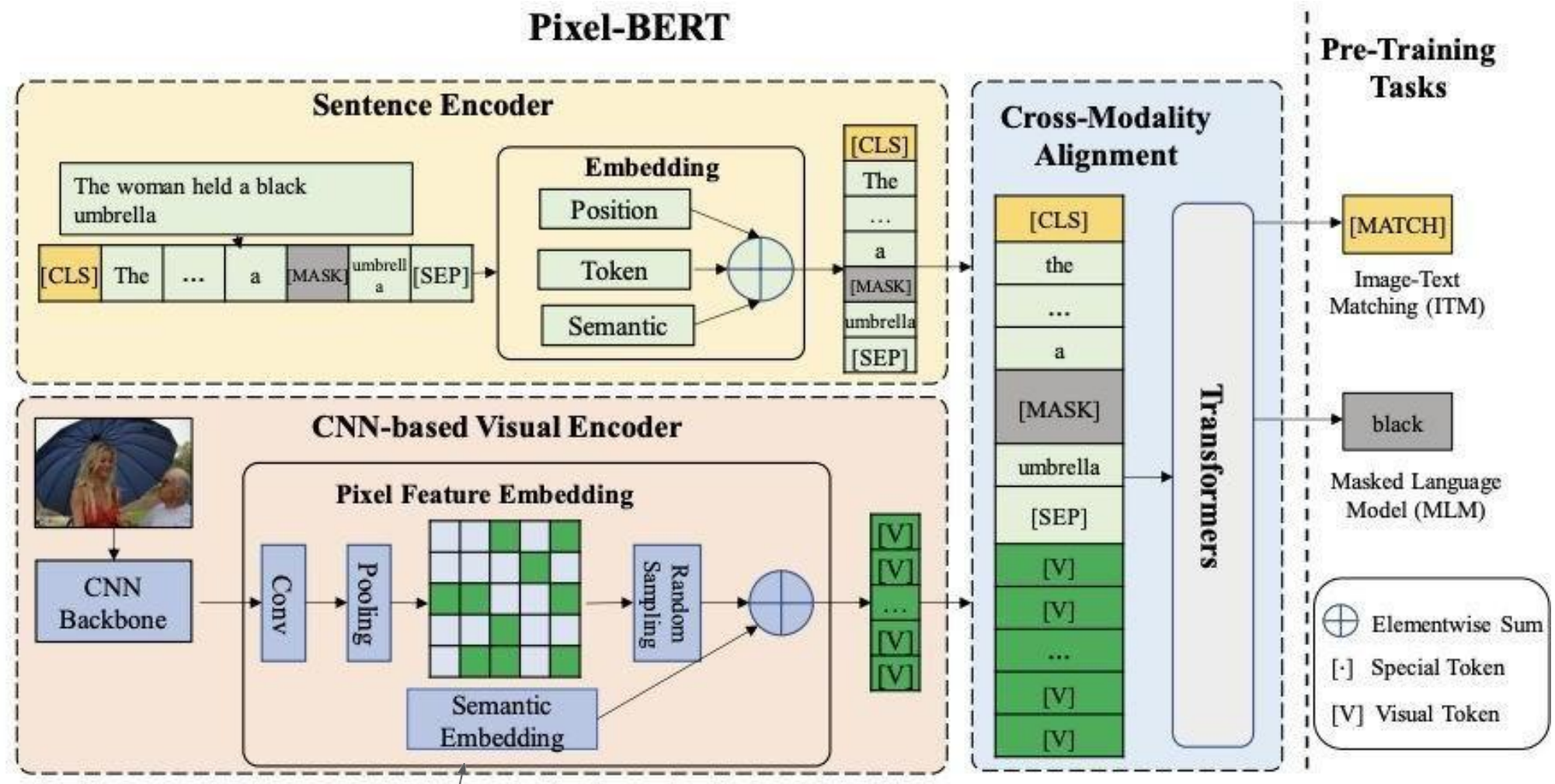


Figure 1: Illustration of the multimodal bitransformer architecture.

MMBT Kiela et al. 2019

Visual BERTs: PixelBert



Misnomer: they mean segment embedding

PixelBert Huang et al. 2020

FLAVA (Facebook AI)

- FLAVA = **Fusion of Language And Vision Architecture**
- Handles both unimodal and multimodal tasks
- **Three towers:** vision, language, multimodal
- Self-supervised objectives: masked modeling, contrastive learning

Capabilities:

- Text classification
- Image classification
- Visual Question Answering (VQA)
- Image captioning

Paper: FLAVA: A Foundational Language And Vision Alignment Model

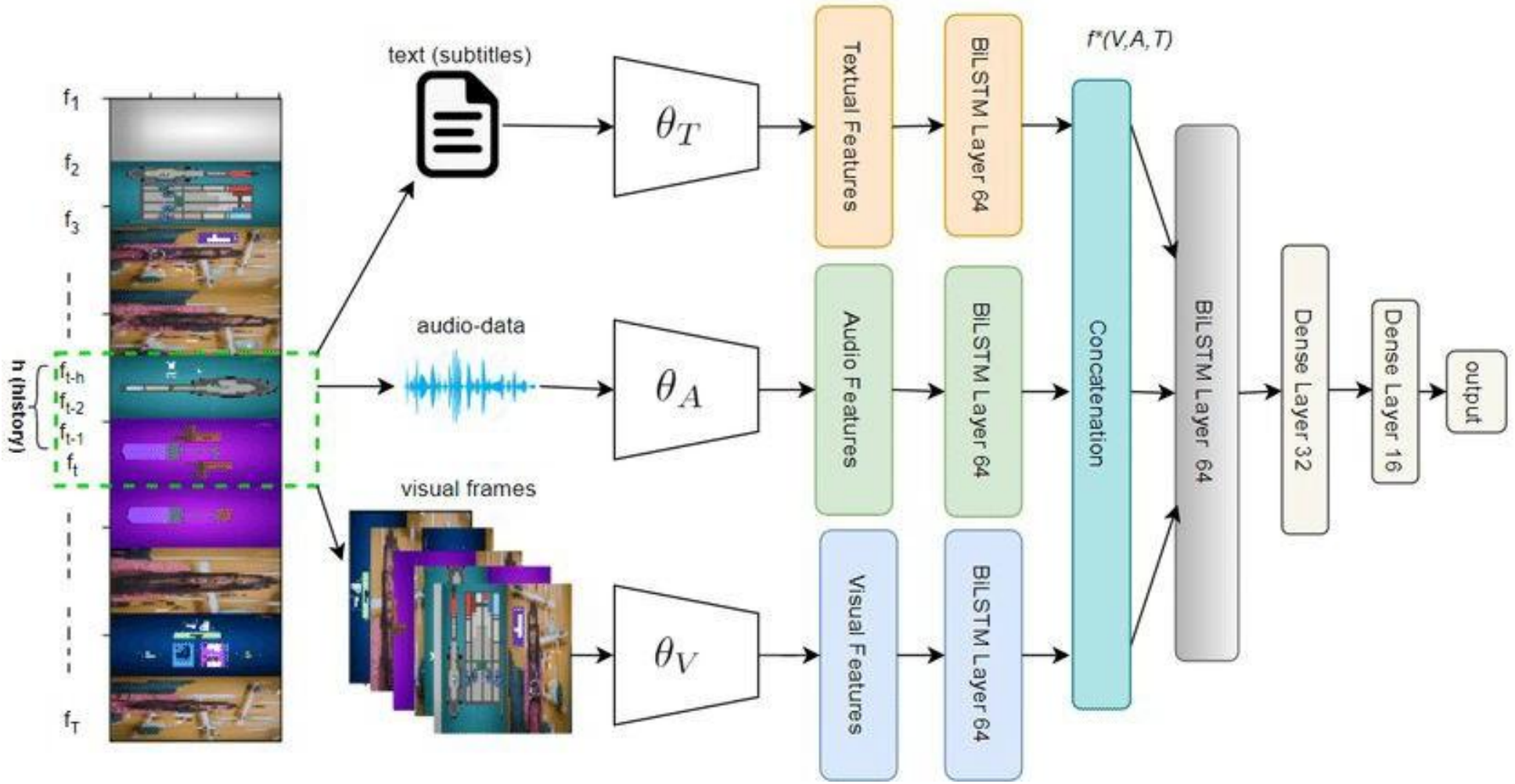
FLAVA (Singh et al., 2021)

Holistic approach to multimodality.
One foundation model spanning V&L, CV and NLP. Jointly pre-trained on:

- unimodal text data (CCNews + BookCorpus)
- unimodal image data (ImageNet)
- public paired image-text data (70M)

All data/models are publicly released.

Multimodal Architectures


text (subtitles)
audio-data
visual frames
f_1
f_2
f_3
f_{t-h}
f_{t-2}
f_{t-1}
f_t
f_T
h (history)
θ_T
θ_A
θ_V
Textual Features
Audio Features
Visual Features
BiLSTM Layer 64
BiLSTM Layer 64
BiLSTM Layer 64
f*(V,A,T)
Concatenation
BiLSTM Layer 64
Dense Layer 32
Dense Layer 16
output

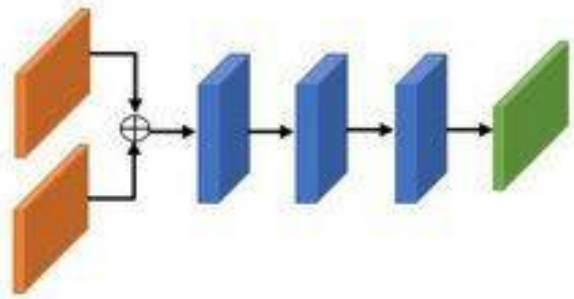
Categories of Multimodal Architectures

- **Early Fusion:** Combine raw modalities at the input level.
  Example: VisualBERT adds image embeddings as tokens to the Transformer.
- **Late Fusion:** Process each modality separately, then merge representations.
  Example: CLIP encodes images and text independently, then compares embeddings.
- **Hybrid / Cross-modal Attention:** Enable interactions between modalities at multiple stages using attention mechanisms.

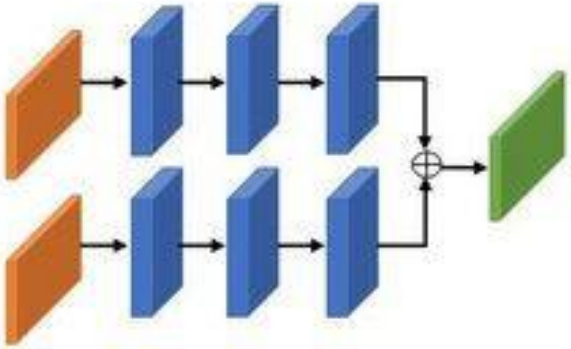
Design Considerations:

- **Alignment:** How to align representations across modalities.
- **Modality Dropout:** Handling missing or noisy modalities during training or inference.
- **Shared vs. Modality-Specific Layers:** Deciding which layers are shared and which are unique to each modality.

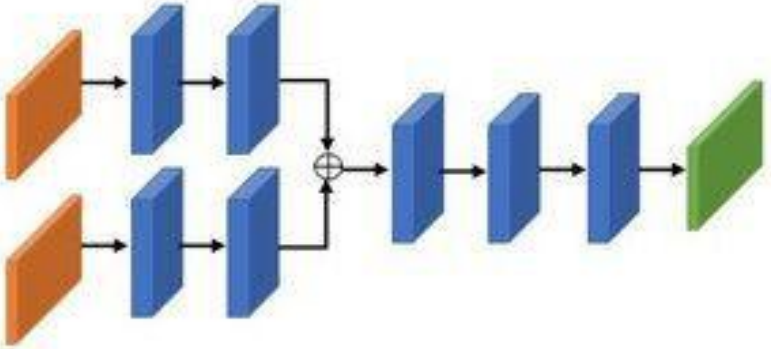
Categories of Multimodal Architectures



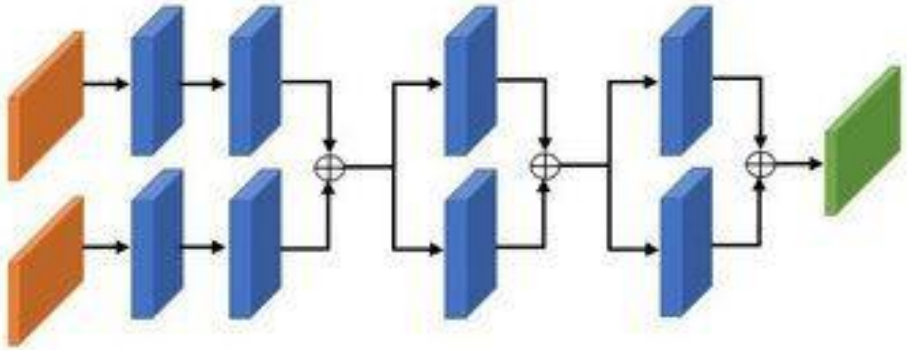
(a) Early Fusion



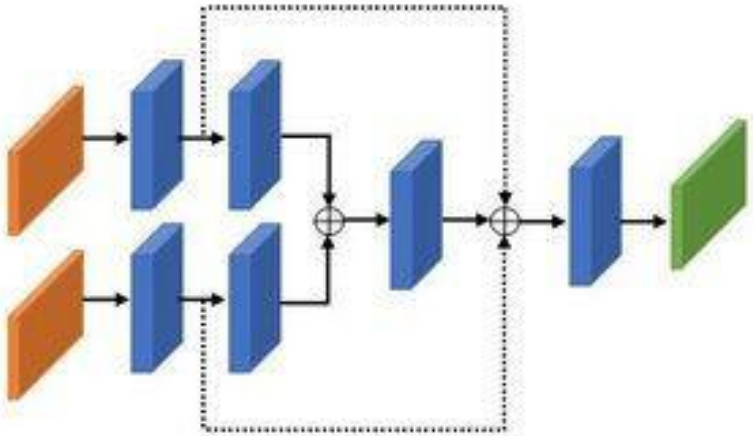
(b) Late Fusion



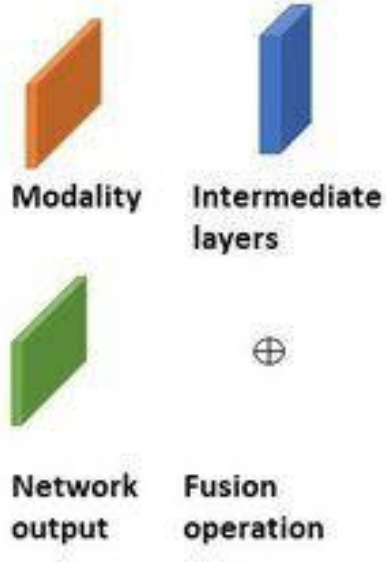
(c) Middle Fusion - fusion in one layer



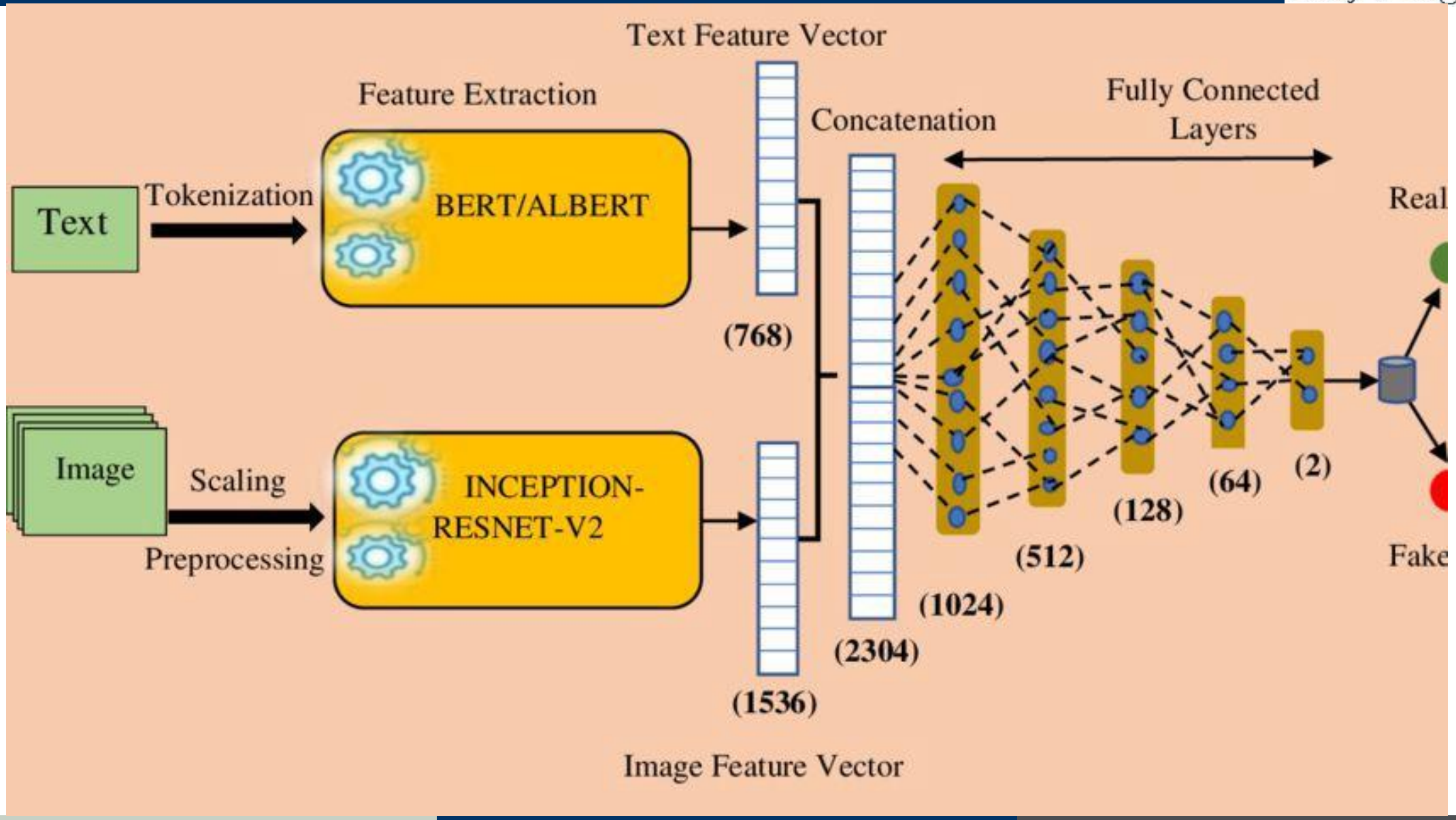
(d) Middle Fusion - deep fusion



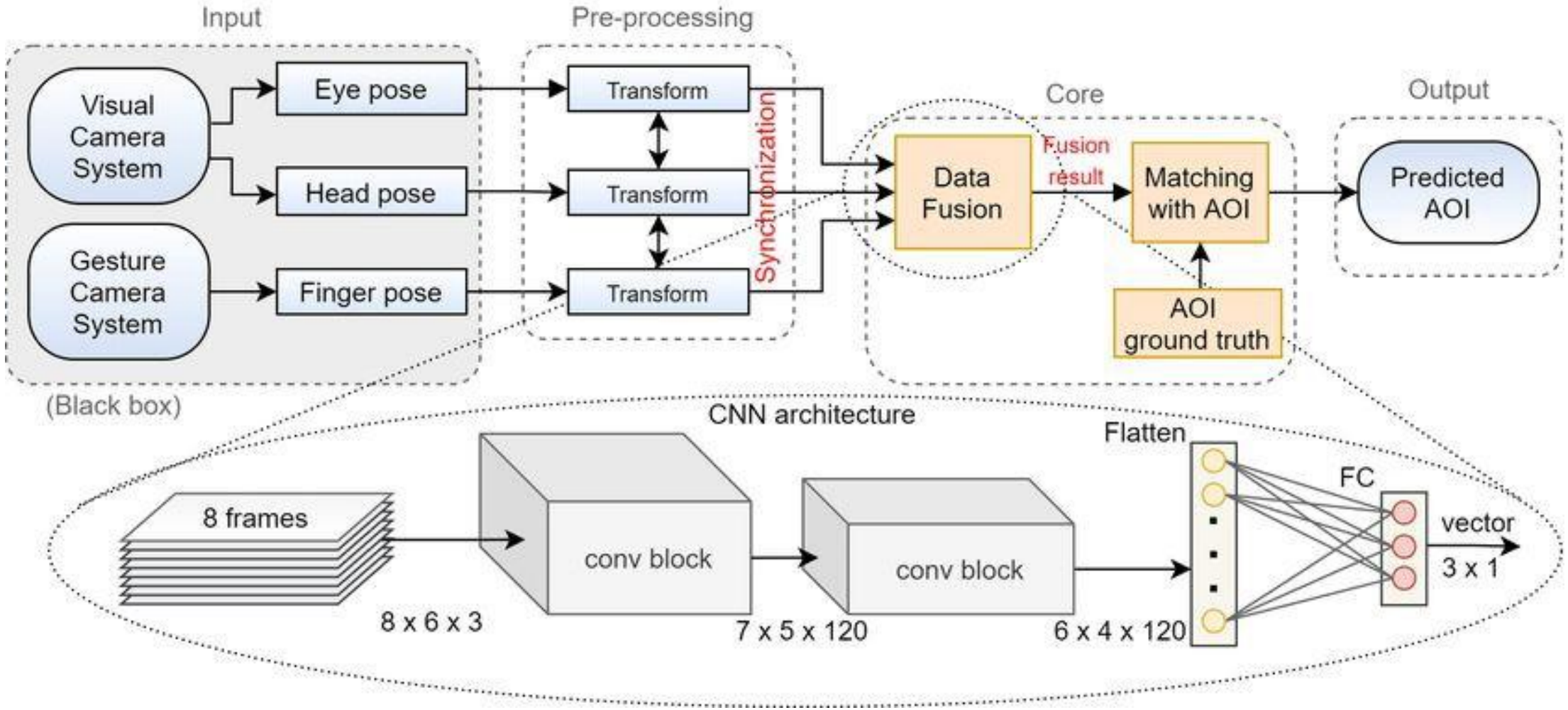
(e) Middle Fusion – short-cut fusion



Early Fusion



Late Fusion


Input
Pre-processing
Core
Output
Visual Camera System
Gesture Camera System
Eye pose
Head pose
Finger pose
Transform
Transform
Transform
Synchronization
Data Fusion
Fusion result
Matching with AOI
AOI ground truth
Predicted AOI
(Black box)
CNN architecture
8 frames
8 x 6 x 3
conv block
7 x 5 x 120
conv block
6 x 4 x 120
Flatten
FC
vector
3 x 1

Challenges in Multimodal Integration

أكاديمية كاوست
KAUST ACADEMY

LMH
Lady Margaret Hall

MultiModal Learning Challenges

Representation

Finding a good representation is crucial for the success of multimodal models that combine different modalities such as image, text, and audio.

Fusion

Fusing information from different modalities to perform a prediction task is the core of multimodal learning and can be challenging due to the diverse nature of multimodal data.

Alignment

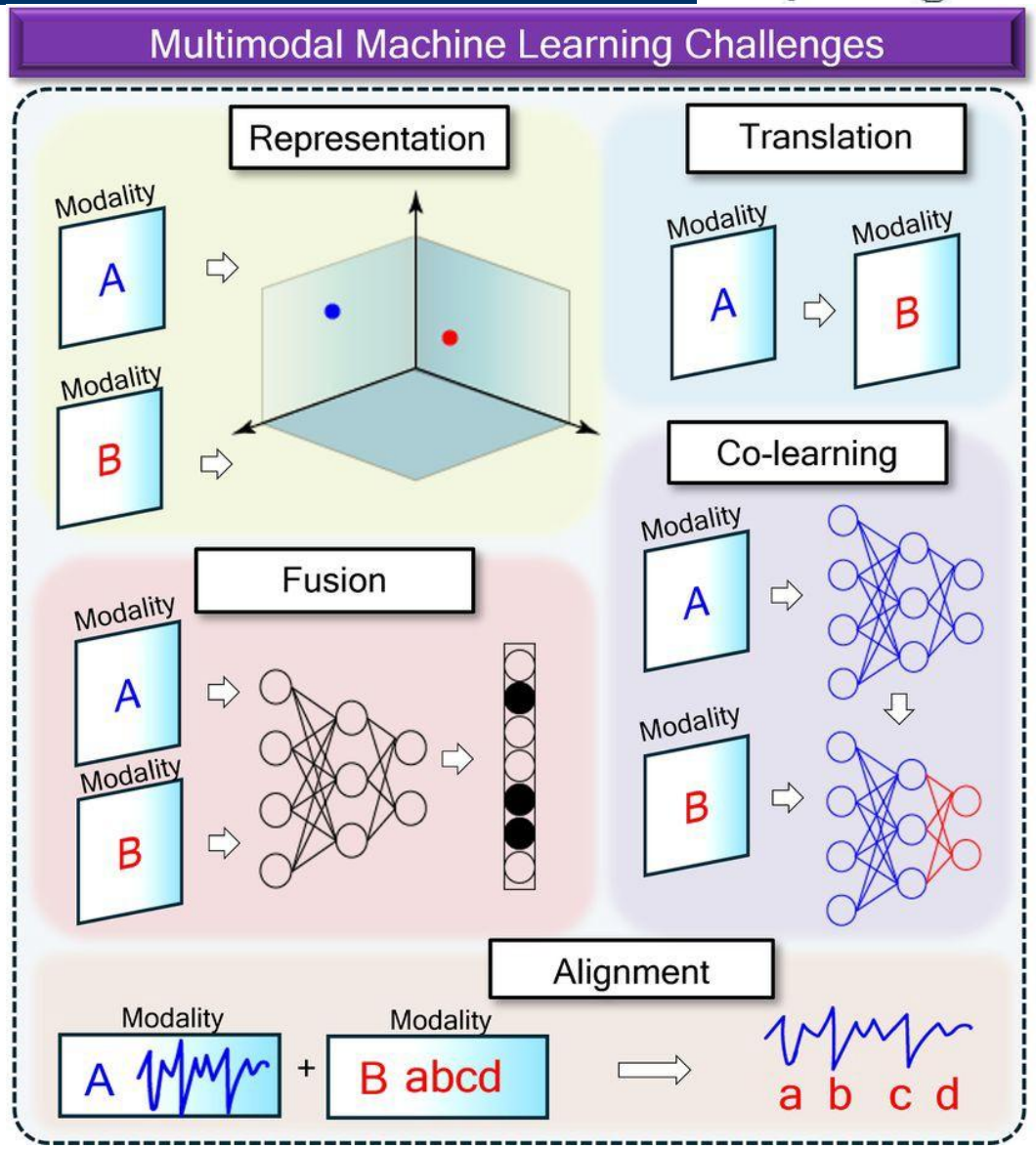
Aligning data from different modalities to create modality-invariant representations is necessary to combine information effectively.

Translation

Translating data from one modality to another is a crucial task in multimodal learning, but can be difficult due to the subjective and open-ended nature of translations.

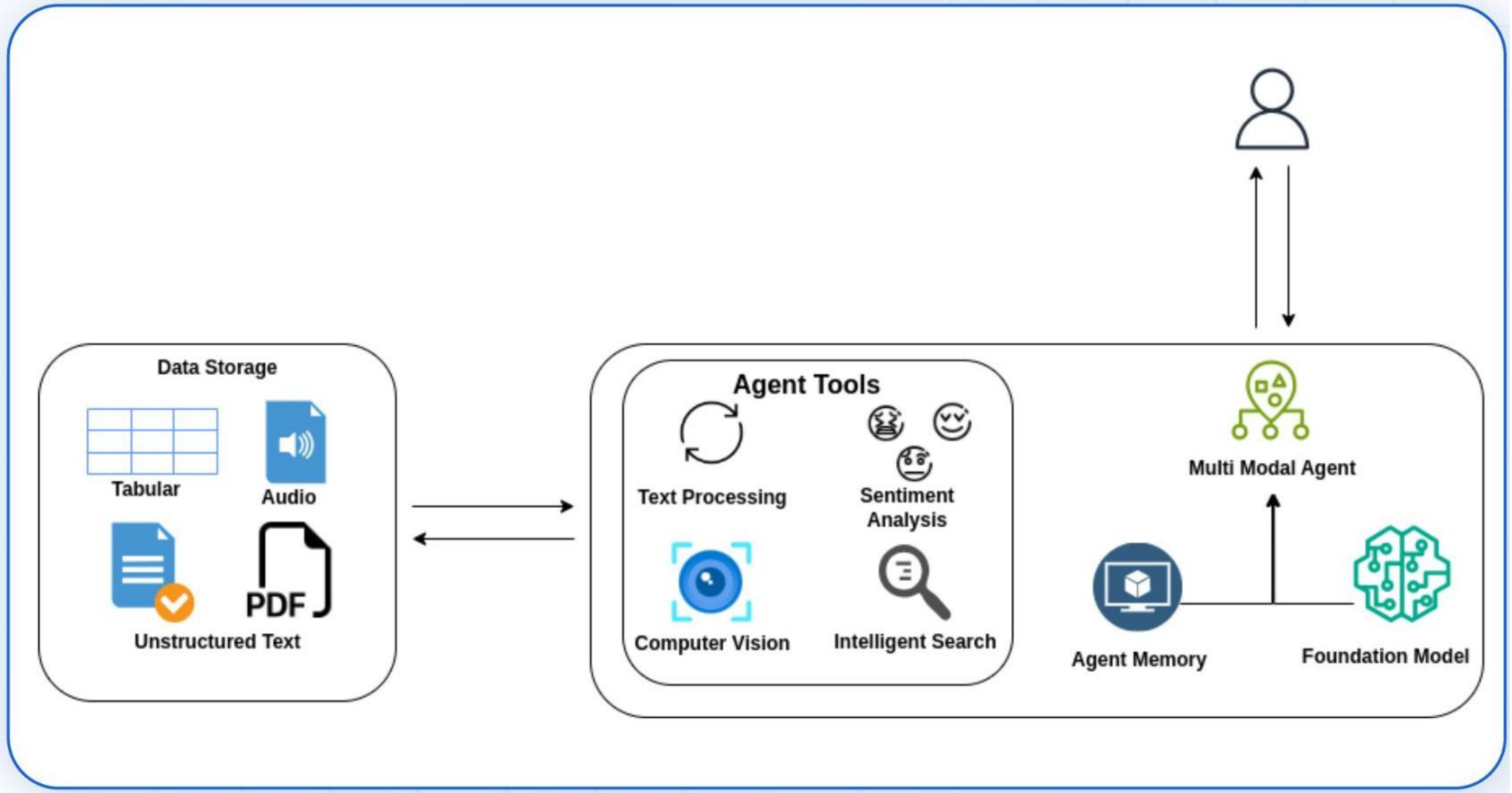
Co-learning

Co-learning aims to transfer knowledge learned through one or more modalities to tasks involving another. It is useful in medical diagnosis combining CT and MRI scans.

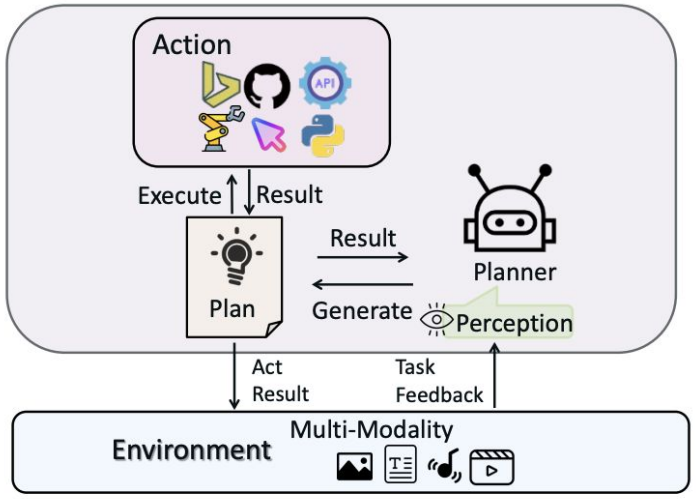


Multimodal Agents

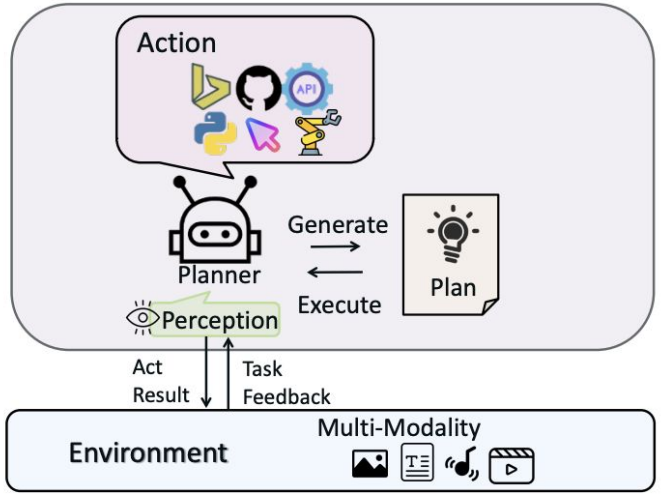
Introduction



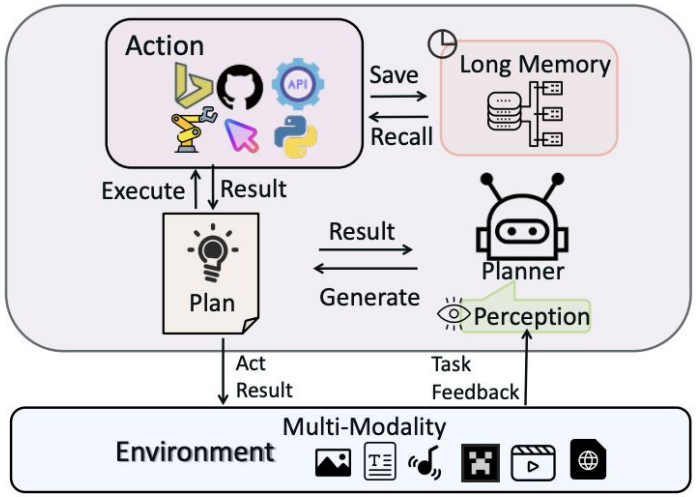
Agentic Framework



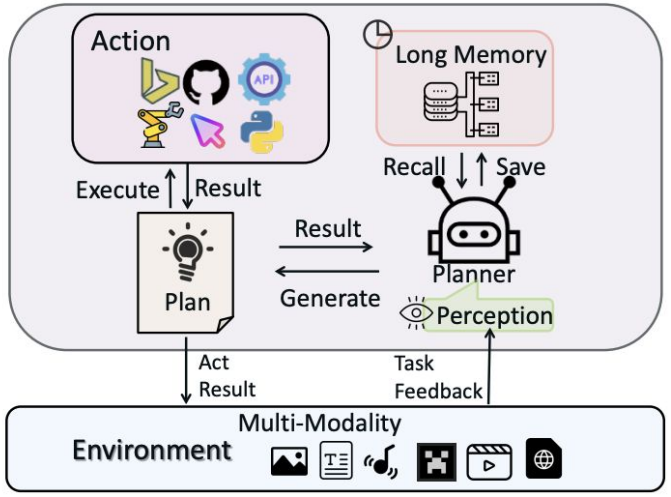
(a)



(b)



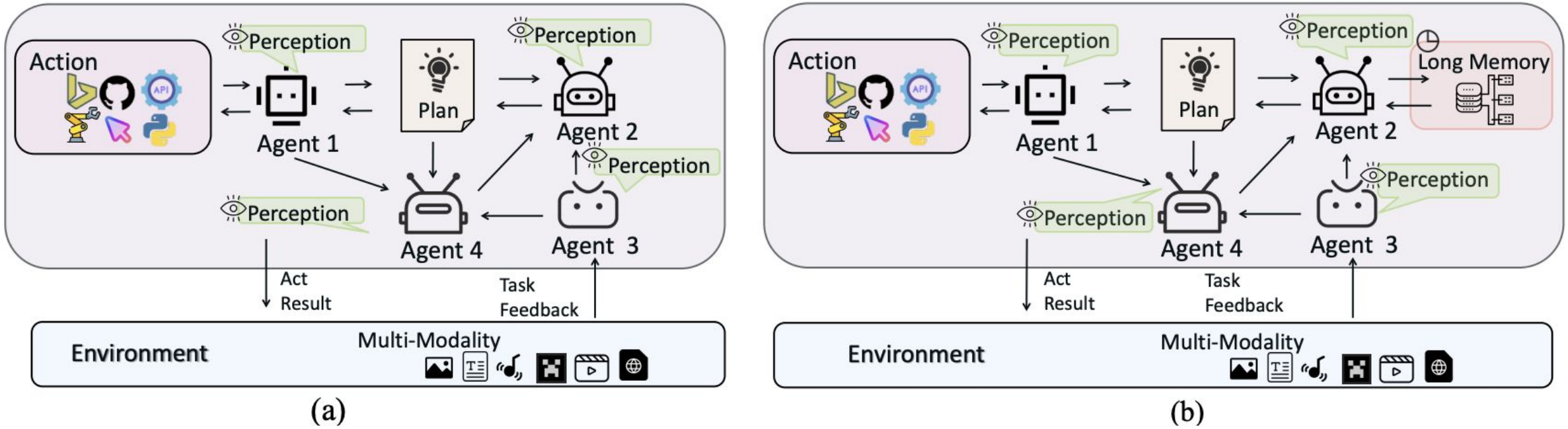
(c)



(d)

(a) Type I: Closed-source LLMs as Planners w/o Longterm Memory. They mainly use prompt techniques to guide closed-source LLMs in decision-making and planning to complete tasks without long memory. (b) Type II: Finetuned LLMs as Planners w/o Long-term Memory. They use action-related data to finetune existing open-source large models, enabling them to achieve decision-making, planning, and tool invocation capabilities comparable to closed-source LLMs.(c) and (d) introduce long-term memory functions, further enhancing their generalization and adaptation abilities in environments closer to the real world. However, because their planners use different methods to retrieve memories, they can be further divided into: (c) Type III: Planners with Indirect Long-term Memory; (d) Type IV: Planners with Native Long-term Memory.

Multi-agent frameworks



Illustrations on two types of multi-agent frameworks: in these two frameworks, facing tasks or instructions from the environment, the completion relies on the cooperation of multiple agents. Each agent is responsible for a specific duty, which may involve processing environmental information or handling decision-making and planning, thus distributing the pressure that would otherwise be borne by a single- agent to complete the task. The unique aspect of framework (b) is its long-term memory capability.

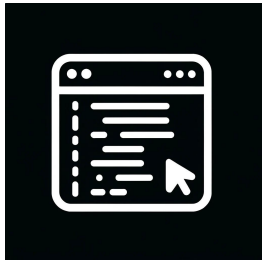
Code Generation

Code Generation

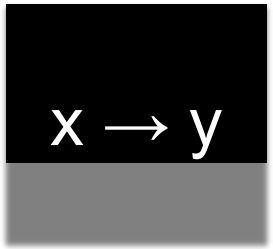
- Generate executable code as an interface to a program
- Why do it?
  - Accelerates software engineering
  - Enables models to access tools (more in a later class)

Inputs and Outputs

Natural language



Code context



Inputs and outputs

Screenshots

Code

```python
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def display(self):
        print(f"Name: {self.name}, Age: {self.age}")

    def years_until_100(self):
        years_left = 100 - self.age
        if years_left <= 0:
            print(f"{self.name} is already 100 years old or more!")
        else:
            print(f"{self.name} has {years_left} years until they turn 100.")
```

Example: Github Copilot

أكاديمية كاوست
KAUST ACADEMY

LMH
Lady Margaret Hall

- A code completion engine within your IDE (VSCode)

```
# load hello.csv
data = np.loadtxt('hello.csv', delimiter=',', skiprows=1)

# add together the two largest numbers in the first column
result = np.sum(np.sort(data[:, 0])[-2:])

# print how much bigger these two numbers are than the average of the first column
result = result - np.mean(data[:, 0])
```

https://youtu.be/xe3oI0rTeaQ

Example: Claude 3

G Create a React app that looks like this screenshot.

To create a React app that looks similar to the screenshot, you can follow these steps:

1. Set up a new React project using Create React App or your preferred method.
2. Create a component for the chat interface, such as `ChatInterface`.
3. Inside the `ChatInterface` component, add a header with the app name "Claude"

Here's a simplified example of the `ChatInterface` component:

jsx Copy code

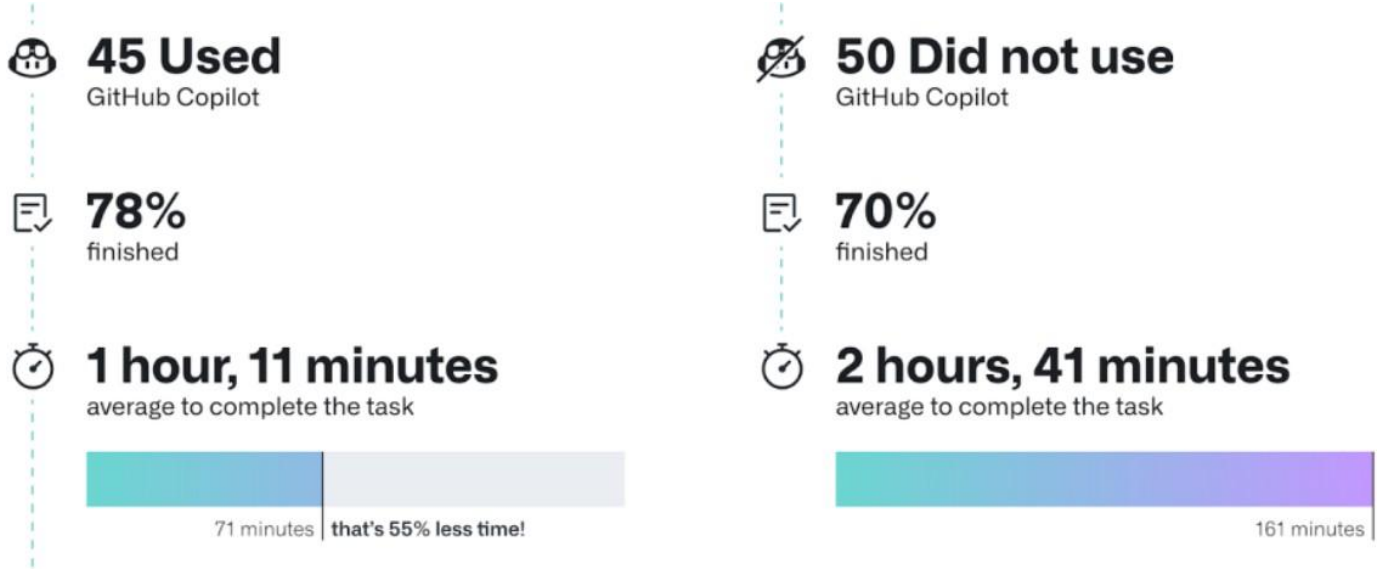
```jsx
import React from 'react';
import styled from 'styled-components';

const ChatContainer = styled.div`
  /* Add styles for the chat container */
`;

const Header = styled.header`
  /* Add styles for the header */
`;

const ChatSection = styled.section`
  /* Add styles for the chat section */
`;

const CapabilityCard = styled.div`
  /* Add styles for the capability cards */
`;

const ChatInterface = () => {
  return (
    <ChatContainer>
      <Header>
```

Why do we do it?

أكاديمية كاوست
KAUST ACADEMY

LMH
Lady Margaret Hall

- Making software is important!
  - US software developers make $175B per year
- Code generation leads to large improvements in productivity (Github 2023)



Differences Between code and Natural Language

- Code has strict grammar
- We know the code semantic flow
- Code is executable
- Code is created more incrementally

Code Generation -

Subtasks and Datasets

HumanEval (chen et al. 2021)

أكاديمية كاوست
KAUST ACADEMY

Lady Margaret Hall

- Examples of usage of the Python standard library
- 164 test examples
- Includes docstring, some example inputs/outputs, and tests

```
def solution(lst):
    """Given a non-empty list of integers, return the sum of all of the odd elements
    that are in even positions.

    Examples
    solution([5, 8, 7, 1]) ==>12
    solution([3, 3, 3, 3, 3]) ==>9
    solution([30, 13, 24, 321]) ==>0
    """
    return sum(lst[i] for i in range(0,len(lst)) if i % 2 == 0 and lst[i] % 2 == 1)
```

```
def encode_cyclic(s: str):
    """
    returns encoded string by cycling groups of three characters.
    """
    # split string to groups. Each of length 3.
    groups = [s[(3 * i):min((3 * i + 3), len(s))] for i in range((len(s) + 2) // 3)]
    # cycle elements in each group. Unless group has fewer elements than 3.
    groups = [(group[1:] + group[0]) if len(group) == 3 else group for group in groups]
    return "".join(groups)

def decode_cyclic(s: str):
    """
    takes as input string encoded with encode_cyclic function. Returns decoded string.
    """
    # split string to groups. Each of length 3.
    groups = [s[(3 * i):min((3 * i + 3), len(s))] for i in range((len(s) + 2) // 3)]
    # cycle elements in each group.
    groups = [(group[-1] + group[:-1]) if len(group) == 3 else group for group in groups]
    return "".join(groups)
```

Metric: Pass@K (Chen et al. 2021)

- Basic idea: “if we generate *K* examples, will at least one of them pass unit tests”
- Generating only *K* will result in high variance, so we generate *N* > *K* with *C* correct answers, and then calculate expected value

$$\text{pass@}k := \mathop{\mathbb{E}}_{\text{Problems}} \left[1 - \frac{\binom{n-c}{k}}{\binom{n}{k}} \right]$$

Broader Domains: CoNaLa/ODEX

(Yin et al. 2018, Wang et al. 2022)

- CoNaLa: Broader data scraped from StackOverflow
- ODEX: Adds execution-based evaluation
- Wider variety of libraries

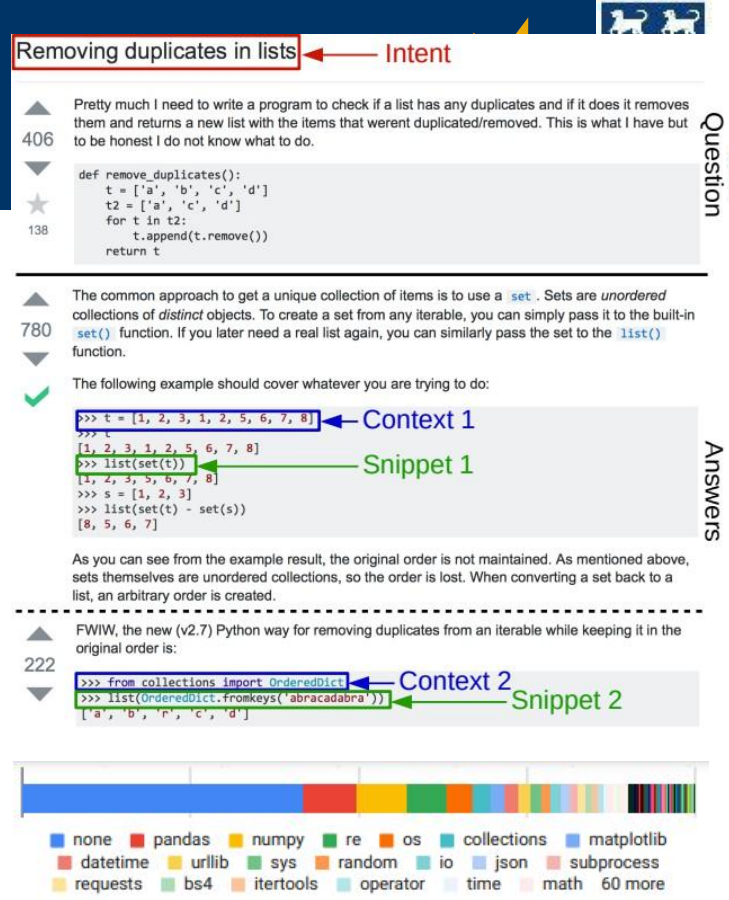
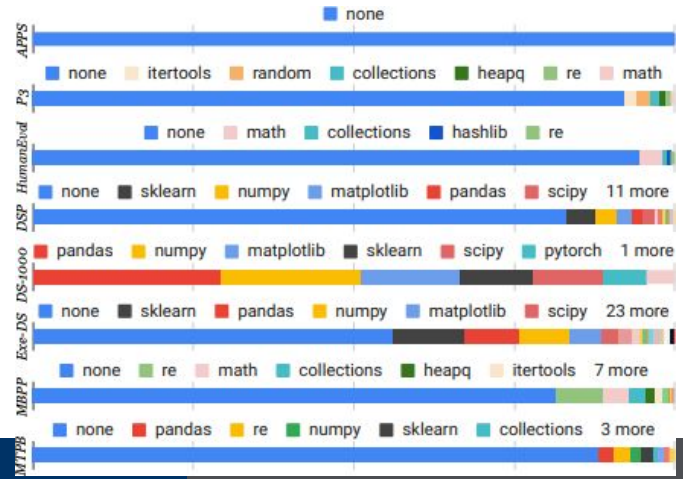
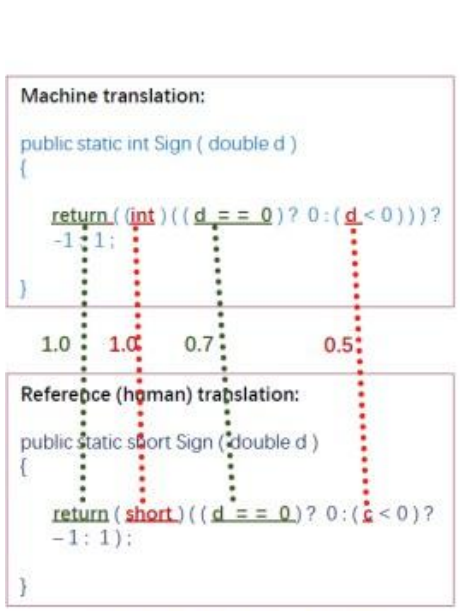


Figure 3: ODEX library distribution.

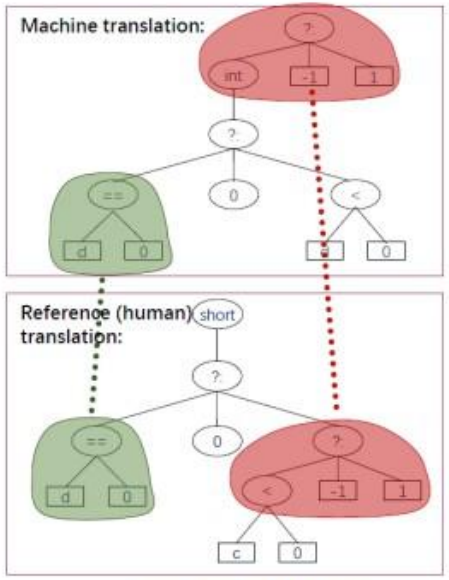


Metric: BLEU, CodeBLEU

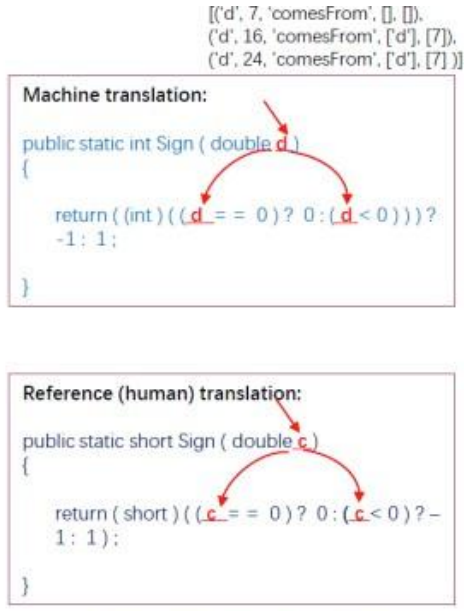
- Issues w/ execution-based evaluation:
  - Requires that code be easily executable (requires unit tests and hard in large libraries)
  - Ignores stylistic considerations
- **BLEU:** consider text n-gram overlap with human code
- **CodeBLEU:** also considers syntax and semantic flow (Ren et al. 2020)



Weighted N-Gram Match



Syntactic AST Match



Semantic Data-flow Match

Metric: CodeBERTScore

- It is also possible to use embedding-based methods to compare code
- **CodeBERTScore:** BERTScore with CodeBERT trained on lots of code (Zhou et al. 2023)
- Leads to better correlation w/ human judgement and execution accuracy

Metric	Java		C++		Python		JavaScript	
	τ	r_s	τ	r_s	τ	r_s	τ	r_s
BLEU	.481	.361	.112	.301	.393	.352	.248	.343
CodeBLEU	.496	.324	.175	.201	.366	.326	.261	.299
ROUGE-1	.516	.318	.262	.260	.368	.334	.279	.280
ROUGE-2	.525	.315	.270	.273	.365	.322	.261	.292
ROUGE-L	.508	.344	.258	.288	.338	.350	.271	.293
METEOR	**.558**	**.383**	.301	.321	.418	.402	**.324**	**.415**
chrF	.532	.319	.319	.321	.394	.379	.302	.374
CrystalBLEU	.471	.273	.046	.095	.391	.309	.118	.059
CodeBERTScore	**.553**	.369	**.327**	**.393**	**.422**	**.415**	**.319**	.402

Table 1: Kendall-Tau (τ) and Spearman (r_s) correlations of each metric with the functional correctness on HumanEval in multiple languages. The correlation coefficients are reported as the average across three runs. Standard deviation is provided in Table 3.

Metric	τ	r_p	r_s
BLEU	.374	.604	.543
CodeBLEU	.350	.539	.495
ROUGE-1	.397	.604	.570
ROUGE-2	.429	.629	.588
ROUGE-L	.420	.619	.574
METEOR	.366	.581	.540
chrF	.470	.635	.623
CrystalBLEU	.411	.598	.576
CodeBertScore	**.517**	**.674**	**.662**

Table 2: The Kendall-Tau (τ), Pearson (r_p) and Spearman (r_s) correlation with human preference. The best performance is **bold**. The correlation coefficients are reported as the average across three runs. Standard deviations are provided in Table 4.

Data Science Notebooks: ARCADE

(Yin et al. 2022)

- Data science notebooks (e.g. Jupyter) allow for incremental implementation
- Allows evaluation of code in context

[1] c_1
```
import pandas as pd

df = pd.read_csv('dataset/Gamepass_Games_v1.csv')
```

[2] u_1 Extract min and max hours as two columns
c_2
```
def get_avg(x):
  try: return float(x[0]) , float(x[1])
  except: return 0, 0

df['min'], df['max'] = zip(*df['TIME'].str.replace(
  ' hours','').str.split("-").apply(get_avg))
```

[3] c_3
```
df['ADDED'] = pd.to_datetime(
    df['ADDED'],format="%d %b %y",errors='coerce')
```

[4] u_2 In which year was the most played game added?
c_4
```
df['GAMERS']=df['GAMERS'].str.replace(
                                    ',','').astype(int)
added_year=df[df['GAMERS'].idxmax()]['ADDED'].year
```

[5] u_3 For each month in that year, how many games that has a rating of more than four?
c_5
```
df[(df['ADDED'].dt.year== added_date.year) &
(df['RATING']>4)].groupby(
                df["ADDED"].dt.month)['GAME'].count()
```

[6] u_4 What is the average maximum completion time for all fallout games added in 2021?
c_6
```
fallout=df[df['GAME'].str.contains('Fallout')]
fallout.groupby(fallout['ADDED'].dt.year).get_group(
  2021)['max'].mean()
```

[7] u_5 What is the amount of games added in each year for each month? (show a table with index as years, columns as months and fill null values with 0)
c_7
```
pd.pivot_table(df, index=df['ADDED'].dt.year, ...,
  aggfunc=np.count_nonzero,
  fill_value='0').rename_axis(
    index='Year', columns='Month')
```

Figure 1: An example of a computational notebook adapted from our dataset, with examples of reading and preprocessing data (cell c_1), data wrangling (cell c_2, c_3), and data analysis (cells $c_3 - c_7$). Annotated NL intents are shown in green.

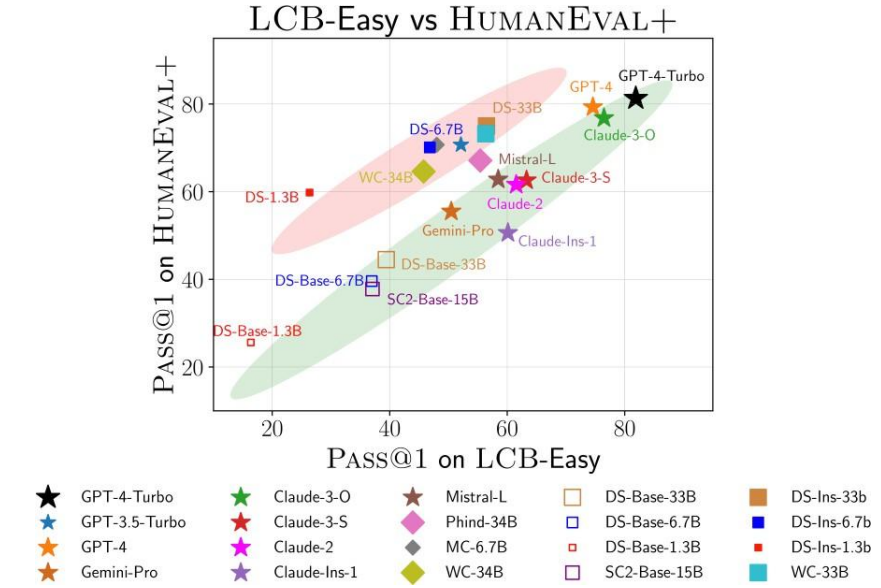
An Aside: Dataset Leakage

أكاديمية كاوست
KAUST ACADEMY

LMH
Lady Margaret Hall

- Leakage of datasets is a big problem
- ARCADE shows that novel notebooks are harder than online notebooks
- LiveCodeBench (Jain et al. 2023) shows that some code LMs outperform on HumanEval

pass@k	Existing 5	New 5
InCoder 1B	30.1	3.8
InCoder 6B	41.3	7.0
CodeGen$_{multi}$ 350M	13.3	1.0
CodeGen$_{multi}$ 2B	25.0	2.7
CodeGen$_{multi}$ 6B	28.0	3.0
CodeGen$_{multi}$ 16B	31.2	4.6
CodeGen$_{mono}$ 350M	18.9	1.9
CodeGen$_{mono}$ 2B	35.8	6.5
CodeGen$_{mono}$ 6B	42.1	8.9
CodeGen$_{mono}$ 16B	46.7	12.0
PaLM 62B (1.3T Tokens)	49.7	12.5
+ Python Code	58.8 +9.1	21.4 +8.9
+ Notebooks (PaChiNCo)	64.6 +7.8	30.6 +9.2
− Schema Description	60.5 −4.1	22.7 −7.9



Dataset: SWEBench

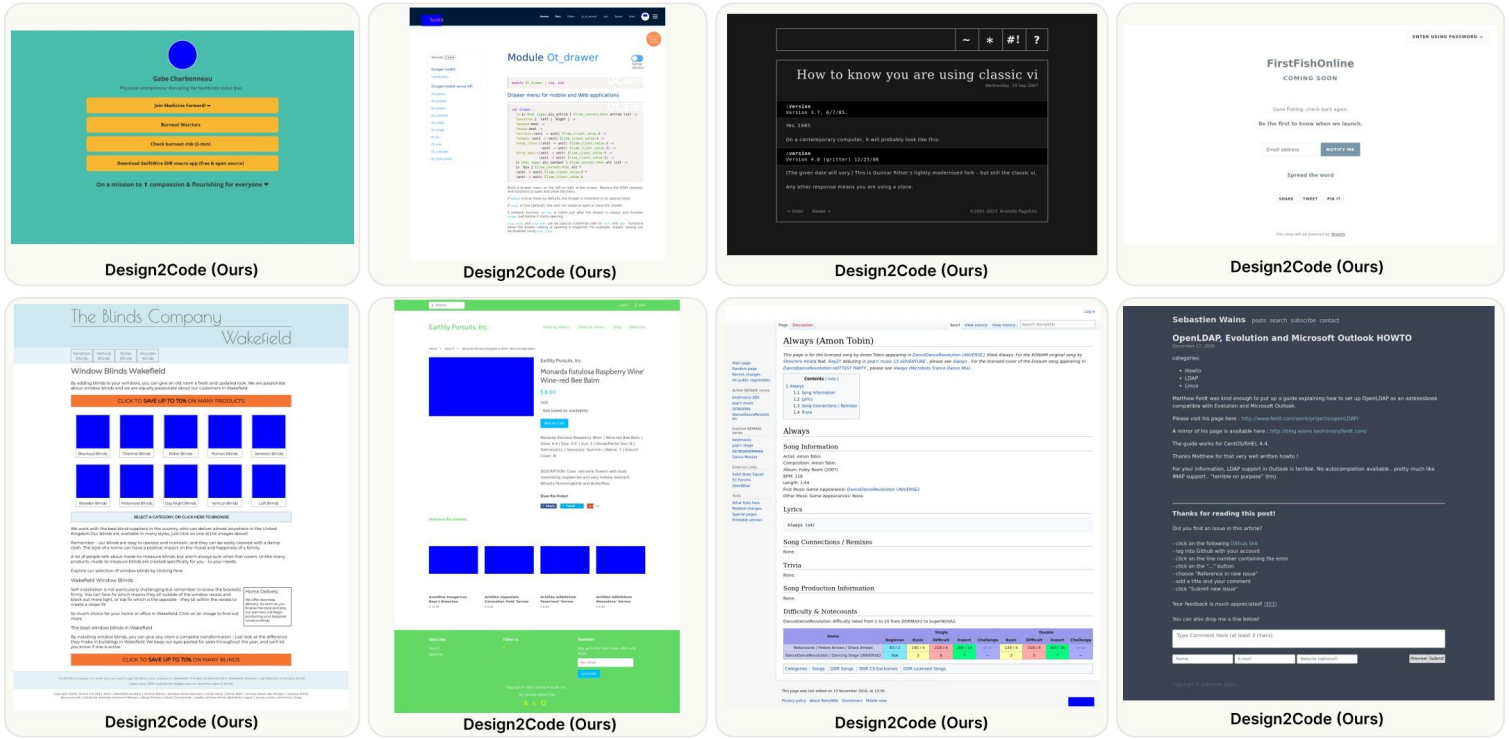
(Jiminez et al. 2023)

- Issues from GitHub + codebases -> pull request

- Requires long-context understanding, precise implementation

Dataset: Design2Code (Si et al. 2024)

- Code generation from web sites



- Also proposed Design2Code model

Metric: Visual Similarity of Web Site

أكاديمية كاوست
KAUST ACADEMY

LMH
Lady Margaret Hall

- Design2Code evaluates by two metrics
- **High-level visual similarity:** Similarity between visual embeddings of the generated sites
- **Low-level element similarity:** Recall of each individual element

Code Generation - Methods

Basic Method: Code-generating LM

- Feed the previous code to an LM
- Virtually all serious LMs are trained on code nowadays, and have the ability to complete queries
- More on specific LMs later!
- Note: temperature settings are important, set to lower values

Code Infilling

أكاديمية كاوست
KAUST ACADEMY

LMH
Lady Margaret Hall

(Fried et al. 2022)

- In code generation, we often want to fill in code
- Solution: train for infilling

Training

Original Document

```
def count_words(filename: str) -> Dict[str, int]:
    """Count the number of occurrences of each word in the file."""
    with open(filename, 'r') as f:
        word_counts = {}
        for line in f:
            for word in line.split():
                if word in word_counts:
                    word_counts[word] += 1
                else:
                    word_counts[word] = 1
    return word_counts
```

Masked Document

```
def count_words(filename: str) -> Dict[str, int]:
    """Count the number of occurrences of each word in the file."""
    with open(filename, 'r') as f:
        <MASK:0> in word_counts:
                    word_counts[word] += 1
                else:
                    word_counts[word] = 1
    return word_counts
<MASK:0> word_counts = {}
        for line in f:
            for word in line.split():
                if word <EOM>
```

Lots of Available Information for Coding!

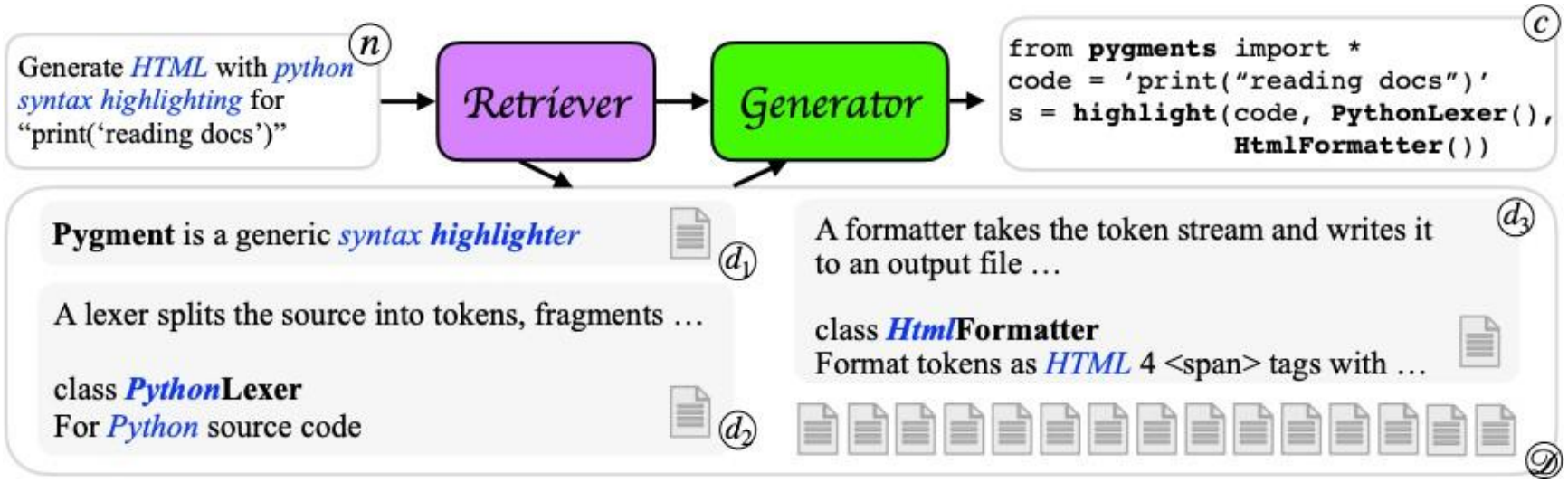
- Current code context
- Description of issue to fix
- Repo context
- Open tabs

Example: Copilot Prompting Strategy (Thakkar 2023)

- **Extract prompt** given current doc and cursor position
- Identify **relative path and language**
- Find **most recently accessed** 20 files of the same language
- **Include:** text before, text after, similar files, imported files, metadata about language and path
- **TL;DR:** lots of prompt engineering to get most useful context tin the prompt

Retrieval-based Code Generation

- Retrieve similar code from online, and fill it in with a retrieval-augmented LM (Hayati et al. 2018)
- Particularly, in code there is also documentation, which can be retrieved (Zhou et al. 2022)

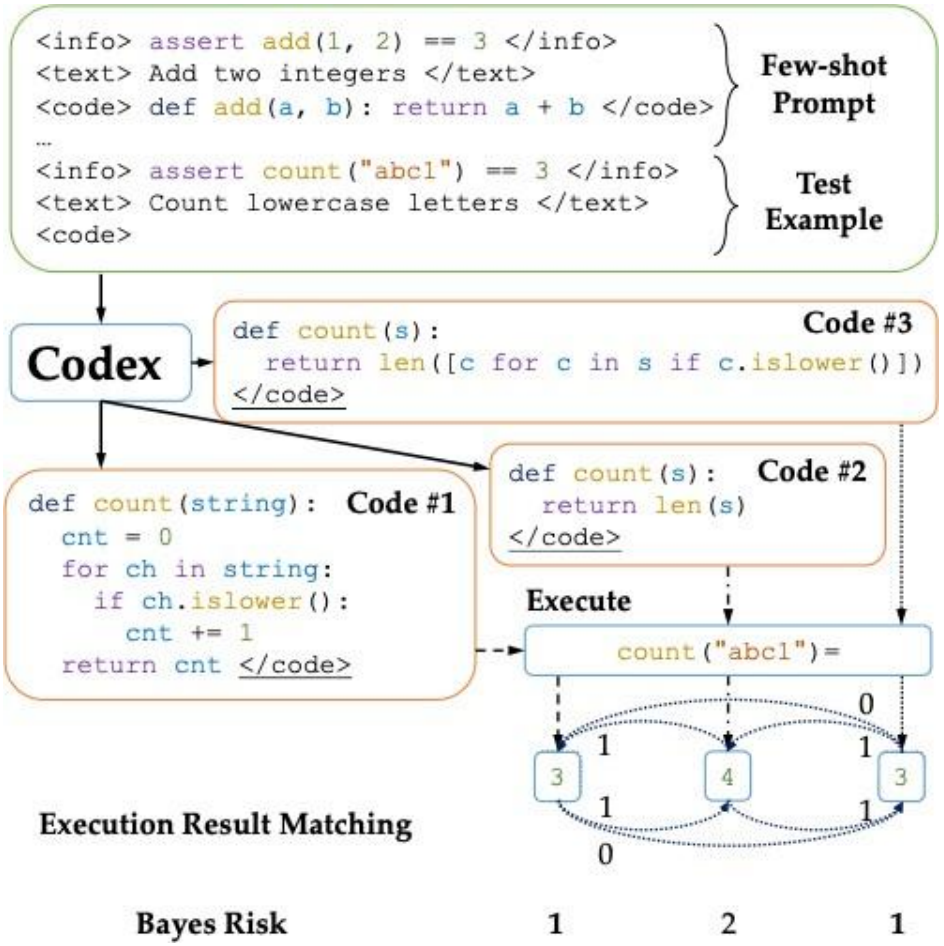


Execution Feedback (Shi et al. 2022)

أكاديمية كاوست
KAUST ACADEMY

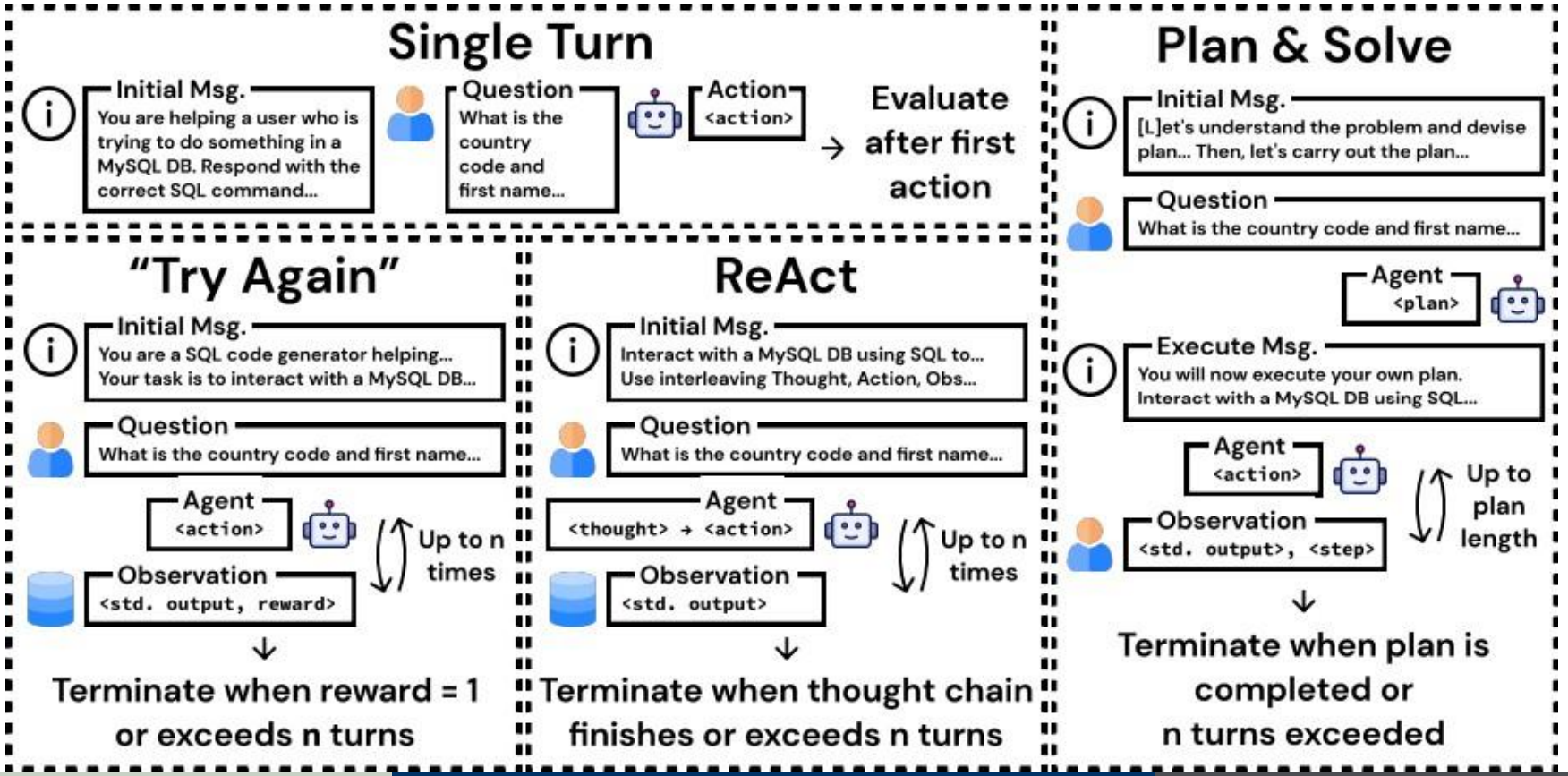
Lady Margaret Hall

- Code can be executed, so we can use this to check results!



Fixing Based on Error Messages

- e.g. InterCode (Yang et al. 2023)



Code Synthesis from Input/ Output Examples

- It is also possible to “guess” programs from input-output examples
- FlashFill induces programs to fill in Excel sheets (Gulwani 2011)
- Terpret compares many different methods for synthesis (Gaunt et al. 2016)
- Generally work with *domain-specific languages* (DSLs) due to problem difficulty

Code Generation -

Representative Code LMs

Codex

(Chen et al. 2022)

- **Creator:** OpenAI
- **Model:** Originally continued training from GPT-3
- **Data:** lots of data from GitHub
- Powers (powered?) GitHub Copilot

StarCoder 2 (Lozhkov et al.

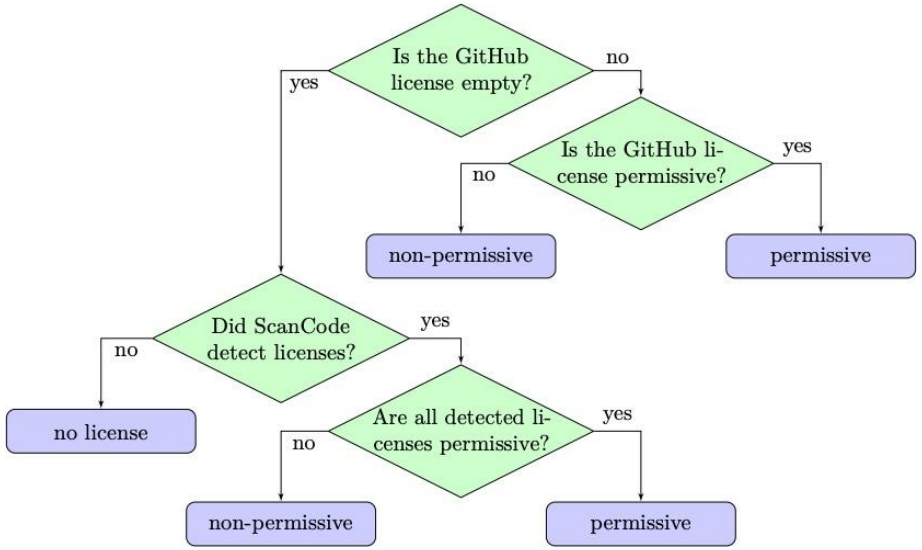
- **Creator:** Big Science (Hugging Face + Service Now)
- **Architecture:** Mostly LLaMa-style, w/ 3B, 7B, 15B variants, reconfigure RoPE for longer context
- **Data:** Trained on code (the stack), GitHub issues, pull requests, jupyter notebooks, Kaggle notebooks, documentation, intermediate representations like LLVM, NL datasets
- **Preproc:** Add metadata tags (e.g. repo name and file name) 50% of the time (to allow it to be used at test). Allow for infilling
- **Training:** Train for 4-5 epochs, 3T+ tokens total

The Stack 2

أكاديمية كاوست
KAUST ACADEMY

Lady Margaret Hall

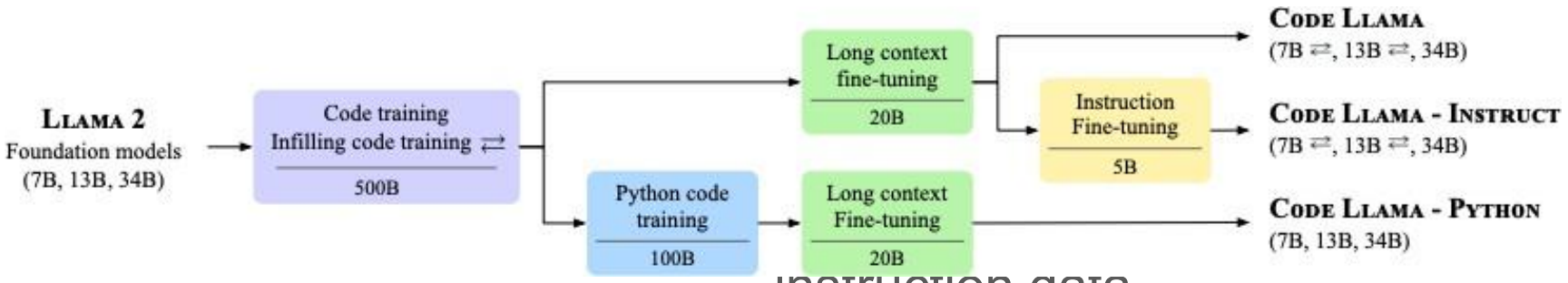
- Code pre-training dataset w/ license considerations



Language	The-stack-v1-dedup Size (GB)	The-stack-v1-dedup Files (M)	The-stack-v2-dedup Size (GB)	The-stack-v2-dedup Files (M)	The-stack-v2-swh-full Size (GB)	The-stack-v2-swh-full Files (M)
Assembly	1.58	0.25	13.02	0.77	7.74	0.70
Batchfile	0.29	0.25	2.11	1.13	1.02	0.99
C	57.43	8.53	202.05	20.78	114.92	19.18
C#	46.29	10.84	239.89	51.23	169.75	48.49
C++	50.89	6.37	353.89	43.18	211.33	42.23
CMake	0.45	0.19	2.58	1.74	2.27	1.70
CSS	22.61	2.99	161.68	23.87	8.00	1.88
Dockerfile	0.572	0.42	1.27	1.90	1.21	1.88
Fortran	0.17	1.84	4.66	0.27	3.61	0.26
Go	25.74	4.73	54.60	9.30	25.83	8.62
Haskell	2.36	0.54	5.11	1.25	4.17	1.23
HTML	146.76	9.53	2,419.87	90.23	99.09	5.23
Java	89.30	20.15	548.00	154.28	199.68	62.27
JavaScript	141.65	21.11	1,115.42	108.87	199.99	66.91
Julia	1.54	0.30	6.12	0.45	1.83	0.43
Lua	3.28	0.56	33.91	2.35	15.22	2.24
Makefile	1.49	0.66	21.30	4.22	5.19	2.78
Markdown	75.25	21.0	281.04	82.78	244.17	81.42
Perl	2.63	0.39	7.82	1.15	5.66	1.06
PHP	66.84	15.90	224.59	46.03	183.70	45.14
PowerShell	1.25	0.27	3.97	0.68	2.46	0.66
Python	64.30	12.96	233.29	56.93	191.61	56.19
R	0.30	0.04	22.39	5.15	19.05	4.29
Ruby	7.14	3.41	31.70	17.79	23.38	17.51
Rust	9.53	1.38	15.60	2.22	12.43	2.19
Scala	4.86	1.36	12.73	4.45	11.30	4.32
Shell	3.38	22.69	19.82	10.68	13.51	10.01
SQL	12.22	0.99	281.45	5.29	35.75	4.52
Swift	0	0	23.76	7.23	22.32	7.16
TeX	5.44	0.55	35.86	3.19	30.01	2.86
TypeScript	28.82	10.64	61.01	23.85	49.14	23.28
Visual Basic	1.49	0.16	16.63	1.06	7.48	0.81
Total	875.85	181.00	6,457.14	784.30	1,922.82	528.44

CodeLLaMA (Roziere et al. 2023)

- `Creator:` Meta
- `Architecture:` Same as LLaMa 2 (7B, 13B, 34B and 70B), but trained on longer input contexts (100k) with longer RoPE



instruction data

DeepSeek Coder

أكاديمية كاوست
KAUST ACADEMY

LMH
Lady Margaret Hall

(Guo et al. 2024)

- **Data:** 87% source, 10% English from Markdown and StackExchange, 3% Chinese
- **Preproc:** Standard preproc, but also include library dependencies
- **Architecture:** LLaMa-like, 1.3B, 6.7B, and 33B, including reconfigured RoPE
- **Training:** Overall 2T tokens

Which to Use?

- All have somewhat similar performance, and are compared in StarCoder paper
- DeepSeekCoder seems to be strong on standard programming tasks
- StarCoder seems to be strong on data science notebooks

Future Directions

Open Problems and Research Directions

- **Multimodal grounding in real-world environments:** How can agents robustly connect language, vision, and action in dynamic, noisy settings?
- **World model integration:** Combining temporal and spatial memory for persistent, context-aware reasoning.
- **Agent collaboration:** Enabling effective communication and coordination in multi-agent systems.
- **Scaling memory-efficient agents:** Designing architectures that scale to long-term, large-scale memory without prohibitive costs.
- **Robust reward models:** Developing reliable reward and feedback mechanisms for open-ended, real-world tasks.
- **Meta-cognition:** Building agents that can introspect, self-monitor, and adapt their own reasoning processes.

Summary

- Multimodal models (CLIP, VisualBERT, FLAVA) integrate perception and language for richer AI understanding.
- Agentic AI enables proactive, goal-driven, and autonomous behavior.
- Open-source frameworks (Auto-GPT, BabyAGI) are accelerating research and applications.
- Ethical, safe, and explainable deployment is essential for real-world impact.
- The future of AI lies in continual learning and intelligent autonomy.

References

[1] Radford, A., Kim, J. W., Hallacy, C., et al. (2021).

CLIP: Learning Transferable Visual Models From Natural Language Supervision.

arXiv:2103.00020.

[2] Li, L. H., Yatskar, M., Yin, D., Hsieh, C. J., Chang, K. W. (2019).

VisualBERT: A Simple and Performant Baseline for Vision and Language.

arXiv:1908.03557.

[3] Singh, A., Goyal, N., Goswami, V., et al. (2021).

FLAVA: A Foundational Language And Vision Alignment Model.

arXiv:2112.04482.

[4] Yao, S., Zhao, J., Yu, D., et al. (2023).

ReAct: Synergizing Reasoning and Acting in Language Models.

arXiv:2210.03629.

References

[5] Nakajima, Y. (2023).

BabyAGI.

https://github.com/yoheinakajima/babyagi

[6] Torantulino (2023).

Auto-GPT.

https://github.com/Torantulino/Auto-GPT

[7] Ahn, M., Brohan, A., Chebotar, Y., et al. (2022).

Do As I Can, Not As I Say: Grounding Language in Robotic Affordances.

arXiv:2204.01691.

[8] OpenAI Blog.

Agent Simulations and Superalignment.

https://openai.com/blog/

Credits

Dr. Prashant Aparajeya

Computer Vision Scientist — Director(AISimply Ltd)

p.aparajeya@aisimply.uk

This project benefited from external collaboration, and we acknowledge their contribution with gratitude.