VT INTERNATIONAL PRESS
VTIPJMRR | Journal of Multidisciplinary Research and Review

(Review Article)

Edge AI for Personal Desktop Assistants: A Review of Innovative Local Automation and Intelligent Interaction

Vasudha Patil[1], Vaibhav Holkar[2], Gore Pranjali[3], Savekar Shravani[4], Patil Yash[5]

[1]*Assistant Professor & HOD, Artificial Intelligence & Data Science, Shri Chhatrapati Shivaji Maharaj College of Engineering, Nepti, Ahilyanagar, India*

[2,3,4,5]*BE Scholar, Department of Artificial Intelligence & Data Science, Shri Chhatrapati Shivaji Maharaj College of Engineering, Nepti, Ahilyanagar, India*

Corresponding Author: Holkarvaibhav19@gmail.com

Received: 01.09.2026 | Accepted: 20.09.2026 | Published: 28.09.2026

ABSTRACT

Advances in Artificial Intelligence (AI) have enabled new forms of human–computer interaction through intelligent personal assistants. Conventional desktop assistants are constrained by their dependence on the cloud, data-privacy concerns, limited system-level automation and a lack of multimodal interaction. Edge AI addresses these problems by running inference on the user's own machine, which reduces response latency and keeps personal data on the device. This article presents a structured review of the techniques and tools that enable edge-based personal desktop assistants. Following a PRISMA-guided search of IEEE Xplore, the ACM Digital Library, ScienceDirect, arXiv and Google Scholar, 45 studies and software resources were included and organised into nine themes: intelligent personal assistants, edge AI and on-device inference, on-device automatic speech recognition, intent classification, small and quantised local large language models, retrieval-augmented generation over local documents, operating-system-level desktop agents, and vision-based authentication and gesture interaction. The reviewed approaches are compared qualitatively in terms of functionality, processing location, latency and resource usage, data privacy, reliability and ease of integration. Building on this synthesis, a conceptual reference architecture and formulation is presented for a multimodal desktop assistant, covering intent routing, local retrieval, a sensitivity-aware execution policy, a latency model and a multi-objective trade-off function. The discussion highlights the trade-offs between local inference and cloud-based reasoning, the cost of running several AI models on one machine, and the point at which privacy breaks down when local retrieval is paired with a cloud-hosted language model. Finally, research gaps and future directions are identified in secure command execution, private document retrieval, multimodal interaction and efficient on-device processing.

Keywords: *Edge AI, Personal desktop assistant, Local automation, Intelligent interaction, Speech recognition, Large language models (LLMs), Retrieval-augmented generation (RAG), Computer vision, Multimodal AI, Data privacy*

I. INTRODUCTION

Artificial Intelligence (AI) has changed the way people interact with computers by improving how machines perceive, interpret and execute commands with less manual input. Personal desktop assistants are an important application of AI that allow users to operate a computer through voice and text commands, from launching applications to running multi-step processes and retrieving information stored on the system [1, 2]. Commercial voice assistants made this style of interaction familiar, but they are designed mainly for general queries and offer limited control over the desktop operating system, limited access to local documents and little use of visual cues for security [1, 3]. Assistants that depend on cloud services may also suffer from network delays and expose confidential information to third parties [3].

Edge Artificial Intelligence (Edge AI) moves AI processing close to where data is generated, so that inference runs on the end-user device instead of a remote server [4–6]. On a personal computer, this means that speech recognition, language understanding and vision models can run locally, which improves autonomy and reduces the exposure of sensitive data over the network [7]. However, running several AI models at the same time can place a heavy load on the processor and memory of an ordinary personal computer [7, 8].

Large Language Models (LLMs) and Retrieval-Augmented Generation (RAG) have made natural-language question answering over private collections practical [9, 10]. Combined with document parsing and vector search, a desktop assistant can answer questions from the user's notes, reports and research papers. The degree of privacy, however, depends on which stages of this pipeline run locally and whether an external language model is used for complex queries [11]. Computer vision adds further capabilities: face recognition can be used for authentication, presence detection can identify when an enrolled user is in front of the screen, and hand tracking enables gesture-based commands [12–14].

 © 2026 VT International Press. All rights reserved. E-ISSN: 3139-7026

Although each of these technologies has been studied separately, there is no consolidated review of how they can be combined in a privacy-preserving, edge-based desktop assistant. The main contributions of this review are as follows:

- A structured, PRISMA-guided review [15] of 45 studies and software resources on edge-based personal desktop assistants, organised into nine technical themes.
- A qualitative comparison of representative assistants and frameworks against the criteria of functionality, processing location, latency and resource usage, privacy, reliability and ease of integration (Table II).
- A conceptual reference architecture and formulation that links the reviewed techniques into one pipeline, together with an evaluation protocol for future experimental work.
- A discussion of the main trade-offs and a set of research gaps and future directions.

The remainder of this paper is organised as follows. Section II describes the review methodology. Section III reviews the literature by theme. Section IV presents the comparative analysis. Section V describes the conceptual reference architecture and formulation. Section VI discusses the findings and the limitations of the review. Section VII outlines the research gaps, Section VIII the future scope, and Section IX concludes the paper.

II. REVIEW METHODOLOGY

The review followed the reporting principles of the PRISMA 2020 statement [15], adapted to a narrative-synthesis review in a fast-moving engineering field.

A. Search Strategy

Five sources were searched: IEEE Xplore, the ACM Digital Library, ScienceDirect, arXiv and Google Scholar. Search strings combined one term from each of three groups with Boolean operators: (i) “personal assistant” OR “voice assistant” OR “desktop assistant” OR “desktop agent” OR “GUI agent”; (ii) “edge AI” OR “on-device” OR “local inference” OR “quantization” OR “offline”; and (iii) “speech recognition” OR “intent classification” OR “large language model” OR “retrieval-augmented generation” OR “face recognition” OR “gesture recognition” OR “privacy”. The time window was January 2016 to June 2025; earlier seminal works were added when they are still the standard reference for a technique. Backward and forward citation searching was applied to the included studies, and the official documentation of widely used open-source tools (e.g., Vosk, faster-whisper, llama.cpp and Ollama) was consulted where no peer-reviewed description exists.

B. Inclusion and Exclusion Criteria

Studies were included if they (i) described an assistant, agent, model or tool applicable to personal-computer interaction; (ii) addressed on-device or privacy-preserving processing, or provided a component that can run locally; and (iii) were published in English in a peer-reviewed venue, as a widely cited preprint, or as official documentation of an open-source tool. Studies were excluded if they targeted only cloud data-centre deployment, smart-speaker hardware with no desktop relevance, or vehicles and robots; if they were duplicates or earlier versions of an included work; or if the full text was not available.

C. Study Selection

Titles and abstracts were screened first, followed by full-text assessment against the criteria above. Data extracted from each included study comprised the task addressed, the processing location, reported latency or resource figures where available, privacy and security properties, and stated limitations. These attributes form the comparison criteria used in Section IV. Table I summarises the 45 included studies and resources by theme and source type.

TABLE I. DISTRIBUTION OF THE INCLUDED STUDIES AND RESOURCES BY THEME AND SOURCE TYPE

Sec.	Theme	Journal	Conference	Preprint	Software	Total	References
III-A	Intelligent personal assistants and desktop automation	2	1	0	0	3	[1, 2, 16]
III-B	Edge AI and on-device inference	4	1	1	0	6	[4, 5, 6, 7, 17, 18]
III-C	On-device automatic speech recognition	0	3	0	2	5	[19, 20, 21, 22, 23]
III-D	Intent classification and spoken language understanding	0	0	2	0	2	[24, 25]
III-E	Small and quantised local LLMs and runtimes	0	3	5	2	10	[8, 27, 28, 29, 30, 31, 32, 33, 34, 35]
III-F	Retrieval-augmented generation over local documents	1	3	1	0	5	[9, 10, 26, 36, 37]
III-G	Operating-system-level desktop agents	0	3	2	0	5	[38, 39, 40, 41, 42]
III-H	Face authentication and gesture recognition	1	2	3	0	6	[12, 13, 14, 43, 44, 45]
III-I	Security and privacy of voice and LLM assistants	1	2	0	0	3	[3, 11, 46]
	Total	**9**	**18**	**14**	**4**	**45**	

© 2026 VT International Press. All rights reserved.

D. Synthesis

Because the included works differ widely in task, hardware and evaluation protocol, a quantitative meta-analysis was not possible. A thematic narrative synthesis was therefore used, organised into the nine themes of Section III, followed by a qualitative cross-comparison (Section IV).

III. LITERATURE REVIEW

This section reviews the included studies by theme, from assistant-level systems to the individual components that an edge-based desktop assistant requires.

A. Intelligent Personal Assistants and Desktop Automation

Silva *et al.* [2] conducted a systematic literature review of intelligent personal assistants, summarising their applications, architectures and capabilities and analysing usability, security and privacy. The study shows that natural-language interfaces offer intuitive control of computer operations, but that usability, security and privacy strongly affect user adoption. Hoy [1] describes the mainstream commercial assistants (Siri, Alexa and Cortana) and notes that they process voice queries on remote servers and are oriented towards information retrieval and simple tasks rather than full desktop control.

Dobriyal *et al.* [16] designed an AI-driven personal desktop assistant in Python that accepts voice input, executes commands such as launching applications and web searches, and replies through text-to-speech. The authors note that its functional scope is limited and point to more advanced AI and wider connectivity as future directions. Together, these works show the potential of voice-commanded desktop automation, but also that the commands supported by existing assistants are mostly simple, while operations such as document editing still need extensive manual interaction. A more capable assistant therefore needs multimodal interaction and deeper control over the operating system.

B. Edge AI and On-Device Inference

Edge computing moves computation from the cloud towards the devices that produce data, reducing latency, bandwidth use and privacy exposure [4]. Zhou *et al.* [5] and Deng *et al.* [6] describe edge intelligence as the combination of edge computing and AI, and classify deployments by how much training and inference take place on the device. Wang *et al.* [7] provide a recent survey of on-device AI models, covering model compression, hardware acceleration and system-level optimisation. Model-compression techniques such as pruning and quantisation [17] and efficient architectures such as MobileNets [18] make it possible to run speech, language and vision models within the memory and power limits of personal devices. The main constraint highlighted across these works is that several models running at once compete for the same CPU, GPU and memory, so a desktop assistant must balance speed, accuracy and resource use.

C. On-Device Automatic Speech Recognition

Automatic Speech Recognition (ASR) is the entry point of a voice-driven assistant. Self-supervised models such as wav2vec 2.0 [19] and convolution-augmented Transformers such as Conformer [20] substantially improved recognition accuracy. Whisper [21], trained on about 680,000 hours of weakly supervised multilingual audio, is robust to accents and background noise and is released in several model sizes, so that a smaller model can be chosen when computing resources are limited. Faster-whisper [22] re-implements Whisper with the CTranslate2 inference engine and supports 8-bit quantisation, which lowers latency and memory use on CPUs. Vosk [23] provides lightweight offline recognition models that run in real time on modest CPUs, at the cost of lower accuracy than large Transformer models. These tools make fully local speech input feasible on ordinary personal computers.

D. Intent Classification and Spoken Language Understanding

After transcription, the assistant must determine what the user wants. The Snips Voice Platform [24] showed that wake-word detection, ASR and natural-language understanding can run entirely on embedded hardware, following a private-by-design approach in which no audio leaves the device. Its intent parser relies on lightweight models trained per domain. Chen *et al.* [25] showed that a fine-tuned BERT model can jointly classify intents and fill slots with high accuracy, while sentence-embedding models such as Sentence-BERT [26] allow similarity-based intent matching with very little training data. A practical edge assistant can therefore combine a rule-based layer for frequent, unambiguous commands with a small learned classifier or a local LLM for ambiguous requests.

E. Small and Quantised Local Large Language Models

Open-weight model families such as LLaMA [27], Mistral 7B [28] and Phi-3 [29] have made capable language models small enough to run on personal computers; the Phi-3 report explicitly demonstrates a model that runs locally on a phone. Post-training quantisation methods such as GPTQ [30] and AWQ [31] reduce weights to 3–4 bits with limited loss of accuracy, and QLoRA [32] enables memory-efficient fine-tuning of quantised models. Runtimes such as llama.cpp [33] and Ollama [34] package these models for CPU and consumer-GPU inference and expose a local API that an assistant can call. Xu *et al.* [8] review on-device language models and identify memory capacity, inference

© 2026 VT International Press. All rights reserved.

speed and energy use as the main barriers. For a desktop assistant, a local LLM removes the need to send queries to a cloud provider, but smaller models reason less reliably than large cloud-hosted models such as Gemini [35].

F. Retrieval-Augmented Generation over Local Documents

Lewis *et al.* [9] introduced retrieval-augmented generation, in which a neural retriever first obtains relevant passages and a language model then generates an answer conditioned on them. Dense Passage Retrieval [36] and Sentence-BERT [26] provide the embedding models used for retrieval, and FAISS [37] supports fast similarity search over large vector collections. Gao *et al.* [10] survey RAG for LLMs and identify chunking, embedding quality and retrieval ranking as the main factors that limit answer quality. For a desktop assistant, document embeddings can be stored in a local FAISS index so that searching remains on the device. However, if the generation step uses a cloud-hosted LLM, the retrieved passages, which may contain private information, still leave the machine; local retrieval alone does not make the pipeline private.

G. Operating-System-Level Desktop Agents

LLM-based agents extend assistants from single commands to multi-step tasks. ReAct [38] interleaves reasoning steps with tool calls, a pattern now widely used in agent design. Mind2Web [39] and OSWorld [40] provide benchmarks for web and full operating-system tasks respectively. In OSWorld, which runs agents in real Ubuntu, Windows and macOS environments, humans completed over 72.36% of tasks whereas the best agent completed only 12.24%, mainly because of failures in GUI grounding and operational knowledge [40]. UFO [41] is a UI-focused agent for Windows that uses a multimodal LLM to observe application screenshots and control interface elements across applications. Li *et al.* [42] survey personal LLM agents and identify capability, efficiency and security as the three main challenges. Most current OS agents rely on large cloud-hosted models and send screenshots off the device, which conflicts with the privacy goals of edge assistants.

H. Face Authentication and Gesture Recognition

Deep face-recognition models such as FaceNet [12] and ArcFace [43] map a face image to a compact embedding, so that identity can be verified by comparing embeddings against an enrolled template. BlazeFace [44] performs face detection in under a millisecond on mobile GPUs, making continuous presence detection inexpensive. Face authentication is, however, vulnerable to presentation attacks using photos, videos or masks; Yu *et al.* [45] review deep-learning anti-spoofing (liveness-detection) methods that address this. For gesture input, the MediaPipe framework [13] and MediaPipe Hands [14] provide real-time on-device hand tracking with 21 landmarks per hand, which can be classified into gesture commands. All of these components can run locally, so biometric data need not leave the user's machine.

I. Security and Privacy of Voice and LLM Assistants

Cheng and Roedig [3] survey security and privacy threats to personal voice assistants, including unauthorised voice commands, voice spoofing and leakage of recorded speech. DolphinAttack [46] showed that inaudible ultrasonic commands can control voice assistants without the user noticing. With LLM-based assistants, Greshake *et al.* [11] demonstrated indirect prompt injection, in which instructions hidden in retrieved documents or web pages manipulate the model. These findings show that an assistant able to execute desktop commands needs an explicit execution policy, user authentication for sensitive actions, and treatment of retrieved content as untrusted data.

IV. COMPARATIVE ANALYSIS

Table II compares representative assistants, frameworks and component technologies against the criteria stated in the abstract. The assessment is qualitative and is based on the properties reported in the cited works; it is not the result of benchmarking by the authors.

TABLE II. QUALITATIVE COMPARISON OF DESKTOP-ASSISTANT APPROACHES AND COMPONENT TECHNOLOGIES

Approach [Ref.]	Main functionality	Processing location	Latency / resource use	Privacy	Reliability / key limitation	Ease of integration
Commercial cloud voice assistants (Siri, Alexa, Cortana) [1, 3]	General Q&A, reminders, simple tasks; limited desktop OS control	Cloud	Network-dependent; low local load	Low–medium: speech and queries processed by provider	Mature; exposed to voice spoofing and inaudible commands [46]	Low: closed ecosystems, little OS-level access
Python script-based desktop assistant [16]	Voice launch of applications, web search, text-to-speech replies	Local execution of commands	Low	Depends on speech/search services used	Small fixed command set; limited functionality	High: standard Python libraries
Snips Voice Platform [24]	Wake word, ASR and NLU for predefined intents	Fully local (embedded)	Low; designed for embedded hardware	High: private by design	Limited to trained domains	Medium: per-domain training needed
Open ASR: Whisper / faster-whisper, Vosk [21–23]	Speech-to-text, multilingual	Local	Whisper: moderate–high; faster-whisper	High	Whisper robust to noise and accents;	High: Python APIs

© 2026 VT International Press. All rights reserved.

			lower with 8-bit; Vosk: light, real-time on CPU		small models trade accuracy for speed	
Quantised local LLMs via llama.cpp / Ollama (LLaMA, Mistral, Phi-3) [27–29, 33, 34]	Intent parsing, dialogue, summarisation, answer generation	Local	Several GB of RAM; speed depends on CPU/GPU	High	Weaker multi-step reasoning than large cloud models; hallucination	High: local REST API
Local RAG: SBERT/DPR + FAISS + LLM [9, 26, 36, 37]	Question answering over personal documents	Retrieval local; generation local or cloud	One-time indexing cost; fast search	High with local LLM; reduced with cloud LLM	Quality depends on chunking and retrieval [10]	Medium–high
LLM-based OS/GUI agents (UFO; agents on OSWorld) [40, 41]	Multi-step GUI and cross-application tasks	Mostly cloud multimodal LLMs	High per-step latency; screenshots uploaded	Low–medium: screen content leaves device	Best agent 12.24% vs. human 72.36% task success on OSWorld	Medium: research prototypes
Vision: MediaPipe, BlazeFace, FaceNet/ArcFace + anti-spoofing [12, 13, 43–45]	Presence detection, face authentication, hand gestures	Local	Real-time on CPU / mobile GPU	High: biometric templates stay local	Sensitive to lighting and pose; needs liveness detection	High: Python bindings
Reference architecture of this review (Fig. 1)	ASR, intent routing, automation, RAG, vision security	Hybrid: local ASR, retrieval, vision; cloud LLM API [35]	Not measured (conceptual)	Medium: retrieved passages reach the cloud LLM	To be validated (Section V-H)	High: Python stack

Three findings emerge from this comparison. First, no single existing approach combines broad desktop automation, local processing and strong security: cloud assistants and LLM-based OS agents offer the widest functionality but send user data off the device, whereas fully local systems such as Snips are private but limited to predefined domains. Second, all of the component technologies needed for a local assistant (ASR, intent classification, quantised LLMs, vector retrieval, face and gesture recognition) are now available as mature, locally executable open-source tools with Python interfaces, so the main challenge is integration rather than the availability of components. Third, reliability remains the weakest point for complex tasks: on OSWorld, the best agent succeeded in only 12.24% of tasks compared with over 72.36% for humans [40], which suggests that safe desktop assistants should combine deterministic automation for routine commands with LLM reasoning only where it is necessary.

V. CONCEPTUAL REFERENCE ARCHITECTURE AND FORMULATION

This section presents a reference architecture that combines the reviewed techniques into one pipeline (Fig. 1). It is intended as a conceptual framework that organises the design space identified in the review. The formulation below is conceptual: no measured results for a prototype are reported in this paper, and the evaluation protocol in Section V-H is provided for future experimental work.

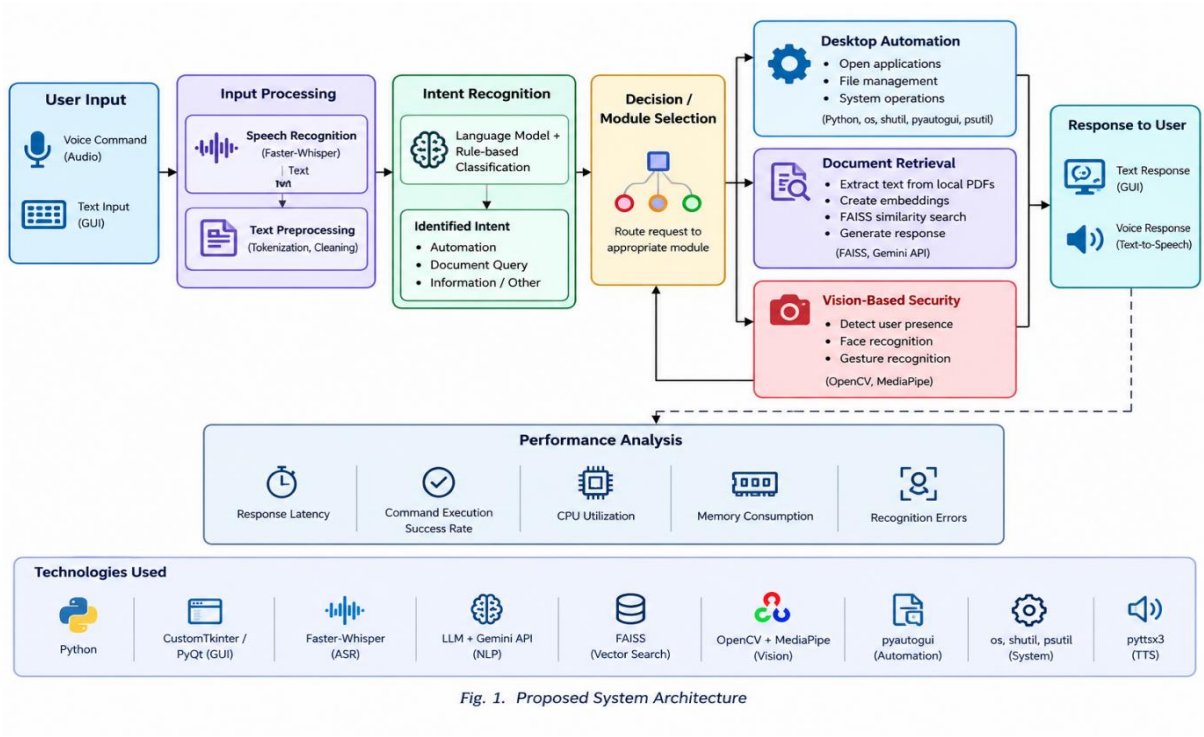


Fig. 1. Reference architecture of an edge-based AI desktop assistant with multimodal interaction and vision-based security. ASR, intent recognition, desktop automation, vector retrieval and vision run locally; in the configuration shown, answer generation for document queries uses a cloud LLM API, which is the point at which retrieved content leaves the device.

© 2026 VT International Press. All rights reserved.

The architecture maps multimodal user inputs (spoken commands, typed text, webcam frames and hand gestures) to desktop actions or natural-language responses, while balancing command accuracy, user privacy and latency under the resource limits of a personal computer.

A. Input Representation

The multimodal input stream is modelled as the set

$$X = \{X_a, X_t, X_v, X_g\} \tag{1}$$

where X_a is microphone audio, X_t is typed text, X_v denotes webcam frames and X_g denotes hand-landmark sequences extracted from X_v by a hand-tracking model [14]. Speech and text are used for commands (Section V-B), X_g for gesture commands (Section V-C) and X_v for authentication (Section V-D).

B. Intent Recognition from Speech and Text

The command text T is obtained from speech by an ASR engine or taken directly from typed input:

$$T = \begin{cases} \mathrm{ASR}(X_a), & \text{if the input is spoken} \\ X_t, & \text{if the input is typed} \end{cases} \tag{2}$$

An intent classification model f_θ with parameters θ then predicts the user's goal:

$$I = f_\theta(T) \tag{3}$$

To route requests, the predicted intent is assigned to one of four functional domains:

$$I \in \{I_{\text{automation}}, I_{\text{document}}, I_{\text{information}}, I_{\text{system}}\} \tag{4}$$

- $I_{automation}$: operating-system actions, e.g., opening applications or managing windows.
- $I_{document}$: queries over local files through RAG (Section V-E).
- $I_{information}$: general-knowledge or web-based look-ups.
- I_{system}: hardware and system-level settings.

C. Gesture Commands

A gesture classifier g_φ maps the landmark sequence X_g to a gesture γ from a fixed vocabulary Γ (e.g., swipe, pinch, open palm), and a mapping h assigns each gesture to an automation intent:

$$\gamma = g_\varphi(X_g) \in \Gamma, \quad I = h(\gamma) \in I_{\text{automation}} \tag{5}$$

Gesture commands therefore bypass ASR and intent classification and are passed directly to the execution policy in Section V-F.

D. Vision-Based Authentication

A face-embedding network e_ψ [12, 43] maps the current webcam frame to an embedding that is compared with the enrolled user template u. The authentication state A is

$$A = \begin{cases} 1, & \text{if } \cos\big(e_\psi(X_v), u\big) \geq \tau \wedge \ell(X_v) = 1 \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

where τ is a similarity threshold chosen on validation data to give the desired false-acceptance rate, and $\ell(X_v) = 1$ when a liveness (anti-spoofing) check is passed [45].

E. Local Document Retrieval (RAG)

Document passages and the user query are converted into dense vector embeddings [26]. For a query embedding q and a passage embedding d_i, semantic similarity is computed with the cosine measure:

$$\mathrm{Sim}(q, d_i) = \frac{q \cdot d_i}{\|q\| \, \|d_i\|} \tag{7}$$

The retriever selects the k most relevant passages from the local vector index D [37]:

$$D_k = \mathrm{TopK}_{d_i \in D}\big(\mathrm{Sim}(q, d_i)\big) \tag{8}$$

The retrieved passages are given as context to the LLM together with the query to generate the response R [9]:

$$R = \mathrm{LLM}(q, D_k) \tag{9}$$

Note on privacy: while a local index (e.g., FAISS) keeps document search on the device, end-to-end privacy depends on where the LLM is hosted. If an external API is used, sensitive retrieved passages cross the network boundary; a local quantised LLM (Section III-E) removes this exposure.

© 2026 VT International Press. All rights reserved.

F. Sensitivity-Aware Execution Policy

Recognising a command does not mean that it should be executed automatically. Each command C is checked by a policy manager $P(C)$, which returns 1 if the command is on the allowed list, and is labelled with a sensitivity flag $s(C)$, which is 1 for actions that are destructive or change system configuration (e.g., deleting files, changing settings). The execution decision $E(C)$ is

$$E(C) = \begin{cases} 1, & \text{if } P(C) = 1 \wedge s(C) = 0 \\ 1, & \text{if } P(C) = 1 \wedge s(C) = 1 \wedge A = 1 \wedge K = 1 \\ 0, & \text{otherwise} \end{cases} \tag{10}$$

where A is the authentication state from (6) and $K = 1$ when the user explicitly confirms the action. Non-sensitive actions (e.g., lowering the volume or opening an application) run without additional checks, whereas sensitive actions require both an authenticated user and explicit confirmation. The same rule is used in Section VII-E.

G. Latency Model

For natural interaction, the response time must stay within acceptable limits. The total latency is modelled as the sum of the delays of the active modules:

$$T_{\text{total}} = T_{\text{capture}} + T_{\text{ASR}} + T_{\text{intent}} + T_{\text{retrieval}} + T_{\text{auth}} + T_{\text{exec}} + T_{\text{response}} \tag{11}$$

where $T_{retrieval}$ and T_{auth} are zero when the corresponding modules are not invoked; for example, a volume change bypasses retrieval. Over a benchmark of N tasks, overall responsiveness is summarised by the mean response time:

$$\overline{T} = \frac{1}{N}\sum_{i=1}^{N} T_i \tag{12}$$

H. Multi-Objective Trade-Off and Evaluation Protocol

Optimising an assistant involves trade-offs: faster responses may reduce retrieval depth, and stricter security policies add friction. Latency L and resource consumption C_r (combined CPU and memory utilisation) are first min–max normalised over the evaluated configurations:

$$\tilde{z} = \frac{z - z_{\min}}{z_{\max} - z_{\min}}, \quad z \in \{L, C_{\text{r}}\} \tag{13}$$

The trade-off is then expressed by the cost function

$$J = \alpha\tilde{L} + \beta(1 - Q) + \gamma(1 - S) + \delta\tilde{C_{\text{r}}} \tag{14}$$

where $Q \in [0, 1]$ is the task-success rate, $S \in [0, 1]$ is security coverage (the fraction of sensitive commands correctly gated by the policy in (10)), and the weights α, β, γ and δ are non-negative and sum to one, so that $J \in [0, 1]$. The weights are selected by a grid search over the weight simplex W (step 0.1) on a validation task set, subject to minimum service levels:

$$w^* = \arg\min_{w \in W} \bar{J}(w) \quad \text{s.t. } \overline{T}(w) \le T_{\max},\ Q(w) \ge Q_{\min} \tag{15}$$

where T_{max} and Q_{min} are the maximum acceptable mean latency and minimum acceptable success rate. Separate weight profiles can be selected for different command classes, for example a latency-focused profile (larger α) for quick desktop controls and an accuracy-focused profile (larger β) for document queries.

For future experimental validation, the following metrics are recommended: mean and 95th-percentile response latency per command class; command-execution success rate; ASR word error rate; intent-classification accuracy and F1 score; RAG answer accuracy on a labelled question set; face-verification false-acceptance and false-rejection rates; peak CPU and RAM usage; and the fraction of requests whose content leaves the device.

VI. DISCUSSION

A. Local versus Cloud Inference

The review shows a clear trade-off between privacy and capability. Local ASR (Whisper, Vosk), local retrieval (FAISS) and local vision (MediaPipe, face embeddings) are mature and run in real time on ordinary hardware, so these stages can be kept on the device with little loss [13, 21, 23]. The language model is the critical stage: quantised 3–8 billion-parameter models make local generation feasible [29–31], but their reasoning is weaker than that of large cloud models, which matters for complex multi-step automation [8, 40]. A practical design is therefore a tiered one, in which routine commands are handled by deterministic automation, document questions by a local LLM, and only non-sensitive, user-approved requests are sent to a cloud model.

B. Running Multiple Models on One Machine

Running ASR, an LLM, an embedding model and vision models together competes for the same CPU, GPU and memory [7, 8]. Strategies identified in the literature include loading models on demand rather than keeping all of them

© 2026 VT International Press. All rights reserved.

resident, using quantised variants [22, 30], sharing one embedding model between intent matching and retrieval, and limiting continuous vision processing to a lightweight detector [44] that triggers heavier models only when needed.

C. Security and Reliability

An assistant that can execute commands is a high-value attack target. Voice spoofing and inaudible commands [3, 46] and indirect prompt injection through retrieved documents [11] show that recognition accuracy alone is not sufficient. The sensitivity-aware policy in (10), combined with liveness-checked face authentication [45], limits the damage from such attacks, at the cost of extra confirmation steps. The low success rates of current OS agents [40] further argue for keeping a human in the loop for consequential actions.

D. Limitations of this Review

This review has several limitations. The field moves quickly, and many relevant works are preprints that have not yet been peer reviewed. Screening was performed by the author team without formal inter-rater agreement measurement. The comparison in Table II is qualitative, because the included works use different hardware, datasets and metrics and therefore cannot be compared numerically. Finally, the reference architecture has not yet been evaluated experimentally, so its latency and accuracy remain to be measured using the protocol in Section V-H.

VII. RESEARCH GAPS AND CHALLENGES

Despite recent advances in AI and desktop automation, several challenges remain in building an efficient, safe and intelligent personal desktop assistant that combines local automation, document retrieval and multimodal interaction.

A. Limited Integration of AI Technologies

Existing desktop assistants usually focus on a narrow set of tasks [16, 24]. Speech recognition and natural-language processing need to be combined with document retrieval and computer vision to enable effective multimodal interaction, and few works evaluate such integrated systems end to end.

B. Privacy and Data Security

When cloud-based language models are used, personal data may be exposed. Local document retrieval is relatively secure, but generating responses through an external API can leak retrieved content [11]. Privacy-preserving generation with local models, and clear user control over what leaves the device, remain open problems.

C. Computational Resource Limitations

Running speech recognition, language understanding, vector search and computer vision simultaneously can consume significant CPU and memory and increase response time [7, 8]. Model optimisation, on-demand loading and scheduling of models on the local machine are needed.

D. Speech and Intent Recognition

The effectiveness of a desktop assistant depends heavily on the accuracy of its speech-recognition component. Background noise, different accents and variations in speaking rate affect recognition quality [21], and natural-language commands are often ambiguous. More robust intent recognition, including clarification dialogues for ambiguous requests, is needed.

E. Security of Desktop Automation

Desktop automation executes user-initiated commands but offers limited safety guarantees, and personal files and applications may be deleted or modified without authorisation. Consistent with the execution policy in Section V-F, sensitive commands (e.g., deleting files or changing system configuration) should require user authentication and explicit confirmation, while non-sensitive commands can run without additional steps to keep interaction fluent.

VIII. FUTURE SCOPE

Edge AI can be used to develop intelligent, secure and efficient personal desktop assistants. The following research directions are recommended.

A. Advanced Local AI Processing

Future research can use lightweight, quantised language models [29, 31] for local processing to improve confidentiality and remove the dependence on cloud services.

B. Intelligent Desktop Automation

Desktop assistants can be extended to interpret and execute complex commands, manage files intelligently and automate multi-step tasks, building on agent frameworks [38, 41] while keeping humans in the loop for sensitive actions.

© 2026 VT International Press. All rights reserved.

C. Enhanced Multimodal Interaction

Voice commands, face authentication and gesture control can be integrated for intuitive, natural interaction, with liveness detection to secure biometric authentication [14, 45].

D. Standardised Evaluation

A shared benchmark for local desktop assistants that reports latency, success rate, resource use and privacy exposure on common hardware would allow future systems to be compared directly.

IX. CONCLUSION

This review examined how Edge AI can be used to build intelligent personal desktop assistants. Using a PRISMA-guided search, 45 studies and software resources were reviewed across nine themes covering speech recognition, intent classification, local large language models, retrieval-augmented generation, operating-system-level agents, and face and gesture recognition. The comparative analysis shows that the necessary components are now available as locally executable tools, but that no existing approach combines broad desktop automation, local processing and strong security. Although Edge AI reduces the amount of data sent to the cloud, end-to-end confidentiality requires that the language model also run locally or that only non-sensitive content be sent externally. The conceptual architecture and formulation presented here provide a framework for designing such assistants, with a sensitivity-aware execution policy and an explicit latency–accuracy–security–resource trade-off. Experimental studies following the proposed evaluation protocol are the next step towards dependable, efficient and intuitive edge-based desktop assistants.

ACKNOWLEDGMENT

The authors thank the HOD, Dr. Vasudha Patil for the opportunity to carry out this work, the project coordinator, G. P. Joshi, for guidance and cooperation, and the faculty members and Shri Chhatrapati Shivaji Maharaj College of Engineering, Nepti, Ahilyanagar, for providing the necessary resources.

REFERENCES

[1] M. B. Hoy, “Alexa, Siri, Cortana, and more: An introduction to voice assistants,” *Medical Reference Services Quarterly*, vol. 37, no. 1, pp. 81–88, 2018, doi: 10.1080/02763869.2018.1404391.

[2] A. de Barcelos Silva *et al.*, “Intelligent personal assistants: A systematic literature review,” *Expert Systems with Applications*, vol. 147, Art. no. 113193, Jun. 2020, doi: 10.1016/j.eswa.2020.113193.

[3] P. Cheng and U. Roedig, “Personal voice assistant security and privacy—A survey,” *Proceedings of the IEEE*, vol. 110, no. 4, pp. 476–507, Apr. 2022, doi: 10.1109/JPROC.2022.3153167.

[4] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, “Edge computing: Vision and challenges,” *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, Oct. 2016, doi: 10.1109/JIOT.2016.2579198.

[5] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang, “Edge intelligence: Paving the last mile of artificial intelligence with edge computing,” *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1738–1762, Aug. 2019, doi: 10.1109/JPROC.2019.2918951.

[6] S. Deng, H. Zhao, W. Fang, J. Yin, S. Dustdar, and A. Y. Zomaya, “Edge intelligence: The confluence of edge computing and artificial intelligence,” *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 7457–7469, Aug. 2020, doi: 10.1109/JIOT.2020.2984887.

[7] X. Wang, Z. Tang, J. Guo, T. Meng, C. Wang, T. Wang, and W. Jia, “Empowering edge intelligence: A comprehensive survey on on-device AI models,” *ACM Computing Surveys*, vol. 57, no. 9, Art. no. 228, 2025, doi: 10.1145/3724420.

[8] J. Xu *et al.*, “On-device language models: A comprehensive review,” arXiv:2409.00088, 2024, doi: 10.48550/arXiv.2409.00088.

[9] P. Lewis *et al.*, “Retrieval-augmented generation for knowledge-intensive NLP tasks,” in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 9459–9474, doi: 10.48550/arXiv.2005.11401.

[10] Y. Gao *et al.*, “Retrieval-augmented generation for large language models: A survey,” arXiv:2312.10997, 2023, doi: 10.48550/arXiv.2312.10997.

[11] K. Greshake, S. Abdelnabi, S. Mishra, C. Endres, T. Holz, and M. Fritz, “Not what you’ve signed up for: Compromising real-world LLM-integrated applications with indirect prompt injection,” in *Proc. 16th ACM Workshop on Artificial Intelligence and Security (AISec)*, 2023, pp. 79–90, doi: 10.1145/3605764.3623985.

[12] F. Schroff, D. Kalenichenko, and J. Philbin, “FaceNet: A unified embedding for face recognition and clustering,” in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 815–823, doi: 10.1109/CVPR.2015.7298682.

[13] C. Lugaresi *et al.*, “MediaPipe: A framework for building perception pipelines,” arXiv:1906.08172, 2019, doi: 10.48550/arXiv.1906.08172.

 © 2026 VT International Press. All rights reserved.

[14] F. Zhang *et al.*, "MediaPipe Hands: On-device real-time hand tracking," arXiv:2006.10214, 2020, doi: 10.48550/arXiv.2006.10214.

[15] M. J. Page *et al.*, "The PRISMA 2020 statement: An updated guideline for reporting systematic reviews," *BMJ*, vol. 372, Art. no. n71, 2021, doi: 10.1136/bmj.n71.

[16] A. Dobriyal, D. Rawat, S. Vats, and V. Sharma, "AI-driven personal desktop assistant," in *Proc. 2024 IEEE 2nd Int. Conf. Innovations in High Speed Communication and Signal Processing (IHCSP)*, 2024, doi: 10.1109/IHCSP63227.2024.10959800.

[17] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," in *Proc. Int. Conf. Learning Representations (ICLR)*, 2016, doi: 10.48550/arXiv.1510.00149.

[18] A. G. Howard *et al.*, "MobileNets: Efficient convolutional neural networks for mobile vision applications," arXiv:1704.04861, 2017, doi: 10.48550/arXiv.1704.04861.

[19] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 12449–12460, doi: 10.48550/arXiv.2006.11477.

[20] A. Gulati *et al.*, "Conformer: Convolution-augmented Transformer for speech recognition," in *Proc. Interspeech*, 2020, pp. 5036–5040, doi: 10.21437/Interspeech.2020-3015.

[21] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, "Robust speech recognition via large-scale weak supervision," in *Proc. 40th Int. Conf. Machine Learning (ICML)*, PMLR vol. 202, 2023, pp. 28492–28518, doi: 10.48550/arXiv.2212.04356.

[22] SYSTRAN, "faster-whisper: Faster Whisper transcription with CTranslate2," GitHub repository (software; no DOI). [Online]. Available: https://github.com/SYSTRAN/faster-whisper (accessed Oct. 2026).

[23] Alpha Cephei, "Vosk speech recognition toolkit: Offline speech recognition API," software documentation (no DOI). [Online]. Available: https://alphacephei.com/vosk/ (accessed Oct. 2026).

[24] A. Coucke *et al.*, "Snips Voice Platform: An embedded spoken language understanding system for private-by-design voice interfaces," arXiv:1805.10190, 2018, doi: 10.48550/arXiv.1805.10190.

[25] Q. Chen, Z. Zhuo, and W. Wang, "BERT for joint intent classification and slot filling," arXiv:1902.10909, 2019, doi: 10.48550/arXiv.1902.10909.

[26] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using Siamese BERT-networks," in *Proc. Conf. Empirical Methods in Natural Language Processing and 9th Int. Joint Conf. Natural Language Processing (EMNLP-IJCNLP)*, 2019, pp. 3982–3992, doi: 10.18653/v1/D19-1410.

[27] H. Touvron *et al.*, "LLaMA: Open and efficient foundation language models," arXiv:2302.13971, 2023, doi: 10.48550/arXiv.2302.13971.

[28] A. Q. Jiang *et al.*, "Mistral 7B," arXiv:2310.06825, 2023, doi: 10.48550/arXiv.2310.06825.

[29] M. Abdin *et al.*, "Phi-3 technical report: A highly capable language model locally on your phone," arXiv:2404.14219, 2024, doi: 10.48550/arXiv.2404.14219.

[30] E. Frantar, S. Ashkboos, T. Hoefler, and D. Alistarh, "GPTQ: Accurate post-training quantization for generative pre-trained transformers," in *Proc. Int. Conf. Learning Representations (ICLR)*, 2023, doi: 10.48550/arXiv.2210.17323.

[31] J. Lin *et al.*, "AWQ: Activation-aware weight quantization for on-device LLM compression and acceleration," in *Proc. Machine Learning and Systems (MLSys)*, vol. 6, 2024, doi: 10.48550/arXiv.2306.00978.

[32] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, "QLoRA: Efficient finetuning of quantized LLMs," in *Advances in Neural Information Processing Systems*, vol. 36, 2023, doi: 10.48550/arXiv.2305.14314.

[33] G. Gerganov and contributors, "llama.cpp: LLM inference in C/C++," GitHub repository (software; no DOI). [Online]. Available: https://github.com/ggml-org/llama.cpp (accessed Oct. 2026).

[34] Ollama, "Ollama: Run large language models locally," software documentation (no DOI). [Online]. Available: https://ollama.com (accessed Oct. 2026).

[35] Gemini Team, Google, "Gemini: A family of highly capable multimodal models," arXiv:2312.11805, 2023, doi: 10.48550/arXiv.2312.11805.

[36] V. Karpukhin *et al.*, "Dense passage retrieval for open-domain question answering," in *Proc. Conf. Empirical Methods in Natural Language Processing (EMNLP)*, 2020, pp. 6769–6781, doi: 10.18653/v1/2020.emnlp-main.550.

[37] J. Johnson, M. Douze, and H. Jégou, "Billion-scale similarity search with GPUs," *IEEE Transactions on Big Data*, vol. 7, no. 3, pp. 535–547, Jul. 2021, doi: 10.1109/TBDATA.2019.2921572.

[38] S. Yao *et al.*, "ReAct: Synergizing reasoning and acting in language models," in *Proc. Int. Conf. Learning Representations (ICLR)*, 2023, doi: 10.48550/arXiv.2210.03629.

© 2026 VT International Press. All rights reserved.

[39] X. Deng *et al.*, "Mind2Web: Towards a generalist agent for the web," in *Advances in Neural Information Processing Systems (Datasets and Benchmarks Track)*, vol. 36, 2023, doi: 10.48550/arXiv.2306.06070.

[40] T. Xie *et al.*, "OSWorld: Benchmarking multimodal agents for open-ended tasks in real computer environments," in *Advances in Neural Information Processing Systems (Datasets and Benchmarks Track)*, vol. 37, 2024, doi: 10.52202/079017-1650.

[41] C. Zhang *et al.*, "UFO: A UI-focused agent for Windows OS interaction," arXiv:2402.07939, 2024, doi: 10.48550/arXiv.2402.07939.

[42] Y. Li *et al.*, "Personal LLM agents: Insights and survey about the capability, efficiency and security," arXiv:2401.05459, 2024, doi: 10.48550/arXiv.2401.05459.

[43] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, "ArcFace: Additive angular margin loss for deep face recognition," in *Proc. IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 4690–4699, doi: 10.1109/CVPR.2019.00482.

[44] V. Bazarevsky, Y. Kartynnik, A. Vakunov, K. Raveendran, and M. Grundmann, "BlazeFace: Sub-millisecond neural face detection on mobile GPUs," arXiv:1907.05047, 2019, doi: 10.48550/arXiv.1907.05047.

[45] Z. Yu, Y. Qin, X. Li, C. Zhao, Z. Lei, and G. Zhao, "Deep learning for face anti-spoofing: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 5, pp. 5609–5631, May 2023, doi: 10.1109/TPAMI.2022.3215850.

[46] G. Zhang, C. Yan, X. Ji, T. Zhang, T. Zhang, and W. Xu, "DolphinAttack: Inaudible voice commands," in *Proc. ACM SIGSAC Conf. Computer and Communications Security (CCS)*, 2017, pp. 103–117, doi: 10.1145/3133956.3134052.

© 2026 VT International Press. All rights reserved.