BENCHCOUNCIL™ AISYS-IQ

STANDARD SPECIFICATION

VERSION 1.0

AUGUST, 2024

International Open Benchmark Council (BenchCouncil)
http://www.benchcouncil.org
benchcouncil@gmail.com

http://www.benchcouncil.org/aisys-iq

Acknowledgements

The BenchCouncil acknowledges the substantial contribution of AISys-IQ proposal principal submitters: Institute of Computing Technology (ICT), Chinese Academy of Sciences(CAS), and QiYuan Lab, and the proposal reviewer BenchCouncil AI committee.

AISys-IQ Proposal Principal Submitter: Institute of Computing Technology, Chinese Academy of Sciences

AISys-IQ Proposal Reviewer: BenchCouncil AI Committee

Contact: benchcouncil@gmail.com (BenchCouncil)

BenchCouncil Membership

(as of August 2024)

Full Members

Associate Members

Document Revision History

Date	Version	Description
20 July 2024	Draft 1.0	Mail ballot version
22 August 2024	Version 1.0	Revision version (proposed standard)

Contents

1 Introduction

BenchCouncil™ AISys-IQ is a standardized benchmarking specification and methodology designed for evaluating the IQ of intelligent systems. This framework comprises two tiers of IQ evaluation: one for intelligent algorithms and the other for intelligent systems. Within both algorithmic and systemic IQ assessments, the focus lies on three categories of intelligence: single-task intelligence, industrial intelligence, and general intelligence. This approach caters to various benchmarking needs, ensuring thoroughness, diversity, and scalability. Single-task intelligence pertains to AI algorithms and systems specialized in executing singular tasks like image classification or object detection. Industrial intelligence refers to AI algorithms and systems tailored for intricate industrial contexts or entire life-cycle applications characterized by long workflows and multiple components, such as search engines or social networks. General intelligence encompasses AI algorithms and systems capable of addressing a broad spectrum of tasks with versatility, like the functionality exhibited by ChatGPT.

AISys-IQ is maintained by Institute of Computing Technology (ICT), Chinese Academy of Sciences(CAS), and QiYuan Lab. This specification defines the terminology and background, AISys-IQ methodology, AISys-IQ benchmarks, system under test, life of running a workload, quality target, scenario and metrics, models and datasets, implementation constraints, measure and report procedure, and reference implementation.

2 Terminology and Background

2.1 Terminology

The terminologies used in this specification are as follows.

AI is short for Artificial intelligence. This specification focuses on single-task AI, industrial AI, and general AI.

IQ refers to the intelligence quotient of intelligent algorithms and systems. Within this specification, AISys-IQ provides methodology, benchmarks, and metrics for the IQ assessment of intelligent algorithms and systems.

Benchmark refers to the process of running a specific program or workload on a specific machine or system and measuring the resulting performance [1].

Benchmark Suite refers to a suite of benchmarks.

Single-task intelligence refers to AI algorithms and systems specialized in executing singular tasks like image classification or object detection.

Industrial intelligence refers to AI algorithms and systems tailored for intricate industrial contexts or entire life-cycle applications characterized by long workflows and multiple components, such as search engines or social networks.

General intelligence refers to AI algorithms and systems capable of addressing a broad spectrum of tasks with versatility, like the functionality exhibited by ChatGPT.

An individual can be defined as an object described by a set of data. A system is a unified entity formed by a group of interacting or interdependent individuals, regardless of whether they are of the same type or different types. Systems can exhibit recursive structures, where a higher-level system is composed of interacting or interdependent lower-level systems.

A population refers to the entire group of individuals or systems that are the subject of study and understanding, whereas a sample represents a smaller subset of individuals or systems from the population. Variables or quantities are attributes of individuals or systems. Parameters are numbers that describe certain attributes of the population, while statistics are numbers that describe certain attributes of a sample. Inference is the process of making conclusions about parameters of a population based on statistics derived from sample data.

A function (denoted as f) is a rule that assigns a unique element from set R (referred to as f(x)) to each element in set D. In this context, the domain (denoted as D) refers to the set of all possible values for which the function is defined. On the other hand, the range of a function (denoted as f(x)) includes all

possible values that f(x) can take as x varies within the domain. An independent variable is symbolically represented and can take any number within the domain.

A model is a simplified version of a system that cannot be extensively analyzed. Models can be physical or mathematical. A mathematical model is a mathematical description of a system, typically represented by functions or equations, aimed at understanding the system and predicting its behavior. Throughout this article, unless explicitly stated otherwise, the terms ”system” and ”model” will be used interchangeably.

Treatment refers to specific conditions or interventions applied to the subjects of study. An observational study involves observing individuals or systems and measuring variables of interest without attempting to influence their responses, while an experiment is designed to deliberately apply treatments to individuals or systems to measure and analyze their responses. In experiments with multiple independent variables, treatments consist of specific combinations of values assigned to these variables. In understanding causal relationships, experiments are crucial as opposed to observational studies. Even though observational studies are based on random samples, they are still insufficient for effectively measuring the impact of one variable on another. Experiments provide us with convincing and conclusive data, making them the only true source of evidence for believing in causal relationships.

Confounding refers to the situation where two independent variables are associated in a way that makes it difficult to distinguish their specific effects on the dependent variable. In other words, the effects of these independent variables become entangled, making it challenging to attribute specific effects to each independent variable. In such cases, the independent variable responsible for this confounding effect is termed a confounding variable. Control entails keeping other independent variables that may influence the response constant, with the primary purpose of a control group being to provide a baseline for the evaluation and comparison of alternative treatments.

Quality refers to the ability of a AI model producing correct results.

Tail Latency refers to the latency values associated with the slowest responses or operations in a system. It represents the delay experienced by a small percentage of requests or tasks, often measured at the upper percentiles of latency distribution, such as the 99th percentile or 90th percentile. Intelligent systems focus on critical execution paths, where end-to-end execution time serves as a vital metric.

Tail Quality refers to the impact of model quality impacted by the tail latency. Under the influence of tail latency, the tail quality of intelligent systems can be constrained by the limitations imposed by long tail latency. To simultaneously address the impacts of latency and algorithmic quality, we utilize tail quality metrics, such as tail quality under 99th percentile tail latency constraints, tail quality under 90th percentile tail latency constraints, and others.

Quality of Service (QoS) Tasks refers to a task whose latency or quality performance meets certain threshold requirements. Here, a QoS task refers to a task with application execution demands that are not system tasks, unfinished tasks, or repetitive tasks.

Goodput refers to the number of QoS tasks completed per unit time. Assuming the system executed M QoS tasks within time period T, the goodput is calculated as M / T.

Yield refers to the ratio of completed QoS tasks to total tasks.

Reference Implementation refers to the official benchmark implementation provided for reference.

System Under Test (SUT) refers to a system that is being tested for correct operation [2].

2.2 Background

As the AI advancement has brought breakthroughs in processing images, video, speech, and audio [3], more and more domains and fields pervasively employ AI techniques to augment their products and services, and hence boost the deployments of massive AI algorithms, systems and architectures. For example, Alibaba proposes a new DUPN network for more effective personalization [4]. Facebook integrates AI into many essential products and services like news feed [5]. Google proposes the TensorFlow [6] system and the tensor processing unit (TPU) [7] to accelerate the service performance. Amazon adopts AI for intelligent product recommendation [8].

Although there is considerable evaluation of intelligent algorithms or systems in terms of model quality and performance, little work has focused on assessing the model's intelligence quotient (IQ). Meanwhile, in the context of AI's rapid evolution, the diversity of intelligent algorithms and systems poses a profound challenge in measuring their "IQ". Unlike human intelligence, which is generally assessed through standardized tests focusing on aspects like logic, reasoning, and problem-solving, AI encompasses a vast array of abilities tailored to different tasks. For example, single-task intelligent algorithms or systems might demonstrate exceptional performance in specialized areas, such as image recognition or language translation, while industry-specific intelligent algorithms or systems are optimized for particular end-to-end applications with long and complex execution path, like unmanned systems or search engine. General-purpose AI, on the other hand, strives to achieve a broader range of competencies but often lacks the fine-tuned proficiency of specialized systems. This variation in objectives and performance criteria makes it difficult to define a single, unified measure of AI intelligence that is as meaningful and standardized as human IQ scores.

To solve these problems, this specification presents BenchCouncil AISys-IQ as the first benchmark specification and methodology for the IQ assessment of intelligent algorithms and systems.

3 AISys-IQ Evaluatology

AISys-IQ follows the Evaluatology proposed by Prof. Zhan et al [9]. This section illustrates the background of evaluatology and AISys-IQ evaluatology.

3.1 Background of Evaluatology

Evaluatology [10] involves assessing an individual or system, referred to as the subject. The subject can be a single entity or a more intricate scenario. Each subject has stakeholders who determine the requirements and define value functions for measurement and outcome comparison. The essence of evaluation is to apply a well-defined evaluation condition (EC) to a well-defined subject, which forms an evaluation model (EM). Subsequently, measurements or tests are conducted to evaluate the subject using the value functions defined by the stakeholders. Among them, the EC contains five basic components [10]: "(1) a set of equivalent definitions of problems or tasks; (2) the set of a collective of equivalent problem or task instances; (3) the algorithms or algorithm-like mechanisms; (4) the implementations of algorithms or instantiations of algorithm-like mechanisms; (5) support systems".

In evaluating a single subject, it is essential to follow the systematic methodology, that is, to utilize a reference evaluation model (REM) that establishes equivalent evaluation conditions (EECs) or least equivalent evaluation conditions (LEECs) to ensure fair and unbiased evaluations across different subjects. The REM serves as a controlled environment within an EM element, where one independent variable is manipulated while the others are held constant. EEC represents a strict form of equivalence that requires every individual component in each level of the two ECs to be equivalent. As a more relaxed form of equivalence, LEEC ensures equivalence at the first two levels of ECs.

When it comes to complex scenarios, the evaluation methodology aims to create a set of EMs that maintain transitivity, from a real-world evaluation system to a perfect evaluation model and a pragmatic evaluation model [10].

Besides, evaluatology, benchmarkology is considered the engineering of evaluation. The essence of a benchmark is "a simplified and sampled EC, specifically a pragmatic EC, which ensures different levels of equivalency, ranging from LEECs to EECs" [10].

3.2 AISys-IQ Evaluatology

The AISys-IQ Evaluatology framework presents a comprehensive approach to assessing the intelligence quotient (IQ) of both intelligent algorithms and intelligent systems. It is structured around the evaluation requirements of stakeholders, focusing on two primary areas: algorithmic IQ assessment and system IQ

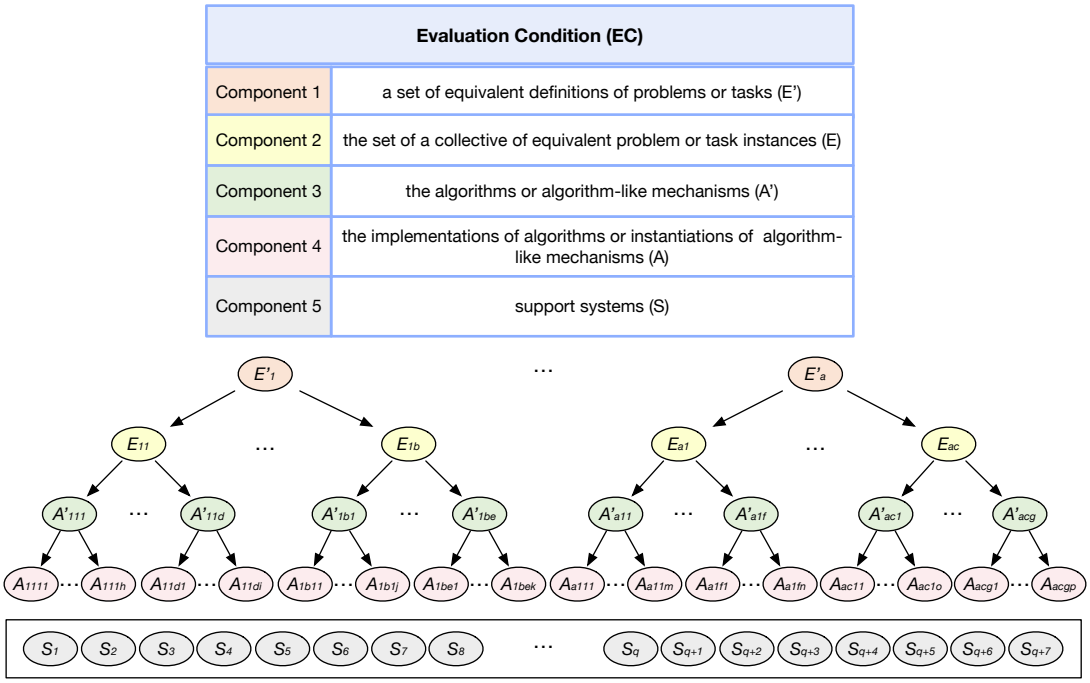

Figure 1: The Hierarchical Definition of an EC. [10]

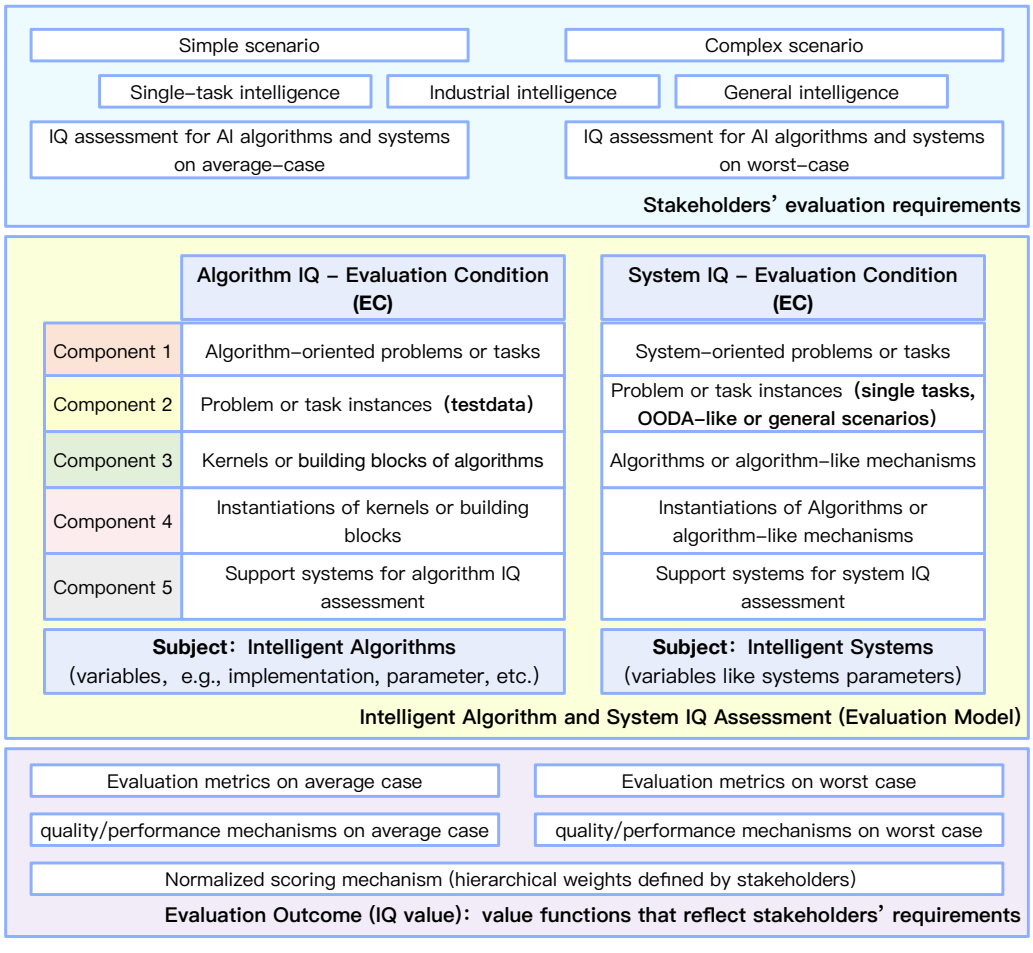

Figure 2: AISys-IQ Evaluatology.

assessment. The framework emphasizes the necessity to accommodate various scenarios, from simple to complex, and to consider both single-task, multi-task with intricate interactions, and general scenarios.

3.2.1 Stakeholders' Evaluation Requirements

At the top of the framework, the stakeholders' evaluation requirements are highlighted, indicating that the entire evaluatology process is driven by the needs and expectations of relevant parties. These requirements guide the development of evaluation conditions tailored to assess intelligent algorithms and systems effectively.

3.2.2 Algorithm IQ Evaluatology

The Algorithm IQ Evaluatology component focuses specifically on evaluating the intelligence of algorithms. This evaluation is structured through several key components:

- Algorithm-oriented Problems or Tasks: Identifying specific problems that the algorithms are designed to solve.
- Problem or Task Instances (Test Data): Utilizing various instances of problems to evaluate algorithm performance, ensuring robustness and adaptability.
- Kernels or Building Blocks: Analyzing the fundamental components that make up the algorithms, which can influence their overall performance.
- Instantiations of Kernels and Building Blocks: Evaluating how different configurations of these components impact algorithm effectiveness.
- Support Systems for Algorithm IQ Assessment: Establishing auxiliary systems that assist in the evaluation process, providing necessary tools and methodologies.

The evaluation metrics for algorithms consider average-case performance, ensuring a holistic view of their effectiveness across various scenarios.

3.2.3 System IQ Evaluatology

On the other hand, the System IQ Evaluatology addresses the intelligence assessment of entire systems. This evaluation follows a parallel structure:

- System-oriented Problems or Tasks: Defining the specific problems the systems are designed to tackle.
- Problem or Task Instances: Similar to algorithms, various instances are used to assess system performance under different conditions—this includes simple tasks and more complex OODA-like (Observe, Orient, Decide, Act) scenarios.
- Algorithms or Algorithm-like Mechanisms: Focusing on the algorithms that drive the systems and their effectiveness in achieving the desired outcomes.
- Support Systems for System IQ Assessment: Implementing systems that aid in the evaluation of intelligent systems, providing necessary frameworks and tools.
- Variables and System Parameters: Considering the different variables that can affect system performance, ensuring a comprehensive assessment.

The evaluation metrics for systems also focus on worst-case scenarios, providing insights into their robustness and reliability under challenging conditions.

3.2.4 Evaluation Outcomes - IQ

At the bottom of the framework, the value functions and evaluation outcomes reflect the stakeholders' requirements. This section describes a multi-layered and multi-level normalization scoring mechanism that aggregates the results from both algorithm and system assessments. The scoring mechanism accounts for various scenarios, including:

Average-case and worst-case evaluations. Single-task and multi-task interactions. General tasks and specific problem instances. The ultimate goal is to derive an IQ assessment result that aligns with the core evaluation needs of the stakeholders.

In summary, the AISys-IQ Evaluatology framework provides a structured and comprehensive methodology for assessing the intelligence of both algorithms and systems. By focusing on specific evaluation conditions and components, it ensures that the assessments are relevant and aligned with stakeholders' needs. This dual approach enhances the understanding of both algorithmic and systemic intelligence, ultimately contributing to the development of more effective and intelligent solutions.

4 Design

The design of AISys-IQ is illustrated from the perspectives of system under test (SUT), life of running a workload, benchmark problem domains, quality target, scenario and metrics, implementation constraints, measure and reporting procedure.

4.1 System under Test

AISys-IQ system contains multiple components for AI algorithm IQ assessment and AI system IQ assessment, which are provided for all AISys-IQ users. According to stakeholders' benchmarking requirements and considerations, AISys-IQ users are feasible to implement an SUT.

4.1.1 AI Algorithm IQ System under Test

Algorithm IQ system under test consists of stakeholders' inputs, algorithm evaluation conditions, a system under test (SUT), monitoring tools, and AI algorithm IQ output components. Fig. 3 shows Algorithm IQ SUT overview.

Stakeholders' inputs is to receive the user-defined settings according to stakeholders' requirements. For example, the intelligence types like algorithms for single tasks, for industrial tasks, or for general tasks, the evaluations under different cases like average-case, worst-case, or hybrid-case, the weight vectors for different settings. This module considers the stakeholders' evaluation requirements and application scenarios, which have an impact on the IQ output.

Algorithm evaluation conditions (EC) define the five levels of EC components.

Component One: define the problems or tasks solved by intelligent algorithms, segmented by application type, mainly encompassing singular problems or tasks such as image classification, natural language processing, speech recognition, industry/scenario-level problems or tasks like search engines, e-commerce, dual-use intelligent scenarios, and general problems or tasks like reasoning, understanding, common sense addressed by large language models. The construction of Evolutionary Computation (EC) varies depending on different application types.

Component Two: define the instantiation of intelligent algorithms for the given problem or task. In the realm of intelligent algorithms, problems are abstract concepts that are difficult to define mathematically. Typically, a specific instance of a problem or task is represented by a dataset with real labels. For instance, ImageNet in the image domain is a notable example of a problem or task instance. It is important to note that the training and validation sets within a dataset are variables and conditions of the evaluation object and are part of the evaluation subject. The validation set within the dataset truly represents the instance of the problem or task. During evaluation, the intelligent algorithm is tested by inputting test data, generating

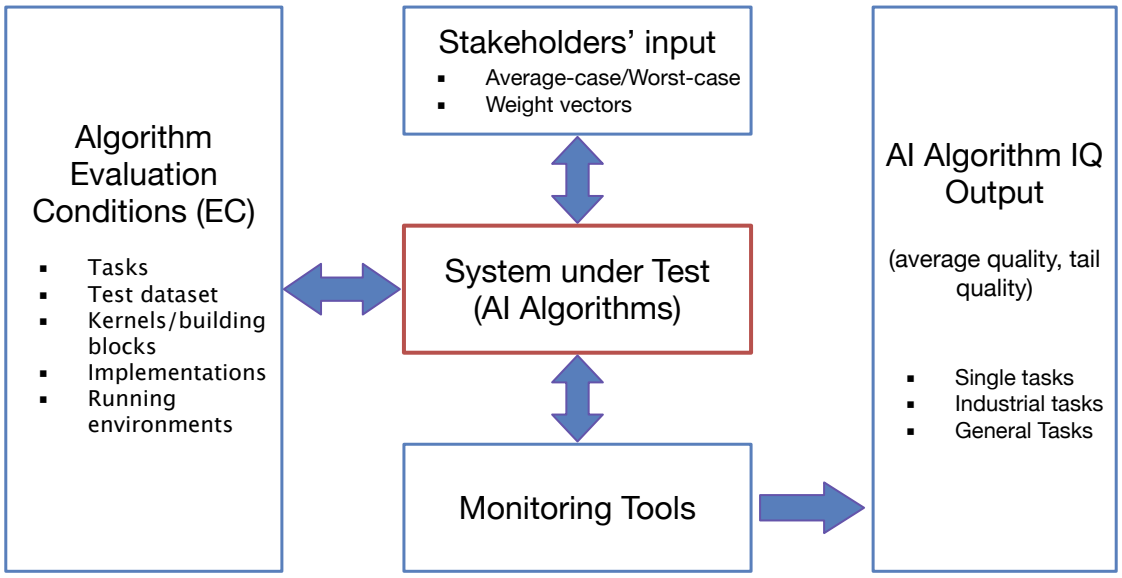

Figure 3: AI Algorithm IQ System under Test.

an output, and comparing it with the real label. This testing process is based on an evaluation model (EM) that includes EC and the subject, assessing the evaluation object through measurement or testing.

Component Three: collection of algorithmic solutions or similar algorithmic mechanisms tailored to the given problem. In intelligent algorithm evaluation, the typical application features and building blocks of abstract intelligent algorithms are abstracted. Primitive intelligent algorithms are formed through various combinations of different building blocks, enabling precision inference based on the combination of building blocks. Building blocks vary across different application types. For instance, in single-task intelligent algorithms, the building block is a sub-code block within the algorithm, like a sub-network in a neural network. Building blocks for industry/scenario intelligence are combinations tailored to single-task intelligent algorithms. The predictive capabilities of the algorithm are based on a core set of similar algorithm mechanisms.

Component Four: implementation of algorithms or instantiation of similar algorithm mechanisms, representing the instantiation of the building blocks from Component Three.

Component Five: support system for the evaluation of intelligent algorithms. The evaluation of intelligent algorithms requires implementation and execution on a computer system, encompassing all systems involved in the process.

System under test (SUT) for AI algorithms defines the subjects to be evaluated. The evaluation of the intelligence quotient (IQ) of intelligent algorithms targets the algorithms themselves. Moreover, intelligent algorithms encompass numerous variables pertinent to evaluation, such as the specific implementation of the algorithm, parameter configurations, and the training and validation datasets required for algorithm model training. The intelligent algorithms themselves, along with associated variables and conditions, collectively form a diverse set of evaluation objects.

Monitoring tools provide a suite of scripts that monitor the running performance and running status. When execution ends, the results are generated, including the model qualities on average-case and worst-case and IQ results.

4.1.2 AI System IQ System under Test

AI System IQ system under test consists of dataset, a system under test (SUT), monitoring tools, and result output components. Fig. 4 presents AI system IQ SUT overview.

Stakeholders' inputs is to receive the user-defined settings according to stakeholders' requirements. For example, the targets of the intelligent systems like for completing single tasks, industrial tasks, or general tasks, the evaluations under different cases like average-case, worst-case, or hybrid-case,

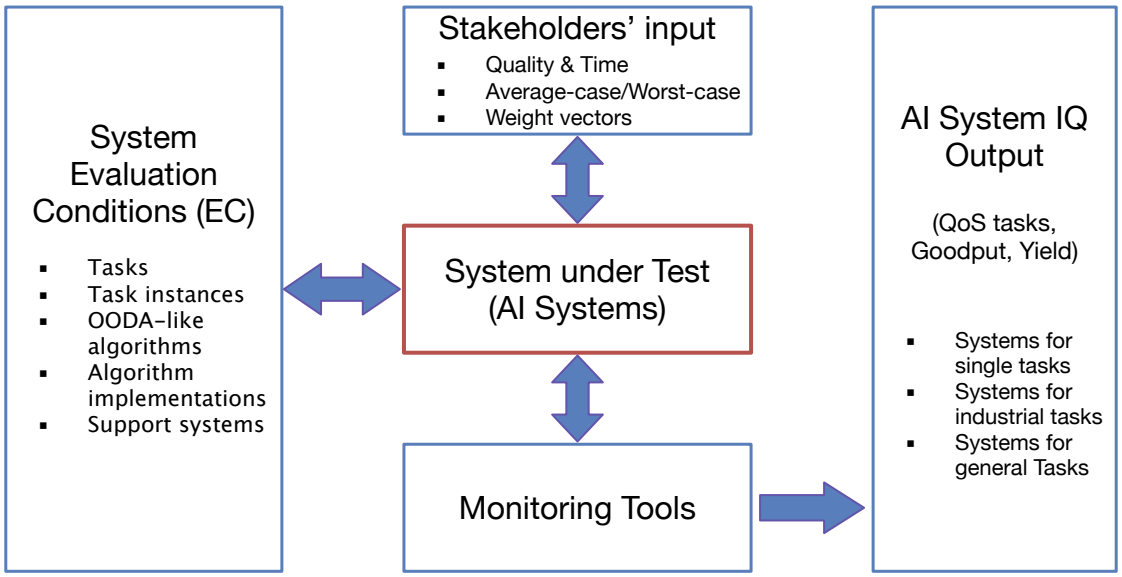

Figure 4: AI System IQ System under Test.

the weight vectors for different settings, and the quality and time requirements according to different application scenarios. This module considers the stakeholders' evaluation requirements and application scenarios, which have an impact on the system IQ output.

System evaluation conditions (EC) define the five levels of EC components.

Component One: define the problems or tasks addressed by intelligent systems, categorized by application type, primarily encompassing singular tasks such as image classification, natural language processing, speech recognition, and industry/scenario-level problems or tasks like search engines, e-commerce, dual-use intelligent scenarios, as well as general problems or tasks like reasoning, understanding, and common sense targeted by large language models. The Evolutionary Computation (EC) construction varies across different application types.

Component Two: define the instantiation of intelligent systems for the given problem or task. Within intelligent systems, problems are abstract concepts that are challenging to precisely define through a singular mathematical approach. Typically, a specific instance of a problem or task is represented by a dataset with real labels.

Component Three: Collection of algorithmic solutions or similar algorithmic mechanisms tailored to the given problem.

Component Four: implementation of algorithms or instantiation of similar algorithmic mechanisms, representing the instantiation of building blocks from Component Three.

Component Five: support system. It furnishes the necessary resources and environment on which the evaluation depends.

System under test (SUT) for AI systems defines the subjects to be evaluated. The evaluation subject of intelligent systems is the systems themselves, encompassing the overall architecture, design and implementation of individual modules, the operational environment, and external factors interacting with the system. The evaluation of intelligent systems extends beyond functional performance to include hardware support, operational conditions, input-output data, system parameter configurations, as well as the training and validation datasets used. These elements, intertwined with the design, implementation, and operation of intelligent systems, form a multidimensional and multivariable evaluation subject.

Monitoring tools provide a suite of scripts that monitor the running performance and running status. When execution ends, the results are generated, including the QoS (quality of service) tasks, goodput, yield, and IQ results.

4.2 Life of Running a Workload

4.2.1 AI Algorithm IQ Evaluation

At startup, the data/query generator send requests to SUT according to the five components of evaluation conditions (EC). Then the SUT receives the input data and processes them. During processing, the data are preprocessed and flowed through one or multiple AI related and non-AI related modules. The generator sends data to be processed according to specified configurations. The configuration designates parameters like data size, concurrency, distribution, user thinking time, and the ratio of different data types, e.g., text items and image items, to depict realistic evaluation scenarios.

The SUT processes the input data and returns the results. During the run, monitoring tools are started in background and sample algorithm-level, system-level, and microarchitecture-level performance every specific seconds. Also, the response time of each data input is tagged and recorded. After the run, an script outputs the running results.

4.2.2 AI System IQ Evaluation

AI system IQ evaluation consist of running scripts including various tasks on different datasets, and monitoring tools.

At startup, the SUT loads samples or batches into memory and runs selected workloads with specific AI tasks. During the run, monitoring tools are started in background and sample system and micro-architecture performance every specific seconds. Also, the running time and model quality to achieve stakeholders' requirements will be recorded. After the run, an script checks whether the model quality and time satisfies the rules and outputs the running performance.

4.3 AISys-IQ Problem Domains

This section illustrates the problem domains of AISys-IQ from the perspectives of AI algorithm and AI systems.

4.3.1 Problem Domains for AI Algorithm Evaluations – Dataset-defined Problems

To achieve intelligence evaluation of any architectural model, it is crucial to represent the model architecture in a suitable manner. ONNX is an open deep learning model interchange standard that provides a universal language for any machine learning framework to describe its models, defining all necessary operations for model inference. Therefore, we select models in the ONNX format or convertible to ONNX format, abstract their architectures as directed acyclic graphs (DAGs), and represent their nodes with information about different operators under the ONNX standard.

Given the need for a substantial amount of data to train the intelligence evaluation model, and as far as we know, such datasets are currently unavailable, we must construct a dataset from scratch. This dataset should include numerous models along with their trained accuracies. Through algorithmic statistical analysis of these model architectures and corresponding dataset and accuracy information, we train the intelligence prediction model to evaluate the intelligence of input model architectures. In this process, we identify building blocks (subgraphs recurring in models), cluster different combinations of building blocks to understand their relationships and impact on model intelligence, and ultimately predict accuracy based on these building block combinations.

To build the aforementioned dataset, training a vast number of models on various datasets is undeniably a challenging task. After conducting research, we discovered a plethora of well-trained models available for reuse in Hugging Face. Additionally, many model repositories' README files document the tasks, datasets, and corresponding metric values associated with the models. Furthermore, a significant portion of model README files contain Model Card information provided by the authors, detailing the Task, Dataset, Metric, and their respective values.

Therefore, we download all ONNX-compatible models stored in the Hugging Face Hub, including repositories such as Timm, Diffusers, Transformers, Sentence-Transformers, and pre-trained models labeled with ONNX tags. These models support a comprehensive and integrated evaluation of model architectures' performance for different tasks and datasets. Building upon this foundation, we design a complete process to annotate their Task, Dataset, Metric, and training information.

Our evaluation program aims to predict the metrics and corresponding values of intelligent models on different tasks and datasets by inputting the architectures of these models. Our dataset comprises directed acyclic graphs (DAGs) representing machine learning model architectures from various domains. Hence, we opt for graph convolutional neural networks as the evaluation model suited for this requirement.

The design of graph convolutional neural networks involves constructing features at the node level, graph level, and establishing connectivity among nodes within the entire graph. We choose operators at the node level as node features and metrics and corresponding values of model architectures represented by DAGs as graph-level features. Common methods for representing connections between nodes in a graph include adjacency matrices and adjacency lists. However, since the DAGs in our dataset encompass varying numbers of nodes from dozens to tens of thousands, adjacency matrices become sparse and challenging to represent large-scale graph structures. Therefore, we utilize the COO format adjacency list to depict the connectivity of the entire graph, utilizing two arrays to record the starting and ending nodes.

In terms of model design, we primarily employ graph convolution and differentiable pooling to accomplish tasks such as node representation and building block extraction. This approach enables us to predict the accuracy of intelligent model architectures composed of combinations of building blocks.

In the realm of public model repositories, numerous isomorphic neural network architectures abound, necessitating the curation of unique architectures within datasets. To ensure the distinctiveness of each architecture, a critical filtering process is essential. Leveraging the Weisfeiler Lehman (WL) graph hash algorithm, we compute the derived Directed Acyclic Graphs (DAGs) to pinpoint heterogeneous architectures. The WL algorithm, known for its ability to produce identical results for isomorphic graphs, provides a robust mechanism to guarantee unique hashes for divergent graphs. In our approach, we designate operator type and attributes embodied in nodes as the focal points for the WL hash algorithm iterations, thereby safeguarding the dataset's architectural heterogeneity concerning hyperparameters and operator types. Following meticulous filtering procedures, we successfully identified 7629 heterogeneous neural network architectures from a pool of 174K real-world models.

During the inception of the initial dataset-defined problems (Younger dataset), a staggering 743.5K publicly accessible models existed, with 341K models being convertible to the ONNX format. By the time of the Younger dataset's inaugural release, we had isolated and processed 174K models. This observation underscores a striking fact: despite the substantial base and rapid growth of deep learning models, the proportion of effective and diverse neural network architectures amounts to less than 1% of the total model count.

4.3.2 Problem Domains for AI System Evaluations

Image Classification

Image classification is to extract different thematic classes within an image, which is a supervised learning problem to define a set of target classes and train a model to recognize.

ResNet-50 [11]: A milestone in Image Recognition [11], marking the ability of AI to identify images beyond humans. It solves the degradation problem, which means in the very deep neural network the gradient will gradually disappear in the process of propagation, leading to poor performance. Due to the idea of ResNet, researchers successfully build a 152-layer deep CNN. This ultra deep model won all the awards in ILSVRC'15.

ImageNet Dataset [12]: A large visual database designed for use in visual object recognition software research. More than 14 million images have been hand-annotated by the project to indicate what objects are pictured and in at least one million of the images, bounding boxes are also provided. The data size is more than 100 GB.

Reference Quality: The reference implementation of ResNet-50 on ImageNet dataset achieves Top-1 accuracy 74.9%.

Natural Language Processing Task

We select the Vicuna-13B-v1.3 as the test model and process it using a custom dialogue dataset. This task is primarily aimed at evaluating the system's performance in generative language modeling tasks. The Vicuna-13B-v1.3 model, known for its advanced capabilities in natural language processing, stands as a cutting-edge choice for our evaluation task. Leveraging its vast parameter size and pre-trained knowledge, this model excels in generating coherent and contextually appropriate responses in dialogue scenarios. Its proficiency extends to a wide range of generative language modeling tasks, showcasing remarkable fluency and contextual understanding.

In our endeavor to evaluate the system's performance, we employ a meticulously curated custom dialogue dataset that reflects the intricacies and nuances of real-world conversational exchanges. By subjecting the Vicuna-13B-v1.3 model to this rich dataset, we aim to gauge its ability to generate responses that not only align with grammatical correctness but also exhibit a deep comprehension of context, coherence, and conversational flow.

Through this evaluation, we seek to uncover the model's strengths and areas for improvement, shedding light on its adaptability to diverse dialogue contexts and its potential for enhancing user experiences in natural language processing applications.

Unmanned Systems and Navigation Tasks

Unmanned systems and navigation tasks is a typical safety-critical and latency-critical application. It requires not only millisecond-level response time from the data perception to decision feedback but also near-100% model quality and human reactions. We construct an navigation workloads for unmanned systems based on our methodology. The benchmark consists of six modules and involves five iterative execution paths. In addition, to support evaluation with different and user-defined configurations, we provide diverse deployment settings using different web servers and protocols and supporting user-defined parameters, as shown in Fig. 5.

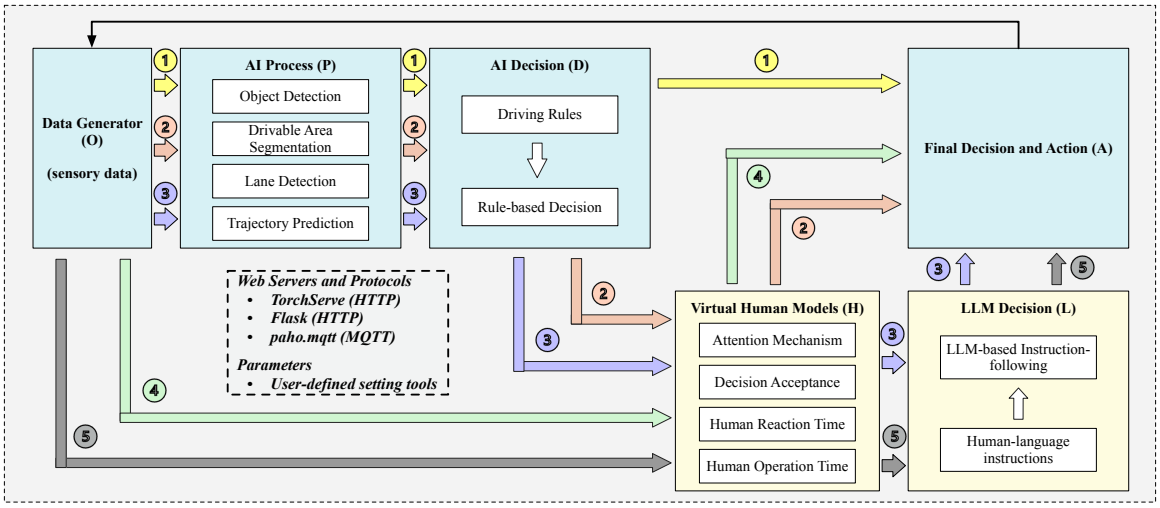

Figure 5: Navigation Workloads for Unmanned Systems.

Six modules. Our benchmark combines the up-to-date techniques like large language model (LLM) and contains six modules. They are data generator, AI process, AI decision, virtual human, LLM decision, and action modules.

The data generator module (*Observe*) simulates the vehicle sensors to obtain and send sensory data. It uses publicly available datasets as input and sends the data frame to the other modules. We implement the

data generator module using Python. We use the publicly available datasets including BDD100K [13] and LMDrive dataset [14].

The AI process module (*Process*) processes the sensory data sent by the data generator and perform object detection, drivable area segmentation, lane detection, and trajectory prediction. We use the state-of-the-art AI models for autonomous driving. Specifically, we use HybridNets [15] for object detection, drivable area segmentation, and lane detection. We use QDTrack [16] for trajectory prediction.

The AI decision module (*Decide*) receives the outputs of AI process module, and generates a rule-based decision according to a series of driving rules.

The virtual human module (*Human*) simulates the behavior of a driver or passenger and interacts with AI models. We introduce attention mechanism and simulate the human's decision and decision time. Among them, the human's decision is generated by a decision acceptance model which accept or reject the AI model's decision. In emergency, it may take the sensory data as input and make a decision directly without an AI model's decision. The human reaction time and human operation time refer to a learned probabilistic perceptual model [17] which characterize the human reaction time after observing a visual target.

The LLM decision module (*LLM*) uses the large language model within LMDrive [14] to receive human-language instructions. The module would follow the human-language instructions or reject the instructions according to its prediction result.

The action module (*Action*) receives the final decision, simulates the action, and outputs the statistical results.

Five execution paths. According to the different interaction modes between human and AI, and the potential values of LLM for autonomous driving, we integrate five representative execution paths in HiTL Auto-driving benchmark. The five execution paths are as follows. Note that for clarity, we use O, P, D, H, L, and A to represent the data generation, AI process, AI decision, virtual human, LLM decision, and action modules, respectively.

(1) OPDA path (172 in Fig. 5) indicates a regular execution path of autonomous driving, which completely adopts AI models to receive and process the sensory data, and further makes a decision and performs action accordingly.

(2) OPDHA path (173 in Fig. 5) incurs a human intervention built upon the regular OPDA path. The AI decision is reviewed and adjusted by human to avoid potential risks. The human decision is the final decision to be performed.

(3) OPDHLA path (174 in Fig. 5) exploits LLM for better explainability and natural interaction with humans [14] based on the OPDHA path. In this condition, the human decision are delivered to LLM in the form of human-language instructions. Then the LLM takes the instructions and sensory data as input and makes the final decision.

(4) OHA path (175 in Fig. 5) indicates an emergency situation in which human makes the decision directly according to his/her visual data without the output and decision of AI modules. For example, if the driving environment changes suddenly like a rushing out pedestrian, the human will perform emergency treatment other than waiting for the decision of AI models.

(5) OHLA path (176 in Fig. 5) indicates a similar situation with OHA path. The difference is that the human says the language instructions and the LLM comprehends the instructions and performs actions. This is possible since the human may be not able to operate quickly, for example, when sitting in the back seat of the vehicle.

Large Language Models - TinyLlama-1.1B-Chat

TinyLlama-1.1B-Chat is a lightweight language model developed by the research team at the Singapore University of Technology and Design (SUTD). With 1.1 billion parameters, it falls within the realm of smaller-scale models compared to current large language models. However, it showcases capabilities on par with larger models in terms of performance. We have selected InfiniLM as the execution framework, proposed by the YuanYuan Lab, as it is designed to adapt to multiple domestic platforms such as FeiTeng.

In the realm of artificial intelligence, the evaluation of general intelligence systems holds paramount

significance. These systems are designed to exhibit a broad spectrum of cognitive abilities, enabling them to adapt to diverse tasks, learn from new experiences, and make decisions in complex environments. By assessing the performance of such systems, we gain insights into their proficiency across a range of domains, facilitating advancements in AI research and applications.

TinyLlama-1.1B-Chat, a remarkable creation by the research team at the Singapore University of Technology and Design (SUTD), stands out as a noteworthy addition to the landscape of language models. Despite its relatively modest size of 1.1 billion parameters compared to larger models, TinyLlama-1.1B-Chat demonstrates a remarkable capacity to understand and generate coherent text. Its efficiency in handling various language tasks, including dialogue generation, sentiment analysis, and language understanding, showcases the model's adaptability and effectiveness in real-world applications.

Moreover, the utilization of InfiniLM as the execution framework for TinyLlama-1.1B-Chat underscores the importance of platform compatibility and efficiency in deploying AI models on diverse hardware architectures. With its ability to cater to domestic platforms like FeiTeng, InfiniLM enhances the accessibility and usability of models like TinyLlama-1.1B-Chat, paving the way for broader adoption and integration of advanced AI technologies in various settings.

4.4 Scenario and Metrics

AISys-IQ specifies different scenarios metrics for three benchmark levels (i.e., single-level, industrial-level, and general-level AI algorithms and systems). AISys-IQ focuses on realistic scenarios and a series of metrics covering model quality, performance, and quality of service on average and/or worst-case evaluations, which are major stakeholders' concerns.

Server. This scenario represents online server applications that simulate concurrent users with different query configurations. The server scenario models an execution path containing either one or more AI components, and measures the end-to-end performance containing AI related and non-AI related modules and critical path. A data/query generator is provided to simulate concurrent users and send query requests to SUT based on a specific configuration. Note that the query item provides diverse data types to reflect different requirements of applications. For this scenario, both the latency, tail latency (user perceived metric), throughput (service/producer concerned metric), model quality, number of quality of service tasks, goodput, yield are important metrics. In addition, the quality deviation should satisfy the stakeholders' input settings, which are called quality-ensured [1]. So the metrics of server scenario are quality-ensured response latency, tail latency, latency-bounded throughput, number of quality of service tasks, goodput, and yield.

Offline. This scenario represents batch processing mode. In this scenario, offline training and offline inference are included, i.e., offline training and offline inference for single-level, industrial-level, and general-level workloads. The metrics are tail latency, tail quality, No. of QoS tasks, goodput, and yield.

4.5 Implementation Constraints

AISys-IQ implementation should satisfy the following constraints.

AISys-IQ implementation should follow the define of five components of EC for both algorithm and system IQ evaluations, to assure consistent and comparable evaluations.

AISys-IQ implementation disallows inconsistent settings for comparisons such as the input data and the model implementations.

4.6 Measure and Reporting Procedure

AISys-IQ provides running scripts, source code, dataset, and monitoring tools for each benchmark. In addition, AISys-IQ also provides user manuals for deployments and execution. After downloading the AISys-IQ, users can directly run the benchmarks in their own environments and submit results to BenchCouncil AI committee.

[1] Quality-ensured means the quality (e.g., accuracy, BLEU) deviation with target quality is within user-acceptable range.

AISys-IQ result submission should contain the following information: SUT description, including the hardware and software configurations, benchmarking scores, e.g., achieved quality, latency, benchmark configurations, including hyper-parameters like batch size, learning rate.

After the submission, BenchCouncil AI community will check the validity, repeatability, and authenticity of the results. If the results are abidance by the rules, the submission will be reported online.

References

[1] R. H. Saavedra and A. J. Smith, "Analysis of benchmark characteristics and benchmark performance prediction," *ACM Transactions on Computer Systems (TOCS)*, vol. 14, no. 4, pp. 344–384, 1996.

[2] https://en.wikipedia.org/wiki/System_under_test.

[3] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[4] Y. Ni, D. Ou, S. Liu, X. Li, W. Ou, A. Zeng, and L. Si, "Perceive your users in depth: Learning universal user representations from multiple e-commerce tasks," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 596–605, ACM, 2018.

[5] K. Hazelwood, S. Bird, D. Brooks, S. Chintala, U. Diril, D. Dzhulgakov, M. Fawzy, B. Jia, Y. Jia, A. Kalro, J. Law, K. Lee, J. Lu, P. Noordhuis, M. Smelyanskiy, L. Xiong, and X. Wang, "Applied machine learning at facebook: A datacenter infrastructure perspective," in *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pp. 620–629, IEEE, 2018.

[6] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mane, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viegas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.

[7] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers, R. Boyle, P.-l. Cantin, C. Chao, C. Clark, J. Coriell, M. Daley, M. Dau, J. Dean, B. Gelb, T. V. Ghaemmaghami, R. Gottipati, W. Gulland, R. Hagmann, C. Ho, D. Hogberg, J. Hu, R. Hundt, D. Hurt, J. Ibarz, A. Jaffey, A. Jaworski, A. Kaplan, H. Khaitan, D. Killebrew, A. Koch, N. Kumar, S. Lacy, J. Laudon, J. Law, D. Le, C. Leary, Z. Liu, K. Lucke, A. Lundin, G. Mackean, A. Maggiore, M. Mahony, K. Miller, R. Nagarajan, R. Narayanaswami, R. Ni, K. Nix, T. Norrie, M. Omernick, N. Penukonda, A. Phelps, J. Ross, M. Ross, A. Salek, E. Samadiani, C. Severn, G. Sizikov, M. Snelham, J. Souter, D. Steinberg, A. Swing, M. Tan, G. Thorson, B. Tian, H. Toma, E. Tuttle, V. Vasudevan, R. Walter, W. Wang, E. Wilcox, and D. H. Yoon, "In-datacenter performance analysis of a tensor processing unit," in *Proceedings of the 44th Annual International Symposium on Computer Architecture*, pp. 1–12, ACM, 2017.

[8] B. Smith and G. Linden, "Two decades of recommender systems at amazon. com," *Ieee internet computing*, vol. 21, no. 3, pp. 12–18, 2017.

[9] J. Zhan, L. Wang, W. Gao, and R. Ren, "Benchcouncil's view on benchmarking ai and other emerging workloads," *arXiv preprint arXiv:1912.00572*, 2019.

[10] J. Zhan, L. Wang, W. Gao, H. Li, C. Wang, Y. Huang, Y. Li, Z. Yang, G. Kang, C. Luo, *et al.*, "Evaluatology: The science and engineering of evaluation," *BenchCouncil Transactions on Benchmarks, Standards and Evaluations*, vol. 4, no. 1, p. 100162, 2024.

[11] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[12] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and F.-F. Li, "Imagenet: A large-scale hierarchical image database," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pp. 248–255, IEEE, 2009.

[13] F. Yu, H. Chen, X. Wang, W. Xian, Y. Chen, F. Liu, V. Madhavan, and T. Darrell, "Bdd100k: A diverse driving dataset for heterogeneous multitask learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 2636–2645, 2020.

[14] H. Shao, Y. Hu, L. Wang, S. L. Waslander, Y. Liu, and H. Li, "Lmdrive: Closed-loop end-to-end driving with large language models," *arXiv preprint arXiv:2312.07488*, 2023.

[15] D. Vu, B. Ngo, and H. Phan, "Hybridnets: End-to-end perception network," *arXiv preprint arXiv:2203.09035*, 2022.

[16] J. Pang, L. Qiu, X. Li, H. Chen, Q. Li, T. Darrell, and F. Yu, "Quasi-dense similarity learning for multiple object tracking," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 164–173, 2021.

[17] B. Duinkharjav, P. Chakravarthula, R. Brown, A. Patney, and Q. Sun, "Image features influence reaction time: A learned probabilistic perceptual model for saccade latency," *ACM Transactions on Graphics (TOG)*, vol. 41, no. 4, pp. 1–15, 2022.