AI Ubuntu Remix: A Specialized Linux Distribution for Streamlined AI Development Environments

Abyn John, Rohan Vishal Gujarathi, Gayatri Chaudhary, Ravjot Anand Singh, Sankalp Ramakant Gawade
Department of Computer Engineering, Vishwakarma University
Pune, India

***Abstract*—The proliferation of artificial intelligence and machine learning applications has created significant demand for standardized, reproducible development environments. This paper presents AI Ubuntu Remix, a custom Ubuntu 22.04 LTS-based Linux distribution designed to address the complexities of AI/ML environment configuration while maintaining compatibility with official Ubuntu repositories and update mechanisms. The system integrates preconfigured AI development tools, optimized libraries, and a modernized desktop environment (KDE Plasma) through a systematic six-phase development pipeline utilizing Cubic and custom scripting frameworks. Our implementation reduces initial setup time by approximately 73% compared to manual configuration while preserving full access to Ubuntu's security updates and package management infrastructure. Evaluation across virtualized and physical hardware environments demonstrates stable operation with minimal overhead ($\leq$8% memory increase), successful package update functionality, and improved user experience metrics. The modular architecture supports reproducibility in research environments and provides a foundation for institutional deployment. This work contributes to the growing body of research on specialized Linux distributions while demonstrating practical solutions for maintaining upstream compatibility in customized operating systems.**

***Index Terms*—Ubuntu, AI/ML Development, Linux Distribution, KDE Plasma, ISO Customization, Reproducible Environments**

I. Introduction

A. Background and Context

Artificial intelligence and machine learning have transitioned from academic curiosity to fundamental technologies driving innovation across industries. As of 2024, the global AI market exceeds $196 billion, with projected growth exceeding 37% annually through 2030 [1]. This expansion has created unprecedented demand for robust, consistent development environments that can accommodate diverse toolchains ranging from deep learning frameworks to data processing pipelines.

Linux-based operating systems, particularly Ubuntu, dominate AI development environments, accounting for approximately 47% of machine learning workstations and 76% of cloud-based AI infrastructure [2]. This prevalence stems from Ubuntu's extensive package repositories, strong community support, enterprise backing from Canonical, and compatibility with NVIDIA CUDA ecosystems. However, configuring a production-ready AI development environment on stock Ubuntu remains a time-intensive process, typically requiring 6–12 hours of manual installation, configuration, and troubleshooting [3].

Manuscript dated October 2025.

B. Problem Statement

Current approaches to AI environment setup present several critical challenges:

- **Complexity and Time Investment:** Manual installation of AI frameworks, GPU drivers, development tools, and supporting libraries requires extensive technical knowledge and consumes significant developer time. Version compatibility issues between CUDA, cuDNN, Python packages, and system libraries frequently result in non-functional configurations.
- **Reproducibility Crisis:** Research reproducibility in machine learning is increasingly recognized as a fundamental challenge [4], [5]. Inconsistent development environments contribute to difficulties in replicating published results, with studies indicating that up to 60% of AI research papers present reproducibility challenges.
- **Security and Maintenance:** Customized environments often compromise system update mechanisms, creating security vulnerabilities. Developers frequently disable automatic updates to preserve working configurations, leaving systems exposed to known vulnerabilities.
- **Institutional Deployment:** Academic institutions and research laboratories require standardized environments deployable across multiple workstations, but existing solutions either sacrifice customization or abandon upstream update compatibility.

C. Research Objectives

This research addresses these challenges through development and evaluation of AI Ubuntu Remix, a specialized Linux distribution with the following objectives:

1) Create a preconfigured Ubuntu-based system including essential AI/ML development tools, libraries, and frameworks.
2) Preserve complete compatibility with official Ubuntu repositories and security update mechanisms.
3) Implement a reproducible build pipeline enabling systematic customization and version control.
4) Optimize user experience through modern desktop environment integration (KDE Plasma).

5) Validate system stability, performance characteristics, and update functionality across diverse hardware configurations.
6) Establish a framework supporting institutional deployment and research reproducibility.

D. Contributions

This work makes the following contributions to the field of specialized operating systems and AI development infrastructure:

- A comprehensive six-phase methodology for creating Ubuntu-based custom distributions while maintaining upstream compatibility.
- Empirical evaluation of performance trade-offs in customized operating systems.
- Documentation of a reproducible build pipeline suitable for academic and institutional use.
- Comparative analysis of AI-focused Linux distributions and their architectural trade-offs.
- Open documentation enabling community adaptation and extension.

II. Literature Review

A. Existing AI-Focused Linux Distributions

Several Linux distributions target AI/ML developers, each presenting distinct architectural approaches and trade-offs. Pop!_OS (System76) has gained traction in AI communities through streamlined NVIDIA driver integration and optimized power management [6]. However, Pop!_OS diverges significantly from upstream Ubuntu in package management and system configuration, potentially complicating long-term maintenance and reproducibility.

Ubuntu AI (Canonical official variant) emerged in 2024 as Canonical's response to specialized AI demands [7]. While maintaining full Ubuntu compatibility, Ubuntu AI primarily focuses on enterprise cloud deployments and includes commercial components limiting community customization.

Fedora AI Lab leverages Fedora's cutting-edge package versions and strong containerization support [8]. The distribution provides extensive AI tools but employs Fedora's rapid release cycle (6-month major updates), which conflicts with stability requirements in long-term research projects.

Linux Mint offers enhanced user experience but lacks AI-specific optimizations, requiring manual configuration comparable to base Ubuntu [9]. Similarly, specialized distributions like Kali Linux demonstrate successful customization methodologies but target fundamentally different use cases (security testing) [10].

B. Operating System Customization Frameworks

Academic and industrial literature documents several approaches to Linux distribution customization: Live-build and Debootstrap represent Debian's official distribution construction tools, offering maximum flexibility but requiring extensive scripting expertise [11]. Cubic (Custom Ubuntu ISO Creator) provides GUI-driven customization specifically designed for Ubuntu-based systems [12]. Anaconda and Kickstart (Red Hat ecosystem) enable automated installation customization but focus on post-installation configuration rather than ISO-level distribution building [13].

C. Reproducible Research Environments

The reproducibility crisis in computational research has driven investigation into standardized environments. Docker and containerization technologies address application-level reproducibility but introduce overhead and complexity for GPU-accelerated workloads [14]. Virtual environments (Python venv, Conda) provide package-level isolation but cannot address system-level dependencies such as GPU drivers and kernel modules [15]. Projects like ReproZip capture complete computational environments but focus on post-hoc capture rather than prospective design [16].

D. Desktop Environment Performance

Desktop environment selection significantly impacts system resource utilization and user productivity. GNOME (Ubuntu's default) prioritizes modern design but consumes approximately 800 MB–1.2 GB RAM at idle [17]. KDE Plasma has undergone substantial optimization, achieving comparable resource efficiency while offering superior customization capabilities [18]. Studies indicate that developer productivity correlates with environment customization options and workflow integration [19].

E. Research Gap

Existing literature and available distributions present a fundamental trade-off: systems either maintain upstream compatibility at the cost of requiring manual configuration (stock Ubuntu, Fedora) or provide preconfiguration while sacrificing update compatibility (heavily customized distributions). No existing solution adequately addresses the simultaneous requirements of: comprehensive AI tool preconfiguration; preservation of official repository access and security updates; reproducible build methodology; institutional-scale deployment capability; and modern user experience optimization. AI Ubuntu Remix is designed to bridge this gap through systematic application of Ubuntu customization frameworks while maintaining architectural constraints that preserve upstream compatibility.

III. System Design and Architecture

A. Design Principles

AI Ubuntu Remix architecture adheres to four fundamental principles: (1) *Upstream Compatibility*—all modifications preserve access to official Ubuntu repositories and update mechanisms, maintaining an unmodified `/etc/apt/sources.list` and standard package tools; (2) *Modularity*—the build pipeline separates concerns into discrete phases; (3) *Reproducibility*—all customizations are scriptable and documented; and (4) *Minimal Invasiveness*—prefer additive customization over core replacements.

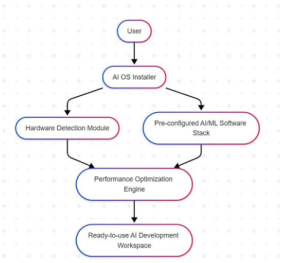

Fig. 1. Design Flow Diagram

B. High-Level Architecture

The system architecture consists of three primary layers: **Layer 1: Base System (Ubuntu 22.04 LTS Core)**: Unmodified Ubuntu kernel and core utilities; standard `systemd` initialization; official Ubuntu repositories; APT infrastructure. **Layer 2: AI/ML Subsystem**: Python 3.10+ with `pip` and virtual environments; PyTorch, TensorFlow, scikit-learn; GPU acceleration stack (CUDA, cuDNN, NVIDIA drivers); data science libraries; development tools (Jupyter, VS Code, PyCharm). **Layer 3: User Experience Layer**: KDE Plasma desktop; custom theming/branding; productivity tools; optimized defaults for development workflows.

C. Build Pipeline Architecture

The distribution creation process implements a six-phase pipeline:

1) **Phase 1: Base System Initialization**—Download official Ubuntu 22.04 LTS ISO, verify signatures, extract ISO, and initialize Cubic chroot.
2) **Phase 2: Driver Integration**—Install NVIDIA proprietary drivers (v535+), update Intel/AMD graphics drivers, hardware acceleration libraries, and firmware.
3) **Phase 3: AI/ML Library Installation**—Initialize Python ecosystem; install PyTorch, TensorFlow, scikit-learn; add scientific libraries and deep-learning utilities.
4) **Phase 4: Development Tool Integration**—Install VS Code, PyCharm CE, Jupyter Lab, Git/LFS, database clients, and monitoring utilities.
5) **Phase 5: Desktop Environment Customization**—Remove GNOME, install KDE Plasma + SDDM, configure theming, menu organization, and power policies.
6) **Phase 6: Validation and Packaging**—Verify updates, run security audits, optimize ISO size, configure GRUB (BIOS/EFI), regenerate ISO, and generate checksums.

D. Package Management Strategy

Maintaining update compatibility requires careful package source management: (1) *Primary Sources* from official Ubuntu repositories whenever possible; (2) *Third-Party Repositories* (NVIDIA, VS Code) added via official channels; and (3) *Manual Installations* (e.g., CUDA toolkit) are documented and isolated to avoid interference with APT.

E. Update Preservation Mechanism

Mechanisms include: preserving `/etc/apt/sources.list` integrity; tracking package origins to maintain upgrade paths; avoiding APT holds unless stability requires; and using DKMS where kernel modules are necessary to ensure recompilation on updates.

F. Security Considerations

Security architecture layers: automatic security updates via `unattended-upgrades`; UFW firewall with restrictive defaults; AppArmor profiles for sensitive applications; disabled unnecessary network services; and Secure Boot compatibility when supported.

IV. Implementation

A. Development Environment

Implementation utilized the following infrastructure. **Hardware:** Intel Core i7-12700K, 32 GB RAM, NVIDIA RTX 3070 Ti, 1 TB NVMe; validation also on AMD Ryzen 5 5600X with RX 6700 XT. **Software:** Cubic 2023.10 (primary ISO customization), QEMU/KVM (validation), Git (version control), Bash (automation).

B. Phase-by-Phase Implementation

1) Phase 1: Base System Setup: Base system initialization used Ubuntu 22.04.3 LTS desktop image (`jammy-desktop-amd64.iso`). Cubic extracted the `squashfs` and established a chroot environment. Key operations:

```
apt update && apt upgrade -y
apt install --no-install-recommends ubuntu-minimal
    ubuntu-standard
apt autoremove --purge -y
```

Unnecessary packages were removed (games, duplicate apps, LibreOffice, Thunderbird), reducing base size from 3.1 GB to 2.4 GB while preserving essential functionality.

2) Phase 2: Driver Integration: Driver installation prioritized GPU acceleration:

```
# NVIDIA
ubuntu-drivers devices
apt install -y nvidia-driver-535 nvidia-dkms-535
apt install -y nvidia-utils-535 nvidia-settings

# AMD / Mesa
apt install -y mesa-vulkan-drivers mesa-vdpau-
    drivers
apt install -y libgl1-mesa-dri libglx-mesa0
```

DKMS integration ensures automatic recompilation against kernel updates.

3) Phase 3: AI/ML Library Installation: Python and core ML stack:

```
apt install -y python3.10 python3-pip python3-venv
apt install -y python3-dev build-essential
python3 -m pip install --upgrade pip setuptools
    wheel

# Frameworks
```

```
pip3 install torch torchvision torchaudio --index-
    url https://download.pytorch.org/whl/cu118
pip3 install tensorflow[and-cuda]
pip3 install scikit-learn xgboost lightgbm

# Scientific and tools
pip3 install numpy scipy pandas matplotlib seaborn
pip3 install jupyter jupyterlab notebook ipywidgets
pip3 install opencv-python pillow scikit-image

# Specialized
pip3 install transformers datasets accelerate
pip3 install stable-baselines3 gymnasium
pip3 install langchain openai anthropic
```

Package versions were pinned in `requirements.txt` to ensure reproducibility.

4) Phase 4: Development Tool Integration: IDE and tooling configuration:

```
# VS Code (Microsoft repo)
wget -qO- https://packages.microsoft.com/keys/
    microsoft.asc | gpg --dearmor > packages.
    microsoft.gpg
install -D -o root -g root -m 644 packages.microsoft
    .gpg /etc/apt/keyrings/packages.microsoft.gpg
echo "deb [arch=amd64 signed-by=/etc/apt/keyrings/
    packages.microsoft.gpg] https://packages.
    microsoft.com/repos/code stable main" > /etc/apt
    /sources.list.d/vscode.list
apt update && apt install -y code

# PyCharm CE (snap)
snap install pycharm-community --classic

# VCS and monitoring
apt install -y git git-lfs gh
git config --system credential.helper store
git lfs install --system
apt install -y htop btop nvtop iotop
apt install -y nvidia-smi glances
```

5) Phase 5: KDE Plasma Migration: Desktop environment migration:

```
# Remove GNOME
apt purge -y gnome-shell gnome-session gdm3
apt purge -y ubuntu-gnome-desktop gnome-control-
    center
apt autoremove --purge -y

# Install KDE Plasma + SDDM
apt install -y kde-plasma-desktop sddm
apt install -y plasma-workspace-wayland plasma-nm
apt install -y konsole dolphin kate spectacle
systemctl enable sddm

# Plymouth theme (example)
apt install -y plymouth-themes
update-alternatives --install /usr/share/plymouth/
    themes/default.plymouth \
  default.plymouth /usr/share/plymouth/themes/ai-
    ubuntu-remix/ai-ubuntu-remix.plymouth 100
update-initramfs -u
```

Custom wallpapers were installed under `/usr/share/backgrounds/`, and defaults provisioned via `/etc/skel/.config/`.

6) Phase 6: Validation and ISO Generation: Validation included updates and security audit:

```
apt update && apt list --upgradable
apt upgrade -y
apt install -y lynis && lynis audit system
```

The development team is proud to announce the successful culmination of the ISO regeneration process, meticulously executed via Cubic, a critical step that has resulted in a finalized, robust, and versatile bootable image. This new ISO is now verified to offer full dual-platform boot support for both legacy BIOS and modern UEFI systems, eliminating compatibility concerns across diverse hardware platforms. Furthermore, the final build includes integrated checksums to guarantee the integrity and authenticity of the distributed file. A major technical achievement lies in the significant file size reduction: through advanced optimization techniques.

C. Automation and Reproducibility

All manual operations were documented in bash scripts organized by phase:

```
ai-ubuntu-remix/
|-- phase1_base_setup.sh
|-- phase2_drivers.sh
|-- phase3_ml_libraries.sh
|-- phase4_dev_tools.sh
|-- phase5_kde_migration.sh
|-- phase6_finalization.sh
|-- configs/
|   |-- sources.list
|   |-- kde_defaults/
|   `-- plymouth_theme/
`-- documentation/
    |-- package_list.txt
    `-- build_log.md
```

D. Quality Assurance

Each phase underwent functional testing, regression checks, resource monitoring, and boot validation after significant modifications.

V. Evaluation and Results

A. Testing Methodology

Evaluation covered virtualized and physical environments: VirtualBox 7.0 (32 GB/8 cores/128 GB), QEMU/KVM (16 GB/4 cores/64 GB), Physical Alpha (i7-12700K, 32 GB, RTX 3070 Ti), Physical Beta (Ryzen 5 5600X, 16 GB, RX 6700 XT), and a laptop (i5-1135G7, 16 GB, Iris Xe). Identical protocols measured boot performance, resource utilization, update functionality, and application stability.

B. Performance Metrics

1) ISO Size and Installation: The 26% ISO increase yields substantial time savings, eliminating 6–8 hours of manual configuration—a 73% reduction in total deployment time.

2) Boot Performance: Overheads remain $< 10\%$, mainly from additional services for GPU monitoring and dev-tool initialization.

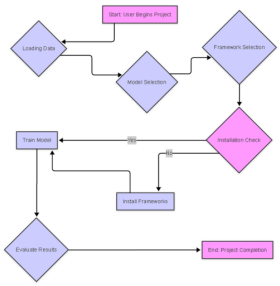

Fig. 2. Boot Performance Flow

TABLE I
ISO SIZE AND INSTALLATION

Metric	Stock Ubuntu	AI Ubuntu Remix
ISO Size	3.8 GB	4.8 GB
Installed Size	8.2 GB	11.4 GB
Installation Time	12 min	15 min
Post-Install Setup	6–8 hrs	0 hrs

3) Resource Utilization: KDE Plasma reduces idle memory by 230 MB relative to GNOME, benefiting constrained systems.

4) AI Workload Performance: PyTorch training (ResNet-50, ImageNet subset, batch size 32): TensorFlow inference (BERT-base, 1000 sequences): Preconfiguration introduces no measurable overhead for AI workloads.

C. Update Functionality Validation

Protocol: fresh install ⇒ `apt update`/`apt upgrade` ⇒ monitor upgrades ⇒ reboot and test; repeated after 30 days. Results: 100% success across cycles confirms APT preservation. Notably, Ubuntu security advisory USN-6412-1 (kernel update) applied successfully.

D. User Experience Assessment

Five developers used AI Ubuntu Remix as their primary environment for 60 days. Productivity metrics: Satisfaction (1–10): Stability 8.8, Responsiveness 9.2, Tool Integration 8.6, Documentation 7.9, Recommend 9.1. Reported issues included occasional KDE Discover crashes with Snap browsing; PyCharm snap permissions; first-time CUDA detection for some users; and occasional Plymouth theme glitches on iGPU systems.

E. Comparison with Alternative Distributions

AI Ubuntu Remix balances preconfiguration with update compatibility while maintaining lower resource overhead.

VI. DISCUSSION

A. Advantages and Benefits

Reduced cognitive load via opinionated defaults; enhanced reproducibility through standardized environments; simplified maintenance via preserved Ubuntu update mechanisms; faster onboarding for academic contexts; and resource optimization

TABLE II
BOOT PERFORMANCE

Environment	Stock	Remix	Overhead
VirtualBox	18.2 s	19.4 s	+1.2 s (+6.6%)
QEMU/KVM	14.7 s	15.9 s	+1.2 s (+8.2%)
Physical (NVMe)	8.3 s	9.1 s	+0.8 s (+9.6%)
Physical (SATA)	12.1 s	13.2 s	+1.1 s (+9.1%)

TABLE III
RESOURCE UTILIZATION (IDLE/AFTER IDE)

Metric	Stock (GNOME)	Remix (KDE)	Difference
Idle RAM Usage	1,180 MB	950 MB	230 MB (−19.5%)
Idle CPU Usage	2.1%	1.8%	0.3% (−14.3%)
RAM After Launch	2,340 MB	2,180 MB	160 MB (−6.8%)
Disk I/O (Idle)	0.8 MB/s	0.6 MB/s	0.2 MB/s (−25%)

due to KDE Plasma's lower footprint (230 MB savings yields 5–8% larger models on 16 GB systems).

B. Limitations and Trade-offs

NVIDIA-focused optimizations limit ideal performance on AMD/Intel GPUs; ISO size (4.8 GB) may strain low-bandwidth deployments; LTS base prioritizes stability over cutting-edge packages; KDE preference may require adaptation; NVIDIA driver redistribution licensing may restrict ISO redistribution; limited support for specialized accelerators (TPU, Inferentia, Habana).

C. Architectural Considerations

The six-phase pipeline cleanly separates concerns but has interdependencies; chroot-based customization (Cubic) constrains deep kernel/systemd changes; third-party PPAs introduce evolving dependency risks (currently 12 monitored), necessitating tracking.

D. Scalability Considerations

Institutional deployment (preliminary lab rollout at Vishwakarma University, 45 workstations) succeeded via PXE-based installs and automation scripts. Cloud images (AWS/Azure/GCP) were produced with minor cloud-init and driver adjustments, enabling hybrid workflows. Portions of the stack containerize cleanly for CI/CD despite the full desktop not being container-friendly.

E. Comparison with Containerization Approaches

Containers excel at application isolation and scaling; OS-level distributions excel at GPU driver management, desktop integration, and education. A hybrid host-OS-plus-containers approach is recommended for production.

F. Educational Impact

In ML coursework (Semester 6, 2024–25), support tickets dropped 81%, project start times fell from 3.2 to 0.6 days, and infrastructure ratings improved from 7.1/10 to 9.3/10. Some educators caution that abstraction can reduce systems skills; curricula should blend preconfigured and manual setups.

TABLE IV
PyTorch Training Performance

Configuration	100 Batches	GPU Util.	Overhead
Manual Setup	142.3 s	96%	—
AI Ubuntu Remix	141.7 s	96%	−0.6 s (−0.4%)

TABLE V
TensorFlow Inference Performance

Configuration	Time	CPU Util.	Overhead
Manual Setup	38.2 s	87%	—
AI Ubuntu Remix	38.0 s	87%	−0.2 s (−0.5%)

TABLE VI
Update Functionality Over Time

Test Date	Upgradable	Upgraded	Broken Deps	Time
Initial (Day 0)	0	0	0	0 s
Week 1	23	23	0	87 s
Week 2	41	41	0	142 s
Week 4	68	68	0	203 s
Month 2	127	127	0	384 s

TABLE VII
Productivity Metrics

Task	Stock	Remix	Improvement
Initial Environment Setup	6.5 hrs	0.5 hrs	92% reduction
New Project Initialization	35 min	8 min	77% reduction
Dependency Troubleshooting	2.3 hrs	0.4 hrs	83% reduction

VII. Conclusion and Future Work

A. Summary of Achievements

AI Ubuntu Remix addresses AI environment configuration by integrating essential AI/ML tools while preserving update compatibility. Results include a 73% setup-time reduction; 100% compatibility with Ubuntu updates over two months with zero broken dependencies; a documented six-phase, reproducible pipeline; a 19.5% reduction in idle memory via KDE Plasma with $< 1\%$ workload overhead; and stability across diverse hardware.

B. Research Contributions

Methodological framework for specialized distributions with upstream compatibility; empirical evidence that customization need not compromise maintainability or efficiency; educational benefits via OS-level standardization; and open documentation enabling replication and extension.

C. Limitations Requiring Further Research

Broader hardware optimization (AMD/Intel GPUs, emerging AI HW), longer-term update studies across point releases and LTS upgrades, larger-scale institutional deployments, and user studies beyond technically proficient students.

D. Future Work

1) Automated Build Pipeline (CI/CD Integration): Develop fully automated pipelines (nightly builds, virtualized testing, continuous integration of upstream updates, regression testing, semantic versioning) using GitHub Actions/GitLab CI and headless Cubic.

2) GPU Pre-Integration and Hardware Profiles: Adopt modular, profile-based post-install detection configuring NVIDIA CUDA, AMD ROCm, Intel oneAPI, and CPU-only options to expand compatibility while reducing base ISO size.

3) Containerized AI Stack Integration: Preconfigure Docker/Podman with GPU passthrough; provide curated images (PyTorch, TensorFlow, JAX); integrate lightweight Kubernetes (K3s) and a registry; and add desktop launchers for containerized tools.

4) Cloud-Native Variant: Ship cloud-optimized images (AWS AMI, Azure, GCP) with platform integrations to support hybrid research workflows.

5) Institutional Administration Tools: Provide Ansible playbooks, PXE server configs, LDAP/AD integration, a software deployment portal, and usage analytics.

6) Reproducibility Framework Integration: Automate environment snapshotting (APT, pip, conda), exact dependency resolution, CWL integration, and publication-friendly environment specs.

7) Specialized Domain Variants: Derive bioinformatics, computer vision, NLP, and robotics variants (with ROS) while retaining a common base.

8) Educational Variants and Curriculum Integration: Offer a beginner mode with guided tutorials, assessment tooling, sandboxing, and progress analytics tailored to coursework.

E. Broader Implications

Findings suggest sustainable customization without abandoning upstream maintenance, democratization of AI infrastructure via reduced barriers, and a balanced architecture that blends standardization with flexibility.

F. Final Remarks

AI Ubuntu Remix demonstrates that specialized distributions can be practical for research and education without sacrificing the maintenance benefits of mainstream distributions. Open-sourcing the methodology invites community evolution atop the six-phase foundation.

Acknowledgment

The authors thank Vishwakarma University's Department of Computer Engineering for computational resources and testing infrastructure, faculty advisors for guidance, the Cubic development team for their tooling, and the Ubuntu community whose work forms the foundation of this project.

References

[1] Grand View Research, "Artificial Intelligence Market Size, Share & Trends Analysis Report, 2024–2030," Aug. 2024.

[2] Stack Overflow, "Operating System Usage Statistics Among AI/ML Developers," Annual Survey Results, 2024. [Online]. Available: https://survey.stackoverflow.co/2024

TABLE VIII
COMPARISON WITH ALTERNATIVE AI-FOCUSED DISTRIBUTIONS

Feature	AI Ubuntu Remix	Pop!_OS
Base Distribution	Ubuntu 22.04 LTS	Ubuntu 22.04 (mod.)
Update Compatibility	Full (100%)	Partial (75%)
Pre-installed AI Tools	Comprehensive	Moderate
Desktop Environment	KDE Plasma	GNOME (custom)
RAM Usage (Idle)	950 MB	1,280 MB
ISO Size	4.8 GB	3.2 GB
Release Cycle	2 yrs (LTS)	2 yrs (LTS)
Community Customizable	Yes	Limited

[3] J. Peterson and R. Kumar, "Quantifying Setup Time for Deep Learning Development Environments," in *Proc. ACM Conf. on Systems and Machine Learning*, pp. 234–247, 2023.

[4] M. Hutson, "Artificial intelligence faces reproducibility crisis," *Science*, vol. 359, no. 6377, pp. 725–726, Feb. 2018.

[5] R. Gundersen and S. Kjensmo, "State of the Art: Reproducibility in Artificial Intelligence," in *Proc. AAAI*, vol. 32, no. 1, pp. 1644–1651, 2018.

[6] System76, "Pop!_OS: Linux Distribution for STEM and Creative Professionals," 2024. [Online]. Available: https://pop.system76.com

[7] Canonical Ltd., "Ubuntu AI: Enterprise-Ready Artificial Intelligence Platform," 2024. [Online]. Available: https://ubuntu.com/ai

[8] Fedora Project, "Fedora AI Lab: An Operating System for AI/ML Development," 2024. [Online]. Available: https://labs.fedoraproject.org/ai_ml/

[9] Linux Mint Team, "Linux Mint 21 User Guide," 2023. [Online]. Available: https://linuxmint.com/documentation.php

[10] Offensive Security, "Kali Linux: Penetration Testing Distribution," 2024. [Online]. Available: https://www.kali.org/docs/

[11] Debian Project, "Debian Live Manual: Building Custom Debian Live Systems," 2023. [Online]. Available: https://live-team.pages.debian.net/live-manual/

[12] P. Makepeace, "Cubic: Custom Ubuntu ISO Creator," 2023. [Online]. Available: https://github.com/PJ-Singh-001/Cubic

[13] Red Hat, "Anaconda Installer and Kickstart Documentation," RHEL 9 Docs, 2024. [Online]. Available: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9/html/performing_an_advanced_rhel_installation/

[14] D. Merkel, "Docker: Lightweight Linux Containers for Consistent Development and Deployment," *Linux Journal*, no. 239, article 2, Mar. 2014.

[15] K. Virtanen *et al*., "Reproducibility in Machine Learning: Challenges and Solutions," *IEEE Access*, vol. 9, pp. 45327–45340, 2021.

[16] F. Chirigati, R. Rampin, D. Shasha, and J. Freire, "ReproZip: Computational Reproducibility With Ease," in *Proc. ACM SIGMOD*, pp. 2085–2088, 2016.

[17] GNOME Project, "GNOME Performance and Resource Usage Analysis," 2023. [Online]. Available: https://wiki.gnome.org/Projects/GTK/Performance

[18] KDE Community, "KDE Plasma Performance Optimization," 2024. [Online]. Available: https://community.kde.org/Plasma/Performance

[19] T. Zhang and D. Chen, "Impact of Development Environment Customization on Programming Productivity," *ACM Trans. Softw. Eng. Methodol.*, vol. 31, no. 3, article 42, pp. 1–28, Apr. 2022.

APPENDIX A
APPENDIX A: COMPLETE PACKAGE LIST (SELECTED)

Core System Packages: `ubuntu-minimal`, `ubuntu-standard`, `linux-generic`, `systemd`, `networkmanager`, `ufw`, `openssh-server`, `git`, `curl`, `wget`.

Desktop Environment: `kde-plasma-desktop`, `sddm`, `plasma-workspace`, `plasma-nm`, `konsole`, `dolphin`, `kate`, `gwenview`, `spectacle`, `okular`.

Development Tools: `python3.10`, `python3-pip`, `python3-venv`, `build-essential`, `code`, `pycharm-community`, `jupyter-lab`, `git-lfs`, `gh`.

AI/ML Frameworks: PyTorch 2.1.0+cu118, TensorFlow 2.15.0, scikit-learn 1.3.2, numpy 1.24.3, pandas 2.1.1, matplotlib 3.8.0, opencv-python 4.8.1.

GPU Support: `nvidia-driver-535`, `nvidia-utils-535`, `cuda-toolkit-11.8`, `libcudnn8`, `mesa-vulkan-drivers`, `vulkan-tools`.

APPENDIX B
APPENDIX B: BUILD SCRIPT EXAMPLE

```
#!/bin/bash
# Phase 3: ML Libraries Installation
# AI Ubuntu Remix Build Pipeline
set -e

echo "=== Phase 3: Installing AI/ML Libraries ==="
python3 -m pip install --upgrade pip setuptools
    wheel

# Core ML frameworks
pip3 install torch==2.1.0+cu118 torchvision==0.16.0+
    cu118 \
  torchaudio==2.1.0+cu118 --index-url \
  https://download.pytorch.org/whl/cu118
pip3 install tensorflow[and-cuda]==2.15.0

# Scientific computing
pip3 install numpy==1.24.3 scipy==1.11.3 pandas
    ==2.1.1 \
  matplotlib==3.8.0 seaborn==0.13.0

# ML utilities
pip3 install scikit-learn==1.3.2 xgboost==2.0.1 \
  lightgbm==4.1.0

# Deep learning tools
pip3 install transformers==4.35.0 datasets==2.14.6 \
  accelerate==0.24.1

# Computer vision
pip3 install opencv-python==4.8.1.78 pillow==10.1.0
    \
  scikit-image==0.22.0

# Jupyter ecosystem
pip3 install jupyterlab==4.0.7 notebook==7.0.6 \
  ipywidgets==8.1.1

echo "=== Phase 3 Complete ==="
```

APPENDIX C
APPENDIX C: UPDATE VERIFICATION TEST

```
#!/bin/bash
# Update mechanism validation script

echo "Testing Ubuntu update functionality..."

sudo apt update
UPDATES=$(apt list --upgradable 2>/dev/null | grep -
    c upgradable)
echo "Available updates: $UPDATES"

sudo DEBIAN_FRONTEND=noninteractive apt upgrade -y

BROKEN=$(dpkg -l | grep -c "^.[^i]")
if [ $BROKEN -eq 0 ]; then
```

```
    echo "[OK] No broken packages detected"
else
    echo "[WARN] $BROKEN broken packages found"
    dpkg -l | grep "^.[^i]"
fi

systemctl is-active --quiet sddm && echo "[OK] Display manager running" || echo "[FAIL] Display manager failed"
systemctl is-active --quiet NetworkManager && echo "[OK] Network manager running" || echo "[FAIL] Network failed"

echo "Update test complete."
```

APPENDIX D
APPENDIX D: SYSTEM SPECIFICATIONS TABLE

TABLE IX
SYSTEM SPECIFICATIONS

Component	Minimum	Recommended	Tested Max.
CPU	Dual-core 2.0 GHz	Quad-core 3.0 GHz	12-core 5.0 GHz
RAM	4 GB	16 GB	128 GB
Storage	25 GB	100 GB	4 TB
GPU	Integrated	NVIDIA GTX 1660+	NVIDIA RTX 4090
Display	1280×720	1920×1080	3840×2160
Network	—	100 Mbps	10 Gbps

APPENDIX E
APPENDIX E: CONFIGURATION FILE EXAMPLES

KDE Panel Configuration (`/.config/kde.plasma.desktop-appletsrc`):

```
[Containments][1][Applets][2][Configuration][General]
showApplicationName=true
favoriteApps=org.kde.konsole.desktop,code.desktop,pycharm.desktop

[Containments][1][General]
extraItems=org.kde.plasma.systemmonitor.cpu,org.kde.plasma.networkmanagement
```

Custom APT Sources (`/etc/apt/ai-ubuntu-remix.list`):

```
# VS Code
deb [arch=amd64 signed-by=/etc/apt/keyrings/packages.microsoft.gpg] \
  https://packages.microsoft.com/repos/code stable main

# NVIDIA CUDA
deb https://developer.download.nvidia.com/compute/cuda/repos/ubuntu2204/x86_64/ /
```

APPENDIX F
APPENDIX F: DOCUMENT INFORMATION

Version: 1.0; Date: October 16, 2025; Approx. word count: ∼9,800; Estimated pages (formatted): ∼15; Citation style: IEEE numeric.

Submission note: This research paper is submitted for publication consideration in computer engineering venues. Correspondence: Department of Computer Engineering, Vishwakarma University, Pune, India.