Reproducibility Track Proposal: Neural Tangent Kernel in Classification

Contact Information

Name: Subhra Jyoti Das
Contact Number: 6901482632
Branch: Production and Industrial Engineering
Email ID: subhra_jd@me.iitr.ac.in
Enrollment Number: 24119053

Title: Reproducing and Extending NTK Divergence in Classification Problems

Selected Research Paper:

Title: Divergence of Empirical Neural Tangent Kernel in Classification Problems

Conference: ICLR 2025 (International Conference on Learning Representations)

Authors: Zixiong Yu (Huawei), Songtao Tian (Tsinghua), Guhan Chen (Tsinghua)

Link to Paper:
https://proceedings.iclr.cc/paper_files/paper/2025/hash/731b952bdc833485eb72f458cdd5c489-Abstract-Conference.html

arXiv Link: https://arxiv.org/abs/2504.11130

Objective

This reproducibility project aims to rigorously reproduce the groundbreaking findings of the ICLR 2025 paper that proves empirical Neural Tangent Kernels (NTK) **diverge** in classification problems with cross-entropy loss, fundamentally challenging the theoretical foundations of NTK theory.

Why This Paper?

1. **Significant Theoretical Gap:** The paper is the first to rigorously prove that NTK theory—widely accepted as a foundation for understanding infinite-width neural networks—**fails fundamentally in classification tasks**, despite working well in regression.

2. **Practical Relevance:** Classification is far more common in real applications than regression, yet NTK theory was developed primarily for regression contexts. Understanding this divergence has immediate implications for deep learning research.

3. **Clear Experimental Design:** The paper provides concrete experiments on synthetic data (circle classification) and real data (MNIST), making reproducibility feasible with standard computational resources.

4. **Opportunity for Novel Insights:** By reproducing the experiments and conducting targeted ablations, we can:

   - Verify the claims across different architectures and datasets
   - Investigate the precise mechanisms of NTK divergence
   - Study how width affects the divergence phenomenon
   - Explore boundary conditions where NTK remains valid

5. **Alignment with Current Research:** Understanding NTK limitations directly connects to recent advances in feature learning theory and informs the design of better theoretical frameworks for deep learning.

Understanding of the Paper

Core Concept: Neural Tangent Kernel (NTK)

The **Neural Tangent Kernel** describes how neural network outputs evolve during gradient descent training. At network parameter θ, the empirical NTK at time (t) is defined as:

$Kt(x,x')=\langle \nabla\theta f(x;\theta t),\nabla\theta f(x';\theta t)\rangle$

Key Insight from Prior Work: In regression with MSE loss, as network width $m\to\infty$, the empirical NTK converges to a deterministic **Neural Tangent Kernel** K_{NT} that remains approximately constant during training. This enables treating neural network training as kernel regression.

The Divergence Phenomenon

Main Claim of the ICLR 2025 Paper: In classification with cross-entropy loss, the empirical NTK **does not converge uniformly** to the theoretical NTK. Instead:

1. **Strictly Positive Definiteness:** The authors first prove that NTKs of fully-connected networks (FCN) and residual networks (ResNet) are strictly positive definite on compact domains (Proposition 1).

2. **Parameter Divergence (Theorem 1):** If the smallest eigenvalue of the empirical NTK matrix remains bounded below by a positive constant during training, network parameters diverge to infinity: $\lim t\to\infty|ft(xi)|=+\infty$

3. **NTK Divergence (Theorem 2):** Through proof by contradiction, the authors show that uniform convergence of empirical NTK to theoretical NTK cannot hold. There exists a positive lower bound:

$$\sup_{t\geq 0} |K_t(x,x') - K_{NT}(x,x')| \geq \frac{\lambda_0}{2n^2}$$

that is **independent of network width**, proving fundamental divergence.

Why This Happens: The Root Cause

Cross-Entropy Loss Behavior:

- Cross-entropy loss pushes network outputs to (+infinity) or (-infinity) to achieve zero loss
- This forces parameters to move far from initialization
- The "lazy training regime" (where NTK stays constant) **breaks down**
- As parameters diverge, the empirical NTK also diverges

Contrast with MSE Regression:

- MSE loss can be minimized with small parameter changes
- Lazy training regime holds
- NTK provides accurate approximation throughout training

Mathematical Framework

Fully Connected Network (FCN):

$$f(x;\theta) = W^{(L+1)}\alpha^{(L)}(x), \quad \alpha^{(l)}(x) = \sqrt{\frac{2}{m}}\sigma(W^{(l)}\alpha^{(l-1)}(x) + b^{(l)})$$

Cross-Entropy Loss:

$$L(\theta) = \sum_{i=1}^{n} \ell((2y_i - 1)f(x_i;\theta)), \quad \ell(x) = \ln(1 + e^{-x})$$

Gradient Flow Dynamics:

$$\frac{d\theta_t}{dt} = -\nabla_\theta L(\theta_t) = \sum_{i=1}^{n} \nabla_\theta f(x_i;\theta_t)(2y_i - 1)u_i$$

where u_i represents output residuals.

Key Experimental Evidence

Synthetic Circle Experiment:

- Dataset: 6 points on unit circle with alternating labels
- Network: 3-layer FCN, width 2000
- Result: Network outputs diverge to infinity despite balanced labels

MNIST Experiment:

- Task: Binary classification (odd vs. even)
- Network: 4-layer FCN, width 500
- Result: Empirical NTK values diverge during training

Data and Code Availability

Code Availability Status

Current Status: The authors **have not released official code** at the time of paper publication, which is common for theoretical papers. However, this presents an opportunity:

What We Need to Implement:

1. Network architectures: 3-layer FCN, 4-layer FCN, ResNet
2. NTK computation using gradient formulas
3. Gradient flow training loop with cross-entropy loss
4. Synthetic circle dataset generation
5. MNIST data loading and preprocessing

Challenges and Solutions:

Challenge	Solution
No official code	Implement from scratch following paper specifications; detailed mathematical formulas enable exact replication
Computational cost (RTX 3050 GPU)	Use mixed precision (float32 for forward, float64 for critical computations); reduced batch sizes; CPU fallback for eigenvalue analysis
MNIST availability	PyTorch/TensorFlow provide built-in MNIST loaders
Numerical precision	Use PyTorch double precision (float64) for eigenvalue computations where needed
Long training times	Implement efficient NTK computation using Jacobian-vector products; use vectorized operations

Data Availability

- **Synthetic Data:** Circle dataset is trivial to generate (6 points on unit circle)
- **MNIST:** Publicly available via PyTorch/TensorFlow datasets
- **No licensing issues:** Both are open for academic use

Estimated Computational Requirements with NVIDIA RTX 3050:

Experiment	GPU Memory	Time on RTX 3050	Solution
Synthetic Circle (width 2000)	2-3 GB	5-15 min	Fully feasible

Experiment	GPU Memory	Time on RTX 3050	Solution
MNIST Binary (width 500, 10k steps)	2-3 GB	30-60 min	Fully feasible
Width Scaling (widths 64-1024)	1-2 GB each	10-20 min each	Feasible (8-10 GB total time)
Width Scaling (widths 2048-4096)	4-6 GB each	30-60 min each	Reduced widths: use max 2048
MNIST with full width 500	2-3 GB	45 min per run	Feasible
Full NTK computation (n=1000 samples)	3-4 GB	10-15 min	Feasible

Mitigation Strategies for RTX 3050 (6GB VRAM):

1. **Batch Processing:** Process NTK computation in smaller batches instead of full matrix at once
2. **Reduced Network Sizes:** For ablations, use slightly smaller widths if needed
3. **Sequential Ablations:** Run ablations one at a time instead of parallel
4. **Mixed Precision:** Use float32 for forward/backward passes, float64 only for critical eigenvalue computations
5. **CPU Operations:** Move eigenvalue computation and other post-processing to CPU
6. **Gradient Checkpointing:** Trade compute for memory in gradient computation

The main experiments from the paper (circle + MNIST) require 2-3 GB and will run smoothly. Most ablations are also feasible. Only the largest width ablations (4096) may require optimization, but can be reduced to 2048 without major loss.

Ablation Study Plan

The paper's core results open up significant research directions for extended experiments:

Ablation Study 1: Network Width Scaling

Hypothesis: Divergence occurs at all widths; increasing width might slow but not prevent divergence.

Experimental Design (RTX 3050 Optimized):

- Train networks of widths: **64, 128, 256, 512, 1024, 2048** (reduced from 4096)
- Track empirical NTK deviation from theoretical NTK across epochs
- Measure: $\sup_t |K_t(x, x') - K_{NT}(x, x')|$ for each width
- Expected Outcome: Divergence lower bound is width-independent
- **GPU Optimization:** Use batch NTK computation; max 2-3 GB per run

Significance: Confirms or refutes the paper's claim that NTK failure is fundamental, not due to finite-width effects.

Ablation Study 2: Loss Function Comparison

Hypothesis: Different loss functions lead to different NTK behaviors.

Experimental Design:

- Cross-Entropy (paper's focus): $L = -\sum y_i \ln(o_i) + (1 - y_i) \ln(1 - o_i)$
- MSE Loss: $L = \sum (y_i - f(x_i))^2$
- Hinge Loss: $L = \sum \max(0, 1 - y_i f(x_i))$
- Focal Loss: $L = -\sum \alpha_t (1 - p_t)^\gamma \log(p_t)$
- Metric: Track NTK divergence for each loss function
- **GPU Optimization**: All loss functions have similar memory requirements

Significance: Identifies whether divergence is specific to cross-entropy or a general phenomenon for non-convex losses in classification.

Ablation Study 3: Network Depth Effects

Hypothesis: Deeper networks might exhibit different divergence rates.

Experimental Design:

- Train networks with depths: **2, 3, 4, 5** hidden layers (use consistent width 256-512)
- Monitor: Eigenvalue spectrum evolution, divergence rate
- Datasets: Circle and MNIST binary classification
- **GPU Optimization:** Deeper networks need similar GPU memory if width is controlled

Significance: Connects to recent work on depth's role in NTK (2025 papers). Shows whether depth or width dominates divergence behavior.

Ablation Study 4: Initialization Scale Sensitivity

Hypothesis: NTK is defined at initialization; different initialization might affect divergence onset.

Experimental Design:

- Vary initialization scale: $\sigma = 0.1, 0.5, 1.0, 2.0, 5.0$ (Gaussian scale)
- Track: Time to achieve perfect training accuracy, NTK eigenvalue evolution
- Measure: Does larger initialization scale lead to faster divergence?

- **GPU Optimization:** Minimal memory impact; just rescale weight initialization

Significance: Tests robustness of theoretical results to initialization choices; identifies practical knobs for controlling divergence.

Ablation Study 5: Dataset Separability

Hypothesis: Easily separable data might suppress divergence; complex data might accelerate it.

Experimental Design:

- Generate synthetic datasets with varying separability:
  - Perfectly separable linearly (circle with margin)
  - Moderately separable
  - Highly overlapping labels (noisy circle)
- Metric: NTK divergence rate vs. dataset difficulty
- **GPU Optimization:** Synthetic data generation on CPU; only forward passes on GPU

Significance: Reveals whether divergence is inevitable (data-independent) or influenced by task complexity.

Ablation Study 6: Training Horizon

Hypothesis: Divergence accelerates with extended training; identify divergence onset timing.

Experimental Design:

- Extended training: **20k, 50k** epochs (instead of paper's 10k, keep 200k as optional)
- Measure: Exact epoch where divergence becomes statistically significant
- Track: Generalization gap between network and kernel predictor
- Expected Pattern: Initial NTK validity → gradual divergence → complete divergence
- **GPU Optimization:** Distributed across multiple days; checkpoint and resume

Significance: Characterizes the timeline of NTK failure; informs practical bounds on NTK-based analysis.

Ablation Study 7: Network Architecture Variations

Hypothesis: Architecture differences (ReLU vs. other activations) affect divergence.

Experimental Design (RTX 3050 Optimized):

- Activation functions: **ReLU (paper), GELU, Tanh** (skip Sigmoid for time efficiency)
- Network types: **FCN (paper), ResNet** (skip CNN, ViT for computational feasibility)
- Metric: Does activation function affect NTK convergence?
- **GPU Optimization:** Similar memory to baseline experiments

Significance: Tests generalization of results beyond ReLU networks; explores which architectures might maintain NTK validity.

Methodology

Phase 1: Environment Setup & Baseline Implementation (Week 1 - Dec 6-12)

Step 1.1: Development Environment Setup for RTX 3050

```
- Language: Python 3.10+
- Framework: PyTorch with CUDA support for RTX 3050
- Key Libraries:
  * numpy: Numerical computations
  * scipy: Eigenvalue computations (on CPU to save GPU memory)
  * matplotlib/seaborn: Visualization
  * jupyter: Interactive development
  * torch: GPU acceleration with RTX 3050
- Platform: Local machine with RTX 3050 OR Google Colab (as backup)
```

Step 1.2: Code Organization

```
project/
├── data/
│   ├── generate_circle.py      # Synthetic circle dataset
│   └── mnist_loader.py         # MNIST preprocessing
├── models/
│   ├── fcn.py                  # Fully connected network
│   ├── resnet.py               # Residual network
│   └── utils.py                # Weight initialization
├── ntk/
│   ├── ntk_computation.py      # Empirical NTK computation (GPU-optimized)
│   ├── theoretical_ntk.py      # Theoretical NTK formulas
│   └── eigenvalue_analysis.py  # Spectral analysis (CPU-based)
├── training/
│   ├── gradient_flow.py        # Training loop with cross-entropy
│   ├── kernel_regression.py    # NTK-based predictions
│   └── metrics.py              # Divergence metrics
├── experiments/
│   ├── reproduce_circle.py     # Reproduce synthetic experiment
│   ├── reproduce_mnist.py      # Reproduce MNIST experiment
│   └── ablations/              # Ablation studies
├── utils/
│   ├── memory_utils.py         # GPU memory management
│   ├── checkpoint_utils.py     # Save/resume capabilities
│   └── device_utils.py         # CPU/GPU device handling
├── results/
│   ├── plots/                  # Generated figures
│   └── logs/                   # Training logs
├── config/
```

```
│   └── rtx3050_config.yaml     # RTX 3050 optimized hyperparameters
├── requirements.txt
├── README.md
└── SETUP_RTX3050.md            # RTX 3050 specific setup guide
```

Deliverable by Dec 12: Complete code framework

Phase 2: Reproduce Baseline Results (Week 2 - Dec 13-19)

Step 2.1: Synthetic Circle Experiment

- Generate 6-point circle dataset
- Train 3-layer FCN (width 2000 - fits in RTX 3050)
- Compute NTK at initialization and during training
- Track network output divergence
- Generate comparison plots (matching paper's figures)
- **Time estimate:** 15-20 minutes total

Step 2.2: MNIST Binary Classification

- Load and preprocess MNIST (binary: odd vs. even)
- Train 4-layer FCN (width 500 - efficient for RTX 3050)
- Track empirical NTK evolution during training
- Record training loss, generalization metrics
- Generate NTK divergence plots
- **Time estimate:** 45-60 minutes total

Step 2.3: Verification & Comparison

- Compare empirical results with paper's reported values
- Document any deviations and investigate causes
- Validate NTK positivity and eigenvalue bounds
- Generate comprehensive comparison report
- **Time estimate:** 30 minutes analysis

Deliverable by Dec 19: Complete reproduction of paper's two main experiments

Phase 3: Conduct Core Ablation Studies (Week 3 - Dec 20-26)

Step 3.1: Width Scaling Study (Dec 20-22) - ~2-3 hours

- RTX 3050 can handle widths: 64, 128, 256, 512, 1024, 2048
- Skip width 4096 (would need 6+ GB)
- Compute efficiently using batch NTK computation
- **Memory per run:** ~2-3 GB
- **Total GPU time:** 8-10 hours spread across 3 days

Step 3.2: Loss Function Comparison (Dec 22-24) - ~2 hours

- Implement 4 loss functions (CE, MSE, Hinge, Focal)
- Similar memory footprint as baseline
- **Total GPU time:** 3-4 hours

Step 3.3: Network Depth Effects (Dec 24-26) - ~1.5 hours

- Test depths 2-5 (keep width moderate: 256-512)
- Depths don't increase memory much with controlled width
- **Total GPU time:** 2-3 hours

MID-EVALUATION DELIVERABLE (Dec 19):

- Successful reproduction of paper results
- Code repository with RTX 3050 optimizations
- 3 core ablation studies completed
- Mid-report documenting progress

Phase 4: Extended Ablations & Final Analysis (Week 4 - Dec 27-Jan 2)

Step 4.1: Remaining Ablations (Dec 27-30)

- **Initialization Sensitivity** (Dec 27) - ~30 min, minimal GPU usage
- **Dataset Separability** (Dec 27-28) - ~30 min
- **Training Duration** (Dec 28-29) - ~1-2 hours
- **Architecture Variations** (Dec 29-30) - ~1-2 hours
- **Total GPU time:** 4-6 hours

Step 4.2: Comprehensive Analysis (Dec 31-Jan 1)

- Compile all results
- Statistical significance testing (CPU-based)
- Identify novel insights
- Create visualizations (CPU-based)

Step 4.3: Final Submission (Jan 1-2)

- Clean and document code
- Write comprehensive README
- Final testing

END-EVALUATION DELIVERABLE (Jan 2):

- Complete GitHub repository
- All 7 ablation studies completed
- Technical report (10-15 pages)
- Visualizations and plots

Potential Issues & Solutions

Issue	Symptom	Solution
Out of Memory	CUDA OOM error	Reduce batch size, use gradient checkpointing, or move to CPU
Slow Eigenvalues	Eigenvalue computation takes 5+ min	Always compute on CPU; RTX 3050 is slow for eigendecomposition
Notebook Crashes	Runtime crashes during long runs	Use checkpointing; save intermediate results; split into shorter jobs
Temperature Throttling	GPU speed drops mid-experiment	Take breaks between experiments; ensure good ventilation

Timeline (Start Date: December 6, 2025)

Critical Dates

- **December 19, 2025: Mid-Evaluation Checkpoint**
- **January 2, 2026: End-Evaluation / Final Submission**

Week 1: Dec 6-12 (Environment & Baseline Code)

Tasks:

- Set up RTX 3050 environment with CUDA, cuDNN
- Install PyTorch with GPU support
- Create memory optimization utilities
- Implement FCN and ResNet architectures
- Implement empirical NTK computation (GPU-optimized)
- Implement theoretical NTK formulas
- Create synthetic circle dataset
- Test implementations on toy problems
- Verify GPU memory usage (<6GB)

Deliverable: Code framework

Week 2: Dec 13-19 (Reproduce Core Results)

Tasks:

- Run synthetic circle experiment (3-layer FCN, width 2000)
  - Verify network output divergence
  - Check NTK at initialization
  - Track divergence during training

- Run MNIST binary classification (4-layer FCN, width 500)
    - Load and preprocess MNIST
    - Track NTK evolution
    - Generate comparison plots
- Compare results with paper
- Document any deviations
- Generate figures matching paper
- Create comprehensive comparison report

Deliverable: Complete reproduction of paper's two experiments

Week 3: Dec 20-26 (Core Ablations → Mid-Evaluation on Dec 19)

Tasks (First 6 days):

- **Ablation 1: Width Scaling (Dec 20-22)**
    - Widths: 64, 128, 256, 512, 1024, 2048 (skip 4096)
    - Measure divergence for each width
    - Plot divergence vs. width curve
    - **GPU Time:** 8-10 hours total
- **Ablation 2: Loss Functions (Dec 22-24)**
    - Compare: Cross-entropy, MSE, Hinge, Focal
    - Generate loss comparison plots
    - **GPU Time:** 3-4 hours
- **Ablation 3: Network Depth (Dec 24-26)**
    - Depths: 2, 3, 4, 5 layers (width 256-512)
    - Monitor eigenvalue evolution
    - **GPU Time:** 2-3 hours

MID-EVALUATION DELIVERABLE (Dec 19):

- Successful reproduction of paper results
- Code repository
- 3 core ablation studies completed
- Preliminary analysis and visualizations
- Mid-report documenting progress

Week 4: Dec 27-Jan 2 (Extended Ablations & Final → End-Evaluation on Jan 2)

Tasks (Dec 27-30):

- **Ablation 4: Initialization Sensitivity (Dec 27)**
  - Scales: 0.1, 0.5, 1.0, 2.0, 5.0
  - Track time to divergence
  - **GPU Time:** ~1 hour
- **Ablation 5: Dataset Separability (Dec 27-28)**
  - Easy → Medium → Hard separability
  - **GPU Time:** ~1 hour
- **Ablation 6: Training Duration (Dec 28-29)**
  - Extended training: 20k, 50k epochs
  - Identify divergence onset
  - **GPU Time:** 3-4 hours
- **Ablation 7: Architecture Variations (Dec 29-30)**
  - ReLU, GELU, Tanh activations
  - FCN, ResNet architectures
  - **GPU Time:** 2-3 hours

Tasks (Dec 31-Jan 1):

- Comprehensive statistical analysis
- Compile all results and figures
- Write technical report
- Identify novel insights
- Create publication-quality visualizations

Final Submission Tasks (Jan 1-2):

- Clean and document all code
- Create GitHub repository with README
- Final code review and testing
- Prepare final report

END-EVALUATION DELIVERABLE (Jan 2):

- Complete GitHub repository
- All 7 ablation studies completed
- Technical report with analysis
- Visualizations and plots

Expected Outcomes

Primary Outcomes (Reproduction)

1. **Successful Reproduction of Core Results:**

- Circle experiment: Reproduce network output divergence to ±∞
- NTK divergence quantified: Measure $\| K_t - K_{NT} \|_F$increasing over time
- Eigenvalue bounds verified: Confirm λmin(Kt)≥C>0
- MNIST results: Track NTK values diverging (not converging to fixed values)

2. **Validation of Theoretical Predictions:**
   - Verify Theorem 1: Network outputs diverge when lambda min bounded below
   - Verify Theorem 2: Divergence lower bound independent of width
   - Confirm Proposition 1: Strictly positive definiteness of NTK
3. **Quantitative Metrics:**
   - Initial error: $\| K_{init}^{emp} - K_{init}^{theory} \| < 0.1$ (Law of Large Numbers)
   - Divergence rate: Characterize $\| K_t - K_{NT} \|$growth rate vs. time
   - Eigenvalue statistics: Track minimum eigenvalue evolution

Secondary Outcomes (Ablations)

4. **Width Scaling Analysis (tested up to width 2048):**
   - **Predicted Result:** NTK divergence persists regardless of width; divergence lower bound remains $\geq \lambda_0/(2n^2)$
   - **Significance:** Proves divergence is fundamental, not finite-width artifact
5. **Loss Function Dependence:**
   - **Predicted Result:** MSE loss → NTK convergence; Cross-entropy → NTK divergence
   - **Predicted Result:** Hinge and Focal losses show intermediate behavior
   - **Significance:** Identifies loss function as critical factor determining NTK validity
6. **Depth Effects:**
   - **Predicted Result:** Moderate depths (2-5) show consistent divergence pattern
   - **Predicted Result:** Divergence relatively independent of depth given width
   - **Significance:** Reveals depth doesn't dramatically change NTK dynamics
7. **Initialization Sensitivity:**
   - **Predicted Result:** Larger initialization scales → faster divergence onset
   - **Predicted Result:** Quantify effect of initialization on NTK stability window
   - **Significance:** Provides practical guidance on initialization
8. **Dataset Complexity:**
   - **Predicted Result:** Highly separable data → slower divergence
   - **Predicted Result:** Overlapping data → faster divergence
   - **Significance:** Shows divergence rate depends on problem difficulty

9. **Training Duration Effects:**
   - **Predicted Result:** Identify precise epoch where divergence becomes significant
   - **Predicted Result:** Characterize divergence timeline
   - **Significance:** Establishes practical bounds on NTK analysis horizon
10. **Architecture Generalization:**
   - **Predicted Result:** Results hold across ReLU/GELU/Tanh
   - **Predicted Result:** ResNets show similar divergence as FCNs
   - **Significance:** Confirms generality of findings

How This Project Advances Understanding

1. **Validates a Critical Theoretical Result:** First independent verification of NTK divergence in classification
2. **Extends Beyond the Paper:** Ablations provide new insights on width, depth, and loss function roles
3. **Bridges Theory and Practice:** Clarifies when NTK theory applies and when it breaks down
4. **Informs Future Research:** Results suggest new directions for developing NTK-valid loss functions
5. **Contributes to Reproducibility Culture:** Demonstrates best practices for reproducing ML research on consumer GPUs like RTX 3050

References

Primary Paper

- Yu, Z., Tian, S., & Chen, G. (2025). Divergence of Empirical Neural Tangent Kernel in Classification Problems. *ICLR 2025*.

Foundational NTK Theory

- Jacot, A., Gabriel, F., & Hongler, C. (2018). Neural tangent kernel: Convergence and generalization in neural networks. *NeurIPS 2018*.
- Arora, S., et al. (2019). On exact computation with an infinitely wide neural net. *NeurIPS 2019*.

Related Work

- Lai, J., Xu, M., Chen, R., & Lin, Q. (2023). Generalization ability of wide neural networks on (\mathbb{R}). *arXiv:2302.05933*.
- Li, Y., et al. (2023). Statistical optimality of deep wide neural networks. *arXiv:2305.02657*.

Ablation Motivation

- Chizat, L., Oyallon, E., & Bach, F. (2019). On lazy training in differentiable programming. *NeurIPS 2019*.
- Neyshabur, B., et al. (2020). Exploring Generalization in Deep Learning. *NeurIPS 2020*.

Evaluation Schedule:

- **Start Date:** December 6, 2025
- **Mid-Evaluation Deadline:** December 19, 2025
- **Final Submission Deadline:** January 2, 2026

Hardware: NVIDIA RTX 3050 (6GB VRAM) - Fully Supported