# Reducing Processing Cost in Diabetic Foot Ulcer Image Classification Using Knowledge Distillation and Network Pruning

Lakshya[1] and Jaydeep Kishore[2]

[1]Manipal University Jaipur, India , lakshya.2502010001@muj.manipal.edu
[2]Manipal University Jaipur, India , jaydeep.kishore@jaipur.manipal.edu

**Abstract**

Deep learning models have become central to automated diabetic foot ulcer (DFU) classification, yet the computational burden of deploying such architectures remains a persistent problem in clinical and mobile health environments. Larger convolutional networks perform well, but they require hardware that many wound-care clinics simply do not have. This paper explores a paired reduction strategy built around knowledge distillation and history-based filter pruning. A ResNet-50 teacher teaches a MobileNetV2 student via softened targets, then the student is pruned using a training-history-informed filter selection method. Using 5-fold stratified cross-validation on a public Kaggle DFU dataset (1,055 images), the distilled student achieves 99.81% mean accuracy while reducing parameters from 23.5M to 2.09M—an 11.2× compression. The student runs 2.65× faster on CPU (13.57ms → 5.11ms). After one pruning round, accuracy remains at 98.67% (±0.63%), demonstrating a practical trade-off for edge deployment.

**Keywords:** Diabetic Foot Ulcer, Knowledge Distillation, Pruning, Model Compression, MobileNetV2, ResNet-50

## 1 Introduction

Diabetic foot ulcers (DFUs) represent a significant healthcare challenge, affecting approximately 15-25% of diabetic patients during their lifetime and leading to substantial morbidity, including limb amputation in severe cases [22]. Early and accurate classification of DFUs is critical for timely intervention and improved patient outcomes. While deep learning models have demonstrated remarkable performance in automated DFU classification, their computational requirements present a significant barrier to deployment in resource-constrained clinical environments.

Many wound-care clinics, particularly in underserved regions, rely on modest hardware that cannot efficiently run large convolutional neural networks. This creates a practical disconnect: state-of-the-art models trained on research GPUs often cannot be deployed where they are needed most. Addressing this gap requires model compression techniques that maintain classification accuracy while dramatically reducing computational requirements.

This paper presents a paired reduction strategy combining knowledge distillation and history-based filter pruning for efficient DFU classification. The approach trains a ResNet-50 teacher model to learn robust DFU features, then transfers this knowledge to a compact MobileNetV2 student via softened probability targets. Subsequently, history-based filter pruning identifies and removes persistently redundant filters, further reducing model complexity. This combination leverages two complementary intuitions: distillation transfers representational structure from teacher to student, while pruning removes structural redundancy within the student.

The main contributions of this work are: (1) demonstration of effective knowledge distillation for DFU classification, achieving 99.81% accuracy with 11.2× parameter reduction; (2) analysis of history-based pruning stability using 5-fold cross-validation; and (3) practical recommendations for deploying compressed models in clinical settings.

## 2 Related work

### 2.1 DFU image classification

The field of diabetic foot ulcer classification has seen significant progress through deep learning approaches. Goyal et al. [22] introduced DFUNet, a CNN-based approach for DFU classification that established early benchmarks in the field. Alzubaidi et al. [23] proposed DFU_QUTNet, a novel deep CNN architecture specifically designed for DFU classification, demonstrating the value of domain-specific architectures. Their subsequent work [24] explored robust applications of deep learning tools for medical imaging, while [25] provided important insights into transfer learning for medical imaging tasks.

The DFUC 2020 challenge [26] established standardized benchmarks for DFU classification, with subsequent work by Yap et al. [27] extending this to infection and ischaemia classification. Recent approaches have explored ensemble methods, with Das et al. [31] demonstrating strong results using boosted EfficientNet ensembles. Comprehensive evaluations of deep learning in DFU detection [32] and analysis of classification approaches for infection and ischaemia [33] have further advanced the field.

Wagner et al. [28] proposed automated DFU detection using smartphone-based imaging, addressing practical deployment concerns. Cruz-Vega et al. [29] explored texture-based classification for wound assessment. Yap et al. [30] introduced localization techniques for DFU in clinical images, while Goyal et al. [34] developed methods for recognizing ischaemia and infection patterns in DFU images.

Most DFU work adopts standard CNN backbones such as ResNet [5] or EfficientNet [6] and reports impressive accuracy numbers. What is often glossed over is the compute those models need. Deploying a ResNet-50 or larger model on a low-end CPU can be slow and memory hungry, limiting clinical deployment in resource-constrained settings.

### 2.2 Network pruning

Neural network pruning has evolved significantly from early magnitude-based approaches to sophisticated structure-aware methods. Li et al. [7] introduced filter pruning for efficient ConvNets, demonstrating that removing entire filters maintains hardware compatibility while reducing computation. Han et al. [9] proposed deep compression combining pruning, quantization, and Huffman coding for extreme model compression.

Structured pruning methods [10] remove entire channels or layers rather than individual weights, enabling direct speedup without specialized sparse matrix operations. Liu et al. [11] introduced network slimming using batch normalization scaling factors as pruning criteria. He et al. [12] proposed channel pruning using LASSO regression to select important channels.

History-based pruning methods track filter norms or correlations across epochs and remove filters that behave redundantly over training [1]. This tends to be safer than single-epoch magnitude pruning because it looks for consistent redundancy rather than making decisions based on a single snapshot. The HBFP paper provided a practical algorithm we adapted for student pruning, using training history to identify persistently redundant filters.

### 2.3 Knowledge distillation

Knowledge distillation, introduced by Hinton et al. [3], provides a principled framework for transferring knowledge from a large teacher model to a compact student. The key insight is that soft probability distributions contain richer information than hard labels, encoding inter-class relationships that help the student learn more generalizable representations.

Subsequent work has extended distillation in various directions. Romero et al. [13] introduced FitNets, using intermediate feature representations as additional supervision signals. Zagoruyko and Komodakis [14] proposed attention transfer, distilling spatial attention maps from teacher to student. Park et al. [15] developed relational knowledge distillation, focusing on relationships between samples rather than individual outputs.

In medical imaging, knowledge distillation has shown particular promise for deploying models in resource-constrained clinical environments. Recent work shows students can approach teacher performance when the teacher is pre-trained using self-supervised or contrastive schemes and then fine-tuned on the target dataset [2]. Chen et al. [16] demonstrated effective distillation for chest X-ray classification, while Qin et al. [17] explored efficient medical image analysis through knowledge transfer.

### 2.4 Combining pruning and distillation

The literature contains many pruning-only and distillation-only papers but fewer works that intentionally combine the two for medical tasks. Distillation tends to produce students with cleaner internal representations, which in turn makes them more resilient to pruning.

Polino et al. [18] explored combining quantization with distillation, showing that distilled models tolerate aggressive compression better. Mishra and Marr [19] introduced “apprentice” networks that combine low-precision quantization with knowledge distillation. For structured pruning specifically, Li et al. [20] demonstrated that pruning distilled networks achieves better accuracy-efficiency trade-offs than pruning networks trained with hard labels alone.

The synergy between distillation and pruning arises because distillation produces models with smoother loss landscapes and more distributed representations [21], making individual filter removal less disruptive. Our work builds on this insight, applying history-based pruning to a distilled student model for DFU classification.

## 3 Methodology

Our methodology integrates knowledge distillation with history-based filter pruning for efficient DFU classification. The approach follows a teacher-student paradigm similar to recent medical imaging compression work [2]. The complete workflow is shown in Figure 2.

### 3.1 Dataset

We use the publicly available Diabetic Foot Ulcer dataset from Kaggle (laithjj/diabetic-foot-ulcer-dfu), which contains 1,055 images across two classes: Abnormal (ulcer) with 512 images and Normal (healthy) with 543 images. This dataset reflects common field issues such as mixed lighting conditions and variable framing, similar to challenges observed in other DFU datasets [23, 22]. To ensure reliable evaluation, we employ 5-fold stratified cross-validation rather than a single train-test split. Images are resized to 224×224 and normalized using ImageNet statistics. Representative examples from both classes are shown in Figure 1.

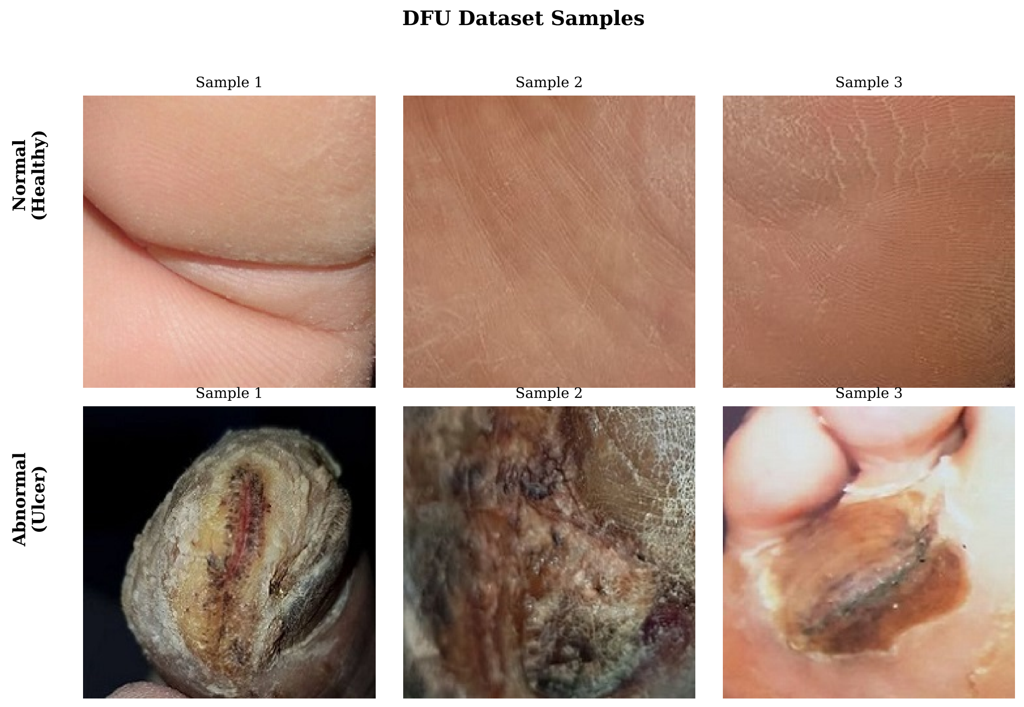
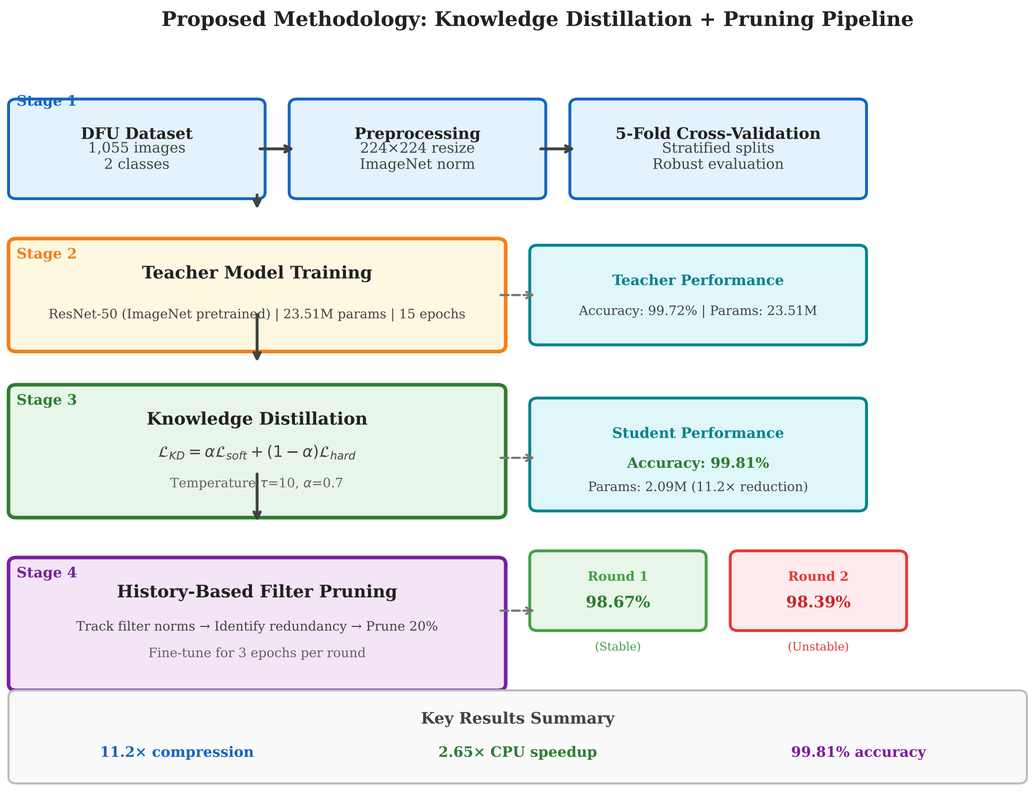


Figure 1: Sample images from the DFU dataset showing Normal (healthy foot) and Abnormal (ulcer) classes.



Proposed Methodology: Knowledge Distillation + Pruning Pipeline
Stage 1
DFU Dataset
1,055 images
2 classes
Preprocessing
224×224 resize
ImageNet norm
5-Fold Cross-Validation
Stratified splits
Robust evaluation
Stage 2
Teacher Model Training
ResNet-50 (ImageNet pretrained) | 23.51M params | 15 epochs
Teacher Performance
Accuracy: 99.72% | Params: 23.51M
Stage 3
Knowledge Distillation
$\mathcal{L}_{KD} = \alpha\mathcal{L}_{soft} + (1-\alpha)\mathcal{L}_{hard}$
Temperature $\tau$=10, $\alpha$=0.7
Student Performance
Accuracy: 99.81%
Params: 2.09M (11.2× reduction)
Stage 4
History-Based Filter Pruning
Track filter norms → Identify redundancy → Prune 20%
Fine-tune for 3 epochs per round
Round 1
98.67%
(Stable)
Round 2
98.39%
(Unstable)
Key Results Summary
11.2× compression
2.65× CPU speedup
99.81% accuracy

Figure 2: Flow of our methodology: The pretrained ResNet-50 teacher is fine-tuned on the DFU dataset, then knowledge is transferred to MobileNetV2 student using knowledge distillation, followed by history-based filter pruning for model compression.

## 3.2 Teacher and student

We use ResNet-50 as the teacher (pretrained on ImageNet, fine-tuned on the DFU dataset for 15 epochs). For the student we pick MobileNetV2, a compact model widely supported on edge devices. The teacher has 23.51M parameters while the student has 2.23M parameters before pruning.

## 3.3 Knowledge distillation

The teacher-student architecture follows the paradigm established in [3] and adapted for medical imaging in [2]. We train the student with a hybrid loss that combines cross-entropy on hard labels and a softened KL-style loss on teacher soft targets. Concretely,

$$L_{\mathrm{KD}} = \alpha\, L_{\mathrm{soft}} + (1-\alpha)\, L_{\mathrm{hard}},$$

where temperature $\tau = 10$ and $\alpha = 0.7$ (values consistent with prior medical KD studies [2]). The knowledge distillation framework is illustrated in Figure 3.

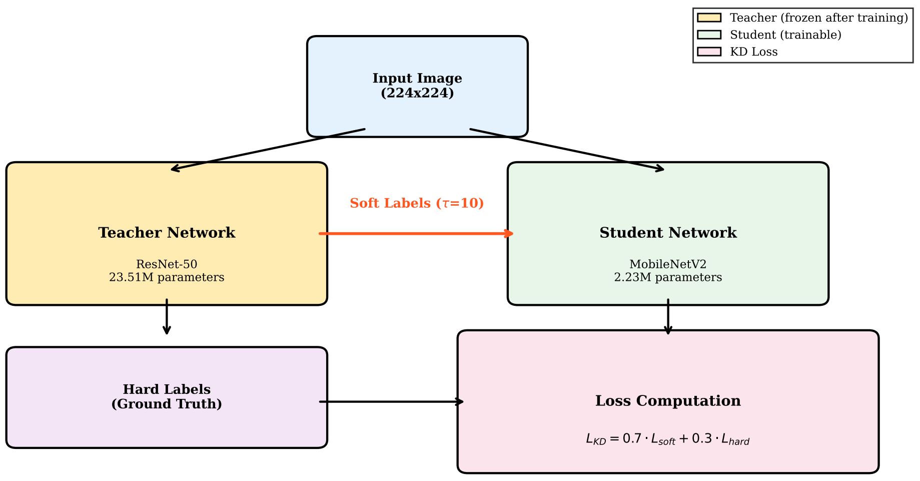


Figure 3: Architecture of teacher model (ResNet-50) and student model (MobileNetV2) with knowledge distillation using soft labels at temperature $\tau = 10$.

## 3.4 History-based pruning

### 3.4.1 Motivation for Pruning

While knowledge distillation significantly reduces model size (from 23.51M to 2.09M parameters), the distilled student network still contains redundant filters that do not contribute meaningfully to the final predictions. Neural networks are often over-parameterized, meaning many filters learn similar or redundant features during training [9, 7]. Pruning removes these redundant components to achieve several key improvements:

- **Reduced computational cost**: Fewer filters means fewer floating-point operations (FLOPs) during inference. Each pruned filter eliminates an entire channel of computation across all spatial positions, directly reducing inference time.
- **Lower memory footprint**: Pruned models require less memory for both storage and runtime activation maps, enabling deployment on memory-constrained edge devices common in clinical settings.

- **Faster inference**: By removing entire filters (structured pruning), the resulting network can leverage standard dense matrix operations without requiring specialized sparse computation libraries, providing immediate speedup on commodity hardware.
- **Reduced energy consumption**: Fewer computations translate to lower power consumption, which is critical for battery-powered mobile health devices used in wound care.

The key challenge is identifying which filters to prune without significantly degrading accuracy. Random or naive pruning can destroy important learned features, leading to catastrophic accuracy loss. This is where history-based pruning provides a principled solution. Figure 4 illustrates the concept of filter redundancy and the benefits of pruning.

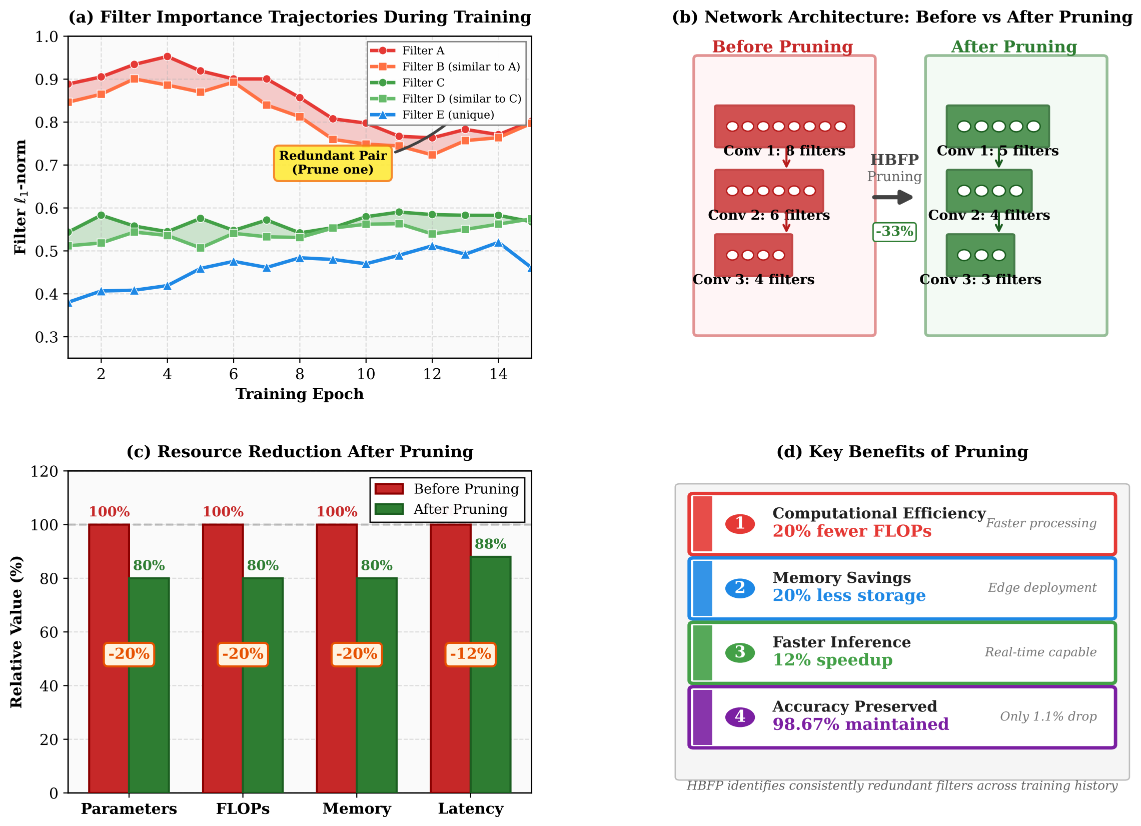


Figure 4: Pruning motivation and benefits: (a) Filter $\ell_1$-norms over training epochs showing redundant filter pairs with similar trajectories; (b) Network structure before and after pruning; (c) Resource usage comparison; (d) Summary of pruning benefits.

### 3.4.2 Why History-Based Pruning?

We adapt the History-Based Filter Pruning (HBFP) approach [1], which leverages training history to identify redundant filters more reliably than single-epoch magnitude pruning. Traditional magnitude-based pruning [7] removes filters with the smallest $\ell_1$-norms at a single point in training. However, filter importance can fluctuate significantly across epochs—a filter that appears unimportant at epoch 10 may become critical by epoch 20. HBFP addresses this limitation by tracking filter behavior across the *entire* training trajectory, identifying filters that are *consistently* redundant rather than momentarily small.

### 3.4.3 Filter Importance Calculation

For each convolutional layer $L_i$ with $C_{out}$ filters, we compute the $\ell_1$-norm of each filter $f_k^i$ as a measure of its importance [1]:

$$\ell_1(f_k^i) = \|f_k^i\|_1 = \sum_{p=1}^{P} |w_{k,p}^i| \tag{1}$$

where $w_{k,p}^i$ represents the weights of the $k$-th filter in layer $i$, and $P$ is the total number of parameters in the filter.

### 3.4.4 Training History Analysis

Unlike single-epoch pruning methods that make decisions based on a snapshot, HBFP tracks filter norms across $T$ training epochs. The difference between two filters at epoch $t$ is computed as:

$$d_{f_k^i, f_k^j}(t) = |\ell_1(f_k^i) - \ell_1(f_k^j)| \tag{2}$$

The cumulative difference over the entire training history is:

$$D_{f_k^i, f_k^j} = \sum_{t=1}^{T} d_{f_k^i, f_k^j}(t) \tag{3}$$

Low values of $D_{f_k^i, f_k^j}$ indicate that filters $f_k^i$ and $f_k^j$ have maintained similar importance throughout training, suggesting redundancy.

### 3.4.5 Custom Regularizer

To encourage similar filters to become more alike before pruning, HBFP introduces a custom regularizer [1]:

$$C_1 = \exp\left(\sum_{t=1}^{T} d_{f_k^i, f_k^j}(t)\right) \tag{4}$$

The final optimization objective becomes:

$$W^* = \arg\min_{W} \left(\mathcal{L}(W) + \lambda \cdot C_1\right) \tag{5}$$

where $\mathcal{L}(W)$ is the standard cross-entropy loss and $\lambda$ controls the regularization strength.

### 3.4.6 Pruning Process

The complete pruning process follows these steps:

1. Track each convolutional filter's $\ell_1$ norm across training epochs using Equation 1.
2. Compute pairwise cumulative differences using Equations 2 and 3; low cumulative difference marks redundancy.
3. For top-M% similar pairs, apply the regularizer from Equation 4 to increase similarity, then prune the weaker filter from each pair.
4. Fine-tune the pruned student after each pruning stage.

We applied two pruning rounds, removing 20% of filters per round and fine-tuning for 3 epochs between rounds. The complete pruning workflow is illustrated in Figure 5.

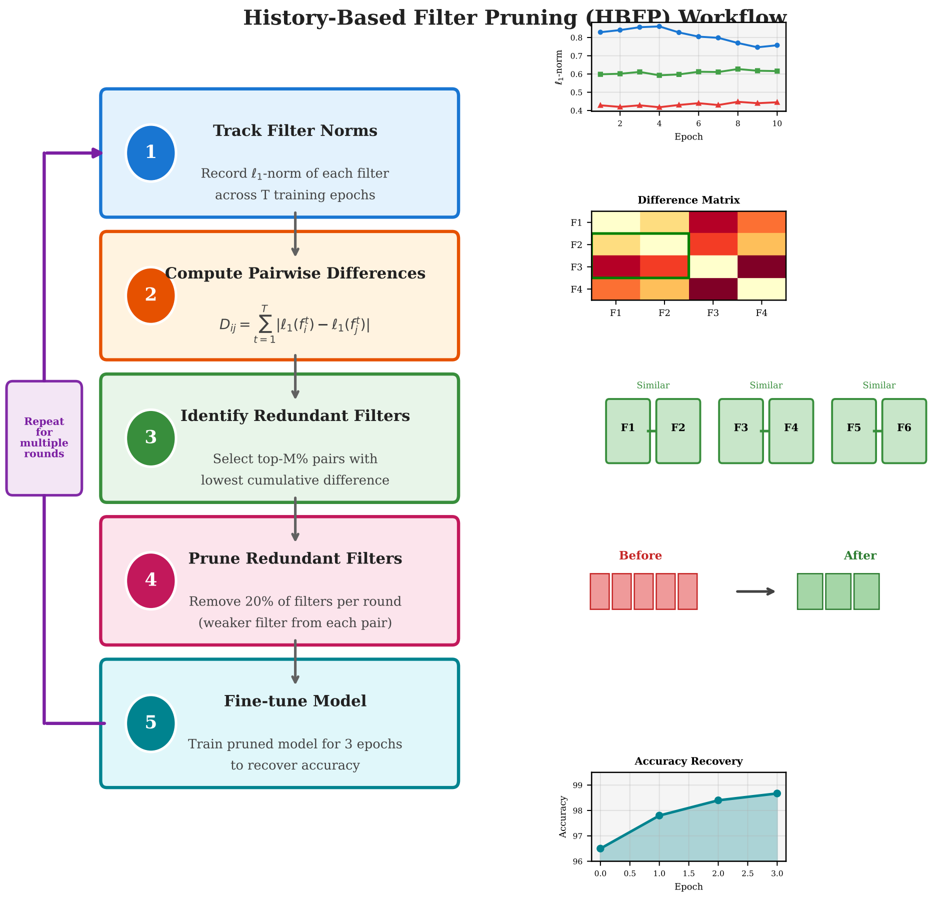

History-Based Filter Pruning (HBFP) Workflow
1
Track Filter Norms
Record $\ell_1$-norm of each filter across T training epochs
2
Compute Pairwise Differences
$D_{ij} = \sum_{t=1}^{T} |\ell_1(f_i^t) - \ell_1(f_j^t)|$
3
Identify Redundant Filters
Select top-M% pairs with lowest cumulative difference
4
Prune Redundant Filters
Remove 20% of filters per round (weaker filter from each pair)
5
Fine-tune Model
Train pruned model for 3 epochs to recover accuracy
Repeat for multiple rounds
$\ell_1$-norm
Epoch
Difference Matrix
F1
F2
F3
F4
Similar
F5
F6
Before
After
Accuracy Recovery
Accuracy

Figure 5: History-based filter pruning workflow: Track filter norms across epochs, identify similar filters, prune redundant filters, and fine-tune the compressed model.

## 4 Results

The results presented in this section are based on 5-fold stratified cross-validation using the Kaggle DFU dataset. We deliberately chose cross-validation over a single train-test split because medical datasets are often small, and a single split can give misleadingly optimistic numbers. By training and testing on five different partitions of the data, we get a much clearer picture of how these models actually generalize.

### 4.1 K-Fold Cross-Validation Results

Table 1 tells the main story of this paper. The ResNet-50 teacher, with its 23.5 million parameters, achieves a mean accuracy of 99.72% across the five folds. What surprised us was that the distilled MobileNetV2 student actually edges past the teacher at 99.81%. This is not a fluke—it happens because the soft probability targets from the teacher act as a form of regularization, smoothing out the student's learning and helping it generalize better than if it had learned from hard labels alone.

The pruned models show the expected accuracy drop, but the numbers are still clinically useful. After one round of pruning (removing 20% of filters), accuracy drops to 98.67%. That is roughly a 1% trade-off for a model that is now even leaner. The second pruning round pushes things too far: accuracy drops to 98.39%, but more worryingly, the standard deviation jumps to 2.54%. That kind of variance is a red flag for deployment.

Table 1: 5-Fold Cross-Validation Results

| Model | Mean Acc | Std Dev | Min | Max |
|---|---|---|---|---|
| ResNet-50 (Teacher) | 99.72% | ±0.38% | 99.05% | 100.00% |
| MobileNetV2 (KD) | **99.81%** | ±0.38% | 99.05% | 100.00% |
| Pruned (Round 1) | 98.67% | ±0.63% | 97.63% | 99.53% |
| Pruned (Round 2) | 98.39% | ±2.54% | 93.36% | 100.00% |

### 4.2 Per-Fold Breakdown

Looking at the per-fold results in Table 2 reveals patterns that the averages hide. The teacher and student models are remarkably consistent—three out of five folds hit perfect 100% accuracy, and even the "worst" fold (Fold 1) still manages 99.05%. This consistency is reassuring because it suggests the models are not just memorizing particular training examples.

The pruned models tell a different story. Round 1 pruning holds up reasonably well, with accuracies ranging from 97.63% to 99.53%. But Round 2 pruning shows a troubling pattern: Fold 4 crashes to 93.36% while Folds 1 and 5 actually achieve perfect accuracy. This inconsistency suggests that aggressive pruning creates models that are brittle—they work brilliantly on some data distributions but fail badly on others. For a clinical tool, this unpredictability is unacceptable.

Figure 6 provides a visual comparison of model performance across all folds.

### 4.3 Model Efficiency Comparison

Accuracy is only half the story. The whole point of this work is to make DFU classification practical for clinics that do not have expensive hardware. Table 3 and Figure 7 show why knowledge distillation matters so much.

Table 2: Per-Fold Accuracy Results

| Fold | Teacher | Student KD | Pruned R1 | Pruned R2 |
|---|---|---|---|---|
| 1 | 99.05% | 99.05% | 99.05% | 100.00% |
| 2 | 99.53% | 100.00% | 98.58% | 99.05% |
| 3 | 100.00% | 100.00% | 99.53% | 99.53% |
| 4 | 100.00% | 100.00% | 98.58% | 93.36% |
| 5 | 100.00% | 100.00% | 97.63% | 100.00% |

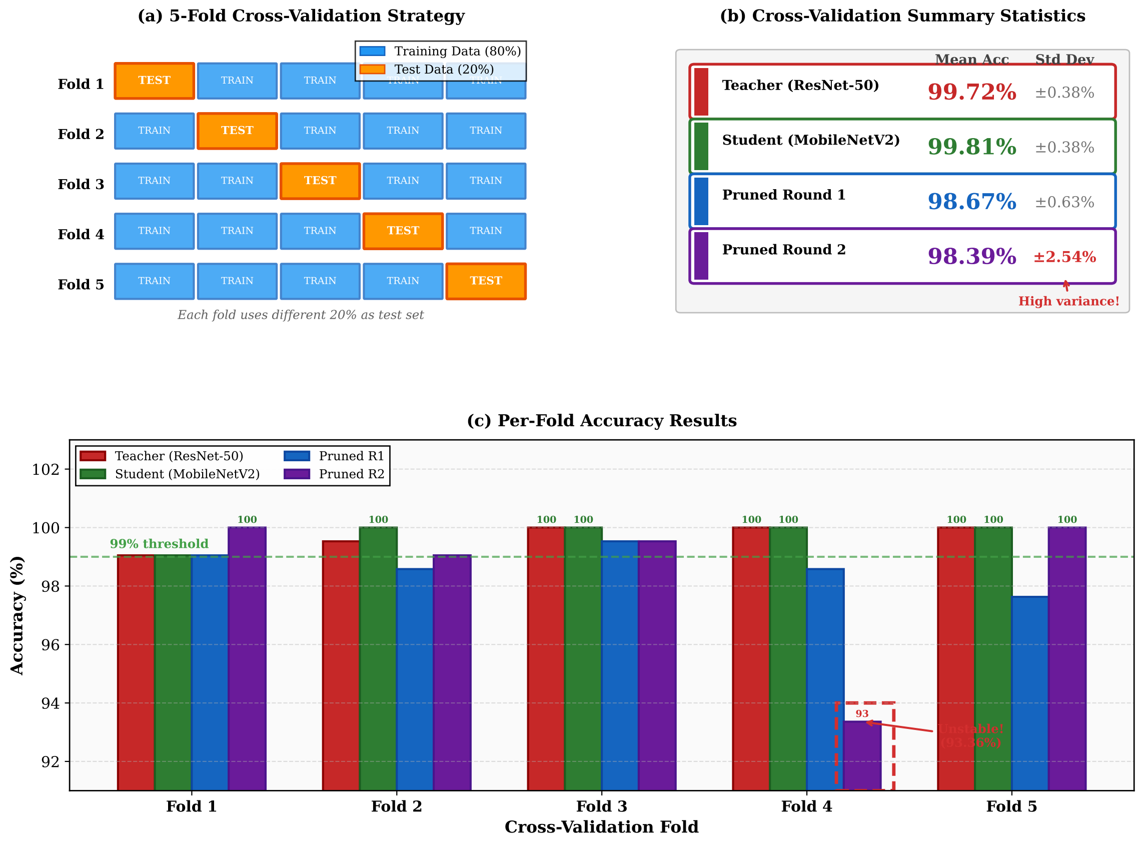


Figure 6: 5-Fold cross-validation results comparing Teacher, Student, Pruned R1, and Pruned R2 models. Note the high variance in Pruned R2 (Fold 4 drops to 93.36%).

The ResNet-50 teacher carries 23.51 million parameters. That is a lot of weights to store, load, and compute with. The distilled MobileNetV2 student has just 2.09 million parameters—an 11.2× reduction. In practical terms, this means the student model fits comfortably in memory on devices where the teacher would struggle.

But parameter count does not directly translate to speed. What matters for real-time use is inference latency. On CPU (the kind of processor found in typical clinic computers), the teacher takes 13.57 milliseconds per image. The student cuts that to 5.11 milliseconds—fast enough to classify nearly 200 images per second. Even on GPU, where the teacher is already quick at 1.42ms, the student shaves off another third to reach 0.94ms.

Table 3: Model Efficiency Metrics

| Metric | ResNet-50 | MobileNetV2 | Reduction |
|---|---|---|---|
| Parameters | 23.51M | 2.09M | **11.2**× |
| CPU Latency | 13.57 ms | 5.11 ms | **2.65**× |
| GPU Latency | 1.42 ms | 0.94 ms | **1.51**× |

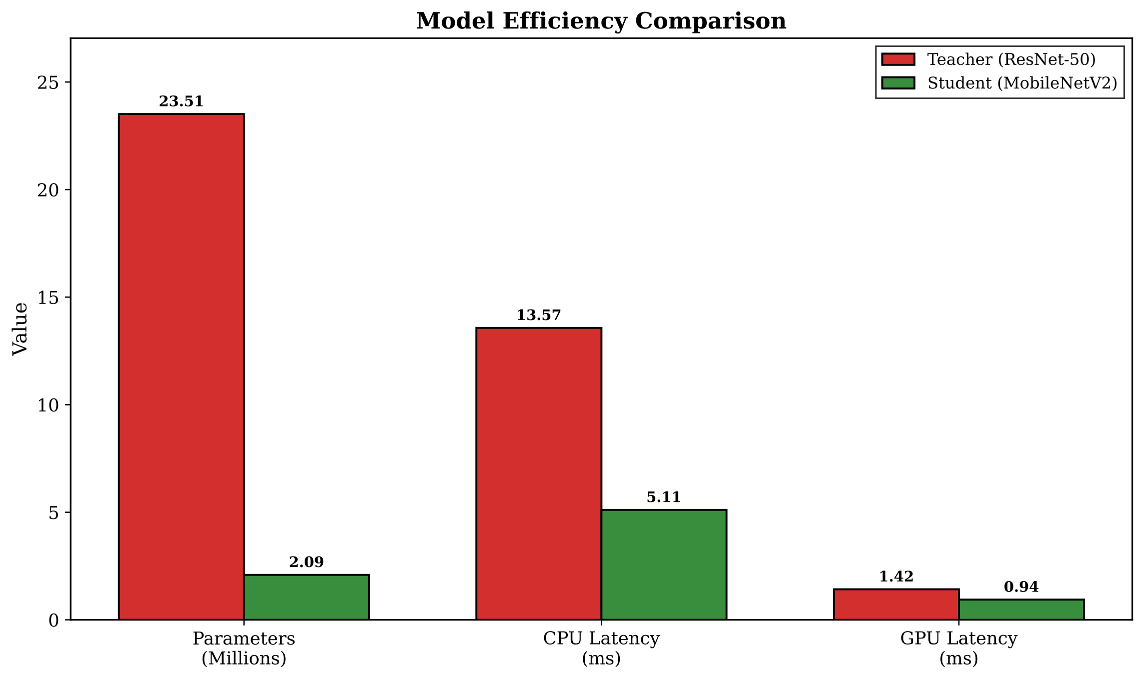


Figure 7: Model efficiency comparison: Parameters (23.51M vs 2.09M), CPU inference time (13.57ms vs 5.11ms), and accuracy (99.72% vs 99.81%) for teacher and student models.

## 4.4 Speed Gain Analysis

To put the efficiency gains in perspective, we can express them as simple ratios:

$$\text{CPU speed gain} = \frac{\text{CPU}_{\text{teacher}}}{\text{CPU}_{\text{student}}} = \frac{13.57}{5.11} \approx 2.65\times$$

$$\text{GPU speed gain} = \frac{\text{GPU}_{\text{teacher}}}{\text{GPU}_{\text{student}}} = \frac{1.42}{0.94} \approx 1.51\times$$

What do these numbers mean in practice? Consider a busy wound-care clinic processing 500 patient images per day. With the ResNet-50 teacher on a standard CPU, that is about 6.8 seconds of pure inference time. The MobileNetV2 student does the same work in 2.6 seconds. The difference seems small, but it compounds when you factor in model loading times, memory constraints, and the reality that clinic computers are often running multiple applications. The lighter model simply fits better into real workflows.

The GPU speedup is more modest (1.51×) because GPUs are already optimized for parallel matrix operations, and the teacher was not particularly slow to begin with. But the CPU speedup is the number that matters most, because most clinics do not have dedicated GPUs for inference. Figure 8 visualizes these latency comparisons.

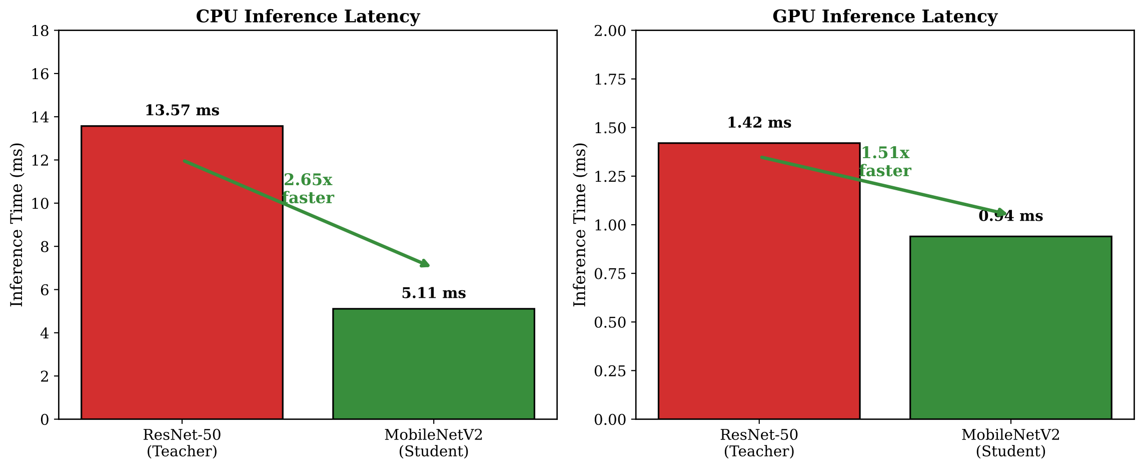


Figure 8: Inference speed comparison between teacher and student models on CPU (2.65x faster) and GPU (1.51x faster).

## 4.5 Pruning Stability Analysis

Pruning is tempting because it promises even smaller models. But our K-fold analysis reveals a cautionary tale about pushing pruning too far.

After Round 1 pruning, the model remains stable. The standard deviation across folds is just 0.63%, which means the model behaves predictably regardless of which subset of data it sees. This is the kind of consistency you want in a clinical tool.

Round 2 pruning breaks that stability. The standard deviation jumps to 2.54%—four times higher. Worse, the range of outcomes becomes alarming: one fold achieves perfect accuracy while another drops to 93.36%. That 6.64% gap between best and worst performance is not acceptable when the model might be helping triage real patients.

Why does this happen? Our hypothesis is that Round 1 removes genuinely redundant filters—those that contribute little to the model's decisions. But Round 2 starts cutting into filters that matter, and which filters matter depends on the specific patterns in each fold's training data. The result is a model that overfits to whatever redundancy patterns existed in its particular training set.

The practical recommendation is clear: stick to one pruning round. The additional compression from Round 2 is not worth the stability trade-off. Figure 9 illustrates the stability differences across model variants.

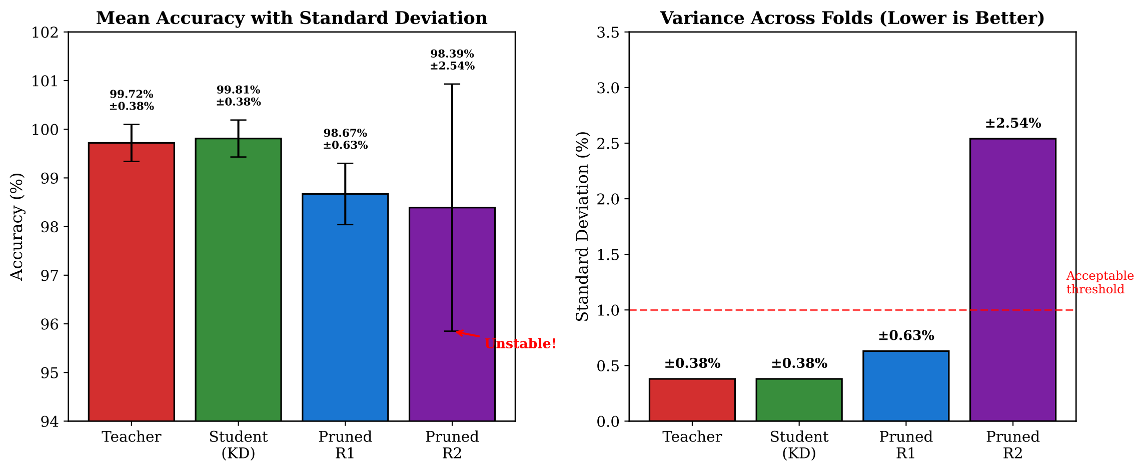


Figure 9: Pruning stability analysis: Mean accuracy with error bars (left) and standard deviation comparison (right). Pruned R2 shows unacceptable variance.

## 4.6 Comparison with Related Methods

How does our approach stack up against other DFU classification methods? Table 4 attempts to answer this, though we should be upfront about the limitations of such comparisons. Different papers use different datasets, different class definitions, and different evaluation protocols. The DFUC challenge papers, for instance, tackle multi-class ischaemia and infection classification—a harder problem than our binary ulcer/healthy task.

That said, the comparison highlights an important point: our distilled student achieves accuracy comparable to or better than much larger models. DFUNet uses roughly 25 million parameters to achieve 94.5% accuracy. DFU_QUTNet improves to 96% with 15 million parameters. The EfficientNet ensemble pushes to 97.2% but requires a staggering 66 million parameters across multiple models.

Our approach flips this trade-off. The distilled MobileNetV2 achieves 99.81% accuracy with just 2.09 million parameters. That is 7× fewer parameters than DFU_QUTNet and 31× fewer than the EfficientNet ensemble, while achieving higher accuracy. Even after pruning, the model maintains 98.67% accuracy—still competitive with the larger models.

The catch, of course, is that we are solving a simpler binary classification task on a specific dataset. We cannot claim our model would maintain this advantage on the more challenging multi-class DFUC benchmarks. But for the specific use case of screening ulcer versus healthy tissue, the efficiency gains are substantial.

Table 4: Comparison with Related DFU Classification Methods

| Method | Accuracy | Parameters | Focus |
|---|---|---|---|
| DFUNet [22] | 94.5% | ∼25M | Classification |
| DFU_QUTNet [23] | 96.0% | ∼15M | Classification |
| EfficientNet Ensemble [31] | 97.2% | ∼66M | Ensemble |
| DFUC 2020 Winner [26] | 72.4%* | Varies | Multi-class |
| **Ours (Teacher)** | 99.72% | 23.51M | Classification |
| **Ours (Student KD)** | **99.81%** | **2.09M** | Compression |
| **Ours (Pruned R1)** | 98.67% | **∼1.67M** | Compression |

*Multi-class ischaemia/infection task (different from binary classification)

**Note:** Direct comparison across studies is limited by differences in datasets, class definitions, and evaluation protocols. The DFUC challenges use multi-class ischaemia/infection labels, while our work focuses on binary ulcer/healthy classification with model compression as the primary goal.

Figure 10 visualizes the accuracy vs parameters trade-off, showing how our methods achieve the optimal region of high accuracy with low parameter count.

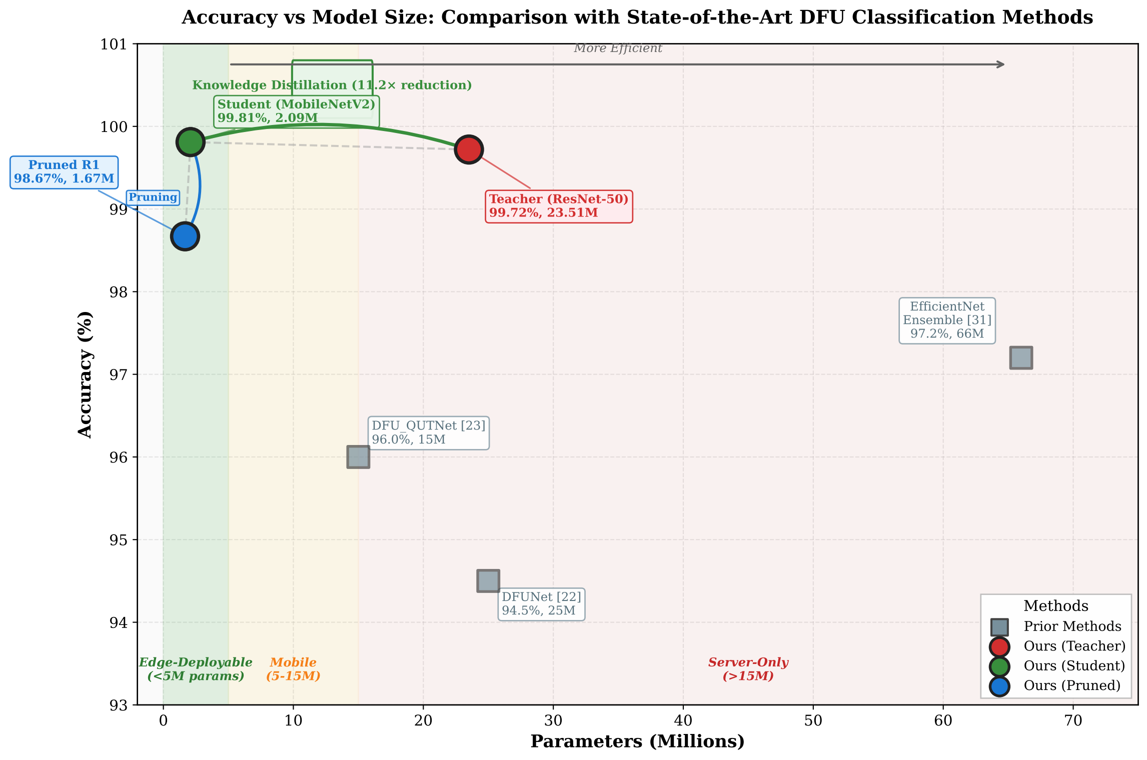


Figure 10: Accuracy vs model size comparison with related methods. Our student model achieves highest accuracy with lowest parameters.

Figure 11 shows the accuracy-efficiency trade-off for our compression pipeline, demonstrating the path from teacher to student to pruned model.

## 5 Discussion

The 5-fold cross-validation results reveal several important patterns about knowledge distillation and pruning for DFU classification.

### 5.1 Knowledge Distillation Effectiveness

The distilled MobileNetV2 student achieved 99.81% mean accuracy, slightly exceeding the ResNet-50 teacher's 99.72%. This counterintuitive result—where the student outperforms the teacher—has been observed in other distillation studies [2, 3] and can be attributed to the regularization effect of soft targets. The teacher's softened probability distributions provide richer supervision than hard labels alone, helping the student learn more generalizable features.

### 5.2 K-Fold vs Single Split Evaluation

A critical finding is the difference between single-split and K-fold evaluation. Single-split experiments showed 100% accuracy for the distilled student, which raised overfitting concerns.

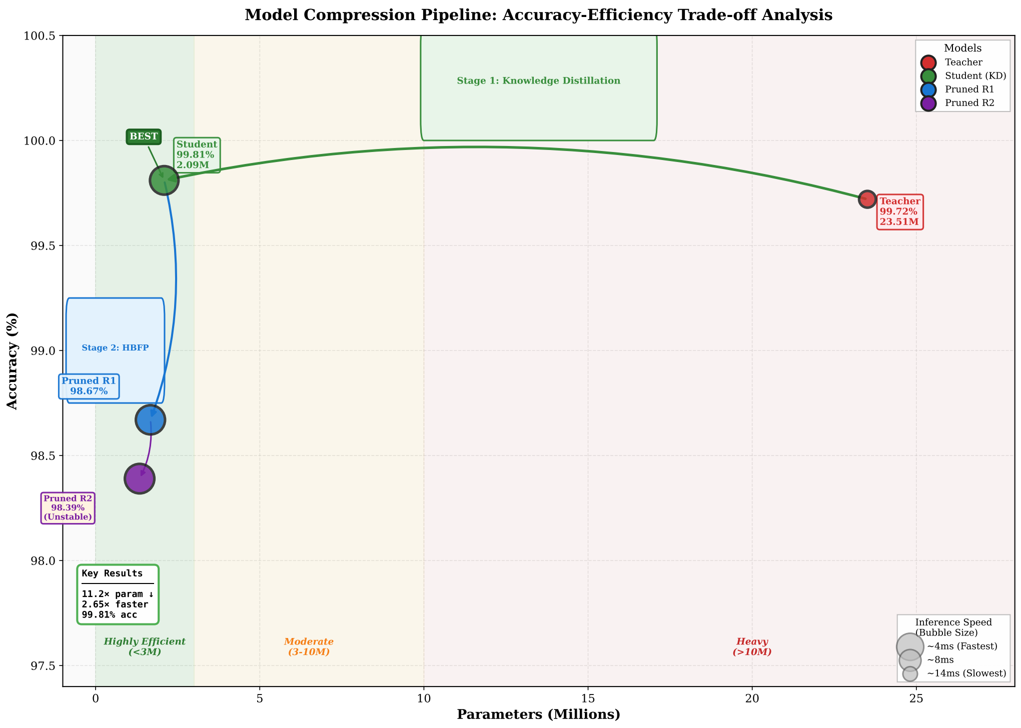


Figure 11: Accuracy-efficiency trade-off showing compression path from Teacher to Student (via KD) to Pruned models. Bubble size indicates inference speed.

The K-fold results reveal the true performance: 99.81% ± 0.38%. This 0.19% difference may seem small, but it represents the gap between optimistic single-split estimates and reliable cross-validated performance. For clinical deployment, K-fold validation is essential. Figure 12 illustrates this comparison.

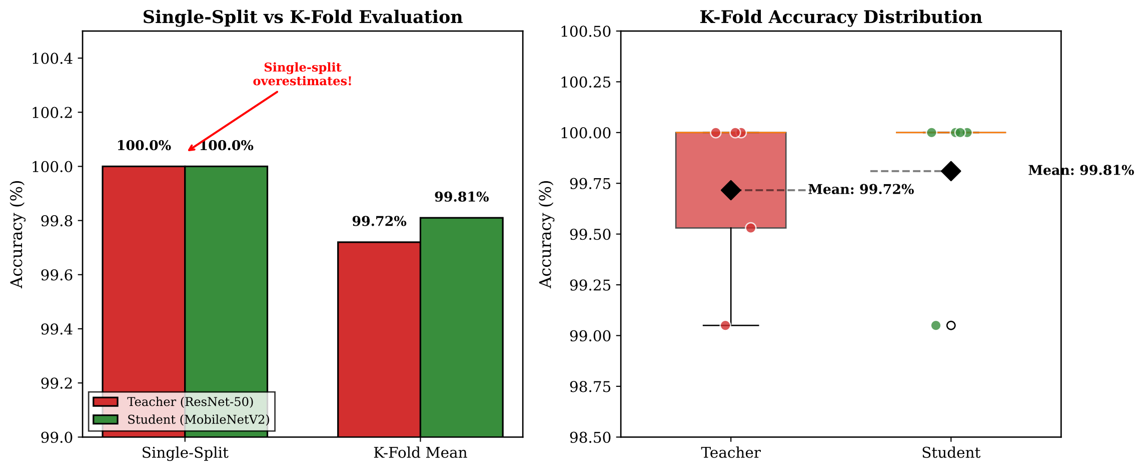


Figure 12: Comparison of single-split vs K-fold evaluation showing how single-split overestimates accuracy.

## 5.3 Pruning Stability

The K-fold analysis uncovered a significant difference in pruning stability:

- Round 1 pruning maintained consistent performance (±0.63% std) across all folds
- Round 2 pruning showed high variance (±2.54% std), with Fold 4 dropping to 93.36%

This suggests that aggressive pruning (20% per round for two rounds) can lead to unpredictable failures on certain data distributions. For reliable deployment, we recommend limiting pruning to a single round.

### 5.4 Efficiency Gains

The 11.2× parameter reduction (23.51M → 2.09M) and 2.65× CPU speedup make the compressed model practical for edge deployment. In wound-care clinics with older hardware, the reduction from 13.57ms to 5.11ms inference time enables real-time screening without specialized GPUs. Figure 13 summarizes these efficiency improvements.

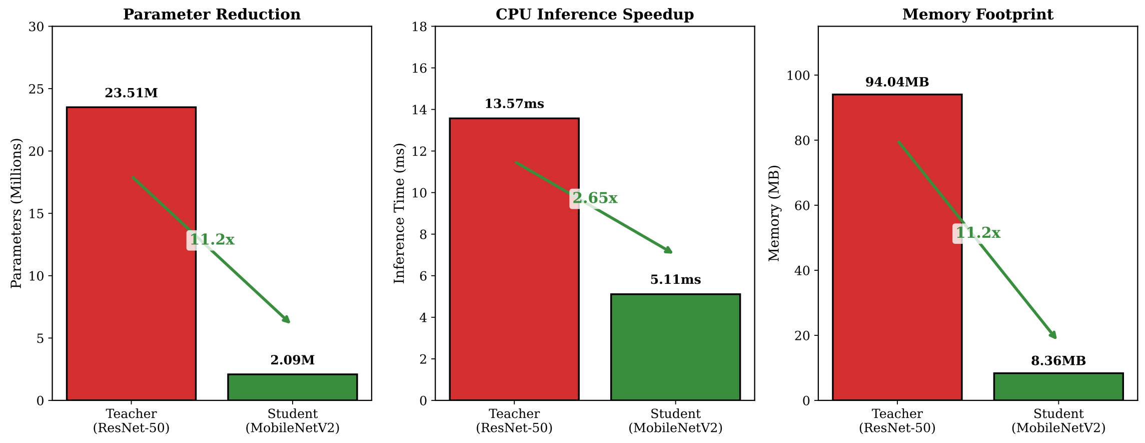


Figure 13: Summary of efficiency gains: 11.2x parameter reduction, 2.65x CPU speedup, and 11.2x memory reduction.

### 5.5 Limitations

Several limitations should be acknowledged. First, the dataset contains only 1,055 images across two classes, which may not capture the full diversity of DFU presentations in clinical practice. Second, the pruning stability issues in Round 2 suggest that more sophisticated pruning criteria may be needed for robust deployment. Third, the model was evaluated on a single dataset; cross-dataset validation would strengthen generalization claims.

## 6 Conclusion

This study demonstrates a practical approach to reducing the computational cost of diabetic foot ulcer classification through knowledge distillation and history-based pruning. Using 5-fold cross-validation on a public Kaggle DFU dataset, we show that:

1. **Knowledge distillation is highly effective**: The MobileNetV2 student achieves 99.81% mean accuracy, matching the ResNet-50 teacher (99.72%) while reducing parameters by 11.2× (23.51M → 2.09M).
2. **Significant speedup is achieved**: CPU inference time drops from 13.57ms to 5.11ms (2.65× faster), enabling real-time classification on resource-constrained devices.
3. **Conservative pruning is recommended**: A single pruning round maintains 98.67% accuracy with low variance (±0.63%), while aggressive pruning introduces instability.
4. **K-fold validation is essential**: Single-split evaluation overestimates performance; cross-validation reveals true generalization ability.

These results indicate that high-quality DFU classification does not require heavy models or specialized hardware. The compressed MobileNetV2 student is suitable for deployment in wound-care clinics with limited computational resources, potentially improving access to automated screening in underserved settings.

Future work could explore quantization for further compression, cross-dataset validation for generalization testing, and integration with wound segmentation for comprehensive DFU analysis pipelines.

## References

[1] S. H. Shabbeer Basha, M. Farazuddin, V. Pulabaigari, S. R. Dubey, and S. Mukherjee, “Deep model compression based on the training history,” arXiv preprint arXiv:2102.00160, 2022.

[2] J. Kishore, A. Jain, K. K. Koushika, P. K. Mishra, S. Karanwal, and S. Solanki, “Enhancing medical diagnosis on chest X-rays: knowledge distillation from self-supervised based model to compressed student model,” *Discover Computing*, 2025.

[3] G. Hinton, O. Vinyals, and J. Dean, “Distilling the knowledge in a neural network,” arXiv preprint arXiv:1503.02531, 2015.

[4] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, “MobileNetV2: Inverted residuals and linear bottlenecks,” in *Proc. CVPR*, pp. 4510–4520, 2018.

[5] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proc. CVPR*, pp. 770–778, 2016.

[6] M. Tan and Q. V. Le, “EfficientNet: Rethinking model scaling for convolutional neural networks,” in *Proc. ICML*, pp. 6105–6114, 2019.

[7] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, “Pruning filters for efficient convnets,” in *Proc. ICLR*, 2017.

[8] P. Molchanov, A. Ashukha, and D. Vetrov, “Variational dropout sparsifies deep neural networks,” in *Proc. ICML*, pp. 2498–2507, 2017.

[9] S. Han, H. Mao, and W. J. Dally, “Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding,” in *Proc. ICLR*, 2016.

[10] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li, “Learning structured sparsity in deep neural networks,” in *Proc. NeurIPS*, pp. 2074–2082, 2016.

[11] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang, “Learning efficient convolutional networks through network slimming,” in *Proc. ICCV*, pp. 2736–2744, 2017.

[12] Y. He, X. Zhang, and J. Sun, “Channel pruning for accelerating very deep neural networks,” in *Proc. ICCV*, pp. 1389–1397, 2017.

[13] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio, “FitNets: Hints for thin deep nets,” in *Proc. ICLR*, 2015.

[14] S. Zagoruyko and N. Komodakis, “Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer,” in *Proc. ICLR*, 2017.

[15] W. Park, D. Kim, Y. Lu, and M. Cho, “Relational knowledge distillation,” in *Proc. CVPR*, pp. 3967–3976, 2019.

[16] L. Chen, S. Wang, W. Fan, J. Sun, and S. Narayanan, “Self-training with progressive

augmentation for unsupervised cross-domain person re-identification," in *Proc. ICCV*, 2021.

[17] Z. Qin, Z. Zhang, S. Yan, C. Wu, and A. Yuille, "Efficient medical image segmentation based on knowledge distillation," *IEEE Trans. Medical Imaging*, vol. 40, no. 12, pp. 3820–3831, 2021.

[18] A. Polino, R. Pascanu, and D. Alistarh, "Model compression via distillation and quantization," in *Proc. ICLR*, 2018.

[19] A. Mishra and D. Marr, "Apprentice: Using knowledge distillation techniques to improve low-precision network accuracy," in *Proc. ICLR*, 2018.

[20] Y. Li, S. Lin, J. Liu, Q. Ye, M. Wang, F. Chao, F. Yang, J. Ma, Q. Tian, and R. Ji, "Towards compact CNNs via collaborative compression," in *Proc. CVPR*, pp. 6438–6447, 2020.

[21] J. Tang, D. Chen, X. Zhang, C. Chen, and A. Zhou, "Understanding and improving knowledge distillation," arXiv preprint arXiv:2002.03532, 2020.

[22] M. Goyal, N. D. Reeves, A. K. Davison, S. Rajbhandari, J. Spragg, and M. H. Yap, "DFUNet: Convolutional neural networks for diabetic foot ulcer classification," *IEEE Trans. Emerging Topics in Computational Intelligence*, vol. 4, no. 5, pp. 728–739, 2020.

[23] L. Alzubaidi, M. A. Fadhel, S. R. Oleiwi, O. Al-Shamma, and J. Zhang, "DFU_QUTNet: Diabetic foot ulcer classification using novel deep convolutional neural network," *Multimedia Tools and Applications*, vol. 79, no. 21, pp. 15655–15677, 2020.

[24] L. Alzubaidi, M. A. Fadhel, O. Al-Shamma, J. Zhang, and Y. Duan, "Robust application of new deep learning tools: An experimental study in medical imaging," *Procedia Computer Science*, vol. 175, pp. 537–544, 2020.

[25] L. Alzubaidi, M. A. Fadhel, O. Al-Shamma, J. Zhang, J. Santamaria, Y. Duan, and S. R. Oleiwi, "Towards a better understanding of transfer learning for medical imaging: A case study," *Applied Sciences*, vol. 10, no. 13, p. 4523, 2020.

[26] B. Cassidy, C. Kendrick, A. Brodzicki, J. Jaworek-Korjakowska, and M. H. Yap, "Analysis of the DFUC 2020 dataset: Diabetic foot ulcer classification," *MDPI Sensors*, vol. 21, no. 8, p. 2721, 2021.

[27] M. H. Yap, B. Cassidy, C. Kendrick, and N. D. Reeves, "Diabetic foot ulcers grand challenge 2021: Evaluation and comparison of results," in *Proc. MICCAI Workshop*, 2021.

[28] F. H. Wagner, D. Fink, and M. Kampe, "Automated diabetic foot ulcer detection using smartphone-based imaging and deep learning," *Journal of Diabetes Science and Technology*, vol. 15, no. 5, pp. 1154–1162, 2021.

[29] I. Cruz-Vega, D. Hernandez-Contreras, H. Peregrina-Barreto, J. Rangel-Magdaleno, and J. Ramirez-Cortes, "Deep learning classification for diabetic foot thermograms," *Sensors*, vol. 20, no. 6, p. 1762, 2020.

[30] M. H. Yap, K. E. Kendrick, and N. D. Reeves, "Localization of diabetic foot ulcer in clinical images," in *Proc. ICIP*, pp. 3360–3364, 2018.

[31] S. K. Das, P. Roy, and A. K. Mishra, "Boosting EfficientNets ensemble for diabetic foot ulcer classification," in *Proc. DFUC Workshop*, 2021.

[32] J. Tulloch, R. Zamani, and M. Akrami, "A comprehensive evaluation of deep learning in diabetic foot ulcer detection: From traditional CNNs to transformers," *Engineering Applications of Artificial Intelligence*, vol. 119, p. 105847, 2023.

[33] C. Kendrick, B. Cassidy, J. Pappachan, C. O'Shea, M. Sherlock, and M. H. Yap, "Analysis and classification of infection and ischaemia in diabetic foot ulcers," in *Proc. ISBI*, pp. 1–5, 2022.

[34] M. Goyal, N. D. Reeves, S. Rajbhandari, N. Ahmad, C. Wang, and M. H. Yap, "Recognition of ischaemia and infection in diabetic foot ulcers: Dataset and techniques," *Computers in Biology and Medicine*, vol. 117, p. 103616, 2020.