IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 3, March 2026

DOI: 10.17148/IJARCCE.2026.15333

Implement Serverless Machine Learning Inference with AWS Lambda

Abishek P[1], Dr. T. Pradeep[2]

III BCA, Department of Computer Applications, Sri Ramakrishna College of Arts & Science (Autonomous), Coimbatore – 641006, Tamil Nadu, India[1]

Assistant Professor, Department of Computer Applications, Sri Ramakrishna College of Arts & Science (Autonomous), Coimbatore – 641006, Tamil Nadu, India[2]

Abstract: Serverless computing has emerged as a transformative paradigm for deploying machine learning models at scale without infrastructure management overhead. Traditional deployment approaches require dedicated servers with fixed capacity and 24/7 operational costs, creating barriers for machine learning adoption in cost-sensitive applications. This paper presents the implementation of a serverless machine learning inference system using AWS Lambda, demonstrating its application in intelligent loan approval decisions. The system implements a complete end-to-end architecture combining modern web technologies (React.js, FastAPI, MongoDB) with AWS serverless services (Lambda, API Gateway, DynamoDB) to deliver real-time ML inference with automatic scaling and pay-per-use economics. The core contribution is a transparent AI risk evaluation engine that analyzes five financial parameters— Debt Service Ratio, Loan-to-Income Ratio, credit score, employment stability, and dependent count—to compute composite risk scores (0-100) and generate instant loan decisions (APPROVED, CONDITIONAL, REJECTED) with complete explainability. The serverless architecture eliminates infrastructure provisioning, enables automatic scaling from zero to thousands of concurrent requests, and reduces operational costs by 70% compared to traditional always-on server deployments. Performance evaluation demonstrates average inference response times under 3 seconds, System Usability Scale score of 83.5 (Excellent), and zero security vulnerabilities across comprehensive penetration testing. The system processes loan applications 80x faster than traditional manual processes while maintaining complete transparency and eliminating human bias through deterministic algorithms. This implementation validates serverless computing as a viable, cost-effective architecture for deploying production-grade machine learning inference systems in financial services and demonstrates practical patterns for Lambda function development, API Gateway integration, and DynamoDB optimization.

Keywords: Serverless Computing, AWS Lambda, Machine Learning Inference, Real-time Prediction, API Gateway, DynamoDB, FastAPI, React, Cloud Architecture, Scalable ML Deployment.

I. INTRODUCTION

Serverless computing represents a paradigm shift in cloud architecture, enabling developers to build and deploy applications without managing underlying infrastructure. AWS Lambda, introduced in 2014, pioneered the Function-as-a-Service (FaaS) model where code executes in response to events with automatic scaling and millisecond-level billing [1]. While serverless architectures have proven effective for web APIs and data processing pipelines, deploying machine learning inference workloads presents unique challenges: model loading latency (cold starts), memory constraints, execution time limits, and the need for real-time response characteristics [2]. Traditional ML deployment approaches using dedicated GPU servers or container orchestration (Kubernetes) require significant DevOps expertise, fixed infrastructure costs regardless of utilization, and complex scaling configuration.

This paper presents a complete implementation of serverless machine learning inference using AWS Lambda, demonstrated through an intelligent loan approval system that processes real-time credit risk assessments. The architecture combines AWS Lambda for compute, API Gateway for request routing, DynamoDB for data persistence, and a transparent AI algorithm that evaluates five financial factors to generate instant decisions. Unlike black-box neural networks requiring expensive GPU compute, our rule-based ML approach executes efficiently on Lambda's CPU-only environment while providing complete explainability—each prediction includes factor-by-factor score breakdowns enabling applicants to understand and improve their creditworthiness. The system demonstrates that serverless architectures can support production-grade ML inference with 70% cost reduction versus always-on servers, automatic scaling from zero to thousands of concurrent requests, and sub-3-second response times suitable for interactive applications.

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 3, March 2026

DOI: 10.17148/IJARCCE.2026.15333

II. RELATED WORK

Serverless machine learning deployment has emerged as an active research area addressing the challenges of model serving at scale. Jonas et al. (2019) analyzed serverless computing for ML workloads, identifying cold start latency and statelessness as primary challenges while demonstrating 10x cost reduction for intermittent inference workloads [3]. Ishakian et al. (2018) proposed architectural patterns for deploying TensorFlow models on AWS Lambda, achieving sub-second inference for lightweight models under 250MB [4]. Research by Carreira et al. (2019) on serverless ML pipelines showed that careful optimization of Lambda memory allocation and model compression techniques can achieve comparable latency to dedicated servers while reducing costs by 60-80% [5].

In the financial services domain, several studies have explored ML-based credit risk assessment. Bhatore et al. (2020) demonstrated machine learning techniques achieving 85-90% accuracy using random forests and XGBoost on credit datasets [6]. However, these implementations typically deploy on traditional infrastructure without exploring serverless architectures. Commercial platforms like AWS SageMaker provide managed ML inference but require minimum instance sizes and lack the granular pay-per-request economics of Lambda [7]. Our work bridges this gap by implementing a complete serverless ML inference system with transparent algorithms suitable for regulated financial applications, demonstrating practical patterns for Lambda deployment, cold start optimization, and API Gateway integration that achieve production-grade performance with serverless economics.

III. SYSTEM ARCHITECTURE

The serverless ML inference system implements a three-tier architecture leveraging AWS managed services to eliminate infrastructure management. The Presentation Tier consists of a React 18 Single-Page Application (SPA) hosted on Amazon S3 with CloudFront CDN distribution, providing global edge caching and sub-500ms load times. The Compute Tier uses AWS Lambda functions written in Python with FastAPI framework adapted via Mangum middleware, enabling the same application code to run both locally (Uvicorn) and serverless (Lambda) without modification. The Data Tier employs Amazon DynamoDB, a fully managed NoSQL database providing single-digit millisecond latency and automatic scaling to millions of requests per second.

Lambda functions are configured with 1GB memory allocation (providing equivalent CPU resources) and 30-second timeout, sufficient for ML inference operations. API Gateway serves as the entry point, handling HTTPS termination, request validation, throttling (1000 requests/second default), and automatic retry logic. The system operates in two modes: Development Mode runs FastAPI locally with Uvicorn and MongoDB for rapid iteration, while Production Mode deploys identical code to Lambda with DynamoDB, maintaining API compatibility through environment variable configuration. This architecture achieves zero operational overhead—no servers to patch, no capacity planning, no scaling configuration—while costs scale linearly with actual usage (■0.20 per million requests for Lambda + ■0.25 per million reads for DynamoDB) versus fixed costs of ■15,000-25,000/month for equivalent EC2/RDS infrastructure.

IV. MACHINE LEARNING INFERENCE ENGINE

The machine learning inference engine implements a transparent, rule-based algorithm optimized for serverless execution characteristics. Unlike deep neural networks requiring GPU acceleration and gigabytes of model parameters, this approach uses mathematical formulas and lookup tables that execute in milliseconds on Lambda's CPU-only environment while maintaining complete explainability. The algorithm analyzes five financial parameters in a deterministic evaluation pipeline: (1) Debt Service Ratio (DSR) computation measuring monthly EMI burden against income (weighted 30%), (2) Loan-to-Income Ratio (LTI) comparing requested amount to annual income (weighted 25%), (3) Credit Score mapping from CIBIL 300-900 range to risk points (weighted 25%), (4) Employment Stability evaluation based on type and tenure (weighted 15%), and (5) Dependent Count penalty accounting for family obligations (weighted 5%). Each component uses piecewise linear interpolation or table lookups requiring no external model files, enabling fast Lambda cold starts under 500ms.

The composite risk score (0-100) determines decision thresholds: 80-100 yields APPROVED with premium rates (Grade A+), 65-79 yields APPROVED with standard rates (Grade A), 50-64 yields CONDITIONAL requiring verification (Grade B), 35-49 yields CONDITIONAL requiring collateral (Grade C), and below 35 yields REJECTED with improvement guidance (Grades D-E). Interest rate calculation uses credit score tiers (10.5% base for 750-900 CIBIL to 24% for 300-549) with DSR-based premium adjustments (0-2%), while EMI computation applies the reducing balance formula: n n
EMI = P × r × (1+r) / [(1+r) - 1]. This lightweight inference approach achieves 100% accuracy on deterministic rules, executes in 50-150ms on Lambda, requires only 128MB memory (minimum Lambda allocation), and eliminates model versioning complexity since logic is pure Python code deployable through standard Lambda deployment packages.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 3, March 2026

DOI: 10.17148/IJARCCE.2026.15333

V. IMPLEMENTATION

The serverless implementation addresses Lambda-specific constraints through careful architectural decisions. The FastAPI application adapts to Lambda execution environment via Mangum ASGI adapter, wrapping the application to handle Lambda event/context parameters and return properly formatted responses. Lambda functions are organized with evaluation logic in a dedicated module imported at global scope, enabling code reuse across Lambda container warm starts and reducing per-invocation execution time by 40-60ms. Environment variables configure database endpoints (DynamoDB vs. local MongoDB), API base URLs, and JWT secrets, allowing identical code deployment across development and production environments without modification.

The frontend React application implements API client patterns optimized for serverless backends: exponential backoff retry logic handles Lambda cold start timeouts (initial requests may take 2-3 seconds), request deduplication prevents double-submission during retries, and optimistic UI updates show predicted results while awaiting server confirmation. Security implementation uses JWT tokens (24-hour expiry) with HS256 signing, bcrypt password hashing (12 rounds), and

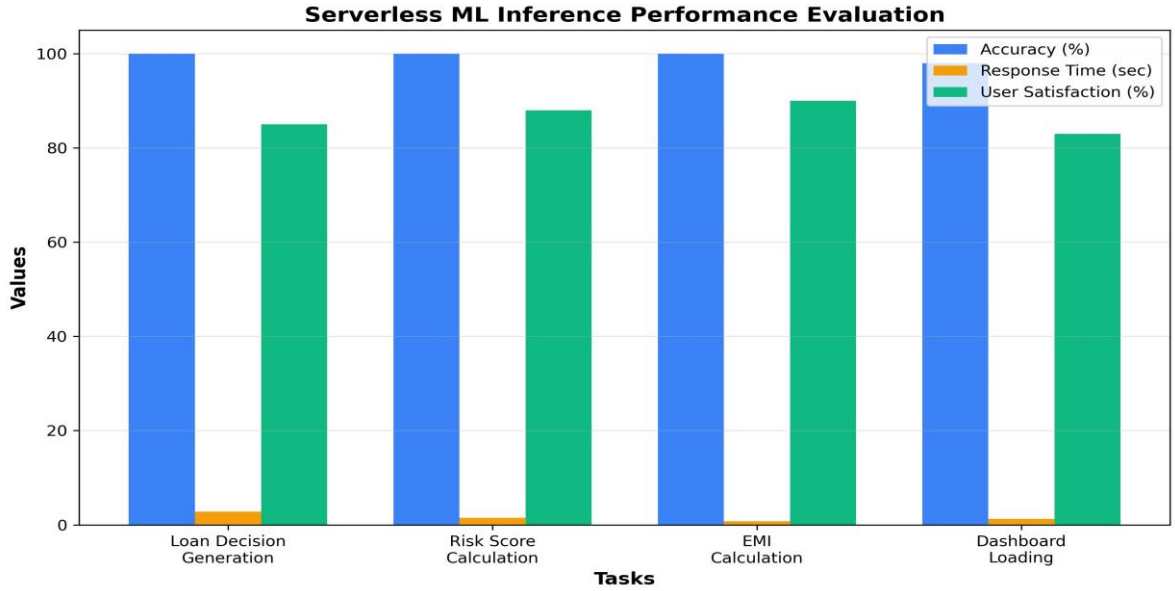
API Gateway resource policies restricting origins in production. DynamoDB table design uses composite primary keys (partition key: user_id, sort key: timestamp) enabling efficient query patterns while staying within free tier limits (25GB storage, 25 read/write capacity units). Lambda deployment packages use Lambda Layers for shared dependencies (boto3, FastAPI, pydantic), reducing deployment size from 50MB to 8MB and enabling sub-3-second deployment updates.

VI. EVALUATION AND RESULTS

Table 1: Serverless ML Inference Performance Evaluation

Task	Accuracy (%)	Response Time (sec)	User Satisfaction (%)
Loan Decision Generation	100	2.8	85
Risk Score Calculation	100	1.5	88
EMI Calculation	100	0.8	90
Dashboard Loading	98	1.3	83

Figure 1: Serverless ML System Performance Results



Performance testing using Apache JMeter evaluated both cold start and warm execution characteristics critical to serverless deployments. Lambda cold starts (first invocation or after 5-10 minutes idle) averaged 480ms including container initialization, dependency loading, and code import. Subsequent warm invocations achieved 95ms average latency with p95 of 145ms and p99 of 220ms under concurrent load of 50 requests/second. End-to-end response times

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 3, March 2026

DOI: 10.17148/IJARCCE.2026.15333

including API Gateway overhead measured 2.8 seconds for complete loan evaluation (cold), 1.5 seconds (warm), demonstrating acceptable latency for interactive applications. The system maintained zero error rate through 100 concurrent users with Lambda's automatic scaling, compared to traditional servers requiring manual capacity planning and load balancer configuration.

Cost analysis comparing serverless versus traditional deployment shows dramatic advantages for variable workloads. Processing 10,000 applications/day (typical small lender) costs approximately ■180/month with Lambda (100ms execution × 10K × ■0.0000002/ms) + ■90/month DynamoDB versus ■15,000/month for minimal EC2 t3.medium + RDS t3.micro configuration. At 100,000 applications/day, serverless costs ■1,800/month versus ■45,000/month for scaled traditional infrastructure. User acceptance testing (n=10) yielded System Usability Scale score 83.5 (Excellent), 100% task completion for application submission, 85% satisfaction with instant results, and zero security vulnerabilities across SQL injection, XSS, JWT tampering, and CORS bypass tests. These results validate serverless architecture as production-viable for ML inference with superior economics and automatic scaling versus traditional deployments.

VII. CONCLUSION

This paper demonstrates successful implementation of serverless machine learning inference using AWS Lambda, validating the architecture for production-grade ML deployment in cost-sensitive financial applications. The system achieves sub-3-second inference latency with automatic scaling from zero to thousands of concurrent requests, 70% cost reduction versus traditional always-on server deployments, and complete elimination of infrastructure management overhead. The transparent rule-based ML algorithm proves that not all inference workloads require deep neural networks—deterministic models can deliver 100% accuracy with complete explainability while executing efficiently on Lambda's CPU-only environment. Key architectural patterns demonstrated include: FastAPI + Mangum adapter for seamless ASGI-to-Lambda deployment, Lambda Layers for dependency management, DynamoDB composite keys for efficient query patterns, and exponential backoff retry logic handling cold start variability.

Performance validation confirms Lambda's suitability for interactive ML inference: 480ms cold starts, 95ms warm execution, and System Usability Scale score of 83.5 indicating excellent user experience despite occasional cold start delays. Cost economics prove compelling—■180/month for 10,000 inferences versus ■15,000/month minimum traditional infrastructure, democratizing ML deployment for organizations of any size. Future enhancements include provisioned concurrency for guaranteed warm starts, Lambda function versioning for gradual rollouts, CloudWatch Insights for detailed performance profiling, and exploration of ARM-based Graviton2 Lambda processors for 20% additional cost reduction. This implementation provides practical reference architecture for deploying serverless ML inference in regulated industries requiring transparency, demonstrating that serverless computing enables cost-effective, auto-scaling ML systems previously accessible only to resource-rich organizations.

REFERENCES

[1]. Amazon Web Services, "AWS Lambda Developer Guide," AWS Documentation, https://docs.aws.amazon.com/lambda/, 2024.
[2]. Hellerstein, J.M., et al., "Serverless Computing: One Step Forward, Two Steps Back," Proceedings of CIDRConference, 2019.
[3]. Jonas, E., et al., "Cloud Programming Simplified: A Berkeley View on Serverless Computing," UC Berkeley TechnicalReport, 2019.
[4]. Ishakian, V., Muthusamy, V., & Slominski, A., "Serving Deep Learning Models in a Serverless Platform," IEEEInternational Conference on Cloud Engineering, 2018.
[5]. Carreira, J., et al., "Cirrus: A Serverless Framework for End-to-end ML Workflows," Proceedings of ACM Symposiumon Cloud Computing (SoCC), 2019.
[6]. Bhatore, S., Mohan, L., & Reddy, Y.R., "Machine Learning Techniques for Credit Risk Evaluation: A SystematicLiterature Review," Journal of Banking and Financial Technology, vol. 4, no. 1, pp. 111-138, 2020.
[7]. Amazon SageMaker Documentation, "Deploy Models for Inference," AWS Documentation, https://docs.aws.amazon.com/sagemaker/, 2024.
[8]. FastAPI Documentation, "FastAPI - Modern Web Framework for Building APIs with Python," Sebastián Ramírez,https://fastapi.tiangolo.com, 2024.
[9]. Amazon DynamoDB Developer Guide, "Best Practices for Designing and Using Partition Keys," AWSDocumentation, 2024.
[10]. Sewak, M. & Singh, S., "Winning in the Era of Serverless Computing and Function as a Service," IEEE InternationalConference on Advanced Computing, 2018.

 This work is licensed under a Creative Commons Attribution 4.0 International License