

Comparative Performance and Cost Analysis of Computer Vision in Edge-Cloud Continuum

Hongyu Guo
Technical University of Munich
Munich, Germany

Navidreza Asadi
Technical University of Munich
Munich, Germany

Giovanni Bartolomeo
Technical University of Munich
Munich, Germany

Patrick Laufer
Technical University of Berlin
Berlin, Germany

Jörg Ott
Technical University of Munich
Munich, Germany

Wolfgang Kellerer
Technical University of Munich
Munich, Germany

Abstract

As Artificial Intelligence (AI) applications increasingly migrate to the cloud, selecting appropriate deployment strategies that ensure performance while minimizing cost has become a key challenge. This paper investigates the deployment of a Computer Vision workload in four representative strategies across the edge-cloud continuum. These include cloud-only, edge-only, and hybrid edge-cloud architectures, incorporating both serverless and server-based computing models. Furthermore, we investigate the feasibility of integrating advanced techniques such as Split Computing and Early Exiting within hybrid architectures. We conduct real-world experiments throughout Germany on AWS cloud *Availability Zones* from Amazon and edge *Wavelength Zones* within Vodafone infrastructure. For each deployment strategy, we measure latency, accuracy, and operational cost. Our key findings reveal a $\approx 40\%$ average lower end-to-end latency on edge deployments. We also outline the trade-offs and application scenarios for each strategy, providing practical guidance for the efficient deployment of computer vision systems in cloud/edge environments.

CCS Concepts

• **Computing methodologies** → **Machine learning**; • **Computer systems organization** → **Cloud computing**; **Distributed architectures**; • **General and reference** → *Evaluation*.

Keywords

Edge-cloud collaboration, computer vision, CV deployment, cloud services, split computing, early exiting

ACM Reference Format:

Hongyu Guo, Navidreza Asadi, Giovanni Bartolomeo, Patrick Laufer, Jörg Ott, and Wolfgang Kellerer. 2025. Comparative Performance and Cost Analysis of Computer Vision in Edge-Cloud Continuum. In *Proceedings of the 2025 ACM CoNEXT Workshop Edge-Cloud Collaboration for AI (ECCAI '25)*, December 1–4, 2025, Hong Kong, Hong Kong. ACM, New York, NY, USA, 8 pages. <https://doi.org/10.1145/3769696.3771218>



This work is licensed under a Creative Commons Attribution 4.0 International License. *ECCAI '25, Hong Kong, Hong Kong*

© 2025 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2242-4/2025/12
<https://doi.org/10.1145/3769696.3771218>

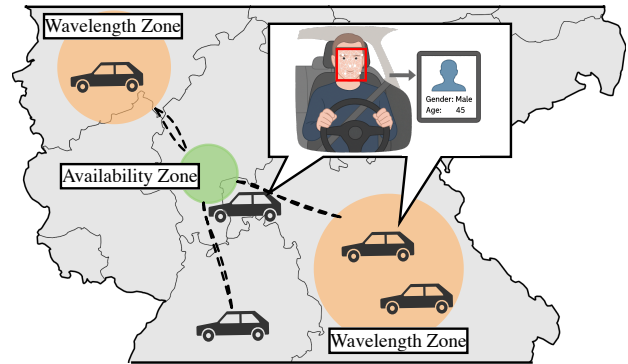


Figure 1: Use Case for In-Cabin System Deployments With the Integration of AWS Availability Zones and Wavelength Zones for Cloud and Edge Computing.

1 Introduction

AI is everywhere; natural language processing (NLP), computer vision (CV), and speech recognition are only a few domains among many [1, 24]. While more accurate AI models emerge on a daily basis, this often comes with a significant increase in the size of the models and the amount of data they process [55]. This creates deployment challenges for resource-limited edge and IoT devices. On one hand, it is convenient to exploit locality and process data close to its source, such as in edge and IoT devices [6, 7, 13, 27–29, 33, 35, 46, 53, 57]. On the other hand, the computational requirements are difficult to meet in these resource-limited devices, necessitating offloading to more powerful infrastructures [19, 48, 49].

Many enterprises face difficulties meeting the resource provisioning requirements due to the upfront costs and uncertain return on investment [16, 17, 21]. Therefore, cloud computing is often the preferred solution to address these challenges [21, 41]. To address these challenges, cloud computing has emerged as an important solution. Cloud providers offer a wide range of services on a pay-as-you-go pricing model, allowing users to avoid heavy initial investments and to scale resources as needed [3, 25].

Serving AI in the edge-cloud continuum. Cloud solutions, while providing flexibility and scalability, are often associated with high

latency and bandwidth costs [47], especially for real-time applications that require low-latency responses. A leading use case are smart vehicles offering entertainment and security features relying on Vehicle-to-Everything (V2X) communications and AI offloading [54]. For this reason, hybrid edge-cloud deployments have become popular [8, 9, 15, 22, 31, 37, 40], as well as cloud providers partnering with telecom operators to offer edge computing services [5]. Such complex offerings increase the challenge for users to select the most suitable deployment strategy and pricing models for their AI applications.

Our Contributions. In this work, we study the performance and costs trade-offs of edge-to-cloud continuum deployments of a computer vision application. We consider an exemplary in-cabin AI system for smart vehicles which offloads the computationally intensive task to external infrastructures such as the Amazon Web Services (AWS) platform. Specifically, we focus on the deployment of Convolutional Neural Networks (CNNs) for computer vision tasks. We limit the scope of this work to a nationwide deployment scenario in Germany, where the AWS Cloud Infrastructure extends to the edge in collaboration with Vodafone as part of their mobile infrastructure [56].

As shown in Fig. 1, the proposed system spans across both cloud and edge infrastructures, utilizing the *availability zone* in Frankfurt for centralized processing and *wavelength zones* [5] in Berlin, Munich, and Dortmund to enable edge computing.

Depending on user location and deployment configuration, inference tasks are executed entirely at the edge (edge-only), in the cloud (cloud-only), or across both (hybrid). To support the edge-cloud continuum, we further investigate two complementary techniques: Split Computing (SC) and Early Exiting (EE) [12, 14, 36]. SC partitions the model between the edge and the cloud to offload computational burden from the edge while maintaining full accuracy. EE introduces lightweight model branches at intermediate layers, enabling early predictions for confident inputs and thereby reducing average inference latency and communication overhead.

In summary, our key contributions are:

- (1) We design and implement four deployment strategies on AWS, spanning cloud, edge, and hybrid topologies with both server-based and serverless options.
- (2) We integrate SC and EE into hybrid architectures and explore the feasibility and effectiveness of these methods through detailed system design and experiments.
- (3) We conduct a comprehensive evaluation of each deployment strategy under real 5G network conditions, focusing on latency, cost, and accuracy.

2 System Design and Deployment Strategies

This section outlines the computer vision task and deployment strategies adopted for the comparative analysis that we propose. First, we choose a Convolutional Neural Network (CNN) model as a representative candidate for computer vision task offloading (Section 2.1). Secondly, we introduce model partitioning to improve adaptability to constrained edge environments and we employ early exiting mechanisms to reduce network overhead (Section 2.2). Finally, we define four architectures and deployment strategies for cloud, edge and hybrid production environments (Section 2.3).

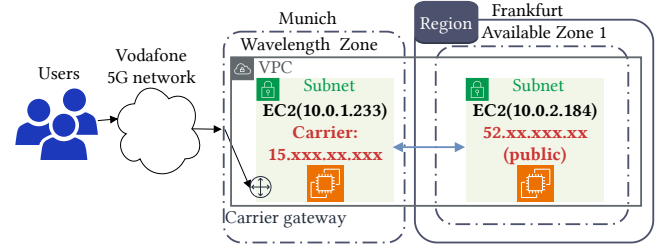


Figure 2: Hybrid Deployment Architecture Example Combining Cloud and Edge EC2 Instances in Germany.

2.1 CNN Model

For the computer vision task, we start from a pre-trained CNN model based on the widely used VGG-style [50] architecture. This model is well-suited for object detection and classification and is commonly employed in many computer vision applications [11, 30, 52]. We fine-tune the model for binary gender prediction, a continuous and multi-class (infant, child, teenager, adult and elderly) age estimation from facial images [32]. Age and gender prediction is a task well represented in the literature, which provides a suitable accuracy comparison of this work with existing solutions. On the other hand, the model architecture makes the proposed measurements representative of real world computer-vision offloading. In fact, models such as VGG-16 are often used as a starting point for transfer learning [38, 51], where the model is fine-tuned on specific datasets to adapt it to new tasks. They are employed in use cases such as in-vehicle entertainment and driver monitoring systems, where real-time processing of visual data is crucial for applications like driver behavior analysis and passenger safety [26, 38, 42]. These contexts require efficient task offloading and low-latency inference for in-vehicle safety and accidents prevention.

The UTK dataset [58], comprising approximately 24,000 frontal face images, is used to train the proposed multi-task CNN model. Experimental results demonstrate that the model achieves about 90% accuracy in gender prediction, 60% accuracy in age group classification, and a mean absolute error of 6 years in regression-based age estimation.

2.2 Distributed Inference

The edge-cloud continuum enables efficient distributed inference by leveraging both local and cloud resources. Techniques such as Split Computing (SC) and Early Exiting (EE) are employed in such scenarios to optimize models deployments, their scalability, reduce the network overhead and to offer collaborative inference. We integrate these techniques to the CNN model proposed in Section 2.1 in order to support distributed hybrid edge-cloud deployments.

Split Computing. We adopt a split computing strategy that partitions the CNN model into two segments: a *head model* deployed at the edge and a *tail model* running in the cloud, as illustrated in Fig. 3. To minimize data transmission while preserving user privacy, the model is split at the `block4_pool` layer, which is a natural bottleneck and deep enough to contain only abstract semantic features, making it less likely to expose personal information. During inference, the edge node processes the input image and transmits

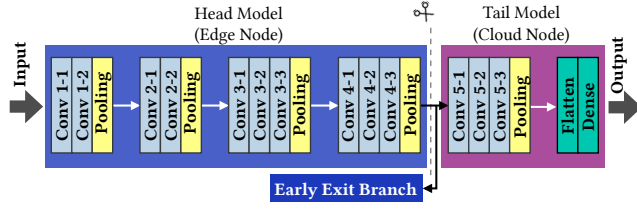


Figure 3: Split Computing Architecture With an Optional Early Exiting Branch.

the intermediate feature map to the cloud for final prediction. This approach maintains model accuracy while enhancing privacy, as raw data remain on the local device. However, it still requires the entire model pipeline to be executed, which may lead to additional latency for easily classifiable samples.

Early Exiting. To enhance the efficiency and adaptability of split computing systems, we add early exit branches by embedding lightweight classifiers at intermediate layers (block2_pool1 and block4_pool1). Branch Classifier 1 (Exit 1) includes two convolutional layers to enhance feature abstraction due to the limited depth of the backbone at that stage. In contrast, Branch Classifier 2 (Exit 2) incorporates only one convolutional layer to balance accuracy and computational efficiency (see Fig. 4). These exits enable fast inference for confident samples relying on entropy-based confidence thresholds. We adopt a separate training strategy: the main model is pretrained and frozen, and early exits are trained individually. Thresholds are empirically selected to balance accuracy and efficiency, set to 0.6 for gender and 0.8 for age at both exits. This mechanism allows confident samples to exit early, reducing computation and communication overhead.

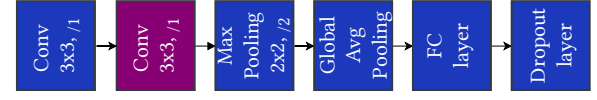
2.3 AWS Deployment

We design and deploy the proposed CNN model with the SC and EE strategies on the AWS platform, focusing on two key dimensions: the compute model (server-based vs. serverless) and the architectural topology (cloud, edge, and hybrid). We choose AWS Lambda and EC2 as representatives for serverless and server-based computing models, respectively.

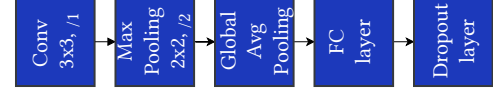
Beyond the traditional cloud infrastructure hosted in Availability Zones, AWS Wavelength extends compute and storage services to the edge of communication service providers' (CSPs) networks, enabling lower data propagation delay.

Considering both compute model and deployment architecture, we evaluate four representative deployment strategies: ① *Lambda in the Cloud*, ② *EC2 in the Cloud*, ③ *EC2 at the Edge*, and ④ *EC2 in an Edge-Cloud Hybrid Architecture*.

① **Lambda in the Cloud.** Since AWS Wavelength Zones do not currently support Lambda service and there is no direct ultra-low-latency link between Lambda and EC2, combining these services does not provide latency benefits, especially for global applications. Therefore, Lambda is evaluated only in a cloud-based setup. In this study, Lambda is deployed via a container image to gain better control over the runtime environment. A public Function URL is used to expose the service, with CORS enabled for compatibility with browser-based clients. A standard `lambda_handler` serves as entry



(a) Design of Branch Classifier 1



(b) Design of Branch Classifier 2

Figure 4: Branch Classifier Design in Early Exiting: the First Classifier Is Placed After Block 2 Pooling, and the Second Classifier Is Placed After Block 4 Pooling.

point. To ensure comparable performance to the EC2 t3.medium instance, memory and ephemeral storage are set to 4 GB.

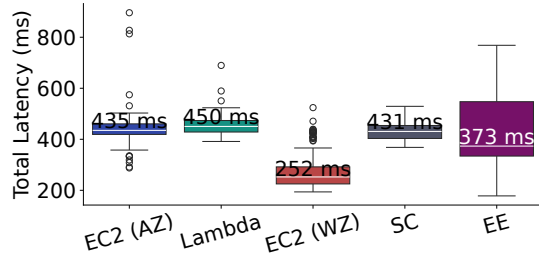
②-④ **EC2 Deployments.** For fair comparison, all EC2-based deployments use the same instance type: t3.medium, which provides 2 vCPUs and 4 GB of memory. Additionally, each instance is configured with a 24 GB General Purpose SSD (gp2) EBS volume for persistent storage. In the *Cloud Deployment*, the EC2 instance is hosted in an Availability Zone (e.g., eu-central-1 in Frankfurt), with an Internet Gateway attached to enable public network access. In the *Edge Deployment*, the EC2 instance is launched in a Wavelength Zone (e.g., eu-central-1-wl1-muc-wl1 in Munich) to achieve low-latency inference. A Carrier Gateway is used to establish connectivity between the Wavelength Zone and the telecommunications network infrastructure. In the *Hybrid Edge-Cloud Deployment*, both edge and cloud EC2 instances are deployed in the same VPC but assigned to different subnets. Requests are initially sent to the edge node, where the model SC head is deployed. This node, based on EE principles, may either process the request locally or forward it to the cloud instance, as illustrated in Fig. 2.

3 Evaluation

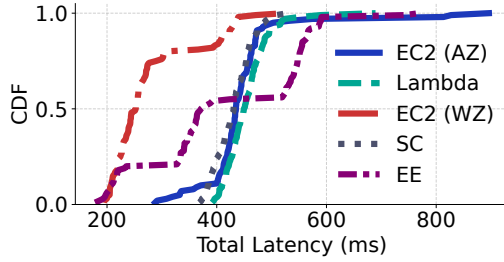
We send traffic from a laptop connected to the Vodafone network via a 5G router. Each experiment consists of 100 balanced samples, with an equal number of instances for each age group and gender. The samples come from a cross-dataset validation involving various publicly available datasets, complemented by our own test subject dataset [32]. We deploy the backend as in §2.3, employing both EC2 instances and a Lambda function. We utilize NTP to ensure time synchronization across all servers. To analyze latency performance, key timestamps were recorded at each request and response stage and embedded within the message payload.

3.1 5G and WAN Network Performance

We present baseline network performance data that is independent of any specific application deployment. These results were collected in Munich using a Vodafone 5G connection at location 48.1833326, 11.533497866. We measure round-trip latency using ping and upload/download bandwidth using `iperf3`, and repeat each experiment ten times, reporting minimum, average, maximum, and standard deviation. Our communication paths include:



(a) Latency Distribution with Median Highlighted.



(b) CDF of Total Latency.

Figure 5: Total Latency Distribution.

- (a) From a user device located in a Wavelength Zone to an EC2 instance in the Wavelength Zone.
 (b) From a user device (not in the Wavelength Zone) to an EC2 instance in the Availability Zone.
 (c) Between an EC2 instance in the Availability Zone and another in the Wavelength Zone.

The results in Table 1 show that Wavelength Zones offer lower latency and higher bandwidth than Availability Zones, due to their proximity and 5G integration. However, in our test regions (Munich for WZ and Frankfurt for AZ), the performance gap is moderate. Communication between WZ and AZ is highly efficient, with average upload speeds over 1.35 Gbps and download speeds around 240 Mbps, suggesting that hybrid edge-cloud deployments are feasible with minimal communication overhead.

Table 1: Network Performance Metrics

Metrics		Min.	Avg.	Max.	Std. dev.
Latency (ms)	(a)	19.61	24.16	47.48	6.21
	(b)	25.74	31.38	50.58	6.72
	(c)	6.51	6.64	6.71	0.04
Upload Bandwidth (Mbit/s)	(a)	24.10	34.50	48.10	7.30
	(b)	8.38	30.43	53.70	13.97
	(c)	1180	1356	1700	165.9
Download Bandwidth (Mbit/s)	(a)	–	–	–	–
	(b)	–	–	–	–
	(c)	208.0	239.7	273.0	21.29

3.2 Latency

In this section, we evaluate latency from the perspective of the specific application we deployed. The total latency is defined as the time from the request being sent by the client to the final response being received. Figure 5 shows the total latency distribution for each deployment strategy. Deploying the application on an EC2 instance in the Wavelength Zone results in the lowest latency. Lambda and EC2 perform similarly when deployed in the Availability Zone. For hybrid strategies like Split Computing (SC) and Early Exiting (EE), latency falls between pure edge and cloud deployments. Notably, EE achieves lower average latency than SC, as some samples exit early at the edge, while others are forwarded to the cloud, resulting in a wider distribution.

It is worth noting that serverless platforms such as AWS Lambda typically experience cold start issues. In our experiments, we observed that the end-to-end latency of requests affected by cold start was approximately 30 seconds. This overhead can be mitigated by enabling features like provisioned concurrency [2]. We exclude cold start scenarios from our latency analysis in this study.

To further analyze latency components, we separate the compute and communication latencies. Compute latency refers to the total processing time on the backend server. In hybrid architectures, we further split it into edge-side (Compute1) and cloud-side (Compute2) components. To ensure precise time synchronization, we consider communication latency as the residual difference.

Figure 6 illustrates the breakdown of latency. Based on these results, several key insights can be drawn:

- ① Lambda maintains lower compute latency than EC2 in the same region. With 4 GB memory, Lambda provides slightly more computing power than a t3.medium EC2 instance, leading to faster inference time;
- ② Lambda, however, has the highest communication latency. The network overhead to invoke and receive results from Lambda introduces noticeable delays;
- ③ EC2 in the Wavelength Zone performs better than the Availability Zone. Despite identical configurations, variations in hardware and resource availability impacts the performance, and edge-local deployment yields better efficiency;
- ④ Early Exiting increases edge compute time but reduces cloud compute load. Due to the added branching logic and multiple sub-models, EE introduces extra processing at the edge. However, samples that exit early avoid cloud computation altogether, lowering the overall latency.

Table 2: EC2 Configuration and Prices [4]

Instance	EC2 in AZ	EC2 in WZ
Instance Type	t3.medium	t3.medium
Location	Frankfurt	Munich
Tenancy Type	Shared Instances	Shared Instances
Operating System	Linux	Linux
Base Price	\$0.048/hour	\$0.060/hour
EBS Price	\$0.119/GB-month	\$0.155/GB-month

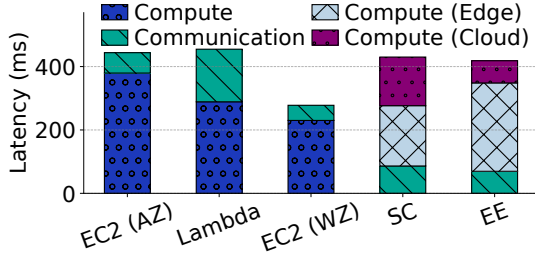


Figure 6: Latency Breakdown.

3.3 Operational Cost

From a practical standpoint, even during periods of low or no traffic, an EC2 instance remains active to handle potential incoming requests, which incurs ongoing charges. Therefore, rather than evaluating cost on a per-request basis, we estimate costs over a longer period, such as daily or monthly.

Table 2 and Table 3 summarize the key pricing factors for EC2 and Lambda, along with their corresponding configurations and associated costs. Based on these configurations, we estimate the monthly cost of each deployment strategy. In addition, using the per-request compute time discussed in §3.2, we approximate the maximum number of requests that each strategy can support per month. Figure 7 presents the average monthly cost across different deployment strategies. For Lambda, the cost increases linearly with request volume, making it more economical under low traffic. The intersection between Lambda’s cost curve and those of other strategies indicates the break-even point, beyond which EC2-based options become more cost-efficient due to better resource utilization. Although EC2 instances in the Availability Zone are cheaper than those in the Wavelength Zone, their slower compute times limit their maximum request handling capacity. For hybrid architectures, where both edge and cloud EC2 instances are used, the total cost includes both components and is constrained by the slower node. In our tests, Early Exiting shows higher edge compute time, slightly reducing its capacity compared to Split Computing. It is worth noting that all our evaluation is based on on-demand pricing in a test environment. In real production settings, AWS offers reserved instances and saving plans that can reduce the cost.

3.4 Accuracy

This section examines how different deployment strategies influence the model’s prediction accuracy. For strategies where the full model is deployed either in the cloud or at the edge, prediction accuracy remains unchanged, as the model architecture and weights are preserved. Similarly, in Split Computing, although the model is

Table 3: Lambda Pricing Configuration [4]

Cost Component	Price
Requests	\$0.20 per 1M requests
Memory Duration	\$0.004002 per min (4096 MB)
Ephemeral Storage	\$0.052848 per min (4096 MB)

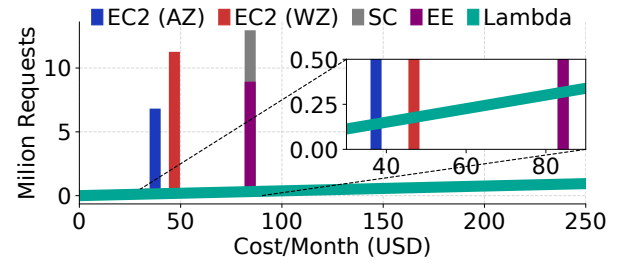


Figure 7: Cost Comparison of Deployment Strategies.

divided between edge and cloud, the full architecture is retained, so accuracy is unaffected. In contrast, Early Exiting introduces intermediate classifiers that allow earlier responses for some inputs based on confidence thresholds. While this reduces latency, it may affect accuracy depending on the placement of exit points, branch design, and threshold settings. As shown in Table 4, EE distributes predictions across three stages: 20% exit early at branch 1, 36% at branch 2, and 44% at the final output. Gender prediction accuracy remains stable at 85%, but age prediction accuracy drops slightly from 61% to 57%.

Figure 8 visualizes the classification result based on input difficulty. In this context, *easy* refers to images that the model is confident at EE branch 1 to produce a prediction; *medium* indicates sufficient confidence at EE branch 2; and *hard* denotes cases where inference must be forwarded to the cloud for full computation. From the results, we observe that most errors in the Early Exiting model occur in the 60+ age group and images of children are frequently routed to the final stage, often due to overly confident early predictions that bypass deeper inference. Additionally, images of children are frequently routed to the final stage, indicating persistent prediction challenges in that age group.

These findings underscore that certain features of the multi-task CNN, such as gender, do not require a deep architecture and can be effectively predicted through early exiting, whereas age estimation benefits from deeper layers for improved accuracy, particularly for edge age groups such as children and elderly individuals.

4 Related Work

Several studies have explored deploying AI applications on cloud platforms such as AWS, leveraging services like EC2, Lambda, and SageMaker to evaluate performance [10] [18], but they lack detailed quantitative experimental analysis.

Table 4: Prediction Accuracy (%) and Exit Rates Across Different Deployment Strategies

Deployment Strategy	Gender	Age	Exit Rate
EC2 (AZ)	85	61	100
EC2 (WZ)	85	61	100
Split Computing	85	61	0, 100
Early Exiting	85	57	20, 36, 44
Lambda	85	61	100

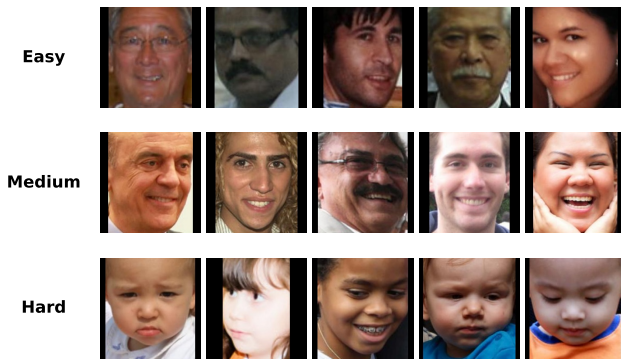


Figure 8: Difficulty-Based Prediction Path Distribution in the EE-Enhanced CNN Model.

Doukha et al. [23] conduct an empirical study on training several CNN architectures on Amazon EC2 instances, identifying scenarios in which upgrading to more powerful GPU instances leads to better performance-cost efficiency, particularly when GPU usage exceeds 90%. Chahal et al. [20] present a comparative analysis of AWS SageMaker and AWS Lambda for deploying a deep learning-based recommender system. Their evaluation focuses on response time and cost efficiency under varying levels of concurrency. Roy et al. [45] developed a real-time fruit recognition and calorie estimation system using a 5-layer CNN model, deployed on AWS EC2 using a Flask-based web service. To our knowledge, these studies primarily focus on cloud-only deployments and do not consider edge or hybrid architectures.

Rao et al. [44] introduce the ECO (Edge-Cloud Optimization) system, which can map services between edge and cloud environments utilizing AWS/Verizon Wavelength edge infrastructure, ECO achieves a twofold improvement in application response time compared to static edge-cloud deployments. [43] introduces ECO-LLM, a hybrid edge-cloud optimization framework using large language models to automate microservice deployment based on natural language inputs. Ogden et al. [39] proposes the Layercake system, a hybrid inference architecture combining mobile devices with AWS EC2 cloud resources. It reduces latency by 20–45% and achieves 30–60% cost savings compared to cloud-only approaches, based on single-request evaluations rather than long-term workloads.

Collectively, prior literature underscores the growing importance of cloud and edge platforms supporting cooperation for scalable, efficient, and low-latency deployment of deep learning models. Nevertheless, comprehensive investigations involving application deployment and empirical analysis on AWS remain relatively limited. We aim to address this lack by presenting a comparative evaluation of diverse deployment strategies, including both edge and cloud setup along with serverless and server-based computing model. Furthermore, we incorporate advanced dynamic serving mechanisms such as SC and EE.

5 Conclusion and Future Work

We presented a comprehensive performance and cost analysis of deploying a computer vision application across the edge-cloud continuum. We designed, implemented, and evaluated four distinct

strategies on the AWS platform, spanning cloud-only, edge-only, and hybrid architectures. The experiments included server-based (EC2) and serverless (Lambda) compute models. Furthermore, we integrated advanced techniques like Split Computing (SC) and Early Exiting (EE) in hybrid deployments to explore adaptive inference.

We observed trade-offs between latency, cost, and accuracy. Edge-native deployments (EC2 in Wavelength Zone) deliver the lowest latency, making them ideal for real-time applications, albeit at a higher operational cost. Cloud-only deployments provide a cost-effective baseline for latency-tolerant services. Serverless computing via Lambda proves most economical for applications with sporadic or low traffic, but its performance is hampered by communication overhead and potential cold starts. Hybrid strategies, particularly Early Exiting, offer a balanced approach by reducing median latency and partially offloading computation to the cloud, with only a minor trade-off in accuracy. Our analysis provides practical guidance for architects and developers in selecting the optimal deployment strategy based on their specific application requirements and budget constraints.

Future Directions. We plan to extend our study in two main directions. First, we aim to explore more recent and larger computer vision models such as Vision Transformers (ViTs). ViTs can achieve performance comparable to or even surpassing that of CNN-based models. However, ViTs generally require significantly higher computational resources than conventional CNNs, which makes them particularly suitable for edge-cloud collaborative computing. For instance, [34] proposed StressViT, a framework that splits and compresses visual Transformers through edge-cloud collaboration, achieving nearly a 50% reduction in end-to-end latency with almost no loss in accuracy. In addition, we will explore more dynamic offloading policies that can switch between deployment strategies in real-time based on network conditions, server load, and application-level quality-of-service requirements.

Acknowledgments

We thank the anonymous reviewers for their helpful comments. This work was supported by the Federal Ministry of Research, Technology and Space of Germany (BMFTR) in the programme of “Souverän. Digital. Vernetzt.” joint project 6G-life, project identification number 16KISK002 and EU Health and Digital Executive Agency (HADEA) program under Grant Agreement No 101092950 (EDGELESS project). We would also like to thank IAV Fahrzeugsicherheit GmbH & Co. KG for their support and collaboration in this research.

References

- [1] Zarif Bin Akhtar. 2024. Unveiling the evolution of generative AI (GAI): a comprehensive and investigative analysis toward LLM models (2021–2024) and beyond. *Journal of Electrical Systems and Information Technology* 11, 1 (2024), 22.
- [2] Amazon Web Services. 2024. Provisioned Concurrency for AWS Lambda. <https://docs.aws.amazon.com/lambda/latest/dg/provisioned-concurrency.html>. Accessed: 2025-06-15.
- [3] Amazon Web Services. 2025. AWS AI Services. <https://aws.amazon.com/ai/services/>. Accessed: 2025-08-02.
- [4] Amazon Web Services. 2025. AWS Pricing. <https://aws.amazon.com/pricing/>. Accessed: 2025-03-30.
- [5] Amazon Web Services. 2025. AWS Wavelength. <https://aws.amazon.com/wavelength/>. Accessed: 2025-06-15.
- [6] Navidreza Asadi, Halil Ibrahim Bengü, Lars Wulfert, Hendrik Wöhrle, and Wolfgang Kellerer. 2025. Gist - Optimizing Segmentation for Decentralized Federated Learning on Tiny Devices. In *Federated Learning and Edge AI for Privacy and*

- Mobility (FLEdge-AI '25) (FLEdge-AI '25). Association for Computing Machinery, New York, NY, USA. doi:10.1145/3737899.3768527
- [7] Navidreza Asadi, Halil Ibrahim Bengü, Lars Wulfert, Hendrik Wöhrle, and Wolfgang Kellerer. 2025. Poster: Road to Tiny Reality: Digital Twins for Decentralized AI on Microcontrollers. In *Proceedings of the 31st Annual International Conference on Mobile Computing and Networking (MOBICOM '25)* (Hong Kong, China). ACM, New York, NY, USA, 3. doi:10.1145/3680207.3765668
 - [8] Navidreza Asadi and Maziar Goudarzi. 2023. An ensemble mobile-cloud computing method for affordable and accurate glucometer readout. *arXiv preprint arXiv:2301.01758* (2023).
 - [9] Navidreza Asadi and Maziar Goudarzi. 2023. Variant parallelism: lightweight deep convolutional models for distributed inference on IoT devices. *IEEE Internet of Things Journal* 11, 1 (2023), 345–352.
 - [10] Rahul Bagai. 2024. Comparative Analysis of AWS Model Deployment Services. *International Journal of Computer Trends and Technology* 72, 5 (May 2024), 102–110. doi:10.14445/22312803/ijctt-v72i5p113
 - [11] Altynzer Baiganova, Zhanar Ubayeva, Zhanar Taskalyeva, Lezzat Kaparova, Roza Nurzhaubaeva, and Banu Umirzakova. 2025. Automated Defect Detection in Manufacturing Using Enhanced VGG16 Convolutional Neural Networks. *International Journal of Advanced Computer Science & Applications* 16, 4 (2025).
 - [12] Arian Bakhtiarnia, Nemanja Milošević, Qi Zhang, Dragana Bajović, and Alexandros Iosifidis. 2023. Dynamic split computing for efficient deep edge intelligence. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 1–5.
 - [13] Colby Banbury, Chuteng Zhou, Igor Fedorov, Ramon Matas, Urmish Thakker, Dibakar Gope, Vijay Janapa Reddi, Matthew Mattina, and Paul Whatmough. 2021. Micronets: Neural network architectures for deploying tinyml applications on commodity microcontrollers. *Proceedings of machine learning and systems* 3 (2021), 517–532.
 - [14] Giovanni Bartolomeo, Navidreza Asadi, Wolfgang Kellerer, Jorg Ott, and Nitinder Mohan. 2025. {On-Demand} Container Partitioning for Distributed {ML}. In *2025 USENIX Annual Technical Conference (USENIX ATC 25)*. 1481–1500.
 - [15] Giovanni Bartolomeo, Jacky Cao, Xiang Su, and Nitinder Mohan. 2023. Characterizing distributed mobile augmented reality applications at the edge. In *Companion of the 19th International Conference on emerging Networking EXperiments and Technologies*. 9–18.
 - [16] Giovanni Bartolomeo, Patrick Sabanic, Nitinder Mohan, and Jorg Ott. 2025. Supporting Hybrid Virtualization Orchestration for Edge Computing. In *Proceedings of the 8th International Workshop on Edge Systems, Analytics and Networking*. 19–24.
 - [17] Giovanni Bartolomeo, Mehdi Yosofie, Simon Bäurle, Oliver Haluszczynski, Nitinder Mohan, and Jörg Ott. 2023. Oakestra: A lightweight hierarchical orchestration framework for edge computing. In *2023 USENIX Annual Technical Conference (USENIX ATC 23)*. 215–231.
 - [18] Didar Bayazitov, Kanat Kozhakhmet, Aikumis Omirali, and Rakhima Zhumaliyeva. 2024. Leveraging Amazon Web Services for cloud storage and AI algorithm integration: A comprehensive analysis. *Applied Mathematics* 18, 6 (2024), 1235–1246.
 - [19] Romil Bhardwaj, Zhengxu Xia, Ganesh Ananthanarayanan, Junchen Jiang, Yuan-chao Shu, Nikolaos Karianakis, Kevin Hsieh, Paramvir Bahl, and Ion Stoica. 2022. Ekyka: Continuous learning of video analytics models on edge compute servers. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*. 119–135.
 - [20] Dheeraj Chahal, Mayank Mishra, Surya Palepu, and Rekha Singhal. 2021. Performance and Cost Comparison of Cloud Services for Deep Learning Workload. In *Companion of the ACM/SPEC International Conference on Performance Engineering (Virtual Event, France) (ICPE '21)*. Association for Computing Machinery, New York, NY, USA, 49–55. doi:10.1145/3447545.3451184
 - [21] Ka-Ho Chow, Umesh Deshpande, Veera Deenadayalan, Sangeetha Seshadri, and Ling Liu. 2024. Atlas: Hybrid cloud migration advisor for interactive microservices. In *Proceedings of the Nineteenth European Conference on Computer Systems*. 870–887.
 - [22] Yinwei Dai, Rui Pan, Anand Iyer, Kai Li, and Ravi Netravali. 2024. Apparate: Rethinking early exits to tame latency-throughput tensions in ml serving. In *Proceedings of the ACM SIGOPS 30th Symposium on Operating Systems Principles*. 607–623.
 - [23] Rim Doukha, Sidi Ahmed Mahmoudi, Mostapha Zbakh, and Pierre Manneback. 2022. A comparative analysis of Convolution Neural Network models on Amazon Cloud Service. In *AIIPEC 2022: The Third International Conference on Artificial Intelligence, Information Processing and Cloud Computing*. 1–7.
 - [24] Divya Garikapati and Sneha Sudhir Shetiya. 2024. Autonomous vehicles: Evolution of artificial intelligence and the current industry landscape. *Big Data and Cognitive Computing* 8, 4 (2024), 42.
 - [25] Google Cloud. 2025. Vertex AI Platform. <https://cloud.google.com/vertex-ai>. Accessed: 2025-08-02.
 - [26] Maryam Hashemi, Alireza Mirrashid, and Aliasghar Beheshti Shirazi. 2020. Driver safety development: Real-time driver drowsiness detection system based on convolutional neural network. *SN Computer Science* 1, 5 (2020), 289.
 - [27] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. 2019. Searching for mobilenetv3. In *Proceedings of the IEEE/CVF international conference on computer vision*. 1314–1324.
 - [28] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. 2017. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *CoRR abs/1704.04861* (2017). arXiv:1704.04861 <http://arxiv.org/abs/1704.04861>
 - [29] Hossein Katebi, Navidreza Asadi, and Maziar Goudarzi. 2024. FullPack: Full Vector Utilization for Sub-Byte Quantized Matrix-Vector Multiplication on General Purpose CPUs. *IEEE Computer Architecture Letters* 23, 2 (2024), 142–145.
 - [30] Taranjit Kaur and Tapan Kumar Gandhi. 2019. Automated brain image classification based on VGG-16 and transfer learning. In *2019 international conference on information technology (ICIT)*. IEEE, 94–98.
 - [31] Stefanos Laskaridis, Alexandros Kouris, and Nicholas D Lane. 2021. Adaptive inference through early-exit networks: Design, challenges and directions. In *Proceedings of the 5th International Workshop on Embedded and Mobile Deep Learning*. 1–6.
 - [32] Patrick Laufer and Clemens Gühmann. 2025. Vehicle Occupant Age and Gender Estimation Using Convolutional Neural Networks on RGB and NIR Images. In *2025 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE, 1–8. Accepted for publication.
 - [33] Yunsheng Li, Yinpeng Chen, Xiyang Dai, Dongdong Chen, Mengchen Liu, Lu Yuan, Zicheng Liu, Lei Zhang, and Nuno Vasconcelos. 2021. Micronet: Improving image recognition with extremely low flops. In *Proceedings of the IEEE/CVF International conference on computer vision*. 468–477.
 - [34] Changyao Lin, Yi Liu, Xiangyu Li, Chengxiang Li, Hao Zhang, Jing Jin, and Jie Liu. 2025. StressViT: Splitting and Compressing Vision Transformer through Edge-Cloud Collaboration. In *Pattern Recognition (ICPR 2025)*. Springer Nature Switzerland, 406–422.
 - [35] Andrés Marafioti, Orr Zohar, Miquel Farré, Merve Noyan, Elie Bakouch, Pedro Cuenca, Cyril Zakka, Loubna Ben Allal, Anton Lozhkov, Nouamane Tazi, Vaibhav Srivastav, Joshua Lochner, Hugo Larcher, Mathieu Morlon, Lewis Tunstall, Leandro von Werra, and Thomas Wolf. 2025. SmolVLM: Redefining small and efficient multimodal models. *arXiv preprint arXiv:2504.05299* (2025).
 - [36] Yoshitomo Matsubara, Marco Levorato, and Francesco Restuccia. 2022. Split computing and early exiting for deep learning applications: Survey and research challenges. *Comput. Surveys* 55, 5 (2022), 1–30.
 - [37] Diego González Morín, Pablo Pérez, and Ana García Armada. 2022. Toward the distributed implementation of immersive augmented reality architectures on 5G networks. *IEEE Communications Magazine* 60, 2 (2022), 46–52.
 - [38] Abdullah Hafez Nur, Md Simul Hasan Talukder, Shihab Adnan, and Md Razu Ahmed. 2024. A transfer learning approach with modified VGG 16 for driving behavior detection in intelligent transportation systems. In *2024 6th International Conference on Electrical Engineering and Information & Communication Technology (ICEEICT)*. IEEE, 1060–1065.
 - [39] Samuel S. Ogden and Tian Guo. 2023. Layercake: Efficient Inference Serving with Cloud and Mobile Resources. In *2023 IEEE/ACM 23rd International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*. 191–202. doi:10.1109/CCGrid57682.2023.00027
 - [40] Seyed Ali Osia, Ali Shahin Shamsabadi, Sina Sajadmanesh, Ali Taheri, Kleomenis Katevas, Hamid R Rabiee, Nicholas D Lane, and Hamed Haddadi. 2020. A hybrid deep learning architecture for privacy-preserving mobile analytics. *IEEE Internet of Things Journal* 7, 5 (2020), 4505–4518.
 - [41] Anjali Parayil, Jue Zhang, Xiaoting Qin, Íñigo Goiri, Lexiang Huang, Timothy Zhu, and Chetan Bansal. 2025. Towards Workload-aware Cloud Efficiency: A Large-scale Empirical Study of Cloud Workload Characteristics. In *Proceedings of the 16th ACM/SPEC International Conference on Performance Engineering*. 136–146.
 - [42] Kiran Sree Pokkuluri, Manthena Sivanjani, M Prasad, Raja Rao PBV, P JR Shalem Raju, and Anuj Rapaka. 2025. Deep Learning-Based Detection of Traffic Accidents Using CNN and VGG16 on Accident and Foggy Image Datasets. In *2025 3rd International Conference on Intelligent Systems, Advanced Computing and Communication (ISACC)*. IEEE, 406–411.
 - [43] Kunal Rao, Giuseppe Coviello, Priscilla Benedetti, Ciro Giuseppe De Vita, Genaro Mellone, and Srmat Chakradhar. 2024. Eco-llm: Llm-based edge cloud optimization. In *Proceedings of the 2024 Workshop on AI For Systems*. 7–12.
 - [44] Kunal Rao, Giuseppe Coviello, Wang-Pin Hsiung, and Srmat Chakradhar. 2021. Eco: Edge-Cloud Optimization of 5G applications. In *2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*. 649–659. doi:10.1109/CCGrid51090.2021.00078
 - [45] Vikas Kumar Roy, Ganpati Kumar Roy, Vasu Thakur, Nikhil Baliyan, Nupur Goyal, and Yash Chauhan. 2023. Real Time Fruit Recognition and Calories Estimation Using CNN and Model Deployment on Amazon's AWS. In *2023 3rd International Conference on Innovative Sustainable Computational Technologies (CISCT)*. 1–6. doi:10.1109/CISCT57197.2023.10351261

- [46] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. 2018. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 4510–4520.
- [47] Megha Sharma, Abhinav Tomar, and Abhishek Hazra. 2024. Edge computing for industry 5.0: Fundamental, applications, and research challenges. *IEEE Internet of Things Journal* 11, 11 (2024), 19070–19093.
- [48] Sudipta Saha Shubha and Haiying Shen. 2023. Adainf: Data drift adaptive scheduling for accurate and slo-guaranteed multiple-model inference serving at edge servers. In *Proceedings of the ACM SIGCOMM 2023 Conference*. 473–485.
- [49] Sudipta Saha Shubha, Haiying Shen, and Anand Iyer. 2024. {USHER}: Holistic interference avoidance for resource optimized {ML} inference. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*. 947–964.
- [50] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).
- [51] Srikanth Tammina et al. 2019. Transfer learning using vgg-16 with deep convolutional neural network for classifying images. *International Journal of Scientific and Research Publications (IJSRP)* 9, 10 (2019), 143–150.
- [52] Maria Tariq, Usman Ali, Sagheer Abbas, Shahzad Hassan, Rizwan Ali Naqvi, Muhammad Adnan Khan, and Daesik Jeong. 2024. Corn leaf disease: insightful diagnosis using VGG16 empowered by explainable AI. *Frontiers in plant science* 15 (2024), 1402835.
- [53] Yunjie Tian, Qixiang Ye, and David Doermann. 2025. YOLOv12: Attention-Centric Real-Time Object Detectors. *arXiv preprint arXiv:2502.12524* (2025).
- [54] Wang Tong, Azhar Hussain, Wang Xi Bo, and Sabita Maharjan. 2019. Artificial intelligence for vehicle-to-everything: A survey. *IEEE Access* 7 (2019), 10823–10843.
- [55] Xiaoguang Tu, Zhi He, Yi Huang, Zhi-Hao Zhang, Ming Yang, and Jian Zhao. 2024. An overview of large AI models and their applications. *Visual Intelligence* 2, 1 (2024), 34.
- [56] Vodafone. 2025. Distributed Multi-Access Edge Computing (MEC) with AWS. https://www.vodafone.de/business/loesungen/aws-mec/?c_id=content_20emm_737:fq0_B_bxx_sta_Teaser-Portal-AWS-MEC. Accessed: 2025-06-15.
- [57] Lars Wulfert, Navidreza Asadi, Wen-Yu Chung, Christian Wiede, and Anton Grabmaier. 2023. Adaptive decentralized federated gossip learning for resource-constrained iot devices. In *Proceedings of the 4th International Workshop on Distributed Machine Learning*. 27–33.
- [58] Zhifei Zhang, Yang Song, and Hairong Qi. 2017. Age progression/regression by conditional adversarial autoencoder. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 5810–5818.