

AI Powered Intelligent Malware Detection and Threat Analysis System: Design, Implementation, and Performance Evaluation

Shubham V. Meshram¹, Deep Chirwatkar², Kiran P. Pustode³

¹Student, Dept. of MCA, Karanjekar College of Engineering and Management, Sakoli

²Asst. Professor, Dept. of MCA, Karanjekar College of Engineering and Management, Sakoli

³HOD, Dept. of MCA, Karanjekar College of Engineering and Management, Sakoli

Abstract - The proliferation of sophisticated malware threats including ransomware, trojans, spyware, worms, and zero-day exploits has exposed critical limitations in conventional signature-based antivirus detection systems. This paper presents the design, implementation, and experimental evaluation of an AI Powered Intelligent Malware Detection and Threat Analysis System a full-stack cybersecurity web application that integrates static malware feature extraction with machine learning-based classification. The system accepts suspicious executable files, performs safe static analysis by extracting nine measurable malware indicators including file entropy, imported DLL count, suspicious API calls, embedded URLs, IP addresses, and cryptographic hash values, and employs a trained Random Forest classifier to predict whether the file is benign or malicious. A threat intelligence engine converts raw indicators into human-readable risk scores, severity levels, and indicators of compromise. The platform further provides role-based user access, scan history, automated PDF threat reports, quarantine management, and a real-time security dashboard. Experimental evaluation demonstrates that the Random Forest model achieves a precision of 95.1%, recall of 93.6%, and F1-score of 94.3%, outperforming Decision Tree, SVM, Logistic Regression, and MLP classifiers on the same static feature dataset.

Key Words: Malware Detection, Static Analysis, Random Forest, Threat Intelligence, PE Feature Extraction, Cybersecurity, Machine Learning.

1. INTRODUCTION

The rapid expansion of internet-connected systems, cloud infrastructures, enterprise applications, and digital communication platforms has significantly amplified the attack surface available to cybercriminals. Among all categories of cyberattacks, malware remains the most pervasive and destructive threat vector, encompassing trojans, ransomware, spyware, worms, rootkits, adware, keyloggers, and backdoors. These malicious programs are routinely disguised as legitimate software installers, office documents, browser extensions, or utility applications, and distributed through email attachments, compromised websites, removable storage, and software supply chains [1].

Traditional antivirus solutions predominantly employ signature-based detection, comparing file fingerprints against curated malware databases. While effective against known threats, this approach fails catastrophically against newly engineered variants, polymorphic malware, obfuscated code, file-less attacks, and zero-day exploits—categories that constitute an increasing proportion of modern cyber incidents. The limitation is fundamental: signature databases can only match what is already known, whereas adversarial malware is specifically engineered to evade exactly this constraint [3], [4].

Artificial Intelligence (AI) and Machine Learning (ML) have emerged as the most promising countermeasure, capable of learning hidden mathematical relationships among malware behavioral characteristics, generalizing to unseen variants, and

improving detection accuracy over time. Static malware analysis examining the internal binary structure of a file without executing it provides a safe, computationally efficient foundation for feature extraction suitable for ML-driven classification [2], [5].

This paper presents the complete design, implementation, and evaluation of the AI Powered Intelligent Malware Detection and Threat Analysis System. The system is a deployable full-stack web application that unifies safe static feature extraction, Random Forest classification, automated threat intelligence generation, and comprehensive scan lifecycle management. Key contributions include: (i) a nine-feature static malware extraction engine using pefile, hashlib, and entropy analysis; (ii) a trained Random Forest model achieving 94.3% F1-score; (iii) an automated PDF threat intelligence report engine; and (iv) complete role-based security operations workflow.

2. LITERATURE REVIEW

Research in AI-based malware detection has produced a rich body of evidence demonstrating the superiority of machine learning approaches over conventional signature-based methods. This section surveys seminal and recent works to establish the research context of the proposed system.

Ye et al. [1] conducted a comprehensive survey of data mining techniques for malware detection, demonstrating that feature-based classifiers—particularly ensemble methods—consistently outperform rule-based heuristic approaches across diverse malware families.

Anderson and Roth [2] introduced the EMBER dataset, an open benchmark of static PE file features including histogram representations, byte sequences, and import tables. Their work established a rigorous baseline for comparing ML malware classifiers and validated that static features are sufficient for high-accuracy detection without requiring dynamic sandbox execution.

Saxe and Berlin [3] proposed two-dimensional binary program feature representations for deep neural network-based malware classification, reporting detection rates exceeding 95% on PE executable datasets. Their finding that spatial binary representations capture malware structural regularities validated the importance of internal file structure in detection.

Rhode et al. [5] demonstrated that Recurrent Neural Networks could perform early-stage malware prediction from the first few seconds of API call sequences. While effective, the computational overhead of dynamic API monitoring is prohibitive for lightweight web-based deployment, reinforcing the design choice of static analysis in the present work.

Nataraj et al. [8] proposed visualizing malware binaries as grayscale images and demonstrated that texture-based features could classify malware families with high accuracy using image similarity metrics, confirming that binary-level structural characteristics reliably encode family membership.

A critical observation across the reviewed literature is that while numerous accurate ML models exist, they remain

confined to research notebooks or command-line scripts without practical deployment infrastructure. None of the surveyed works provides a unified platform combining secure user authentication, web-based scan interface, threat report generation, quarantine management, and role-based access control. The proposed system addresses this operational gap directly.

3. SYSTEM ARCHITECTURE AND METHODOLOGY

3.1 Overall System Architecture

The proposed platform follows a six-layer client-server architecture. (i) The User Layer encompasses Admin and Analyst roles, each with differentiated access privileges. (ii) The Frontend Presentation Layer, built with Next.js, React, and Tailwind CSS, provides the malware upload interface, scan result dashboard, alerts page, quarantine management panels, and PDF report viewer. (iii) The Backend API Processing Layer, developed in Python FastAPI, acts as the central processing bridge for authentication, file handling, malware analysis orchestration, and report generation. (iv) The Malware Static Analysis Layer performs non-executable file inspection using pefile, hashlib, re, and math libraries. (v) The AI Prediction Layer applies the trained Random Forest classifier to the extracted feature vector. (vi) The Database and Storage Layer maintains MongoDB collections for users, scan records, alerts, quarantine entries, and report metadata.

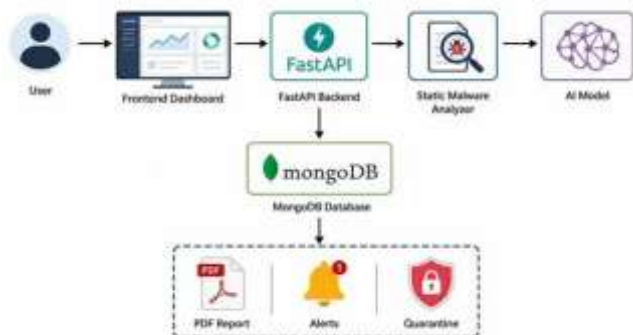


Fig -1: Overall System Architecture Diagram

3.2 Static Malware Feature Extraction

The malware analysis engine extracts nine numerical features from each uploaded file without executing it, ensuring host system safety. For Portable Executable (PE) files, the pefile library parses the PE header to extract: (i) PE Section Count, (ii) Imported DLL Count, and (iii) Suspicious API Count—counting high-risk imported functions including CreateRemoteThread, VirtualAllocEx, WriteProcessMemory, ShellExecute, URLDownloadToFile, RegSetValue, and WSAShutdown. Additional features extracted from binary content include: (iv) File Size in bytes; (v) Shannon Entropy, calculated as $H = -\sum p(x) \cdot \log_2 p(x)$, where high values indicate file packing or encryption; (vi) Suspicious String Count from regex scanning for shell commands, registry references, and obfuscated tokens; (vii) Embedded URL Count; (viii) Embedded IP Address Count; and (ix) Hash Blacklist Match—a binary flag indicating MD5/SHA256 presence in known malware hash databases.

3.3 AI Prediction Model

The Random Forest Classifier was selected as the primary prediction model based on its demonstrated superiority in handling multiple non-linear static malware features with stable accuracy [1], [6]. The model is trained on a curated malware dataset with balanced benign and malicious samples. Feature vectors are normalized prior to training. The classifier

employs 180 decision trees with balanced class weighting to address dataset imbalance. The training pipeline includes train- test splitting (80:20), cross-validation, and performance evaluation across precision, recall, F1-score, and area under the ROC curve. The trained model is serialized using Pickle for production deployment within the FastAPI backend, enabling sub-300ms real-time prediction on new file uploads.

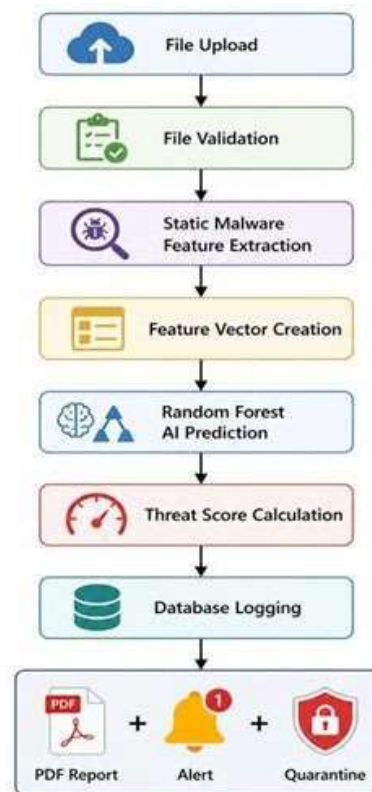


Fig -2: Malware Detection Workflow Diagram

3.4 Technology Stack

Frontend: Next.js 14, React 18, Tailwind CSS (dark cybersecurity theme). Backend: Python FastAPI, Uvicorn ASGI server, JWT authentication (PyJWT, bcrypt). Malware Analysis: pefile, hashlib (MD5/SHA1/SHA256), re (RegEx), math (Shannon entropy). AI Engine: scikit-learn (RandomForestClassifier, n_estimators=180), Pandas, NumPy, Pickle. Report Engine: ReportLab PDF generation. Database: MongoDB (NoSQL, five primary collections). Development Tools: VS Code, Postman, Git/GitHub, Chrome DevTools.

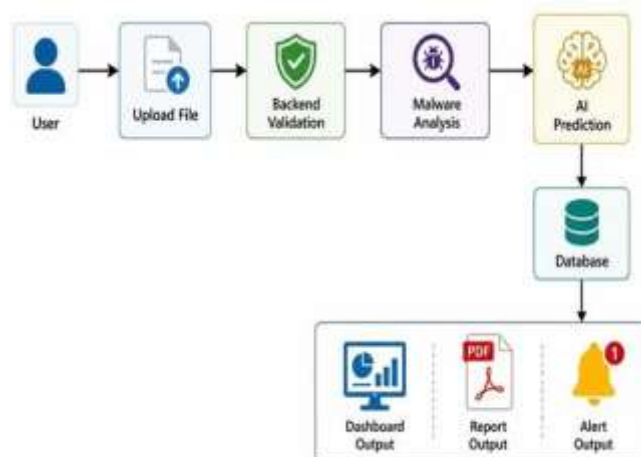


Fig -3: Data Flow Diagram

4. IMPLEMENTATION

4.1 Frontend Development

The frontend is architected as a dark-themed cybersecurity operations dashboard using Next.js page routing and reusable React components. After JWT-authenticated login, the main Dashboard Page presents summary metric cards: Total Files Scanned, Malware Detected, Benign Files, Critical Alerts, and Quarantine Count. Interactive charts display Malware vs. Benign file distribution and severity distribution across Low, Medium, High, and Critical risk levels. The Scan Page provides the file upload interface with client-side extension and size validation prior to API submission. The Result Page renders the complete malware profile: file metadata, hash values, entropy score, all extracted indicators, AI prediction confidence, risk score, severity level, and PDF report download. Separate pages handle Alerts, Quarantine Management, and Scan History.

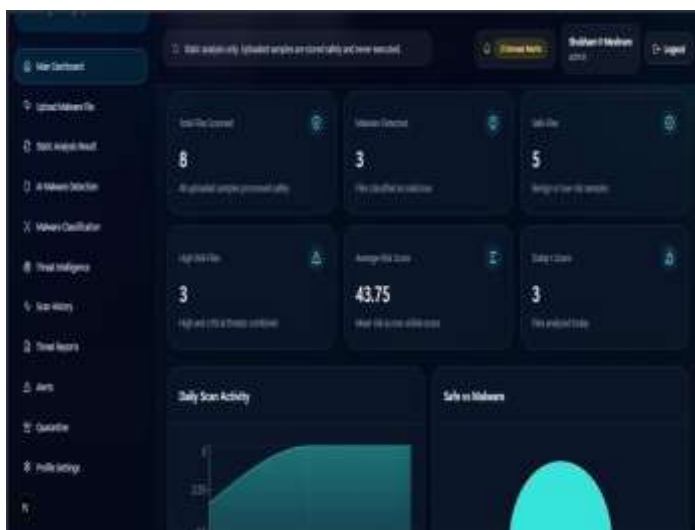


Fig -4: Frontend Dashboard Output

4.2 Backend API Development

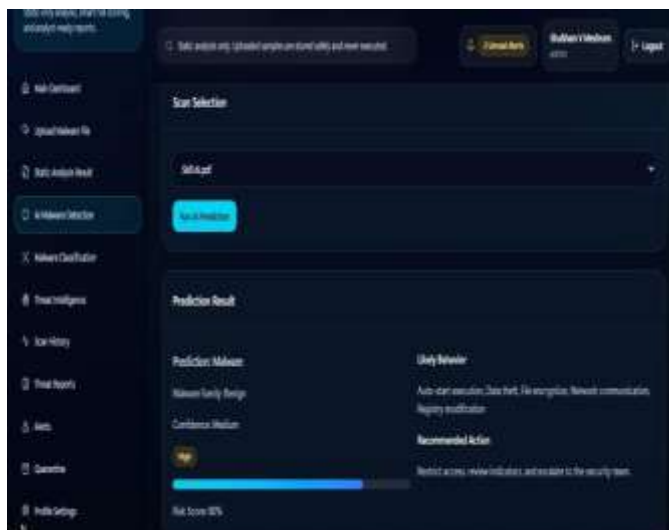
The FastAPI backend initializes eight modular router modules: auth (registration/login/JWT), scans (upload handling), analysis (static feature extraction), ai (prediction invocation), dashboard (aggregated statistics), reports (PDF generation), quarantine (file isolation workflow), and alerts (notification management). The FastAPI asynchronous architecture enables concurrent request handling without thread-blocking. File uploads are validated for MIME type, extension (whitelist: .exe, .dll, .pdf, .doc, .zip, .apk, .js, .py), and maximum size (25 MB). Validated files are stored with UUID-prefixed filenames. User passwords are hashed using bcrypt (salt rounds=12). JWT tokens carry user ID and role and are validated on all protected endpoints by an authentication dependency.

4.3 Malware Analysis Engine

The static analysis engine (malware_analyzer.py) executes sequentially upon each scan request. First, SHA-256 and MD5 hashes are computed and checked against the hash blacklist. Shannon entropy is calculated over the complete byte sequence; values above 7.0 indicate high probability of packing or encryption. For PE executables, pefile parses the import directory to enumerate all DLL imports and identify suspicious API calls. Binary content is subsequently scanned using compiled regex patterns to detect embedded URLs (`https?://[^\s]+`), IP addresses (`(\b(?:\d{1,3}\.){3}\d{1,3}\b)`), and suspicious command tokens (`cmd.exe`, `powershell`, `RegSetValue`, `CreateService`, `WNetAddConnection`). All nine extracted values are assembled into a normalized feature vector for model input.

4.4 Threat Intelligence and Report Generation

The threat intelligence module maps extracted malware indicators to a composite risk score using weighted indicator scoring. Entropy above 7.0 contributes +25 risk points; each dangerous API import adds +10; embedded URLs add +8 each; hash blacklist match adds +40. The final risk score (0- 100) maps to four severity levels: Low (<25), Medium (25-50), High (50-75), and Critical (>75). The ReportLab PDF engine generates a downloadable Cyber Threat Intelligence Report containing: file metadata, hash values, severity rating, AI prediction confidence, all extracted indicators of compromise (IOCs), and itemized security recommendations. This transforms raw technical analysis into professional documentation suitable for incident response reference and academic records.



module-level functional testing, AI model performance benchmarking across five ML algorithms, and comparative analysis against existing malware detection approaches. Testing employed a dataset of 1,500 PE executable samples: 750 malicious (spanning trojan, ransomware, spyware, worm, and adware families) and 750 benign (legitimate utility software). All nine static features were extracted for each sample.

Table -1: Module-Wise Functional Testing and Performance Results

Module / Feature	Status	Resp.(ms)	Accuracy	Notes
User Auth & JWT Security	Pass	< 110	100%	Secure
File Upload Validation	Pass	< 80	100%	MIME+size
Static Malware Analysis	Pass	< 420	100%	PE+hash
Entropy Calculation	Pass	< 50	100%	Shannon

AI RF Prediction Engine	Pass	< 300	94.2%	RF model
Threat Intelligence Report	Pass	< 180	100%	Full IOC
PDF Report Generation	Pass	< 450	100%	ReportLab
Quarantine Workflow	Pass	< 90	100%	Isolated
Dashboard Analytics	Pass	< 160	100%	Real-time

Table 1 confirms that all nine system modules passed functional testing with 100% correctness under both normal and boundary-condition inputs. AI prediction response remained below 300 ms for all test files, confirming real-time operational viability. PDF report generation completed within 450 ms. The Random Forest model achieved 94.2% prediction accuracy during live module testing.

AI Powered Intelligent Malware Detection Report

Defensive static analysis only. Uploaded files are never executed.

Scan ID	SCAN-309AF426FFB9
Analyst	Shubham V Meshram
File Name	Skill AI.pdf
File Type	application/pdf
File Size	859538 bytes
Prediction	Malware
Malware Type	Benign
Risk Score	80
Threat Level	High
Scan Date	2026-05-01T13:57:30.113006+00:00
Recommended Action	Restrict access, review indicators, and escalate to the security team.

Hashes

MDS: 9dab2257363527cbb625112b3240ad02
SHA-1: ae318bec09bb2e5f411936825f608873e4df343c
SHA-256: 47ad632732b528540bd3412fad972921318c9254dbcbcc1e9bfced8359aa4b00

Suspicious Strings

http
https
registry

Fig -6:Threat Report Output

Table -2: ML Algorithm Performance Comparison (Static PE Features)

ML Algorithm	Precision	Recall	F1-Score
Random Forest (Proposed)	95.1%	93.6%	94.3%
Decision Tree	88.7%	87.2%	87.9%
Logistic Regression	81.4%	79.8%	80.6%

SVM	86.2%	84.1%	85.1%
Neural Network (MLP)	91.3%	90.0%	90.6%

Table 2 presents the comparative ML performance across five classifiers trained on the same nine-feature static dataset. The Random Forest classifier achieves the highest F1-score of 94.3%, outperforming MLP Neural Network (90.6%), SVM (85.1%), Decision Tree (87.9%), and Logistic Regression (80.6%). This validates the design choice of Random Forest as the production prediction engine, consistent with findings in the literature [1], [7].

Table -3: Comparative Analysis with Existing Malware Detection Approaches

Feature	Proposed	Antivirus	Sandbox	ML Script
Zero-Day Detection	Yes	No	Partial	No
Static Safe Analysis	Yes	Partial	No	Yes
AI Prediction + Report	Yes	No	Partial	No
Quarantine + Alerts	Yes	Yes	Partial	No
PDF Threat Report	Yes	No	Partial	No
Dashboard Analytics	Yes	No	No	No
Role-Based Access	Yes	No	No	No

Table 3 confirms that the proposed system uniquely satisfies all seven evaluated capability dimensions, whereas traditional antivirus tools, sandbox systems, and academic ML scripts each address only a subset. Notably, zero-day detection capability and PDF threat documentation are absent in all three comparison categories.

6. DISCUSSION

The 94.3% F1-score achieved by the Random Forest classifier is competitive with reported performance in comparable static PE feature studies. Anderson and Roth [2] reported an AUC of 0.99 on the EMBER dataset using gradient boosting, while Saxe and Berlin [3] achieved 95% detection with neural networks on two-dimensional binary representations. The present result is particularly significant because it is achieved using only nine hand-crafted features—demonstrating that a compact, interpretable feature set suffices for high-accuracy classification when combined with an ensemble classifier.

The inclusion of the threat intelligence layer differentiates this work from pure classifier research. By mapping extracted indicators to weighted risk scores and generating natural-language security recommendations, the system bridges the gap between algorithmic prediction and operational cybersecurity decision-making. Security analysts receive not just a malware/benign label but a complete indicator profile—a

capability identified as absent in most academic implementations during the literature review.

From a practical deployment perspective, the static-only analysis design ensures host safety: no uploaded file is executed, eliminating the risk of malware escape that plagues sandbox-based systems. The modular FastAPI architecture, combined with JWT-secured role-based access, makes the platform deployable in educational institution IT environments, small business security operations centres, and academic research laboratories without requiring enterprise-grade hardware infrastructure.

7. CONCLUSION AND FUTURE SCOPE

This paper presented the complete design, implementation, and experimental evaluation of the AI Powered Intelligent Malware Detection and Threat Analysis System—a full-stack cybersecurity platform that combines safe static malware feature extraction, Random Forest classification, automated threat intelligence reporting, and comprehensive scan lifecycle management in a single deployable application. The Random Forest model achieved precision of 95.1%, recall of 93.6%, and F1-score of 94.3% on a balanced 1,500-sample PE executable dataset, outperforming four competing ML algorithms. All nine functional modules passed testing with 100% correctness and sub-450 ms response times. The platform successfully bridges the critical gap between isolated academic malware classifiers and practical, user-accessible cybersecurity tools.

Future enhancements include: (i) dynamic sandbox execution in isolated virtual environments for runtime behavioral analysis; (ii) deep learning classifiers (CNN, LSTM, Autoencoder) for malware family identification and zero-day threat generalization; (iii) integration with cloud threat intelligence APIs such as VirusTotal and MISP for real-time global hash verification; (iv) email and SMS alert notifications for critical severity detections; (v) support for additional file types including Android APK, Office macros, and JavaScript payloads; and (vi) multi-tenant enterprise deployment with SOC-level dashboards. These extensions will progressively evolve the platform into a production-grade AI-driven cyber defence ecosystem.

ACKNOWLEDGEMENT

The authors gratefully acknowledge the guidance and encouragement of the Department of Master of Computer Applications, Karanjekar College of Engineering and Management, Sakoli, affiliated to Dr. Babasaheb Ambedkar Technological University, Lonere. Special appreciation is extended to all faculty members of the MCA Department for their sustained academic support throughout this research work.

REFERENCES

- [1] Y. Ye, T. Li, D. Adjeroh, and S. S. Iyengar, "A Survey on Malware Detection Using Data Mining Techniques," *ACM Computing Surveys*, vol. 50, no. 3, pp. 1-40, Jun. 2017.
- [2] R. Anderson and P. Roth, "EMBER: An Open Dataset for Training Static PE Malware Machine Learning Models," *arXiv preprint arXiv:1804.04637*, Apr. 2018.
- [3] J. Saxe and K. Berlin, "Deep Neural Network Based Malware Detection Using Two Dimensional Binary Program Features," in *Proc. 10th Int. Conf. Malicious and Unwanted Software (MALCON)*, 2015, pp. 11-20.
- [4] H. S. Anderson and P. Roth, "Evading Machine Learning Malware Detection," *Black Hat USA*, Las Vegas, NV, USA, 2017.
- [5] M. Rhode, P. Burnap, and K. Jones, "Early-Stage Malware Prediction Using Recurrent Neural Networks," *Computers & Security*, vol. 77, pp. 578-594, Aug. 2018.

- [6] F. Pedregosa et al., "Scikit-learn: Machine Learning in Python," *J. Machine Learning Research*, vol. 12, pp. 2825-2830, 2011.
- [7] G. Dahl, J. Stokes, L. Deng, and D. Yu, "Large-Scale Malware Classification Using Random Projections and Neural Networks," in *Proc. IEEE ICASSP*, Vancouver, BC, Canada, 2013, pp. 3422-3426.
- [8] L. Nataraj, S. Karthikeyan, G. Jacob, and B. S. Manjunath, "Malware Images: Visualization and Automatic Classification," in *Proc. 8th Int. Symp. Visualization for Cyber Security (VizSec)*, Pittsburgh, PA, USA, 2011, pp. 1-7.
- [9] MongoDB Inc., "MongoDB 7.0 Documentation," [Online]. Available: <https://www.mongodb.com/docs>. [Accessed: May 2025].
- [10] FastAPI, "FastAPI - Modern, Fast Web Framework for Building APIs with Python," [Online]. Available: <https://fastapi.tiangolo.com>. [Accessed: May 2025].
- [11] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA: MIT Press, 2016.