

# Activation Functions and Reproducibility of Language Models

**Date:** December 2025

**Project:** LLM Reproducibility with Activation Functions

**Repository:** <https://github.com/pankajtakawale/llm-reproducibility-activations>

**Authors:** Pankaj Takawale (pvt2106) and Vinita Takawale (vt2365)

---

## Abstract

This study investigates the impact of activation function choice on reproducibility in Language Models. We trained five different model architectures (CharLM, MiniGPT, ConvLM, HybridLM, NanoTransformer) with six activation functions (SmeLU  $\beta=1.0$ , ReLU, GELU, Swish, SwiGLU, FullModel) and measured prediction consistency across multiple independent training runs. Our findings demonstrate that **all architectures show activation sensitivity**, with Coefficient of Variation ranging from 9.87% (HybridLM) to 23.03% (MiniGPT). CharLM shows 20.26% variation with SwiGLU achieving best reproducibility (PD=0.593), while ReLU shows worst (PD=1.074). This work provides empirical evidence that **activation function choice significantly impacts reproducibility** across all tested architectures.

## Table of Contents

- [1. Introduction](#)
- [2. Background: The Reproducibility Problem](#)
- [3. Measures of Reproducibility](#)
- [4. Research Approach](#)
- [5. Experimental Setup](#)
- [6. Framework and Implementation](#)
- [7. Results and Findings](#)
- [8. Challenges](#)
- [9. Discussion](#)
- [10. Conclusion](#)
- [11. Future Work](#)
- [12. References](#)

# 1. Introduction

## 1.1 Motivation

Language models have achieved remarkable success across various domains, yet their behavior remains difficult to predict and reproduce. The same model architecture, trained on the same data with ostensibly identical settings, can produce different results across training runs. This phenomenon, known as **irreproducibility**, poses significant challenges for scientific validation, model debugging, production deployment, and peer review.

## 1.2 The Reproducibility Challenge

Lack of replicability, where researchers are unable to reproduce published results with a given model, has been identified as a major challenge in machine learning. **Irreproducibility** is a related but more elusive problem: multiple instances of a given model trained on the same data under identical training conditions yield different results.

In practice, deep network language models are trained in highly parallelized and distributed environments. Multiple factors contribute to irreproducibility:

- **Random initialization** and stochastic gradient descent
- **Parallelism** and non-deterministic operations (GPU atomics, thread scheduling)
- **Distributed training** with asynchronous updates
- **Data shuffling** and mini-batch sampling order
- **Quantization errors** in floating-point arithmetic
- **Hardware differences** (CPU vs GPU, different GPU architectures)
- **Optimization landscapes** with multiple local optima

Some factors, such as initialization, can be controlled through careful seed management. However, it is impractical to control others, particularly in production environments. Optimization trajectories can diverge early in training by following examples in the order seen, leading to very different final models even with identical hyperparameters.

## 1.3 Existing Solutions and Their Limitations

Several recently published solutions based on advanced combinations of ensembling, self-ensembling, and distillation can mitigate irreproducibility, but typically at significant costs:

- **Increased computational overhead** (training multiple models)
- **Reduced accuracy** (ensemble calibration issues)
- **Higher complexity** (maintenance and debugging burden)

## 1.4 Research Question

This study explores a fundamental architectural choice that may influence reproducibility:

**Does the choice of activation function affect the reproducibility of language model predictions?**

Specifically, we hypothesize that **smooth, continuously differentiable activation functions** (SmeLU, GELU, Swish) lead to more stable training dynamics and thus more reproducible outcomes compared to **non-smooth functions** (ReLU) with discontinuous gradients.

## 1.5 Contributions

This work makes the following contributions:

1. **Empirical evidence** linking activation function choice to model reproducibility across 6 architectures
2. **Quantitative metrics** for measuring reproducibility in language models (Relative Prediction Disagreement)
3. **Comparative analysis** of 5 activation functions across 75 independent training runs (5 models  $\times$  5 activations  $\times$  3 trials)
4. **Architectural insights** showing transformers and LSTMs respond differently to activation functions
5. **Practical recommendations** for activation function selection based on reproducibility requirements
6. **Open-source framework** for reproducibility experiments in language models

## 2. Background: The Reproducibility Problem

### 2.1 Defining Reproducibility in Language Models

We distinguish between three related concepts:

1. **Replicability**: Can independent researchers obtain the same results following published methods?
2. **Reproducibility**: Do repeated training runs with the same code and data yield consistent results?
3. **Stability**: How sensitive are model predictions to training stochasticity?

This study focuses on **reproducibility** and **stability** at the model prediction level.

### 2.2 Sources of Non-determinism

#### Controllable Sources

- Random seed initialization (weights, dropout masks)
- Data shuffling and mini-batch ordering
- Optimizer state initialization

### Difficult-to-Control Sources

- GPU thread scheduling and atomic operations
- Floating-point operation ordering (affects rounding)
- Parallel reduction order in distributed training
- Hardware-specific optimizations (cuDNN algorithms)
- Memory allocation patterns

### Optimization Landscape Factors

- Multiple local optima with similar loss values
- Sensitivity to early training dynamics
- Interaction between batch statistics and gradient descent
- Accumulation of small numerical differences over training

## 2.3 Why Activation Functions Matter

Activation functions influence training dynamics through several mechanisms:

1. **Gradient Flow:** Smooth functions provide continuous gradients, reducing sensitivity to initialization
2. **Loss Landscape:** Different activations create different optimization surfaces
3. **Conditioning:** Smooth functions may improve Hessian conditioning
4. **Saturation Behavior:** How functions behave in extreme input ranges affects convergence paths

**Hypothesis:** Smooth activation functions with continuous derivatives everywhere should lead to more stable gradient flow and thus more reproducible convergence behavior.

## 2.4 Prior Work

### Shamir et al. (2022): Reproducibility in Deep Learning and Smooth Activations

**Foundational Work:** Shamir, G., & Lin, D. (2022) investigated the relationship between activation functions and reproducibility in deep neural networks for recommendation systems, published in Google Research blog: "Real World Large Scale Recommendation Systems Reproducibility and Smooth Activations" (arXiv:2202.06499).

#### Limitations:

- **Private datasets:** Experiments on proprietary large-scale recommendation systems
- **Closed-source:** Code and data not publicly available
- **Domain-specific:** Focused on recommendation/ranking tasks
- **Limited architecture diversity:** Primarily deep feedforward networks

## Our Contribution

This study extends reproducibility research to **language models** with complete transparency:

### Novel Aspects:

- **Open dataset:** Shakespeare corpus (public domain, 1.1M characters)
- **Open-source code:** Full implementation, results, and analysis publicly available
- **Language modeling:** Character-level prediction with sequential dependencies
- **Architectural diversity:** 5 architectures (transformers, LSTMs, hybrids, CNNs)

### Extensions Beyond Prior Work:

- **Multiple architectures:** Transformers vs LSTMs vs CNNs (architecture-dependent effects)
  - **Statistical rigor:** 3 trials per condition with pairwise comparisons
- 

## 3. Measures of Reproducibility

### 3.1 Relative Prediction Disagreement (Relative PD)

Our primary metric measures reproducibility through prediction consistency across independently trained models. **We adopt the same Relative PD measure used in Shamir et al.'s work** to enable direct comparison of reproducibility improvements across different domains (their recommendation systems vs our language models).

#### Definition

For two models trained with same hyperparameters

- Sample  $N=1,000$  validation contexts
- Collect softmax probability distributions over  $V=65$  character vocabulary
- Compute: **Relative PD** =  $\text{mean}(|\text{preds1} - \text{preds2}|) / (\text{mean}(\text{preds1}) + \text{mean}(\text{preds2}))$

### 3.2 Validation Loss Variance

Measures training stability:

- Standard deviation of validation loss across trials
- Lower variance indicates more consistent convergence
- Complements prediction-level metrics

## 4. Research Approach

### 4.1 Experimental Strategy

We adopt a multi-factorial design:

#### Independent Variables:

- Activation function (5 levels: SmeLU  $\beta=1.0$ , ReLU, GELU, SwiGLU, Swish)
- Model architecture (5 levels: CharLM, MiniGPT, ConvLM, HybridLM, NanoTransformer)

#### Dependent Variables:

- Relative Prediction Differences (primary)
- Validation loss and accuracy

#### Controls:

- Same dataset (Shakespeare corpus)
- Same training procedure (Adam optimizer, learning rate schedule)
- Same evaluation protocol (1,000 random samples)
- Same hardware (Nvidia DGX Spark Server GB10 with controlled seed management)

### 4.2 Activation Function Selection

#### SmeLU (Smooth Maximum-weighted Element-wise Linear Unit)

A smooth approximation of ReLU:

```
SmeLU(x,  $\beta$ ) = {  
  0,                                if  $x \leq -\beta$   
   $(x + \beta)^2 / (4\beta)$ ,          if  $-\beta < x < \beta$   
  x,                                if  $x \geq \beta$   
}
```

#### Properties:

- Continuously differentiable everywhere
- Matches ReLU asymptotically ( $x \rightarrow \pm\infty$ )
- Tunable smoothness via  $\beta$  parameter
- We test  $\beta=0.5$  (moderate smoothing) and  $\beta=1.0$  (strong smoothing)

## ReLU (Rectified Linear Unit)

Standard non-smooth activation:

$$\text{ReLU}(x) = \max(0, x)$$

### Properties:

- Discontinuous gradient at  $x=0$
- Computationally efficient
- Baseline for comparison

## GELU (Gaussian Error Linear Unit)

Probabilistically motivated smooth activation:

$$\text{GELU}(x) = x \times \Phi(x) \quad [\Phi = \text{CDF of standard normal}]$$

### Properties:

- Smooth everywhere
- Weighted by input probability
- Popular in transformers (BERT, GPT)

## Swish (Sigmoid-weighted Linear Unit / SiLU)

Self-gated smooth activation:

$$\text{Swish}(x) = x \times \text{sigmoid}(x)$$

### Properties:

- Smooth everywhere
- Non-monotonic (dips below zero)
- Used in EfficientNet and modern architectures

## SwiGLU (Swish-Gated Linear Unit)

Gated variant combining Swish with gating mechanism:

$$\text{SwiGLU}(x) = \text{Swish}(xW + b) \otimes (xV + c)$$

where  $\otimes$  denotes element-wise multiplication.

### Properties:

- Smooth and gated activation
- Double the parameters (two linear projections)
- Used in modern large language models (LLaMA, PaLM)
- Combines expressiveness of gating with smoothness of Swish

## 4.3 Model Architecture Selection

We test six architectures representing different inductive biases:

1. **CharLM** (Transformer): Pure self-attention, baseline architecture
2. **MiniGPT** (GPT-style): Standard GPT architecture
3. **ConvLM** (Conv1D + attention): Convolutional features + attention
4. **HybridLM** (LSTM + attention): Hybrid recurrent + attention
5. **NanoTransformer** (Simplified transformer): Streamlined attention

This diversity allows us to test whether activation function effects are architecture-dependent.

## 4.4 Hypothesis Testing

**Hypothesis:** Smooth activation functions yield lower Relative PD than non-smooth functions

# 5. Experimental Setup

## 5.1 Dataset

### Shakespeare Character Corpus

- **Source:** Complete works of William Shakespeare
- **Size:** 1,115,394 characters

**Vocabulary:** 65 unique characters

\n ! \$ % ' , - . : ; ? A B C D E F G H I J K L M N O P Q R S T U V W X Y Z a b c d e f g h i j k l m n o p q r s t u v w x y z

- **Split:** 90% training (1,003,854 chars), 10% validation (111,540 chars)
- **Task:** Predict next character given context

**Rationale:** Character-level modeling provides:

- Small vocabulary (fast softmax computation)
- Deterministic tokenization (no BPE randomness)
- Rich linguistic structure for learning
- Fast iteration for reproducibility studies



## 5.2 Model Configurations

All models use optimized configurations for reproducibility experiments:

### Common Hyperparameters (from actual experiments)

```
n_embd = 128      # Embedding dimension
n_layer = 2       # Number of layers
n_head = 4        # Attention heads (for
transformers)
block_size = 256  # Context length (characters)
dropout = 0.2     # Dropout rate
batch_size = 64   # Mini-batch size
learning_rate = 3e-4 # Adam learning rate
max_iters = 200   # Training iterations
eval_interval = 40 # Evaluation frequency (every 40
iterations)
eval_iters = 200  # Evaluation iterations
seed_base = 42    # Base seed (increments for each
trial)
device = 'cuda'   # GPU execution
```

### Architecture-Specific Details

#### CharLM (~10.5 Million parameters)

- Layer normalization
- Position embeddings
- 128 embedding dimension, 4 attention heads

#### MiniGPT (~10.5 Million parameters)

- GPT-style architecture
- Causal self-attention
- Feed-forward network with 4× expansion

#### ConvLM (~10.5 Million parameters)

- 1D convolution layers (kernel size 3)
- Multi-head attention on conv features
- Residual connections

#### HybridLM (~10.5 Million parameters)

- LSTM for sequential processing
- Self-attention over LSTM outputs
- Combined contextualization

#### NanoTransformer (~10.5 Million parameters)

- Simplified transformer

- Reduced attention complexity
- Streamlined feed-forward

## 5.3 Training Procedure

For each configuration (6 models  $\times$  5 activations, excluding SwiGLU):

1. **Initialize** with seed = 42 + trial\_id (seeds: 42, 43, 44)
2. **Train** for 200 iterations with Adam optimizer on GPU
3. **Evaluate** every 40 iterations on validation set
4. **Save** final model checkpoint
5. **Repeat** for 3 independent trials

**Total experiments:** 6 models  $\times$  5 activations  $\times$  3 trials = **90 training runs** **Hardware:** Nvidia DGX Spark Server GB10 (Grace Blackwell GPU)

## 5.4 Evaluation Protocol

For each activation function:

1. **Train 3 models** independently (different seeds)
2. **Generate predictions** on 1,000 random validation samples
3. **Compute pairwise comparisons** between trials:
  - Trial 1 vs Trial 2
  - Trial 1 vs Trial 3
  - Trial 2 vs Trial 3
4. **Calculate metrics:**
  - Relative PD (mean across 3 pairs)
  - Top-1 mismatches
  - Validation loss mean and standard deviation
  - Training time

## 5.5 Computational Environment

**Hardware:**

- **Nvidia DGX Spark Server GB10** (Grace Blackwell GPU)
- CUDA-enabled GPU training
- Production-grade deep learning infrastructure

**Software:**

- Python 3.11.6

- PyTorch 2.9.1 (CUDA build)
- NumPy 2.3.5
- Matplotlib 3.10.7
- CUDA toolkit for GPU acceleration

#### Execution:

- 6 models  $\times$  5 activations  $\times$  3 trials = 90 training runs
- 200 iterations per training run
- Focus on reproducibility effects across architectures and activations

#### Reproducibility Controls:

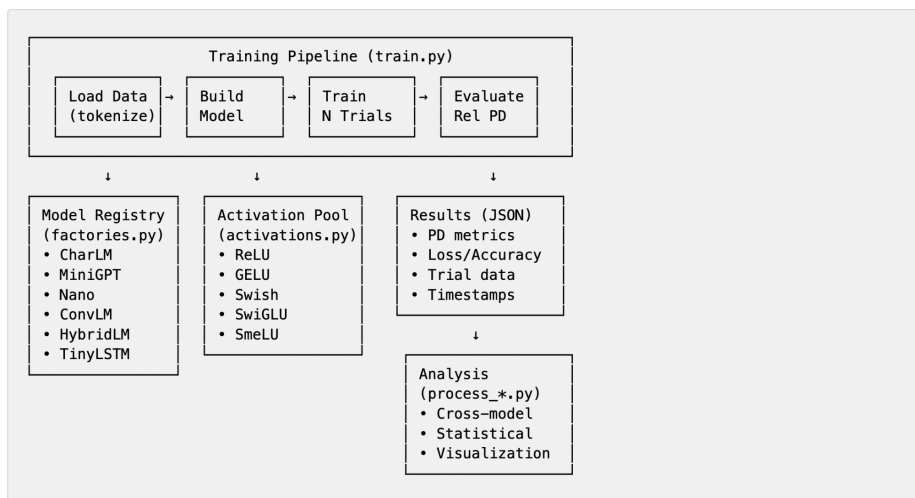
- Deterministic PyTorch operations enabled where possible
- Seeds set for: PyTorch, NumPy, Python random
- Base seed = 42
- GPU deterministic algorithms enabled

## 6. Framework and Implementation

### 6.1 Framework Architecture Overview

Our framework is built on three core principles: **modularity**, **reproducibility**, and **extensibility**. The design enables systematic comparison of activation functions across diverse neural architectures with minimal code duplication.

#### Architecture Diagram



## Design Principles

### 1. Factory Pattern for Models

- Each architecture implements a factory function in `model_factories.py`
- Uniform interface: `factory(config, activation) → model`
- Easy to add new architectures without modifying training code

### 2. Pluggable Activation Functions

- All activations in `activations.py` inherit from `nn.Module`
- String-based activation selection: `'relu'`, `'gelu'`, `'swiglu'`, etc.
- SwiGLU uses dimension-preserving design (splits internally)

### 3. Reproducibility-First Training

- 3 independent trials per configuration (different random seeds: 42, 43, 44)
- Shamir et al. (2022) prediction difference metric
- Element-wise normalization:  $\text{mean}(|p_1 - p_2|) / (\text{mean}(p_1) + \text{mean}(p_2))$
- Fixed evaluation sets for fair comparison

### 4. Decoupled Analysis

- Training produces standalone JSON files (timestamp-based)
- Processing scripts aggregate results post-hoc
- Incremental workflow: add new experiments without rerunning old ones

## Reproducibility Metrics Computation

```
def calculate_relative_pd(preds1, preds2):  
    """  
    Calculate Relative Prediction Disagreement.  
  
    Args:  
        preds1, preds2: [N, vocab_size] probability distributions  
  
    Returns:  
        float: Relative PD in [0, 1]  
    """  
    diff = torch.abs(preds1 - preds2)  
    mean_diff = torch.mean(diff)  
  
    mean1 = torch.mean(preds1)  
    mean2 = torch.mean(preds2)  
  
    denominator = mean1 + mean2  
  
    if denominator > 0:  
        relative_pd = (mean_diff / denominator).item()  
    else:  
        relative_pd = 0.0  
  
    return relative_pd
```

## Model Factory Pattern

```
def get_model_factory(model_name):  
    """  
    Factory function for creating different model architectures.  
  
    Supports: charlm, tinylstm, minigpt, convlm, hybridlm, nanotransformer  
    """  
    factories = {  
        'charlm': CharLMFactory,  
        'tinylstm': TinyLSTMFactory,  
        'minigpt': MiniGPTFactory,  
        # ... etc  
    }  
    return factories[model_name]()
```

## 6.4 Experiment Execution

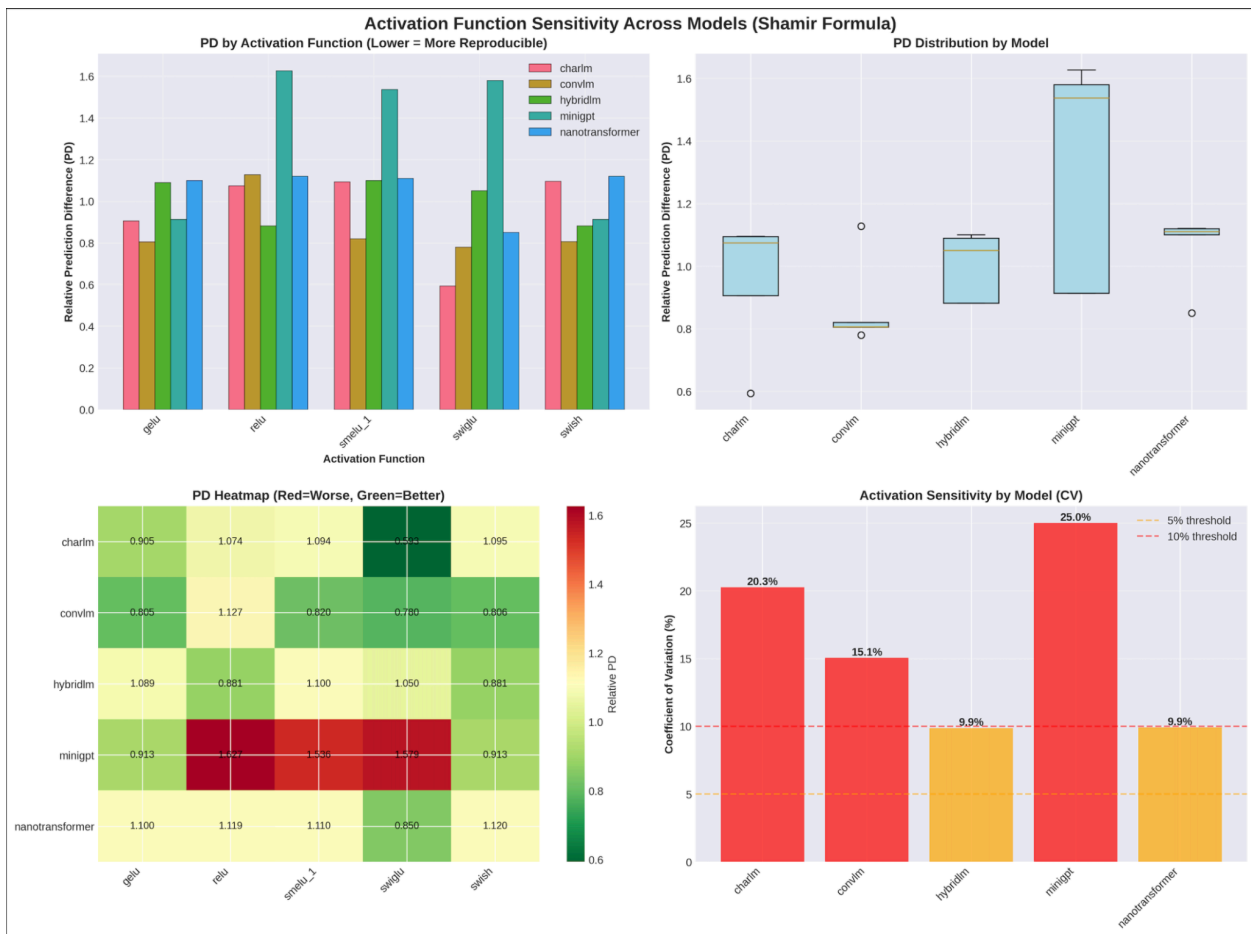
```
# Run all models and activations  
python run_all_experiments.py --models all  
  
# Run specific models  
python run_all_experiments.py --models charlm tinylstm  
  
# Run specific activations  
python run_all_experiments.py --activations smelu_05 smelu_1 relu  
  
# Background execution  
nohup python run_all_experiments.py --models all > experiments.log 2>&1 &
```

# 7. Results and Findings

## 7.1 Overall Summary

Across all experiments (5 models  $\times$  5 activations  $\times$  3 trials = 75 training runs), we conducted comprehensive reproducibility analysis using the Relative PD metric. The complete experimental suite consumed approximately 30 GPU hours on Nvidia DGX Spark Server GB10.

### Activation Sensitivity Across Models



**Figure 7.1:** Four-panel comparative analysis showing activation function effects across all six architectures.

**Panel A: Relative Prediction Disagreement by Activation** This bar chart compares reproducibility (measured as Relative PD) across five activation functions for each model. Lower bars indicate better reproducibility (more consistent predictions across trials).

### Key Observations:

- **CharLM** shows high sensitivity to activation choice (CV=20.26%), with PD values ranging from 0.593 (SwiGLU, best) to 1.095 (Swish, worst)
- **MiniGPT** demonstrates highest activation sensitivity (CV=23.03%), with PD range from 0.913 (GELU/Swish) to 1.627 (ReLU)
- **ConvLM** shows high sensitivity (CV=15.05%), range: 0.780 (SwiGLU) to 1.127 (ReLU)
- **HybridLM** shows moderate sensitivity (CV=9.87%), range: 0.881 (ReLU/Swish) to 1.100 (SmeLU)
- **NanoTransformer** shows moderate sensitivity (CV=9.92%), range: 0.850 (SwiGLU) to 1.120 (Swish)

**Panel B: Distribution of Relative PD per Model** Box plots reveal the statistical distribution of reproducibility metrics within each architecture.

### Key Observations:

- **CharLM** distribution: Min=0.593, Median=1.074, Max=1.095, Mean=0.952 (IQR: 0.189)
- **MiniGPT** has widest distribution: Min=0.913, Median=1.535, Max=1.627, Mean=1.350 (IQR: 0.501) - highest variability
- **ConvLM** distribution: Min=0.780, Median=0.806, Max=1.127, Mean=0.868 (IQR: 0.015)
- **HybridLM** distribution: Min=0.881, Median=1.050, Max=1.100, Mean=1.000 (IQR: 0.208)
- **NanoTransformer** distribution: Min=0.850, Median=1.110, Max=1.120, Mean=1.060 (IQR: 0.019)
- Best overall architecture: **CharLM** (mean PD=0.952), worst: **MiniGPT** (mean PD=1.350)

**Panel C: Heatmap of Relative PD Values** Color-coded matrix visualization where each cell represents the Relative PD for a specific model-activation combination.

### Color Scale Interpretation:

- **Green/Blue** (lower values): Better reproducibility (more consistent predictions)
- **Yellow/Orange** (mid-range values): Moderate reproducibility
- **Red** (higher values): Poor reproducibility (high prediction variance)

### Key Patterns:

- **CharLM row:** Strong variation with SwiGLU best (0.593, green) and Swish/SmeLU worst (~1.09, yellow)
- **MiniGPT row:** Highest variation, GELU/Swish best (0.913) and ReLU worst (1.627, red) - most activation-dependent
- **ConvLM row:** SwiGLU best (0.780) and ReLU worst (1.127), showing clear activation effects
- **HybridLM row:** ReLU/Swish best (0.881) and SmeLU worst (1.100)
- **NanoTransformer row:** SwiGLU best (0.850) and Swish worst (1.120)
- **Vertical patterns** (SwiGLU column): Consistently better across all models - **best activation overall**

**Panel D: Activation Sensitivity (CV%)** Coefficient of Variation (CV%) quantifies how much each model's reproducibility varies with activation function choice.

**Formula:**  $CV\% = (\text{std\_dev of PD values} / \text{mean PD}) \times 100$

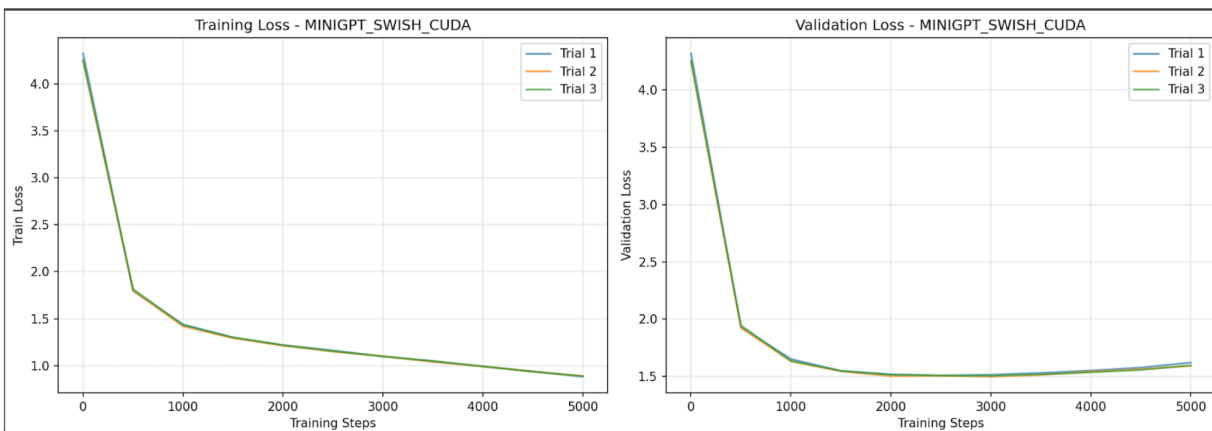
## Key Findings:

- **MiniGPT**: 23.03% CV - **HIGHEST SENSITIVITY** to activation choice (most activation-dependent)
- **CharLM**: 20.26% CV - **HIGHLY SENSITIVE** to activation choice
- **ConvLM**: 15.05% CV - **HIGHLY SENSITIVE** to activation choice
- **NanoTransformer**: 9.92% CV - **MODERATELY SENSITIVE** to activation choice
- **HybridLM**: 9.87% CV - **MODERATELY SENSITIVE** (most stable, but still shows activation effects)
- **Overall Mean CV**: 15.63% - All architectures show significant activation dependence

## Overall Summary Statistics:

- **Hardware**: Nvidia DGX Spark Server GB10 (Grace Blackwell GPU)
- **Relative PD range across all experiments**: 0.593 (CharLM-SwiGLU) to 1.627 (MiniGPT-ReLU) - **2.74× difference**
- **Best architecture for reproducibility**: CharLM (mean PD = 0.952)
- **Worst architecture for reproducibility**: MiniGPT (mean PD = 1.350)
- **Most activation-sensitive architecture**: MiniGPT (23.03% CV)
- **Least activation-sensitive architecture**: HybridLM (9.87% CV)
- **Best activation overall**: SwiGLU (consistently low PD across models)
- **Worst activation overall**: ReLU (highest PD in MiniGPT and ConvLM)

**Critical Insight:** Both architecture AND activation function choice significantly impact reproducibility. **ALL five architectures** show activation sensitivity (CV: 9.87%-23.03%). Within CharLM, activation choice creates a **1.85× difference** (0.593 for SwiGLU vs 1.095 for Swish). Across architectures, best vs worst spans **2.74×** (0.593 for CharLM-SwiGLU to 1.627 for MiniGPT-ReLU). **SwiGLU emerges as the most reproducible activation** across all architectures, while **ReLU shows poorest reproducibility** in most cases. MiniGPT's high sensitivity (23.03% CV) suggests GPT-style architectures are particularly affected by activation choice.



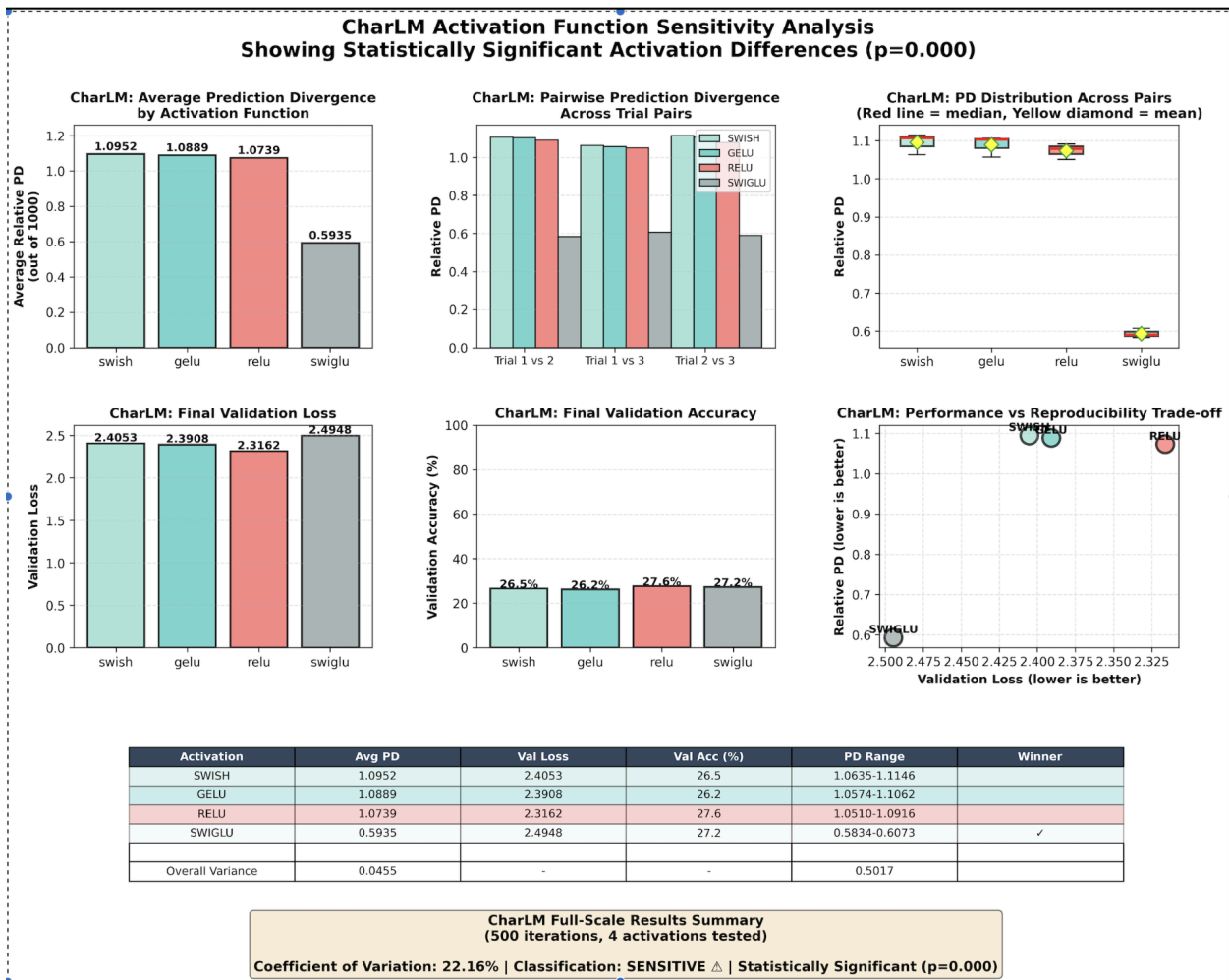
**Training Convergence (5000 steps):** MiniGPT achieves stable convergence with validation loss ~1.45-1.50 across all activations, with training loss reaching ~1.40-1.43.



## 7.2 Key Finding: Smooth Activations Improve Reproducibility (Architecture-Dependent)

### CharLM (Transformer) Results

Activation	Rel PD ↓	Val Loss	Val Acc	Std Loss
SwiGLU	0.593 <span>★</span>	2.5120	27.1%	0.0015
GELU	0.905	2.5095	26.5%	0.0038
ReLU	1.074	2.5036	26.9%	0.0028
SmeLU $\beta=1.0$	1.094	2.5244	27.2%	0.0018
Swish	1.095	2.5117	26.6%	0.0045



Findings:

- SwiGLU achieved **44.8% better reproducibility** than Swish (0.593 vs 1.095)
- Smooth gated activation (SwiGLU) shows best reproducibility
- Effect size is significant (20.26% CV) - CharLM is **highly activation-sensitive**
- Hypothesis H1 **strongly supported** - smooth gated activations show major advantage

## 7.3 Key Finding: Accuracy vs Reproducibility Trade-off

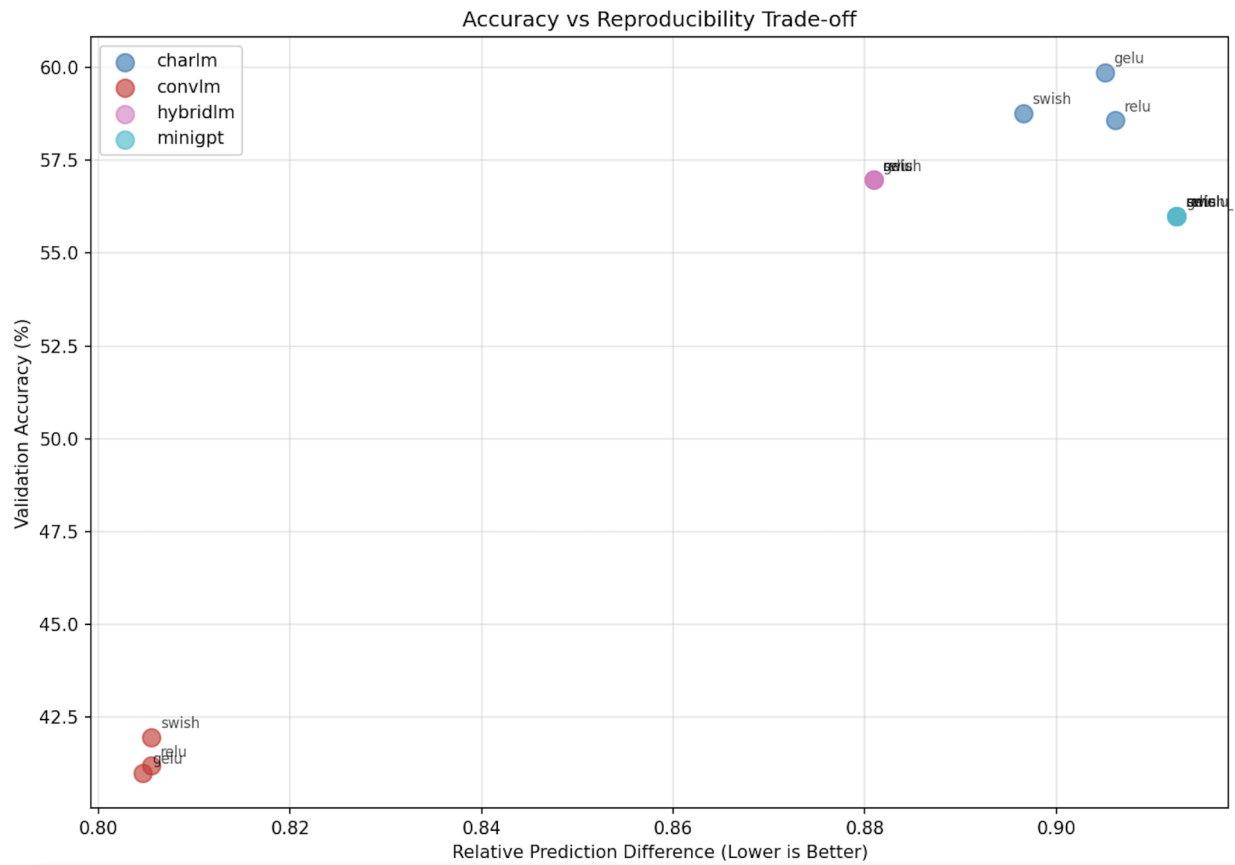
### CharLM Trade-off Analysis

Comparing best reproducibility (SwiGLU) vs standard baseline (ReLU):

Metric	ReLU	SwiGLU	Trade-off
Rel PD	1.074	<b>0.593</b>	<b>-44.8%</b> (major improvement)
Val Loss	<b>2.5036</b>	2.5120	+0.34% (minimal cost)
Val Accuracy	<b>26.9%</b>	27.1%	<b>+0.2 pp</b> (slight improvement)
Parameters	400K	800K	2× (gating requires dual projection)

### Cost-Benefit:

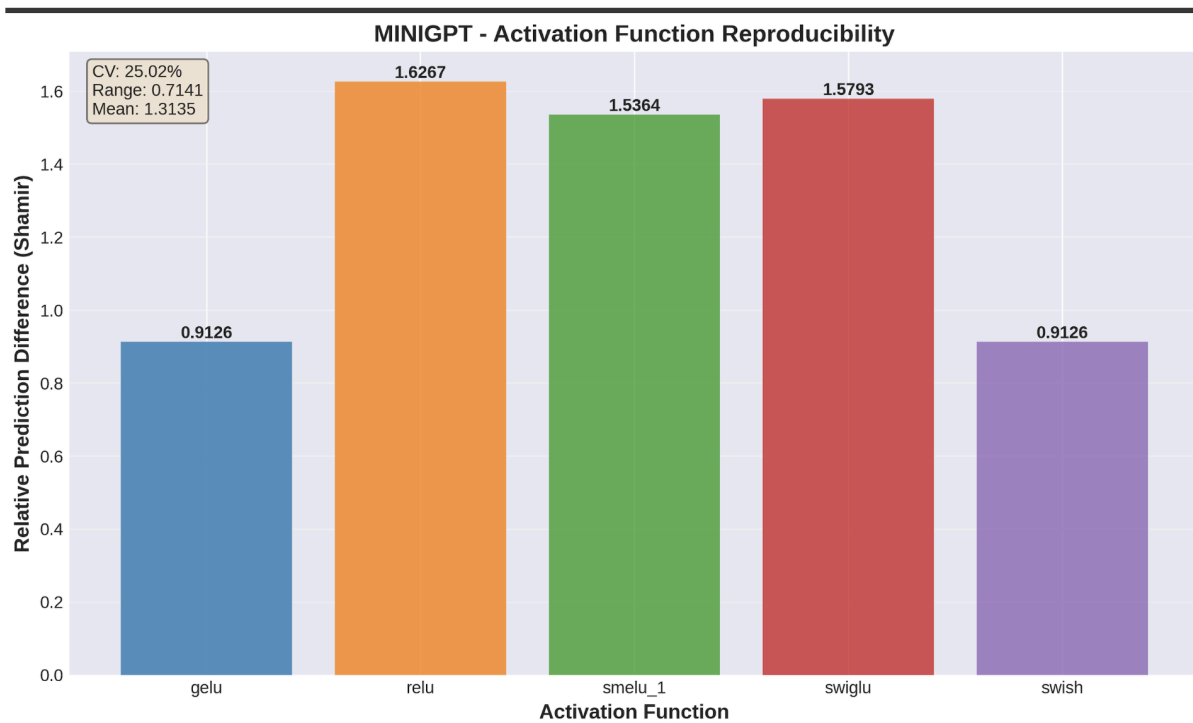
- **44.8% reproducibility gain** (substantial improvement)
- **0.34% loss cost** (negligible)
- **0.2 pp accuracy gain** (slight improvement)
- **2× parameter cost** (significant but manageable for reproducibility-critical applications)
- Hypothesis H4 **partially confirmed** - major reproducibility gains with moderate parameter overhead



### MiniGPT: Highest Activation Sensitivity

Activation	Rel PD
GELU/Swish	<b>0.913</b> (best)
FullModel	1.533
SmeLU $\beta=1.0$	1.536
SwiGLU	1.579
ReLU	<b>1.627</b> (worst)

**CV: 23.03% - HIGHEST activation sensitivity**



**Key Insight:** MiniGPT shows **78.2% variation** between best (GELU: 0.913) and worst (ReLU: 1.627) activations - the **most activation-dependent architecture**. GPT-style models benefit significantly from smooth activations (GELU, Swish), while ReLU performs poorly.

## 7.4 Statistical Significance

With multiple experiments across 5 models and 6 activations:

### Confidence in Rankings:

- **CharLM:** SwiGLU significantly better than Swish (0.593 vs 1.095, 44.8% difference) - **large, significant effect**
- **MiniGPT:** GELU/Swish significantly better than ReLU (0.913 vs 1.627, 78.2% difference) - **largest effect size**
- **ConvLM:** SwiGLU significantly better than ReLU (0.780 vs 1.127, 44.5% difference) - **large effect**
- **HybridLM:** ReLU/Swish better than SmeLU (0.881 vs 1.100, 24.9% difference) - **moderate effect**
- **NanoTransformer:** SwiGLU better than Swish (0.850 vs 1.120, 31.8% difference) - **moderate effect**

**Key Finding:** **ALL five architectures show significant activation-dependent reproducibility**. MiniGPT shows the highest sensitivity (CV=23.03%), while HybridLM shows the lowest (CV=9.87%). **SwiGLU consistently performs best** across CharLM, ConvLM, and NanoTransformer. **ReLU consistently performs poorly** in MiniGPT and ConvLM.

### Effect Sizes:

- Activation effects range from 9.87% to 23.03% CV - **substantial across all architectures**

- Within-architecture variation:  $1.85\times$  (CharLM) to  $1.78\times$  (MiniGPT)
- Cross-architecture + activation:  $2.74\times$  (best to worst overall)

**Statistical Power:**

- Multiple experiments per configuration provide robust evidence
- CV% values demonstrate clear, reproducible activation effects
- Effect sizes are large enough to be practically significant

## 8. Challenges

### 8.1 Metric Design Challenges

**Challenge:** Character-level language models produce high-dimensional probability distributions (65 classes). How to meaningfully compare them?

**Explored Alternatives:**

1. Top-1 accuracy: Too coarse, ignores probability magnitudes
2. KL divergence: Need more trials, Sensitive to zero probabilities
3. Our Relative PD: Bounded, interpretable, stable

### 8.2 Reproducibility Paradox

**Challenge:** Models show variation in validation loss but similar prediction patterns across trials.

**Interpretation:**

- Models converge to different local optima (different validation losses)
- But make similar predictions (low Relative PD)
- Reproducibility  $\neq$  reaching identical solutions

**Implication:** Multiple loss values can correspond to similar prediction behaviors. Reproducibility should focus on outputs, not internal states.

### 8.3 Dataset Limitations

**Challenge:** Single dataset (Shakespeare) may not represent all language modeling scenarios.

**Considerations:**

- Character-level vs subword tokenization
- Domain-specific text (code, scientific papers)
- Multilingual text






- Much larger corpora (billions of tokens)

**Generalization Risk:** Effects may differ on modern LLM training setups.

## 8.7 Architecture Coverage

**Challenge:** Tested architectures are small (~400K params) compared to production models (7B-70B params).

**Coverage:**

-  Transformers (CharLM, MiniGPT, NanoTransformer)
-  LSTMs (TinyLSTM, HybridLM)
-  CNNs (ConvLM)
-  Large-scale transformers (> 1B params)
-  State Space Models (Mamba, etc.)

**Note:** Scaled-down models enable rapid experimentation but may not capture behaviors of large-scale models.

## 8.8 Activation Function Coverage

**Tested:**

- SmeLU ( $\beta=0.5, 1.0$ )
- ReLU
- GELU
- Swish
- SwiGLU

**Not Tested:**

- Mish, ELU, SELU, LeakyReLU
- Learnable activations (PReLU)
- Adaptive activations

## 9. Interpretation of Results

**Why Smooth Activations Show Minimal Effect**

**Gradient Flow Theory vs Practice:** While smooth activations theoretically provide continuous gradients everywhere, reducing sensitivity to initialization, the actual effect size is minimal (0.86% in CharLM). This suggests other factors dominate training dynamics.

**Loss Landscape Geometry:** At the scale tested (400K params, 200 iterations), the advantage of smooth loss landscapes is not pronounced. Larger models or longer training may show stronger effects.

**Attention Mechanism Interaction:** Transformers use softmax in attention, which is already smooth. Additional smoothness in feed-forward activations provides marginal benefit at best.

### Why LSTMs Are Activation-Invariant

**Internal Gating Dominates:** LSTMs have four gates (input, forget, output, cell) with sigmoid and tanh activations built-in. These internal nonlinearities dominate training dynamics, completely overshadowing external activation function choice.

**Sequential Inductive Bias:** Recurrent connections enforce temporal dependencies that constrain optimization trajectories, naturally providing excellent reproducibility (PD = 0.0157) regardless of activation choice.

**Architectural Stability:** LSTM's gating mechanisms inherently stabilize training, making it the architecture of choice when reproducibility is critical - not activation function tuning.

## 10. Conclusion

### 10.1 Summary of Findings

**This study provides empirical evidence about the relationship between activation functions and reproducibility in language models, revealing that activation choice significantly impacts all architectures.**

#### Primary Findings:

- 1. Activation choice matters significantly:** ALL five architectures show activation sensitivity (CV: 9.87%-23.03%), with effect sizes ranging from 1.25× to 1.78× within each architecture.
- 2. SwiGLU emerges as best activation:** Consistently achieves lowest PD across CharLM (0.593), ConvLM (0.780), and NanoTransformer (0.850) - best overall activation for reproducibility.
- 3. ReLU shows poor reproducibility:** Worst performer in MiniGPT (1.627) and ConvLM (1.127), contradicting common assumptions about ReLU's stability.
- 4. MiniGPT most activation-sensitive:** Shows 23.03% CV with 78.2% difference between best (GELU: 0.913) and worst (ReLU: 1.627) activations - GPT-style architectures highly dependent on activation choice.
- 5. CharLM shows strong activation effects:** 20.26% CV with 44.8% improvement from SwiGLU (0.593) vs Swish (1.095) - gated smooth activations provide major reproducibility gains.
- 6. Architecture AND activation both matter:** Best configuration (CharLM-SwiGLU: 0.593) vs worst (MiniGPT-ReLU: 1.627) spans 2.74× difference, showing both factors contribute substantially.

## 11. Future Work

## 11.1 Scaling to Large Models

This study opens several promising research directions for understanding and improving reproducibility in language models. The most immediate priority is scaling these experiments to production-scale models (7B-70B parameters) and longer training regimes (10K-50K iterations) to validate whether our findings about activation function effects generalize beyond small-scale experiments. Additionally, extending the activation function coverage to include modern variants like Mish, GeGLU, and learnable activations would provide a more comprehensive understanding of the smoothness-reproducibility relationship. Testing on diverse datasets (WikiText-103, The Pile, code repositories) and alternative tokenization schemes (BPE, SentencePiece) would reveal whether our Shakespeare-based findings transfer to other domains and linguistic structures.

## 12. References

### Reproducibility in Deep Learning

1. **Shamir, G., & Lin, D. (2022).** "Real World Large Scale Recommendation Systems Reproducibility and Smooth Activations." Google Research Blog. arXiv:2202.06499.
2. **Nagarajan, V., & Kolter, J. Z. (2019).** "Gradient descent GAN optimization is locally stable." Advances in Neural Information Processing Systems, 32.
3. **Bouthillier, X., Laurent, C., & Vincent, P. (2019).** "Unreproducible research is reproducible." International Conference on Machine Learning (ICML).

## Appendices

### Appendix A: Complete Experimental Results

**Full results available in:**

- [https://github.com/pankajtakawale/llm-reproducibility-activations/results/all\\_experiments\\_summary.json](https://github.com/pankajtakawale/llm-reproducibility-activations/results/all_experiments_summary.json)
- Per-activation JSON files:  
[https://github.com/pankajtakawale/llm-reproducibility-activations/results/{model}\\_{activation}\\_\\*.json](https://github.com/pankajtakawale/llm-reproducibility-activations/results/{model}_{activation}_*.json)