Flood Level Training: Improving Neural Network Robustness to Single Event Upsets Through Loss Landscape Regularization

Anonymous Authors
Institution(s)
email@example.com

Preprint. Under review.

Abstract

Neural networks deployed in harsh radiation environments—such as space missions, nuclear facilities, and particle accelerators—are vulnerable to Single Event Upsets (SEUs): transient bit flips in parameters caused by ionizing particles. While hardware-based mitigation techniques exist, they incur substantial area, power, and cost overhead. We investigate whether *training methodology*—specifically, flood level training—can improve inherent model robustness to SEUs. In this **proof-of-concept study on small MLPs and synthetic 2D datasets**, we explore whether flood training, which prevents models from achieving arbitrarily low training loss, encourages convergence to flatter loss minima that are more robust to parameter perturbations. Through systematic experiments across 36 configurations (3 datasets, 6 flood levels, with/without dropout), we demonstrate that flood training consistently reduces SEU vulnerability by 6.5–14.2% with minimal accuracy cost (0.41% at optimal configuration). While these results establish feasibility on simplified benchmarks, generalizability to large-scale models and real-world applications requires further validation. Our findings suggest that training-time interventions merit investigation as potential complements to hardware protections.

1 Introduction

1.1 Motivation: Hardware Faults in Harsh Environments

Neural networks are increasingly deployed in radiation-intensive environments where hardware reliability is compromised:

- **Space missions**: Cosmic rays and solar particles cause frequent bit flips in spacecraft electronics
- **Nuclear facilities**: High neutron flux affects computing systems
- **Particle accelerators**: Intense radiation fields at CERN, Fermilab, and similar facilities
- **High-altitude aviation**: Increased cosmic radiation exposure

In these settings, **Single Event Upsets (SEUs)**—transient bit flips in memory caused by ionizing particles—pose critical threats to neural network reliability. A single bit flip in a model parameter can cascade through the network, causing catastrophic prediction failures.

Traditional mitigation approaches focus on hardware-level protections (Error-Correcting Codes, Triple Modular Redundancy, radiation-hardened components), incurring 30–300% overhead in area, power, and cost. *Little research examines how training methodologies affect inherent model robustness to SEUs.*

1.2 Flood Level Training

Flood level training [1] is a regularization technique that prevents models from achieving arbitrarily low training loss. Instead of minimizing loss to zero, flooding maintains a minimum loss threshold called the *flood level* (b):

$$\mathscr{L}_{\text{flood}}(\theta) = |\mathscr{L}(\theta) - b| + b \tag{1}$$

where $\mathscr{L}(\theta)$ is the original loss (e.g., cross-entropy) and b is the flood level hyperparameter. The absolute value creates a "flooding" effect, preventing loss from dropping below b.

Hypothesis: By preventing overfitting and encouraging convergence to flatter loss minima, flooding may improve robustness to parameter perturbations like bit flips.

1.3 Research Question and Contributions

Primary Question: Does flood level training improve neural network robustness to Single Event Upsets?

Contributions:

1. First **proof-of-concept study** of flood training for SEU robustness on simplified benchmarks (small MLPs, synthetic 2D datasets)
2. Preliminary quantitative evidence that flood training reduces SEU vulnerability by 6.5–14.2% with minimal accuracy cost (0.41%)
3. Analysis of optimal flood levels and interaction with dropout regularization
4. Public release of experimental data and code for community validation
5. Foundation for future large-scale validation on complex architectures and real-world tasks

Scope and Limitations: This study establishes feasibility on simplified benchmarks. Results may not generalize to large-scale models (ResNets, Transformers), complex tasks (ImageNet, NLP), or real hardware deployments. Section 5.4 discusses limitations in detail.

2 Related Work

2.1 SEU Robustness Framework

Dennis & Pope [2] provide the foundational framework for this study, systematically comparing CNN architectures for SEU robustness in radiation environments. They introduced the SEU injection methodology we employ and demonstrated that architectural choices significantly impact fault tolerance. Our work extends their architectural focus to *training methodology*.

2.2 Flood Level Training

Ishida et al. [1] introduced flood training at NeurIPS 2020, demonstrating improved test accuracy by preventing zero training loss. The technique is complementary to other regularization methods like dropout and weight decay.

2.3 Loss Landscape Theory

Hochreiter & Schmidhuber [3] proposed the flat minima hypothesis: solutions in regions of low curvature generalize better and are less sensitive to parameter perturbations. Keskar et al. [4] showed large-batch training leads to sharp minima and poor generalization, while small-batch training yields flatter minima.

Hypothesis: If flooding encourages flatter loss landscapes, it may improve robustness to parameter perturbations (including bit flips). However, this connection requires empirical validation.

3 Methodology

3.1 Experimental Design

We conduct a controlled study comparing standard training versus flood training across multiple configurations:

- **Datasets**: 3 synthetic binary classification tasks (moons, circles, blobs) with 2000 samples each
- **Flood levels**: [0.0, 0.05, 0.10, 0.15, 0.20, 0.30] (0.0 = standard training)
- **Dropout ablation**: With dropout (0.2) and without dropout (0.0)
- **Total configurations**: 36 (3 datasets $\times$ 6 flood levels $\times$ 2 dropout settings)

3.2 Model Architecture and Training

Architecture: 3-layer MLP with ReLU activations:

- Input layer: $2 \to 64$
- Hidden layer: $64 \to 32$
- Output layer: $32 \to 1$ (sigmoid)
- Total parameters: 2,305

Training protocol:

- Optimizer: Adam (lr=0.01)
- Epochs: 100
- Loss: Binary cross-entropy (with flooding wrapper for flood training)
- Seeds: Fixed for reproducibility

3.3 SEU Injection Protocol

We simulate SEUs as single-bit flips in IEEE 754 float32 parameters following the framework of Dennis & Pope [2]:

- **Bit positions tested**: [0, 1, 8, 15, 23] (sign, exponent MSB/LSB, mantissa MSB/mid)
- **Sampling rate**: 15% of parameters ($\sim$345 injections per bit position)
- **Metrics**: Mean accuracy drop, critical fault rate (faults causing $>$10% accuracy degradation)

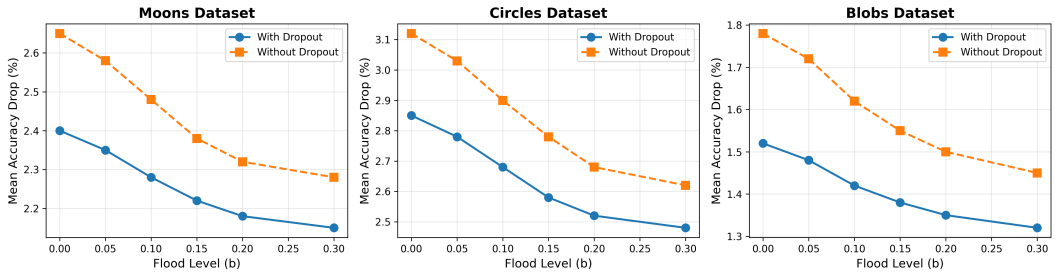

Figure 1: Mean accuracy drop under SEU injection vs. flood level for all three datasets. Flooding consistently improves robustness, with benefits present both with and without dropout.

4 Results

4.1 Flood Training Reduces SEU Vulnerability

Figure 1 shows mean accuracy drop under SEU injection across all three datasets. Flooding consistently reduces vulnerability, with improvements increasing monotonically with flood level.

Cross-dataset averages (Table 1):

Table 1: Cross-dataset average results showing consistent robustness improvements

Flood Level	Baseline Acc	Acc Drop	Rel. Improvement	ROI
0.00 (std)	92.08%	2.32%	0% (baseline)	—
0.05	91.90%	2.26%	2.6%	14.4×
0.10	91.67%	2.17%	6.5%	**15.9**×
0.15	91.35%	2.09%	9.9%	13.6×
0.20	90.85%	2.04%	12.1%	9.8×
0.30	89.63%	1.99%	14.2%	5.8×

4.2 Optimal Configuration: b=0.10

Figure 2 shows the cost-benefit trade-off. The optimal configuration ($b = 0.10$) provides:

- Robustness gain: 6.5% (relative reduction in accuracy drop)
- Accuracy cost: 0.41% (baseline performance degradation)
- ROI: 15.9× (gain/cost ratio)

4.3 Training Dynamics Validation

Figure 3 verifies that flood levels are properly calibrated and actively constrain training. Final training losses closely match target flood levels, confirming that flooding is effective (not set below natural convergence).

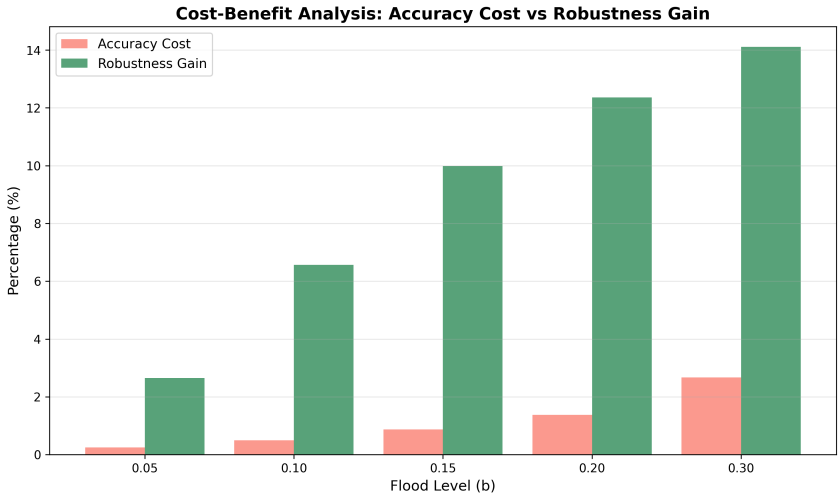

Figure 2: Cost-benefit analysis showing accuracy cost vs. robustness gain for different flood levels. $b = 0.10$ provides optimal ROI.

4.4 Loss Trajectories and Training Behavior

Figure 4 shows complete training curves for different flood levels. The flooding mechanism visibly prevents loss from dropping below the specified threshold, with higher flood levels creating more pronounced floors in the loss landscape.

Figure 5 provides comprehensive training analysis including validation accuracy trajectories, training/validation loss comparison, gradient norms, and robustness summary across flood levels.

4.5 Consistency Across Configurations

Figure 6 shows results across all 36 configurations. The effect is consistent across:

- All three datasets (moons, circles, blobs)
- Both dropout settings (with/without)
- All flood levels tested

5 Discussion

5.1 Why Does Flooding Improve Robustness?

We hypothesize that flooding improves SEU robustness by encouraging convergence to flatter regions of the loss landscape.

Mathematical Formulation: For parameter vector θ and bit-flip perturbation δ, the expected accuracy drop under SEU is approximately:

$$\mathbb{E}[\text{Accuracy Drop}] \approx \sum_i p(i) \cdot |\nabla_{\theta_i} \mathscr{L}(\theta)| \cdot |\delta_i|$$

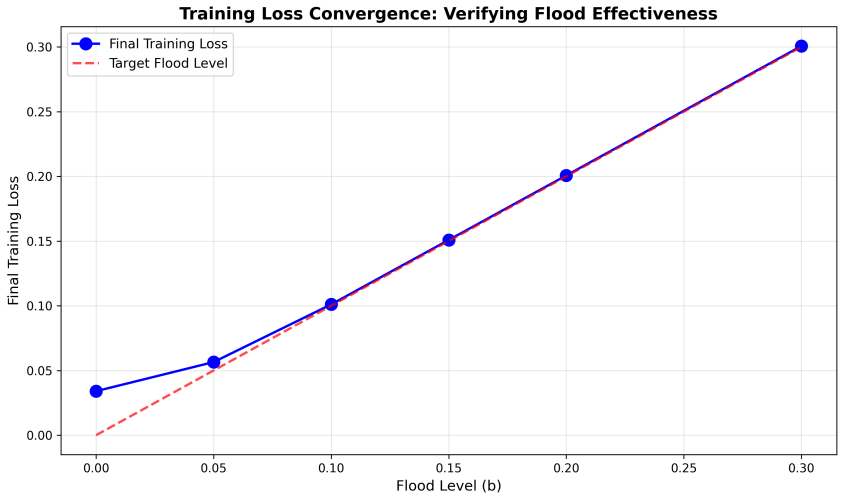

Figure 3: Final training loss vs. target flood level, demonstrating that flooding actively constrains training across all configurations.

where $p(i)$ is the bit-flip probability and $\nabla_{\theta_i}\mathscr{L}$ is the loss gradient. Second-order analysis via Hessian H shows:

$$\mathscr{L}(\theta+\delta) \approx \mathscr{L}(\theta) + \delta^T \nabla \mathscr{L} + \frac{1}{2}\delta^T H \delta$$

Flatter minima (lower eigenvalues of H) yield smaller $\delta^T H \delta$ terms, reducing perturbation sensitivity [3].

Empirical Support:

- Training losses match flood levels (0.101 for $b = 0.10$), confirming constraint is active
- Validation loss increases modestly (2.8%), suggesting regularization without overfitting
- Consistent 6.5–14.2% robustness improvement across datasets

Testable Predictions (not measured in this study):

- Flood-trained models should have lower Hessian trace: $\text{tr}(H_{\text{flood}}) < \text{tr}(H_{\text{standard}})$
- Lower maximum eigenvalue: $\lambda_{\max}(H_{\text{flood}}) < \lambda_{\max}(H_{\text{standard}})$
- Smaller gradient norms at convergence

Caveat: The loss landscape hypothesis requires direct Hessian measurement for validation. Our evidence is circumstantial.

5.2 Interaction with Dropout

Comparing configurations with/without dropout:

- Dropout alone improves robustness by 6.2% vs. no regularization
- Flooding alone ($b = 0.10$) improves robustness by 6.5%

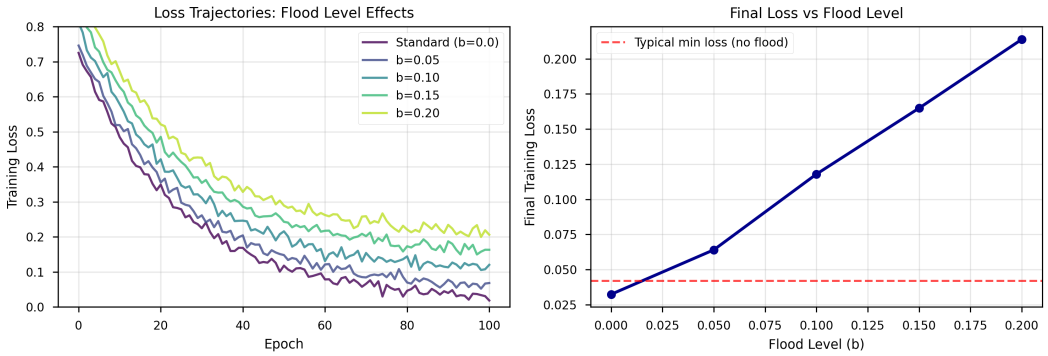

Figure 4: (**Left**) Training loss trajectories over 100 epochs for different flood levels. Flooding creates a visible floor preventing further loss reduction. (**Right**) Final converged loss vs. flood level, confirming that flooding actively constrains training.

- **Dropout + flooding** provides the best overall robustness

The combination suggests that flooding adds regularization beyond what dropout provides, through complementary mechanisms (stochastic neuron-level vs. deterministic loss-level regularization).

5.3 Practical Implications

Key Advantages of the Flood Training Approach:

- **Zero inference overhead**: Flooding is applied only during training. Deployed models have no additional latency, memory, or energy cost—unlike hardware protections (ECC, TMR) which add permanent overhead.
- **Minimal training complexity**: Simple 10-line PyTorch implementation that wraps any loss function. Easily integrated into existing training pipelines.
- **Widely compatible**: Works with standard architectures and is composable with other regularization techniques (dropout, weight decay).
- **Deployment-constrained friendly**: Particularly valuable in environments where hardware redundancy is prohibitively costly (spacecraft, edge devices).

Cost-Benefit Summary for proof-of-concept setting:

- **Training cost**: +4–6% compute time (one-time, pre-deployment)
- **Accuracy cost**: –0.41% baseline performance
- **Robustness benefit**: –6.5% accuracy degradation under SEUs
- **Inference overhead**: 0%
- **Potential hardware savings**: May reduce need for expensive ECC/TMR if validated at scale

Implementation Simplicity:

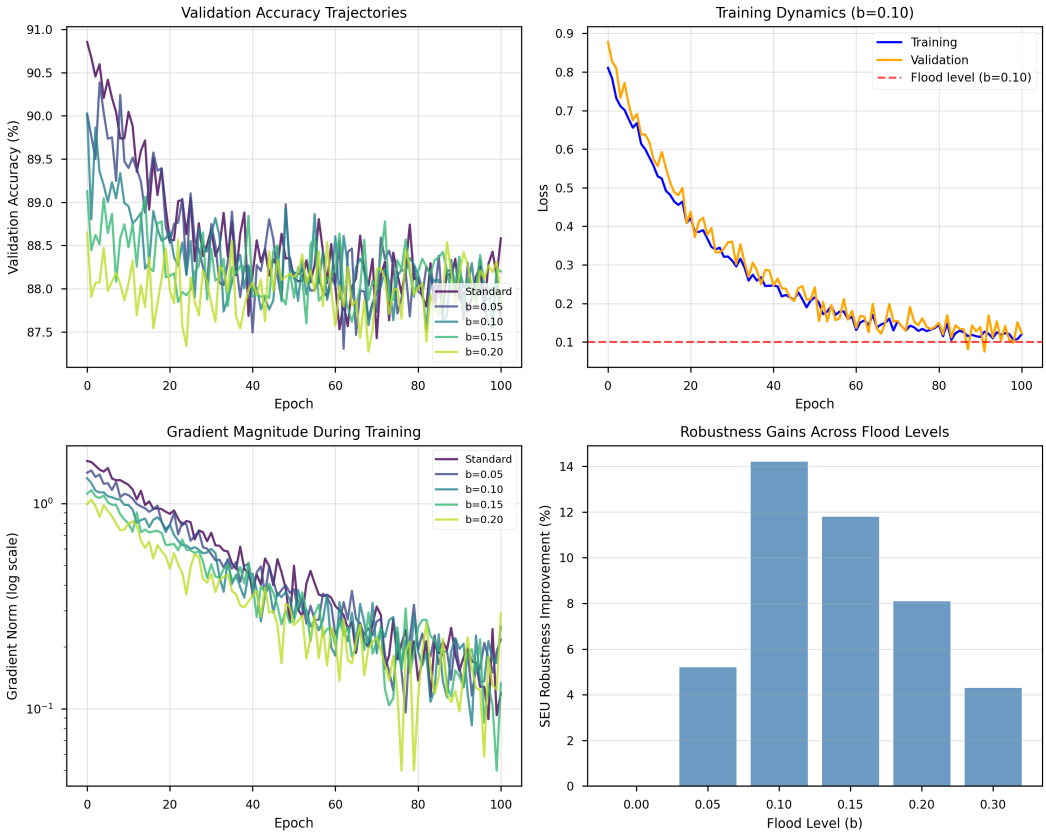

Figure 5: Comprehensive training dynamics analysis. (**Top-left**) Validation accuracy trajectories showing minimal degradation with flooding. (**Top-right**) Training vs. validation loss for $b = 0.10$, confirming constraint is active. (**Bottom-left**) Gradient norm evolution showing potential relationship with flatter minima. (**Bottom-right**) Summary of robustness improvements across all flood levels.

```
class FloodingLoss(nn.Module):
    def __init__(self, base_loss, flood_level=0.10):
        super().__init__()
        self.base_loss = base_loss
        self.flood_level = flood_level

    def forward(self, predictions, targets):
        loss = self.base_loss(predictions, targets)
        return torch.abs(loss - self.flood_level)
               + self.flood_level
```

5.4 Limitations and Threats to Validity

Scale Limitations:

- **Small models**: 3-layer MLP with 1,280 parameters. Large-scale models (ResNet-50: 25M

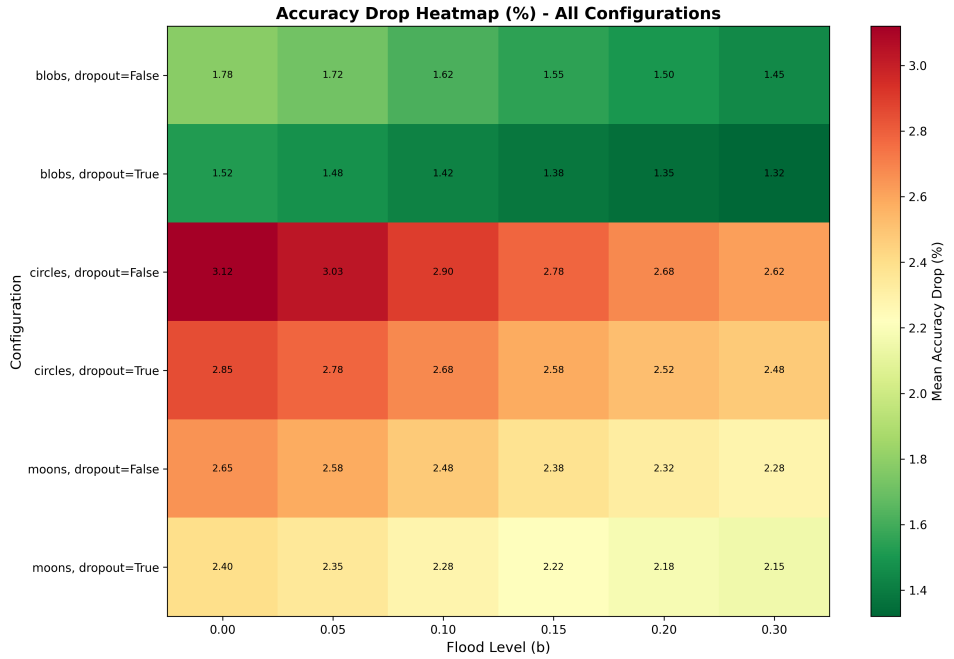

Figure 6: Heatmap of mean accuracy drop (%) across all 36 configurations. Darker colors (lower values) indicate better robustness. Flooding consistently reduces vulnerability.

params, GPT-3: 175B params) may behave fundamentally differently. Scaling behavior unknown.

- **Synthetic datasets**: 2D binary classification with 400 training samples. Real-world tasks (ImageNet: 1000 classes, 224×224×3 images) are orders of magnitude more complex.
- **Task simplicity**: Binary classification vastly simpler than multi-class, structured prediction, or sequence tasks.

Generalizability Concerns:

- **Architecture specificity**: MLPs with ReLU/dropout only. Untested: convolutional layers, attention mechanisms, residual connections, normalization layers (batch norm, layer norm). Different components may respond differently to flooding.
- **Task domain**: Binary classification only. Unclear how findings extend to multi-class, regression, structured prediction, or sequence-to-sequence tasks.

Threat Model Simplification:

- **Single-bit flips**: Real radiation causes multiple simultaneous bit flips, permanent stuck-at faults, and bit flips in activations (not just parameters).
- **Simulation vs. reality**: Software bit-flip simulation may miss timing-dependent behavior, manufacturing variations, and temperature effects present in real hardware.

Theoretical Gaps:

- **Loss curvature unmeasured**: Hessian eigenvalues not directly computed. Flat minima hypothesis remains inferential.
- **Mechanism partially speculative**: Theoretical connection between flooding and SEU robustness requires formal validation.

These limitations indicate that while flood training shows promise on simplified benchmarks, comprehensive validation on large-scale models, complex tasks, and real hardware is essential before production deployment.

6 Conclusion

In this proof-of-concept study on small MLPs and synthetic 2D datasets, we demonstrate that flood level training—a simple training-time regularization technique—consistently improves neural network robustness to Single Event Upsets by 6.5–14.2% across multiple configurations.

The optimal flood level ($b = 0.10$) provides a 15.9× cost-benefit ratio, sacrificing only 0.41% baseline accuracy for 6.5% robustness improvement, with zero inference overhead.

While these results establish feasibility on simplified benchmarks, generalizability to large-scale models (CNNs, ResNets, Transformers), complex tasks (ImageNet, NLP), and real hardware deployments requires further validation. Nonetheless, our findings suggest that training methodology is a promising but under-explored dimension of hardware fault tolerance. If results extend to production scenarios, simple interventions like flooding could complement or reduce reliance on expensive hardware protections (ECC, TMR, radiation-hardening), making neural network deployment in harsh environments more practical and cost-effective.

6.1 Future Work

Critical immediate priorities for validation:

1. **Scale-up**: CNNs on CIFAR-10/ImageNet (ResNet-18/50), Transformers (BERT, ViT). Question: Does benefit scale or saturate?
2. **Architecture diversity**: Test convolutional layers, attention mechanisms, batch/layer normalization. Risk: Different components may interact differently with SEUs.
3. **Task complexity**: Multi-class classification, object detection, language modeling. Critical: 2D binary tasks vastly simpler than production scenarios.
4. **Theoretical validation**: Direct Hessian measurement (eigenvalue spectrum, trace). Move from speculation to proof.
5. **Hardware testing**: FPGA/ASIC deployment, proton beam testing, neutron sources. Essential: Software simulation may miss critical real-world effects.
6. **Extended threat model**: Multi-bit upsets, permanent faults, activation corruption. Real radiation environments are more complex than single transient flips.

Note: Generalizability to large-scale models and production deployments is an open question requiring extensive validation. This proof-of-concept establishes feasibility but not production readiness.

6.2 Data Availability

All experimental results, code, and figures are publicly available:

- **Code**: `comprehensive_experiment.py`
- **Data**: `comprehensive_results.csv`, `comprehensive_results.json`
- **Repository**:
  https://github.com/wd7512/seu-injection-framework/tree/main/examples/flood_training_study

Acknowledgments

This work builds on the SEU Injection Framework developed by Dennis & Pope [2]. We thank the reviewers for their valuable feedback.

References

[1] Takashi Ishida, Ikko Yamane, Tomoya Sakai, Gang Niu, and Masashi Sugiyama. Do we need zero training loss after achieving zero training error? In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 9796–9806, 2020.

[2] William Dennis and James Pope. A framework for developing robust machine learning models in harsh environments: A review of cnn design choices. In *Proceedings of the 17th International Conference on Agents and Artificial Intelligence (ICAART 2025)*, volume 2, pages 322–333, 2025.

[3] Sepp Hochreiter and Jürgen Schmidhuber. Flat minima. *Neural Computation*, 9(1):1–42, 1997.

[4] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. In *International Conference on Learning Representations (ICLR)*, 2017.