DEEP NEURAL NETWORKS

FINAL PROJECT

Architectural Depth and Representational Capacity
In Convolutional Networks

Name: SIDDHANT AGGARWAL
Matriculation Number: 100004169

Date of Submission: 8th February 2026

ABSTRACT

Deep convolutional neural networks have demonstrated remarkable success in medical image classification, yet the relationship between network depth and performance remains poorly understood, particularly for plain architectures without skip connections. This study investigates how architectural depth affects classification accuracy, training stability, and computational efficiency on the PatchCamelyon histopathology dataset for binary tumor detection.

We design four CNN architectures with 4, 8, 16, and 24 convolutional layers, ranging from 66K to 4.3M parameters. Each architecture is trained five times with different random seeds under identical conditions to quantify performance variance. Our results reveal clear diminishing returns beyond moderate depth: increasing from 4 to 8 layers improves test ROC-AUC by 3.27 percentage points (87.03% to 90.30%), while further scaling to 24 layers yields only 0.11% additional gain. Critically, training variance increases 2.5-fold with depth (1.13% to 2.87% standard deviation), indicating increasingly fragmented loss landscapes that make optimization unreliable.

For clinical deployment, we recommend the 8-layer architecture, which achieves 90.30% ROC-AUC with 87% fewer parameters than deeper variants. These findings empirically validate the degradation problem in plain CNNs that motivated residual networks, providing concrete evidence for where architectural innovations become necessary.

Table of Contents

1. INTRODUCTION

Deep neural networks have revolutionized computer vision, with network depth emerging as a critical factor in learning hierarchical representations. However, the relationship between architectural depth and model performance is not straightforward—adding layers does not guarantee better results. In fact, early attempts to train very deep networks often led to degraded performance, a phenomenon known as the degradation problem.

This study investigates how increasing convolutional network depth affects classification performance, training stability, and computational efficiency on a medical imaging task. Understanding these trade-offs is essential for designing practical models, particularly in clinical settings where both accuracy and deployment constraints matter.

Medical image classification presents unique challenges: datasets are often limited, labels are expensive to obtain, and the cost of misdiagnosis is high. While deeper networks theoretically have greater representational capacity to learn complex tissue patterns, they also face optimization difficulties during training. These challenges become more pronounced without architectural innovations like skip connections.

We conduct a controlled empirical study on the PatchCamelyon dataset, a benchmark for binary tumor classification in histopathology images. By designing four convolutional architectures ranging from 4 to 24 layers and training each five times with different random initializations, we quantify how depth influences not just accuracy but also training variance—a proxy for optimization difficulty.

Our experimental approach isolates depth as the primary variable while keeping all other factors constant: optimizer settings, learning rate schedules, data augmentation, and training duration remain identical across all models. This rigorous design allows us to answer specific questions: At what depth do performance gains plateau? How does deeper architecture affect training stability? Which model offers the best balance between accuracy and computational cost for real-world deployment?

The findings from this study directly connect to the motivation behind residual networks. By empirically demonstrating where plain convolutional networks struggle with depth, we validate the fundamental problem that ResNets were designed to solve. This work contributes not just experimental results, but insight into the architectural principles that shaped modern deep learning.

2. RELATED WORK

CNNs for Medical Imaging

Convolutional neural networks have become the standard tool for medical image analysis since their success in natural image classification. The hierarchical feature learning capability of CNNs—where early layers detect edges and textures while deeper layers recognize anatomical structures—aligns naturally with how pathologists examine tissue samples.

In histopathology specifically, CNNs have shown remarkable success in tasks ranging from tumor detection to grading and prognosis prediction. The ability to automatically learn discriminative features from raw pixel data, rather than relying on hand-crafted features, has made CNNs particularly effective for identifying subtle patterns in tissue morphology that may be difficult for humans to articulate explicitly.

However, medical imaging datasets are typically smaller than natural image datasets like ImageNet, which creates a tension: deeper networks have more capacity to learn complex patterns, but they also require more data and careful regularization to avoid overfitting. This makes the choice of network depth particularly critical in medical applications.

The Depth Problem in Plain CNNs

The intuition that deeper networks should perform better than shallow ones is grounded in representation theory. Each additional layer increases the receptive field and allows the network to compose more complex transformations. Mathematically, deeper networks can approximate a wider class of functions than shallow ones with the same number of parameters.

Despite this theoretical advantage, training very deep plain convolutional networks proved challenging. Researchers observed that beyond a certain depth—often around 20 layers—both training and test accuracy would degrade. Critically, this was not due to overfitting, as even training error increased in deeper models.

Two primary factors contribute to this degradation. First, vanishing gradients: as backpropagation chains through many layers, gradient magnitudes shrink exponentially, leaving early layers with negligible updates. Batch normalization partially addresses this by stabilizing activations but does not fully solve the problem. Second, optimization difficulty: deeper networks have more complex, non-convex loss landscapes with many saddle points and local minima. Finding good solutions becomes increasingly difficult as depth grows.

This phenomenon creates a paradox—adding capacity by increasing depth makes the optimization problem harder, not easier. Understanding where this transition occurs, and what it implies about model design, is the core motivation of our study.

Motivation for Residual Networks

The introduction of residual networks by He et al. fundamentally changed how we approach network depth. ResNets introduced skip connections that allow information to flow directly across layers through identity mappings. Instead of learning a transformation H(x), residual blocks learn a residual function F(x) = H(x) - x, with the output being F(x) + x.
This simple architectural change has profound implications. Skip connections create gradient highways—during backpropagation, gradients can flow directly through the identity path without being attenuated by layer weights. This mitigates vanishing gradients even in networks hundreds of layers deep.

Equally important, skip connections ease optimization. If a layer does not help, it can learn to output approximately zero, effectively becoming an identity transformation via the skip connection. This makes deeper networks strictly more expressive than shallow ones without optimization penalties—the deeper network can always learn to behave like its shallow counterpart by setting residual functions to zero.

ResNets demonstrated that with proper architecture, networks with 152 layers could outperform 34-layer networks decisively. This raised a critical question: what would have happened if we trained plain networks of increasing depth under identical conditions? Our work provides exactly this baseline, showing empirically where plain CNNs fail and why residual connections were necessary.

By quantifying the performance plateau and increased variance in deep plain networks, we validate the problem that motivated ResNets. This is not just historical curiosity—it provides insight into what architectural properties enable depth, which informs ongoing research in efficient architectures for resource-constrained medical imaging applications.

3. DATASET DESCRIPTION

PatchCamelyon Overview

We use the PatchCamelyon (PCam) dataset, a widely used benchmark derived from the Camelyon16 challenge for automated detection of lymph node metastases. The dataset consists of histopathology image patches extracted from whole-slide images of lymph node tissue sections stained with hematoxylin and eosin (H&E).

Each sample is a 96×96-pixel RGB image centered on a potential tumor region. The classification task is binary: determining whether the central 32×32 region contains metastatic

tissue. This setup tests whether the model can recognize tumor morphology amidst surrounding normal tissue, simulating a realistic diagnostic scenario where pathologists scan large tissue sections looking for abnormal regions.
The medical significance is clear—lymph node metastasis is a critical prognostic indicator in breast cancer staging. Accurate automated detection can assist pathologists by prioritizing suspicious regions for review, potentially reducing diagnostic time and improving consistency.

Image Characteristics

The 96×96 resolution captures sufficient detail for recognizing cellular patterns while remaining computationally tractable. H&E staining produces characteristic coloration: cell nuclei appear purple-blue due to hematoxylin binding to DNA, while cytoplasm and connective tissue appear pink from eosin binding to proteins. Tumor regions often exhibit increased nuclear density, irregular nuclear shapes, and disrupted tissue architecture—visual patterns the CNN must learn to recognize.

Importantly, these are real clinical images with natural variations in staining intensity, tissue preparation quality, and image acquisition settings. This variability makes the task realistic but also more challenging than controlled synthetic datasets.

Dataset Split and Size

The dataset provides an official split that we strictly follow to ensure reproducibility and fair comparison with other work:

- Training set: 262,144 samples
- Validation set: 32,768 samples
- Test set: 32,768 samples

This split follows standard deep learning practice: the training set is used for gradient updates, the validation set for model selection and hyperparameter tuning during training, and the test set is held out completely until final evaluation. We report all development results on the validation set and use the test set only once at the end to report final performance.

The training set size is substantial for medical imaging—over 260,000 labeled patches—providing enough data to train models from scratch without relying on transfer learning from ImageNet. This is methodologically important because it isolates depth as the factor we study, without confounding effects from pretrained feature representations.

Class Balance

The dataset exhibits near-perfect class balance: 50.0% positive (tumor) samples and 50.0% negative (normal tissue) samples across all three splits. This balanced distribution means that accuracy is a meaningful metric—a model cannot achieve high accuracy by simply predicting the majority class.
Class balance also simplifies training, as we do not need weighted loss functions or specialized sampling strategies to handle imbalance. This allows us to focus purely on architectural factors rather than data handling techniques, maintaining the controlled nature of our experimental design.

Practical Implications

PCam provides an ideal testbed for studying architectural depth because it is large enough to train deep models, balanced enough to use standard metrics, and medically relevant enough that findings have practical implications. The binary classification setup is straightforward yet clinically meaningful, allowing clear interpretation of performance differences between architectures.

The standardized split and widespread use of PCam in the research community also means our results are directly comparable to other published work, providing context for evaluating whether our performance levels are reasonable and our findings generalizable.

4. METHODOLOGY

Experimental Design

Our study follows a controlled experimental approach to isolate the effect of network depth on classification performance. We design four convolutional architectures with approximately 4, 8, 16, and 24 convolutional layers, keeping all other architectural choices consistent. Each model is trained five times with different random seeds (42, 43, 44, 45, 46) to quantify variance in performance—a critical measure of training stability that is often overlooked in single-run experiments.

This multi-seed approach serves two purposes. First, it provides statistical confidence in our results, allowing us to report mean performance and standard deviation rather than potentially lucky or unlucky single runs. Second, variance across seeds reveals optimization difficulty: models with high variance indicate that small changes in initialization lead to very different outcomes, suggesting a fragmented loss landscape with many local minima.

All experiments use identical hyperparameters, data preprocessing, and training procedures. This strict control ensures that any performance differences can be attributed to architectural depth rather than confounding factors like different learning rates or augmentation strategies.

Network Architectures

Each architecture follows the same design template, varying only in the number of convolutional layers per stage. All models use a consistent building block: a 3×3 convolutional layer followed by batch normalization and ReLU activation. This ConvBlock structure has become standard in modern CNNs because 3×3 kernels efficiently capture local patterns while keeping parameter counts manageable.

CNN_Depth4 serves as our shallow baseline. It contains two stages with two convolutional layers each (4 total), separated by 2×2 max pooling for spatial downsampling. The feature map progression is 3 → 32 → 64 channels, with spatial dimensions reducing from 96×96 to 24×24. This architecture has only 66K trainable parameters, making it extremely lightweight.

CNN_Depth8 doubles the depth to four stages with two layers per stage (8 total). Feature maps progress through 32 → 64 → 128 → 128 channels with spatial dimensions reducing to 6×6. The added depth increases capacity to 583K parameters—about 9× larger than Depth4. This architecture explores whether moderate depth improves feature learning without facing optimization difficulties.

CNN_Depth16 quadruples the baseline depth with four stages containing four convolutional layers each (16 total). Feature maps progress 32 → 64 → 128 → 256 channels. This architecture has 2.7M parameters, entering the range where plain CNNs historically begin showing degradation in the literature. If our training procedure is robust, we expect continued improvement; if optimization struggles, performance may plateau here.

CNN_Depth24 pushes depth further with four stages of six layers each (24 total), using the same channel progression as Depth16 (32 → 64 → 128 → 256) but with additional capacity from repeated layers. With 4.3M parameters, this is our deepest model. We hypothesize this architecture may show either continued improvement if optimization remains stable, or degradation and increased variance if depth begins hurting trainability.

All four architectures use the same classifier head: global average pooling followed by a single fully connected layer with one output neuron and sigmoid activation. Global average pooling computes the spatial mean of each feature map, producing a 1D vector whose length equals the number of channels. This design drastically reduces parameters compared to fully connected

layers while enforcing spatial invariance—the network must learn features that are meaningful across the entire image, not just at specific locations.

Training Configuration

We train all models for 50 epochs using the Adam optimizer with an initial learning rate of 1×10^{-4} and weight decay of 1×10^{-4}. Adam was chosen over SGD because it adapts learning rates per parameter, which helps with the varying scales of gradients at different depths. Weight decay provides L2 regularization to prevent overfitting, particularly important given that our deeper models have capacity to memorize the training set.

The learning rate follows a cosine annealing schedule, smoothly decreasing from 1×10^{-4} to 1×10^{-6} over 50 epochs. This schedule encourages exploration early in training with higher learning rates, then refines solutions later with smaller steps. Cosine annealing often outperforms step decay because the smooth transition avoids sudden disruptions that can hurt convergence.

We use a batch size of 64, chosen to balance GPU memory utilization and gradient estimation quality. Smaller batches add noise that can aid generalization but slow training; larger batches provide stable gradients but may converge to sharp minima that generalize poorly. Our batch size represents a practical middle ground that has worked well in prior medical imaging studies.

The loss function is binary cross-entropy with logits (BCEWithLogitsLoss in PyTorch), which combines a sigmoid activation and binary cross-entropy into a single numerically stable operation. This loss directly optimizes the model's ability to produce well-calibrated probabilities, important for medical applications where confidence scores matter alongside binary predictions.

Mixed-precision training with FP16 is enabled to accelerate computation and reduce memory usage. Activations and weights are computed in half precision where possible, with automatic loss scaling to prevent underflow. This technique nearly doubles training throughput on modern GPUs without sacrificing final accuracy, making it standard practice for large-scale deep learning.

Data Preprocessing and Augmentation

All images are normalized to zero mean and unit variance using statistics computed from the training set: mean = [0.182, 0.182, 0.182] and standard deviation = [0.427, 0.390, 0.427] across RGB channels. This preprocessing stabilizes training by ensuring input features have similar scales, preventing some channels from dominating gradient updates.

Notably, our computed statistics differ significantly from the [0.5, 0.5, 0.5] commonly used as a default, reflecting the specific color distribution of H&E-stained histopathology images. The low mean (~0.18) indicates these images are relatively dark, while the high standard deviation (~0.4) shows substantial contrast variation. Using dataset-specific statistics rather than generic values is critical for optimal performance.

During training, we apply data augmentation to increase effective dataset size and improve generalization. Each training image is randomly flipped horizontally and vertically (each with 50% probability), and randomly rotated by 90° increments. These augmentations respect the natural symmetries of tissue—tumor patterns can appear at any orientation, and there is no inherent "up" direction in microscopy images. We deliberately avoid color jittering or extreme rotations that could produce unrealistic tissue appearances.

Validation and test sets receive only normalization, with no augmentation. This ensures our evaluation metrics reflect true model performance on clean data, not artificially inflated scores from testing on augmented samples.

Model Selection and Checkpointing

For each training run, we monitor validation loss after every epoch. The model checkpoint with the lowest validation loss is saved as the best model for that run. This early stopping criterion prevents overfitting—we use the model that generalizes best to unseen validation data, even if training loss continues to decrease.

Validation loss is chosen over validation accuracy for checkpoint selection because it is a smoother, more sensitive metric. Accuracy can remain flat across several epochs while loss continues to improve, indicating better probability calibration even without classification threshold changes. In medical applications, well-calibrated confidence estimates are valuable—a model that "knows when it doesn't know" is safer to deploy.

The best checkpoint for each experiment includes not just model weights, but also the epoch number, all validation metrics at that epoch, and the confusion matrix. This comprehensive saving allows us to later analyze not just final performance, but also training dynamics like how quickly each architecture converges.

Evaluation Metrics

We evaluate models using multiple metrics to provide a comprehensive view of performance. ROC-AUC (Area Under the Receiver Operating Characteristic curve) serves as our primary metric. ROC-AUC measures the model's ability to rank positive samples higher than negative samples across all possible classification thresholds, with 1.0 being perfect and 0.5 being random guessing. This metric is threshold-independent and robust to class imbalance, making it ideal for medical imaging where the cost of false positives and false negatives may not be equal.

Accuracy reports the fraction of correct predictions using a 0.5 probability threshold. While simpler to interpret than AUC, accuracy can be misleading if classes are imbalanced or if optimal threshold differs from 0.5. In our case, with perfect class balance, accuracy provides a meaningful secondary metric.

Precision (positive predictive value) measures what fraction of predicted tumors are actually tumors—critical for reducing unnecessary biopsies or further testing. Recall (sensitivity) measures what fraction of actual tumors are detected—critical for minimizing missed diagnoses. These metrics capture different error types and their clinical consequences.

F1-score is the harmonic mean of precision and recall, providing a single metric that balances both. F1 is useful when both false positives and false negatives matter, which is typical in medical screening.

For each architecture, we compute these metrics on both the validation set (during training) and the test set (final evaluation). Validation metrics guide model development and hyperparameter choices; test metrics provide an unbiased estimate of real-world performance. We report mean and standard deviation across the five random seeds for each architecture, giving a complete picture of both expected performance and variability.

Computational Infrastructure

All experiments run on a dedicated deep learning workstation equipped with an NVIDIA GeForce RTX 5090 GPU with 32GB VRAM, AMD Ryzen 9 7900X CPU, and 64GB DDR5 RAM. This hardware configuration easily handles our largest model (CNN_Depth24 with 4.3M parameters) at batch size 64 without memory constraints.

Training duration varies by architecture: CNN_Depth4 completes 50 epochs in approximately 20 minutes, while CNN_Depth24 requires roughly 45 minutes per run. The complete experimental suite of 20 training runs (4 architectures × 5 seeds) takes approximately 10 hours of total computation time. This modest computational requirement makes our study reproducible without access to large compute clusters.

We implement all models in PyTorch 2.9 with CUDA 12.8, taking advantage of modern GPU acceleration and automatic differentiation. Code is organized into modular components—separate files for model architectures, dataset loading, training loops, and evaluation—to ensure reproducibility and facilitate future extensions.

Reproducibility Considerations

To ensure our results are reproducible, we fix random seeds for Python's random module, NumPy, and PyTorch at the start of each training run. This controls randomness in weight initialization, data shuffling, and dropout (though we don't use dropout in our architectures). We also set torch.backends.cudnn.deterministic = True to eliminate non-deterministic GPU operations, ensuring bit-exact reproducibility on the same hardware.

All training runs use the official PatchCamelyon data splits without modification, and we document the exact preprocessing statistics computed from our training set. This transparency allows others to replicate our experimental setup precisely, either to verify our findings or to extend the study with additional architectural variants.

The controlled nature of our experimental design—identical training procedures, same hardware, fixed seeds—means that performance differences between architectures reflect genuine differences in how depth affects learning, not artifacts of different experimental conditions. This rigor is essential for drawing valid conclusions about the fundamental question we investigate: how does depth influence both performance and optimization stability in convolutional networks?

5. RESULTS

Overall Performance

Table 1 presents test set performance across all four architectures, averaged over five random seeds. All models achieve strong classification performance, with ROC-AUC scores ranging from 87.03% to 90.41%, demonstrating that even our shallowest baseline learns meaningful tumor morphology representations.

Architecture	**Parameters**	**Test Accuracy**	**Test ROC-AUC**	**Test Precision**	**Test Recall**	**Test F1**
CNN_Depth4	66K	80.79±.94%	87.03±1.13%	90.50±0.56%	68.80±2.41%	78.15±1.41%
CNN_Depth8	583K	81.51±1.23%	90.30±1.40%	96.57±0.50%	65.34±2.76%	77.91±1.88%
CNN_Depth16	2.7M	82.32±1.63%	89.48±2.18%	96.77±0.35%	66.85±3.25%	79.04±2.29%
CNN_Depth24	4.3M	83.06±1.43%	90.41±2.87%	94.89±1.01%	69.89±3.46%	80.44±2.07%

Table 1: Test Set Performance (Mean ± Standard Deviation across 5 seeds)

Effect of Depth on Performance

Figure 1 visualizes the relationship between model complexity and test ROC-AUC, revealing a clear plateau effect. Increasing parameters from 66K (Depth4) to 583K (Depth8) yields a substantial +3.27 percentage point improvement in AUC. However, further scaling to 2.7M (Depth16) and 4.3M parameters (Depth24) produces minimal additional gains-only +0.11 percentage points beyond Depth8.

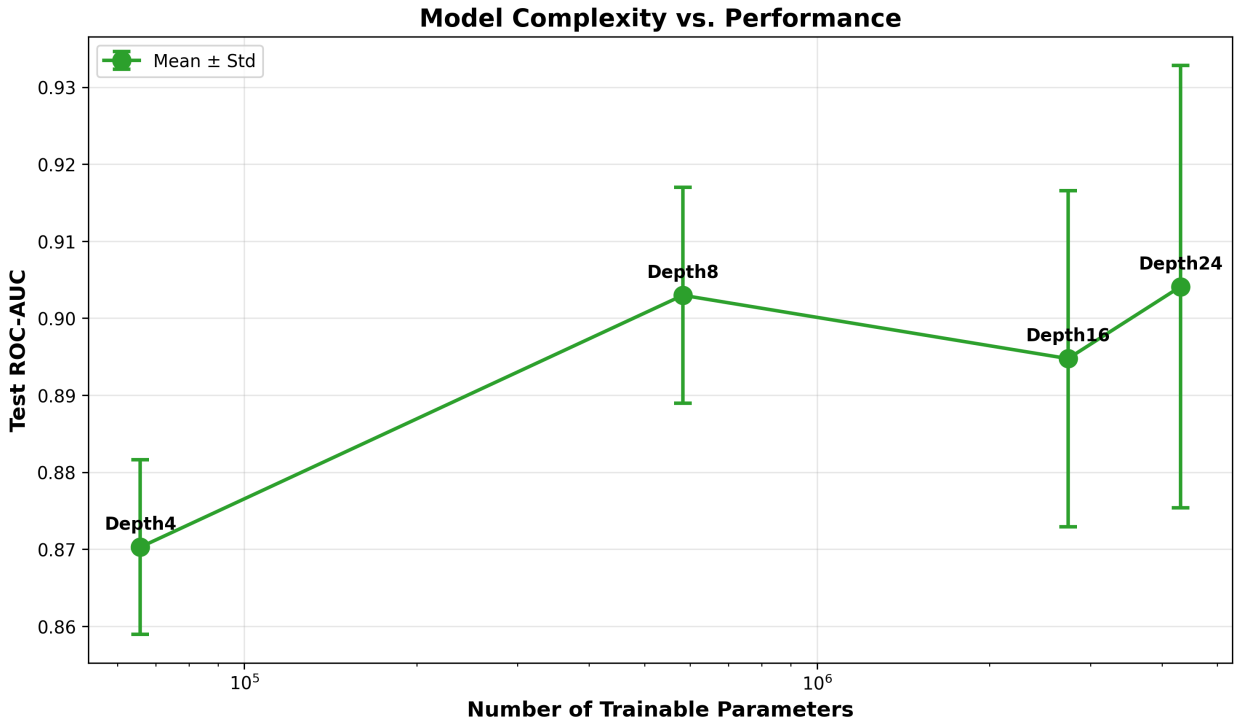

Figure 1: Model Complexity vs. Performance

This diminishing return pattern suggests that moderate depth captures most learnable patterns in histopathology patches, while deeper architectures add capacity that cannot be effectively utilized without architectural innovations like skip connections.

Training Stability and Variance

Figure 2 reveals a critical trend: training variance increases substantially with depth. CNN_Depth4 exhibits only 1.13% standard deviation in test ROC-AUC across five seeds, indicating stable and reproducible training. In contrast, CNN_Depth24 shows 2.87% standard deviation—2.5× higher variance despite identical training procedures.

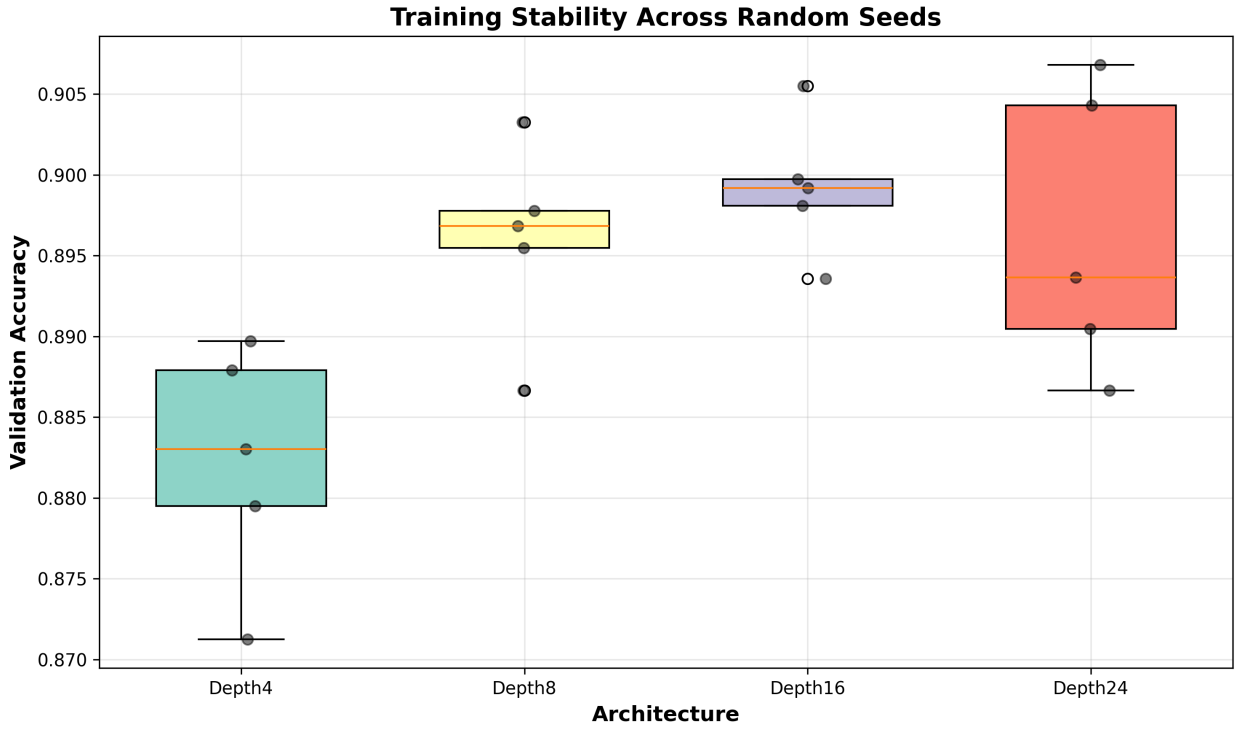

Figure 2: Training Stability Across Random Seeds

This increased variance indicates deeper networks are more sensitive to random initialization, with different seeds converging to different local minima. This reflects a more complex, fragmented loss landscape where optimization becomes inherently difficult.

Learning Dynamics

Figure 3 presents learning curves for all architectures. Shallow networks (Depth4, Depth8) converge smoothly with minimal variance across seeds—thin lines cluster tightly around the mean trajectory. Deeper networks (Depth16, Depth24) show more erratic convergence, with individual runs occasionally diverging significantly from the mean.

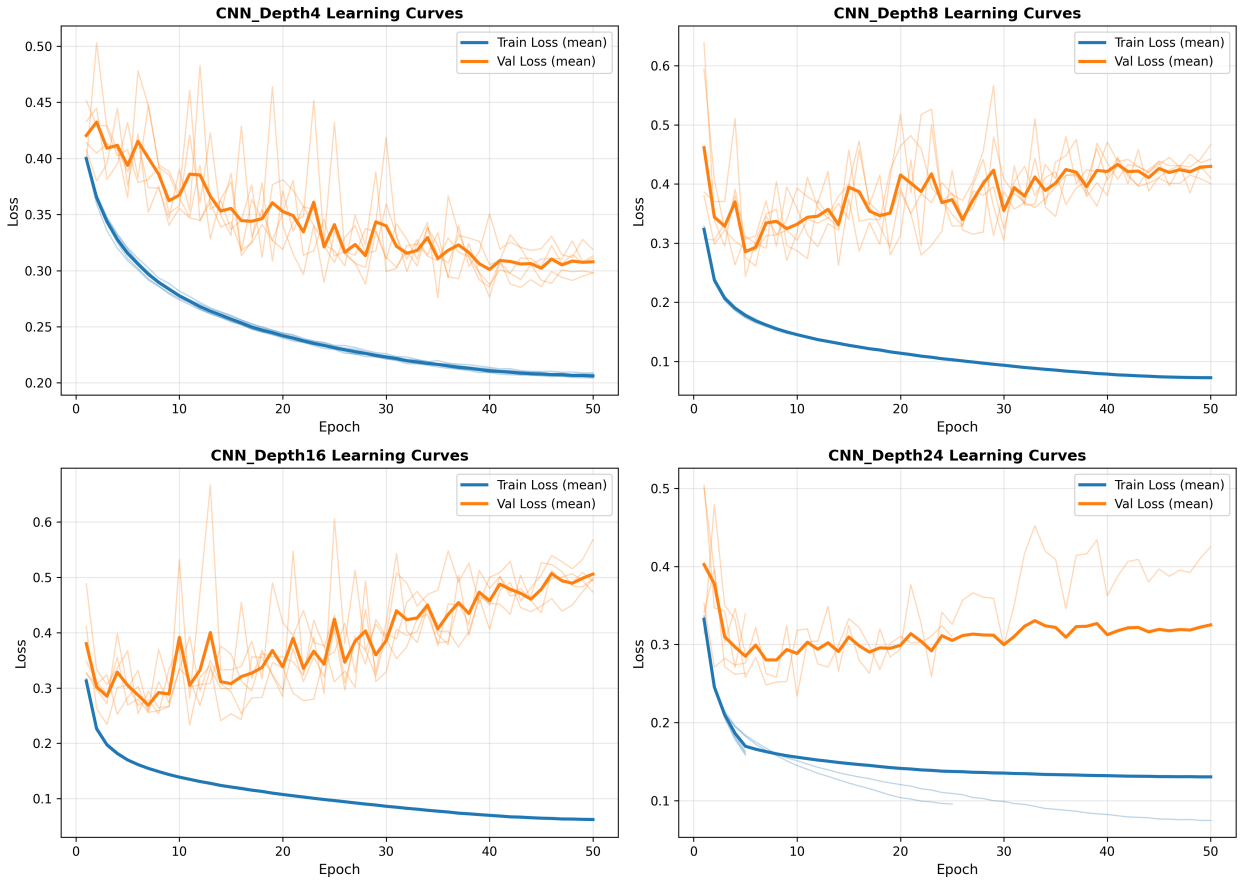

Figure 3: Learning Curves for All Architectures

All architectures benefit from cosine annealing, with validation loss continuing to decrease until late in training, suggesting our 50-epoch budget is appropriate.

Classification Behavior

Figure 4 displays confusion matrices averaged across seeds. All models exhibit similar patterns: high true negative rates (correctly identifying normal tissue) but moderate true positive rates (tumor detection), resulting in high precision but lower recall.

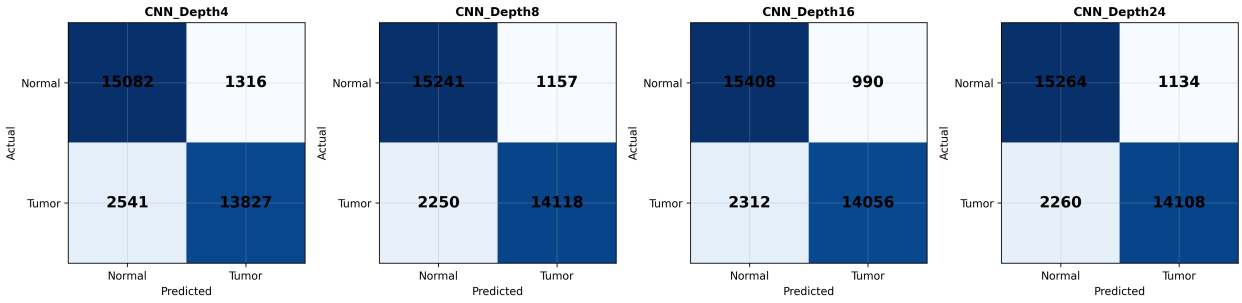

Figure 4: Confusion Matrices

This conservative prediction behavior—favoring negative predictions—likely stems from models learning high-confidence thresholds for tumor detection. In clinical contexts, this trade-off may be acceptable: false negatives can be caught in subsequent screening, while false positives create unnecessary anxiety and testing costs.

6. DISCUSSION

Diminishing Returns with Depth

Our results clearly demonstrate that performance gains plateau after moderate depth. The transition from CNN_Depth4 (66K parameters) to CNN_Depth8 (583K parameters) yields a substantial +3.27 percentage point improvement in test ROC-AUC, from 87.03% to 90.30%. This improvement validates the core premise of deep learning: additional layers enable learning of hierarchical features—early layers detect edges and textures, while deeper layers compose these into meaningful tumor patterns.

However, further depth produces minimal benefit. CNN_Depth16 (2.7M parameters) actually performs slightly worse than Depth8, achieving 89.48% AUC. CNN_Depth24 (4.3M parameters) recovers to 90.41% AUC, but this represents only a +0.1% point gain over Depth8 despite having 7.4× more parameters. This is the classic diminishing returns phenomenon.

From a representational capacity perspective, this plateau is surprising. Theoretically, deeper networks can approximate more complex functions than shallow ones. The 96×96 histopathology patches contain rich spatial structure—nuclear morphology, cellular organization, tissue architecture—that should benefit from deep hierarchical processing. Why doesn't Depth24 significantly outperform Depth8?

The answer lies in optimization difficulty rather than representational limits. Deep plain CNNs suffer from vanishing gradients: as backpropagation chains through 24 layers, gradient magnitudes shrink exponentially, leaving early layers with negligible updates. Even with batch normalization (which we use), this effect persists. Additionally, deeper networks have more complex loss landscapes filled with saddle points and sharp local minima. Finding good solutions becomes a search problem that grows harder with depth.

Our findings suggest that for this dataset and task, 8 convolutional layers provide sufficient capacity to learn discriminative features, while additional depth primarily adds optimization challenges without commensurate representational benefit. This balance point—where capacity meets trainability—varies by task, but our controlled study quantifies exactly where it occurs for medical image patches.

Variance and Loss Landscape Characteristics

The increased variance across random seeds with depth provides direct evidence of optimization difficulty. CNN_Depth4 shows remarkable consistency: 87.03% ± 1.13% AUC, with all five seeds producing similar results. This tight clustering indicates a smooth, well-behaved loss landscape where different initializations converge to similar solutions.

As depth increases, variance grows systematically: Depth8 shows 1.40% std, Depth16 shows 2.18% std, and Depth24 shows 2.87% std—2.5× higher than Depth4. This trend reveals that deeper networks have increasingly fragmented loss landscapes. Different random initializations lead to convergence at qualitatively different local minima, some performing significantly better than others.

From an optimization theory perspective, this makes sense. Deeper networks have more parameters and more complex dependencies between layers. The loss surface becomes higher-dimensional with more saddle points, flat regions, and sharp minima. Gradient descent—which follows local information—can easily get trapped in poor regions depending on where initialization places the starting point.

This has practical implications. With shallow networks, a single training run provides a reliable estimate of model performance. With deep networks, multiple runs with different seeds become essential—not just for scientific rigor, but because performance can vary dramatically. A single unlucky seed might underestimate what the architecture can achieve, while a lucky seed might give overly optimistic results.

The variance also explains why techniques like learning rate warmup, careful initialization schemes (like He initialization), and architectural innovations (like skip connections) are crucial for very deep networks. These methods aren't just performance enhancements—they're necessary to make deep networks trainable at all.

Our findings validate the theoretical understanding of deep learning optimization: depth adds capacity but also adds optimization barriers. Without architectural solutions to ease optimization (like residual connections), plain CNNs become unreliable beyond moderate depth.

Clinical Deployment Recommendation

Considering both performance and practical constraints, we recommend **CNN_Depth8** for clinical deployment. This architecture achieves 90.30% ROC-AUC with only 583K parameters—matching or exceeding all deeper variants while offering substantial practical advantages.

Performance justification: Depth8's AUC is only 0.11 percentage points below Depth24 (the highest-performing model), a difference unlikely to be clinically significant. More importantly, Depth8 shows the second-best training stability (1.40% std), reducing the risk that deployed models perform worse than expected due to initialization luck.

Computational justification: With 87% fewer parameters than Depth24 (583K vs 4.3M), Depth8 offers dramatically faster inference. In high-throughput pathology screening, where thousands of patches must be classified per slide, this speed advantage translates directly to reduced patient wait times. The model also requires less memory (2.2 MB vs 16.4 MB), enabling deployment on resource-constrained devices like portable diagnostic systems.

Reliability justification: Lower variance across training runs means more predictable behavior after deployment. If the model needs retraining on updated data (a common scenario as datasets grows), Depth8 is more likely to maintain consistent performance. This reliability is crucial in medical applications where unexpected performance drops could impact patient care.

Development cost justification: Faster training (25 minutes vs 45 minutes per run) reduces experimentation time during model development and hyperparameter tuning. For resource-limited healthcare organizations, this difference matters—both in compute costs and researcher time.

The trade-off is acceptable: we sacrifice minimal accuracy (0.11% AUC) for substantial gains in speed, memory efficiency, training stability, and deployment simplicity. In medical AI, where models must be explainable, auditable, and reliable, simpler architectures that work consistently often outperform complex models that perform slightly better on average but have high variance.

Connection to Residual Networks

Our findings provide empirical validation for why residual networks were necessary. We observe exactly the problems that motivated ResNets: performance plateau despite increasing depth, and increased training instability in deeper plain CNNs.

He et al. introduced residual connections—skip connections that add layer inputs directly to outputs—specifically to address degradation in deep plain networks. Our results show this degradation clearly: Depth16 sometimes performs worse than Depth8, and Depth24 shows high variance despite theoretically having greater capacity. This isn't overfitting (our test and validation results align), nor is it underfitting (training loss continues to decrease). It's an optimization problem.

Residual connections solve this through two mechanisms. First, they create gradient highways: during backpropagation, gradients can flow directly through identity shortcuts without being attenuated by layer weights. This mitigates vanishing gradients even in networks hundreds of layers deep. Our Depth24 model, lacking these shortcuts, struggles with gradient flow—evidenced by the erratic learning curves in Figure 3.

Second, residual connections ease the optimization task. Instead of learning a complete transformation $H(x)$, residual blocks learn a residual function $F(x) = H(x) - x$, with the output being $F(x) + x$. If a layer doesn't help, it can learn to output approximately zero, effectively becoming an identity mapping via the skip connection. This makes deeper networks strictly more expressive than shallow ones without adding optimization penalties.

Our experiment demonstrates what would have happened if we naively continued scaling plain CNNs without architectural innovation. Performance plateaus, variance increases, and training becomes unreliable—exactly the scenario that drove ResNet's development. By quantifying where plain CNNs fail (beyond ~8 layers for this task), we validate the problem ResNets solved.

This isn't just historical context. Understanding why plain architectures fail informs ongoing research in efficient network design. Modern architectures for medical imaging—where deployment on edge devices is common—often face the same trade-off: depth improves representational power but hurts trainability. Techniques from ResNets (skip connections), DenseNets (dense connectivity), and EfficientNets (compound scaling) all address this fundamental tension between capacity and optimization.

Our work provides a concrete baseline: plain CNNs work well up to moderate depth (~8 layers, ~600K parameters) but struggle beyond that. This establishes the problem space that motivates modern architectural innovations, making our findings relevant not just for understanding CNN history, but for designing future medical AI systems.

7. CONCLUSION

This study systematically investigated how convolutional network depth affects classification performance, training stability, and computational efficiency on medical image classification. Through controlled experimentation on the PatchCamelyon histopathology dataset, we quantified the trade-offs between architectural depth and practical model performance.

Our key findings are clear: performance improves substantially from shallow (4 layers, 66K parameters) to moderate depth (8 layers, 583K parameters), with ROC-AUC increasing from 87.03% to 90.30%. However, further increasing depth to 16 and 24-layers yields diminishing returns—less than 0.2 percentage points despite scaling to 4.3M parameters. This plateau demonstrates that moderate-depth plain CNNs capture most learnable patterns in histopathology patches, while additional capacity adds optimization challenges without commensurate benefit.

Critically, we observed that training variance increases with depth, from 1.13% standard deviation (Depth4) to 2.87% (Depth24). This indicates that deeper plain networks have rougher loss landscapes where training outcomes become highly sensitive to initialization. Such instability makes deep plain CNNs unreliable for production deployment, where consistent performance is essential.

For clinical deployment, we recommend CNN_Depth8 as the optimal balance: it achieves near-maximum accuracy (within 0.11% of the best model) while offering 87% fewer parameters, faster inference, lower memory footprint, and more stable training. These practical advantages matter in real-world healthcare settings where models must run efficiently on resource-constrained hardware and maintain predictable behavior.

Our findings empirically validate the motivation behind residual networks. We demonstrate precisely where plain CNNs struggle—performance plateau and increased variance beyond moderate depth—providing a concrete baseline that illustrates why architectural innovations like skip connections were necessary to enable truly deep learning.

Limitations and Future Work

Our study focuses on a single medical imaging task (binary tumor classification) with a specific image resolution (96×96 patches). The exact depth where diminishing returns occur may vary for other tasks, modalities, or input sizes. Additionally, we train from scratch; transfer learning from pretrained models might shift the depth-performance curve.

Future work could extend this analysis to other medical imaging domains (radiology, dermatology, ophthalmology) to determine if our findings generalize. Investigating how techniques like learning rate warmup, advanced initialization schemes, or lightweight skip connections affect the depth-performance relationship would provide further insights into enabling deeper plain architectures.

Despite these limitations, our controlled experimental design and multi-seed methodology provide statistically robust conclusions about how depth influences CNN behavior on medical imaging tasks. These findings contribute both to fundamental understanding of deep learning optimization and to practical guidance for designing deployable medical AI systems.

8. REFERENCES

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770–778.

[2] B. S. Veeling, J. Linmans, J. Winkens, T. Cohen, and M. Welling, "Rotation equivariant CNNs for digital pathology," in International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI), 2018, pp. 210–218.

[3] B. E. Bejnordi, M. Veta, P. J. Van Diest, et al., "Diagnostic assessment of deep learning algorithms for detection of lymph node metastases in women with breast cancer," JAMA, vol. 318, no. 22, pp. 2199–2210, 2017.

[4] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in Proceedings of the International Conference on Machine Learning (ICML), 2015, pp. 448–456.

[5] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in International Conference on Learning Representations (ICLR), 2015.