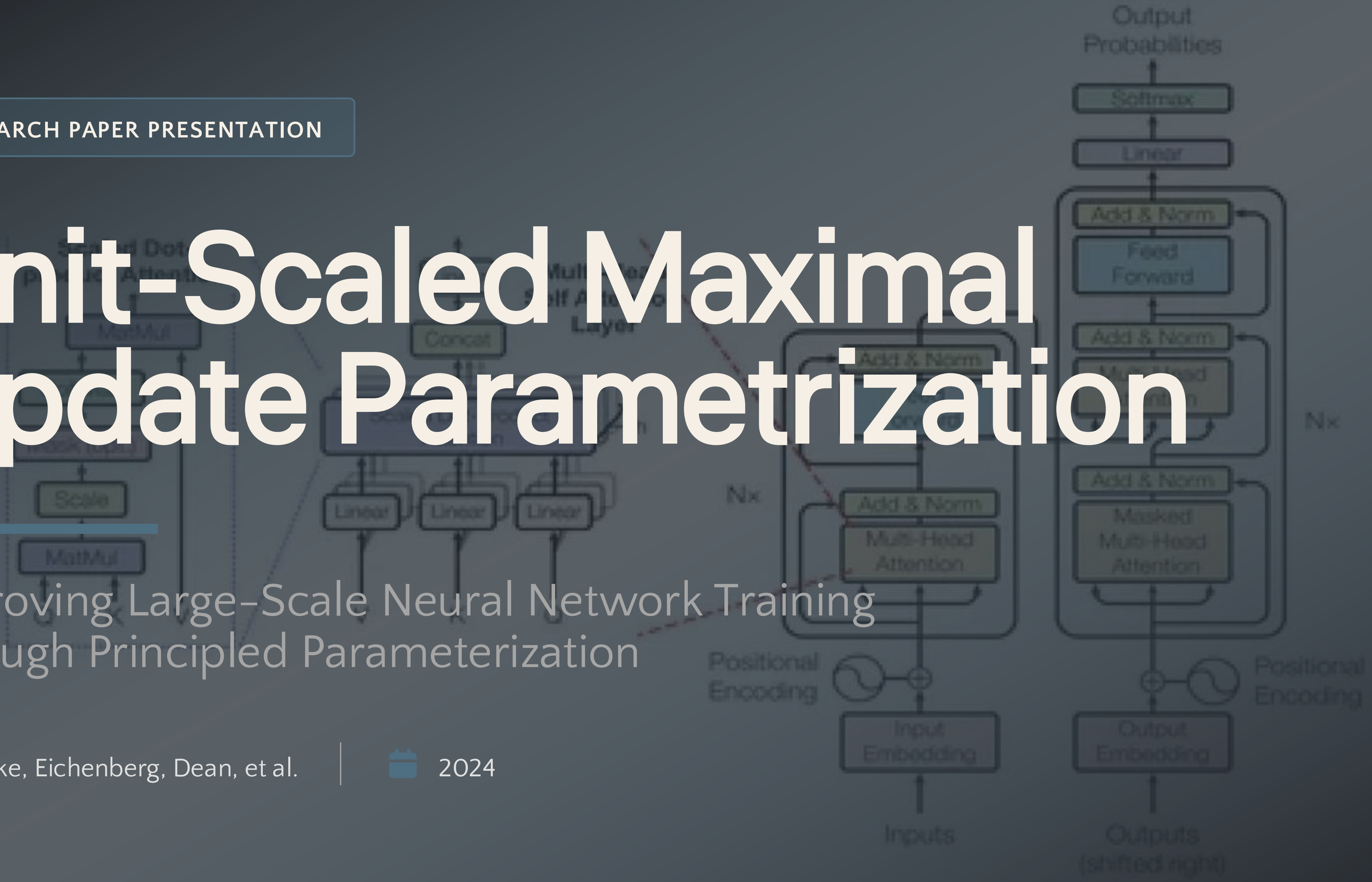


RESEARCH PAPER PRESENTATION

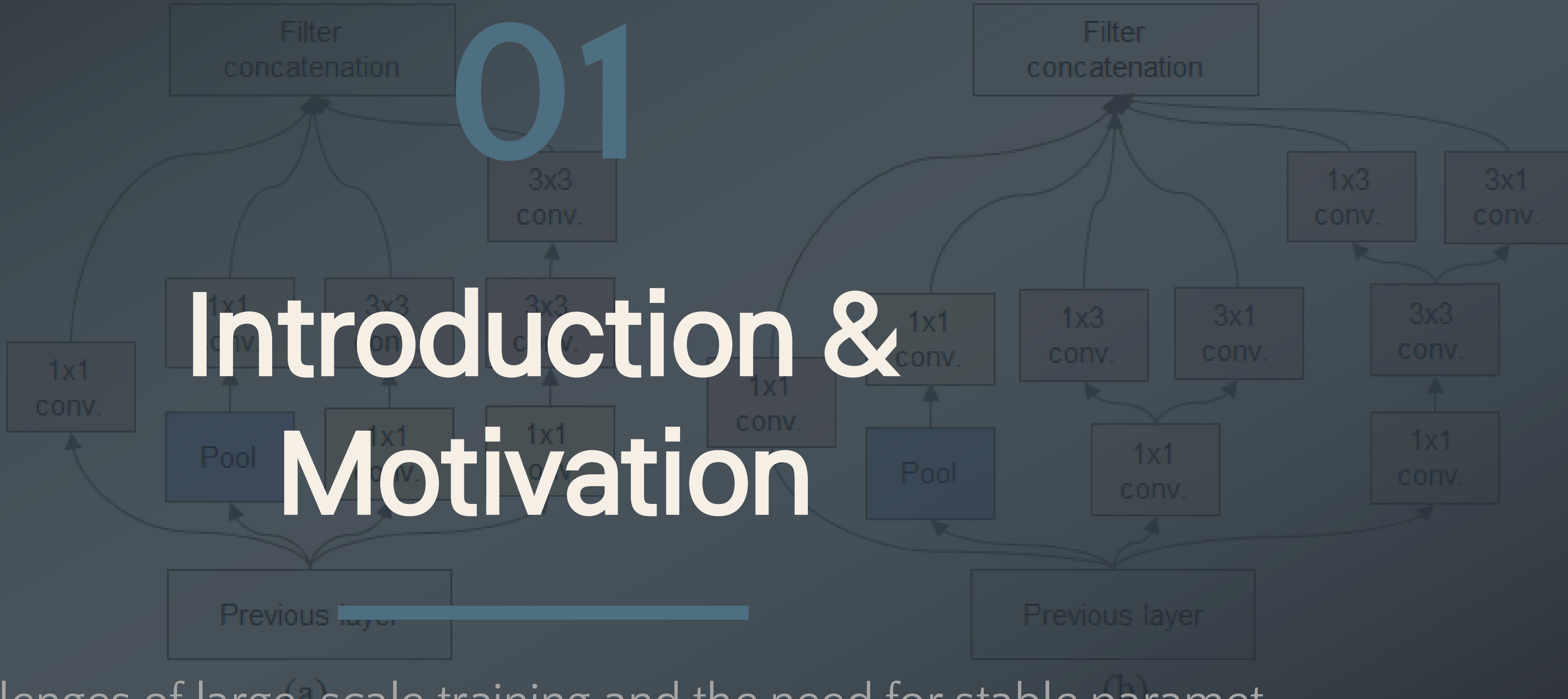
Unit-Scaled Maximal Update Parametrization

Improving Large-Scale Neural Network Training Through Principled Parameterization

Blake, Eichenberg, Dean, et al.

2024

Overview



01

Introduction & Motivation

The challenges of large-scale training and the need for stable paramet erization

INTRODUCTION

The Challenge of Large-Scale Training

Effective approaches for smaller models don't guarantee success at multi-billion-parameter scale

Feature Learning Stability

Ensuring that different parts of the model learn at compatible rates. Some components shouldn't learn too fast while others learn too slow.

CHALLENGE

Imbalanced learning dynamics can cause parts of the network to either blow up or cease learning entirely.

Hyperparameter Stability

Ensuring that optimal hyperparameters for small models remain unchanged as the model size grows.

CHALLENGE

HP search on large models is prohibitively expensive, requiring stable transfer from small proxy models.

Numerical Stability

Ensuring floating-point representations during training stay within the range of a given number format.

CHALLENGE

Low-precision formats like FP8 have limited range, risking overflow/underflow without proper scaling.

The Scale Problem

As models reach **multi-billion parameter scales** , these stability challenges become critical. Traditional training approaches break down, requiring principled parameterization schemes to maintain stability and transferability across model sizes.

The Need for Principled Solutions

Effective large-scale training requires more than just engineering effort—it demands algorithmic innovations that address stability at a fundamental level, ensuring consistent behavior as models continue to grow in size and complexity.

INTRODUCTION

What is Maximal Update Parametrization (µP)?

A theoretical framework for stable scaling of neural networks

Core Goal

Make a model's optimal hyperparameters **independent of its size**, allowing them to be swept using cheap proxy models rather than the full-size target model.

PRACTICAL BENEFIT

Practitioners can reuse HP values found for small proxy models on larger target models, vital for modern LLM training where full HP search is prohibitively expensive.

Theoretical Foundation

µP is derived from **Tensor Programs** theory and is proven to be the **only parametrization** that gives ' maximal feature learning' in the infinite-width limit, unlike standard parametrization which has imba lanced learning.

µTransfer Method

µTransfer describes the process of training many smaller proxy models to evaluate candidate HP va lues, then using the best-performing ones to train a larger target model.

Key Properties

1. **Size Independence**
   Optimal HPs remain consistent across model sizes
2. **Maximal Learning**
   Features continue to improve as models get larger
3. **Cost Efficiency**
   Dramatically reduces HP search cost for large models

Adoption

µP has been adopted by several open LLM training efforts:

- ✓ Cerebras-GPT
- ✓ BTLM-3B-8K
- ✓ LLM360
- ✓ MiniCPM

And indications of use in GPT-4 and Grok

INTRODUCTION

The Gap Between Theory and Practice

Despite strong theoretical foundations, µP faces practical limitations

Efficient HP Search

The standard approach requires **hundreds of random search runs** due to dependencies between hyperparameters, making the search space difficult to explore efficiently.

HP Transfer

µP does not give effective transfer for **Llama-style models** under standard training setups, limiting its practical applicability.

Interpretability

The set of transferable HPs in literature is chosen in an **inconsistent and arbitrary** way, lacking clear justification for which HPs to sweep and how to group them.

Ease-of-Use

Base shape HPs **complicate implementation** and usage, requiring extra 'base' model creation and potentially awkward interactions with other model transforms.

Low-Precision Training

Despite aiming for Θ(1) activations, µP **doesn't leverage** this for actual low-precision training. SP models can train in FP16 while µP models diverge.

The Core Problem

"µP does not necessarily provide the kind of **simple, stable scaling** for which a user might hope."

— Original Paper

The Result: A gap between extensive theory and effective use in practice, motivating the need for an improved approach.

INTRODUCTION

Unit-Scaled Maximal Update Parametrization (u-μP)

Combining μP with Unit Scaling for improved practical performance

A Natural Affinity

u-μP synthesizes two complementary techniques:

μP Maximal Update Parametrization

Ensures that the **scale of activations is independent of model size**, providing consistent training dynamics across different model scales.

Unit Unit Scaling

Ensures that **activations, weights and gradients begin training with a scale o f one**, placing values at the center of floating-point ranges.

Key Insight

This synthesis opens the door to a **simpler scheme** whose default values are nea r-optimal, facilitating a more efficient sweeping strategy.

Key Result

Validation Loss

≤

μP Models

u-μP models reach **equal or lower loss** than comparable μP models

Works out-of-the-box in FP8

More efficient HP search

Better HP transfer

Key Contributions

Six main contributions addressing μP's practical limitations

1 Drawbacks of Standard μP

Show that standard μP has several limitations and **does not give effective transfer** for Llama-style models under standard training setups.

2 Simpler Scaling Rules

u-μP is **easier to implement** than μP, removing unnecessary 'base shape' and initia lization-scale HPs.

3 Out-of-the-Box FP8 Training

u-μP models generate tensors **close to center of floating point range** , enabling si mple .to(float8) casting without dynamic rescaling.

4 Principled, Interpretable HPs

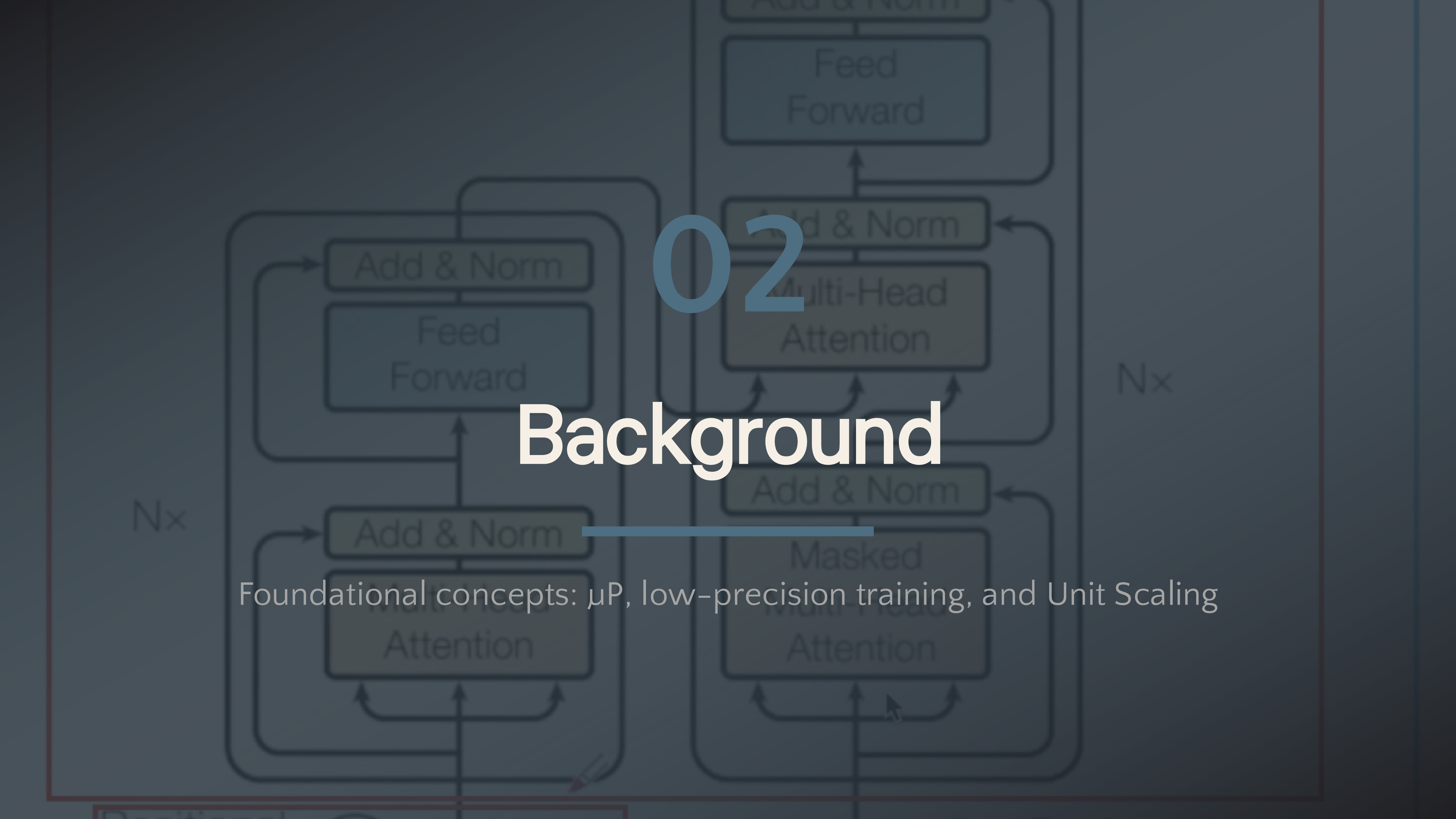
Provide **concrete recommendations** for a good set of transferable HPs, addressing inconsistent choices in literature.

5 Improved HP Transfer

Identify and fix **embedding layer LR scaling problem** , providing better scaling with width.

6 Efficient HP Search

Enable **independent (1D) search** , attaining near-optimal loss when only sweeping the learning rate.



02

Background

Foundational concepts: μP, low-precision training, and Unit Scaling

BACKGROUND

The Maximal Update Parametrization (μP)

Mathematical foundation of abc-parametrizations

ABC-Parametrization Framework

Weights are defined with scaling factors **AW, BW, CW**:

$$w_0 \sim N(0, BW^2)$$
$$W_t = AW \cdot w_t$$
$$w_{t+1} = w_t + CW \cdot \Phi_t(\nabla L_0,\dots,\nabla L_t)$$

Parametrization specifies how **AW, BW, CW** change with model width through width-dependent factors **aW, bW, cW**.

μP Scaling Rules (Table 1)

Input	Hidden	Output
aW: 1	**aW:** 1	**aW:** 1/fan-in
bW: 1	**bW:** 1/√fan-in	**bW:** 1/fan-in
cW: 1	**cW:** 1	**cW:** 1

Weight type determined by fan-in & fan-out dependence on width.

ABC-Symmetry

Key property allowing shifts between scales while preserving learning dynamics:

$$AW \leftarrow AW \cdot \theta$$
$$BW \leftarrow BW / \theta$$
$$CW \leftarrow CW / \theta$$

For any fixed **$\theta > 0$**, network behavior is invariant to these changes.

Complete μP Expressions

With base shapes and HPs:

$$AW \leftarrow \alpha W \cdot (aW / aW\_base)$$
$$BW \leftarrow \sigma W \cdot (bW / bW\_base)$$
$$CW \leftarrow \eta W \cdot (cW / cW\_base)$$

αW: Parameter multiplier

σW: Initialization multiplier

ηW: Adam learning rate multiplier

BACKGROUND

µTransferable Hyperparameters

Which hyperparameters transfer across model sizes?

Three Types of Multipliers

µTransferable HPs function as multipliers contributing to the three abc-multipliers:

αW Parameter Multiplier
Scales the parameter tensor itself: **AW ∝ αW**

σW Initialization Multiplier
Scales initialization variance: **BW ∝ σW**

ηW Learning Rate Multiplier
Scales Adam updates: **CW ∝ ηW**

Base Shape HPs

Two additional (non-µTransferable) HPs:

Base-Width	**Base-Depth**

Specify model shape where µP and SP behavior are same. Implemented by dividing µP scalin g rules by those of fixed-shape model at base-width and base-depth.

The Challenge of Choosing HPs

In theory, search space includes **αW, σW, ηW** for every parameter tensor. In practice, far fe wer HPs are swept:

- Global grouping often used for **σW** and **ηW**
- Many **αW**s dropped or grouped across layers
- Literature shows **variety of approaches** without principled justification

"Practitioners have not justified these choices, appearing to rely on a mixture of precedent and intuition."

HP Interdependencies

Key example: interaction between **σW** and **ηW**

Relative update size determined by:

ηW / σW

Optimal values for σ and η **depend on each other**, making HP search much harder and requiri ng extensive random search.

BACKGROUND

Low-Precision Training

Motivation and challenges for reduced bit-width training

Motivation

All major training bottlenecks see **linear improvements** as bit-width reduces:

Compute	Communication	Storage

With end of Dennard scaling and Moore's law, **low-precision represents one of the most promising avenues** toward increased efficiency in deep learning.

Modern Hardware Support

Recent AI hardware offers **substantial acceleration** for 8-bit FP8 formats:

E4M3	**E5M2**
4 exponent bits, 3 mantissa bits	5 exponent bits, 2 mantissa bits

The Challenge

Reduced range of FP8 formats means they cannot directly represent some values generated during training:

- ↑ **Overflow:** Values too large for format
- ↓ **Underflow:** Values too small for format

Various methods introduced to address this, such as **per-tensor dynamic re-scaling** in Transformer Engine, but with added complexity and potential overheads.

Performance Gains

FP32 (32-bit)	1.0×
FP16 (16-bit)	~2×
FP8 (8-bit)	~4×

Approximate linear improvements in throughput.

BACKGROUND

Unit Scaling

An alternative approach to controlled scaling

Core Principle

Ensure **unit variance** for all tensors at initialization:

Activations	Weights	Gradients
std = 1	std = 1	std = 1

This places values around the **center of floating-point formats' absolute range**, maximizing n umerical stability.

Method Characteristics

Fixed Factors
Calculated at initialization, independent of tensor contents

Lightweight
Only adds overhead of applying static scaling factors

No Dynamic Rescaling
Doesn't re-scale tensors during training

Scaling Rule Example: Matmul

For matrix multiplication, the scaling factor is derived as:

$$1 / \sqrt{\text{fan-in}}$$

This ensures unit-scaled outputs given unit-scaled inputs, applying to all tensors and operations throughout the network.

Empirical Behavior

Unit Scaling provides **ideal starting scale** for gradients, activations and weights. Empirically sho wn to remain stable across multiple architectures, though not guaranteed theoretically.

"Due to its ideal starting scale... this may not be required. Empirically this is shown to be true across multi ple architectures."

Benefits

- Easy to Use
- Low Overhead
- Numerically Stable
- Well-Conditioned

03

Challenges with μP in Practice

Four critical issues limiting μP's practical effectiveness

CHALLENGES WITH µP

Challenge 1: Not All Training Setups Give µTransfer

Standard µP fails on modern LM training without modifications

Three Experimental Setups (Figure 2)

a Tensor Programs V Settings

LR Schedule	Epochs
Constant	High (10000)

Result: Misleadingly good transfer due to **overfitting regime** . Validation loss unusable, transfer artifacts.

b Standard Llama Setup

LR Schedule	Epochs
Cosine	Standard (8192)

Result: Optimal HPs **drift with width** , confirming µP is poor fit for modern LM training.

c Stability Fixes Applied

Norm Layers	Weight Decay

Key Insight

Lingle's finding that directly applying µP to decoder LM **fails to provide LR transfer** across width is significant given µP's primary use in LM training.

"Given that the primary use of µP in the literature has been LM training of this kind, this result suggests a significant limitation."

Reconciliation with Tensor Programs V

Tensor Programs V showed **strong LR transfer**, but their setup had several atypical aspects:

- ✓ Constant LR schedule (not cosine)
- ✓ High number of epochs (overfitting regime)
- ✓ Smaller dataset (WikiText-2)

These make **validation loss unusable** and transfer **misleadingly good**.

Required Stability Fixes

To recover strong transfer:

1. **Non-parametric norms:** Remove trainable parameters from LayerNorm/RMSNorm
2. **Independent weight decay:** Use AdamW with independent weight decay

CHALLENGES WITH μP

Challenge 2: Which Hyperparameters to Sweep?

Lack of principled approach for HP selection

The Problem

Selecting HPs to sweep = choosing subset of per-tensor **α_W, σ_W, η_W** HPs and grouping across/within layers.

Literature shows **variety of approaches** without principled justification—relying on precedent and intuition.

"Practitioners have not justified these choices, appearing to rely on a mixture of precedent and intuition."

Problematic Grouping Example

Consider commonly-used **global σinit HP**:

FFN Swish

std(x_swish) ∝ σW_gate

Linear Effect

Self-Attention

std(x_attn) ∝ σW_Q · σW_K

Quadratic Effect

Global σ HP has **linear effect on FFN** but **quadratic effect on attention**, suggesting this grouping may be unideal.

HP Interdependencies

Not all HPs are independent. Key example: interaction between **σ_W** and **η_W**

Relative weight update size:

$$\eta_W / \sigma_W$$

Not either HP individually

Because of this, **optimal values for σ and η depend on each other**, making HP search much harder.

"This can make the problem of HP search much harder, and may be why hundreds of random-search runs have been required for sweeps in the literature."

Two Major Downsides

1. **Not All Groupings Are Suitable**
   Global HPs can have different effects on different parts of model
2. **Not All HPs Are Independent**
   Interactions make search space difficult to explore

CHALLENGES WITH μP

Challenge 3: Base Shape Complicates Usage

Unnecessary complexity for most practitioners

The Problem

Most practitioners are **unlikely to require alignment with an SP model**, making base-width and base-depth choice unclear.

Literature standard:

256

appears lacking principled motivation

The fact that base shapes are not dropped entirely suggests they may be beneficial, but their ro le is poorly understood.

Implementation Complexity

Engineering complications from base shape HPs (Equation 3):

```
# mup library implementation
proxy_model = MupModel(d_model=128, ...)
base_model = MupModel(d_model=256, ...)
mup.set_base_shapes(proxy_model, base_model)
```

Requires **extra 'base' model creation** and **original model re-initialization**, potentially interacting awkwardly with other model transforms.

Implementation Example

The mup library reflects this complexity:

```
# Must create separate base model
proxy_model = MupModel(d_model=128, ...)
base_model = MupModel(d_model=256, ...)
# Must call set_base_shapes
mup.set_base_shapes(proxy_model, base_model)
# proxy_model is re-initialized here!
```

This can interact awkwardly with:

- Quantization transforms
- Model compilation
- Other model transformations

Practical Questions

What base width to use?
No clear guidance for practitioners not aligning with SP

Why not drop entirely?
If beneficial, unclear why standard value of 256 chosen

CHALLENGES WITH μP

Challenge 4: μP Struggles with Low-Precision

Contradiction between theory and practice

The Contradiction

μP's Stated Goal

μP aims for **Θ(1)-sized coordinates** in the limit, specifically so values can be represented using **finite-range floating-point numbers**.

↓

The Reality

Despite numerical stability being central to μP theory, this is **not leveraged** to ensure μP models can actually be trained in low-precision.

Empirical Evidence

From Tensor Programs V experiments:

✓ **Standard Parametrization** Trains successfully in **FP16**	✗ **μP** Diverges (attributed to **gradient underflow**)

"For the LLM runs in Tensor Programs V the SP model trains successfully in FP16, while the μP model diverges (attributed to underflow of gradients)."

The Core Issue

μP only ensures activations are **of order Θ(1)**, ignoring constant factors. This is not sufficient for reliable low-precision training.

What's missing:

- Precise control of weight scaling
- Precise control of gradient scaling
- Guarantee of optimal numerical range

Unit Scaling addresses this by enforcing **unit scale** across all tensors at initialization.

Setting the Stage for u-μP

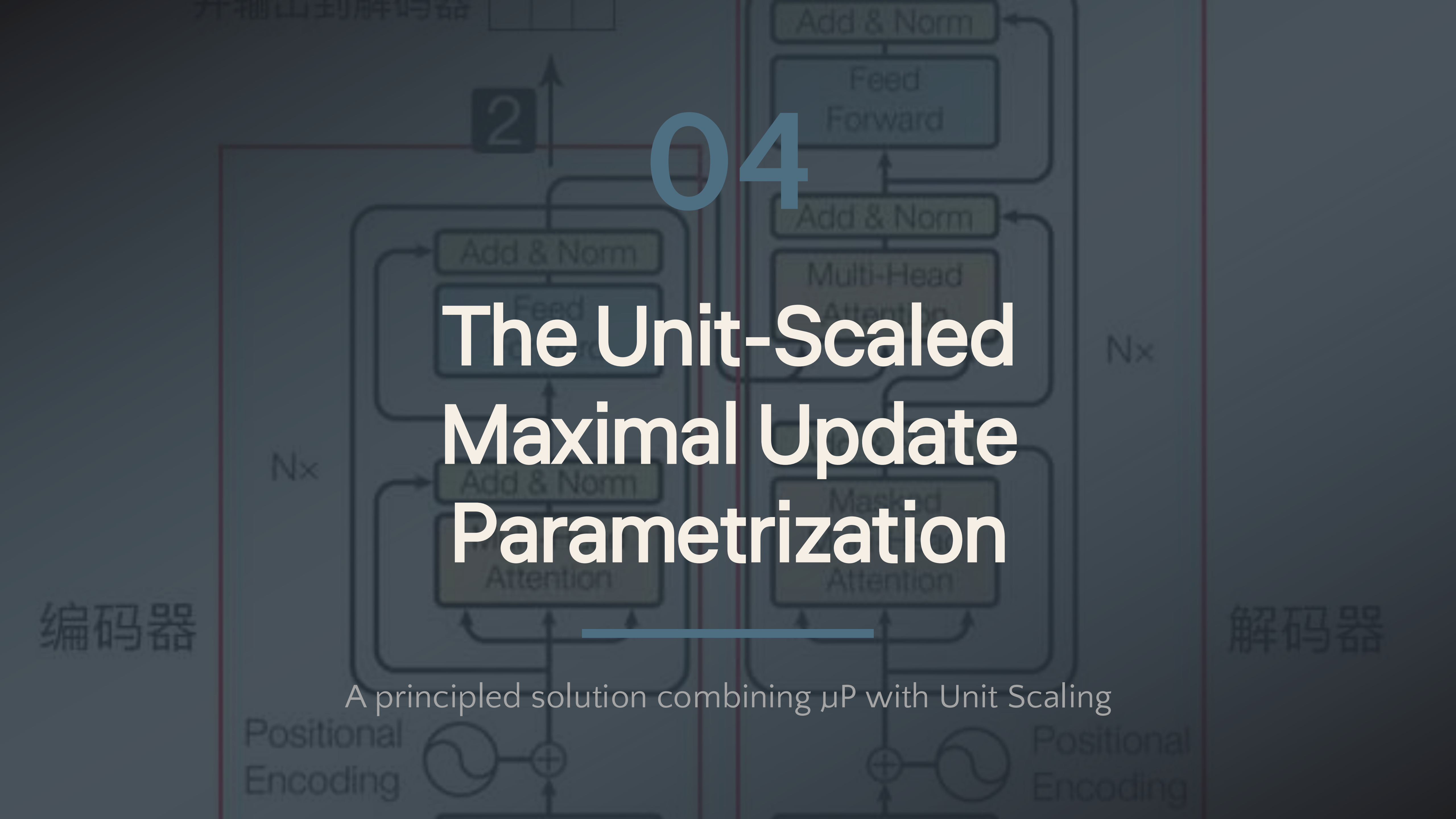
This contradiction motivates combining μP with Unit Scaling:

μP
Size-independent scaling

+

Unit Scaling
Precise numerical conditioning

=



04

The Unit-Scaled Maximal Update Parametrization

A principled solution combining μP with Unit Scaling

THE u-μP SOLUTION

Combining μP with Unit Scaling

Transforming μP into a valid Unit Scaling scheme

Unification Strategy

Unit Scaling provides rules for **all operations** while μP only does so for **parametrized ones**. We need to address the differences.

1 Drop Unnecessary HPs

Drop **σW** and **base-fan-in** HPs entirely. Associate **αW** HPs with functions instead of weights.

2 Apply abc-Symmetry

For hidden weights, shift scales by **√fan-in** to arrive at unit-scaled scheme: **AW←1, BW←1, CW←η/√fan-in**

3 Handle Output Layers

Apply **1/√fan-in** factor in backward pass only (valid via cut-edge rule).

4 Modify Embedding LR Rule

Change input LR scaling rule to **cW←1/√fan-out** for improved transfer.

Key Changes Made

Simplification

Removed σW and base shape HPs

Unit Initialization

All weights start with unit variance

Operation-Associated α HPs

Moved from weights to operations

Embedding LR Fix

New 1/√fan-out rule

Final Result

The final u-μP scheme from Table 2 combines with standard Unit Scaling rules for non-matmul operations (Appendix B).

Note: Scaling rules must be combined with Unit Scaling rules for o ther operations, implemented in the library (Appendix D).

THE u-μP SOLUTION

The Final u-μP Scaling Rules

Complete comparison with μP (Table 2)

Weight Type	Scheme	Parameter (AW)	Initialization (BW)	Adam LR (CW)
Input	μP	αemb/σinit	σinit	η·base-fan-in/fan-in
	u-μP	1	1	η/√fan-out
Hidden	μP	1/σinit	σinit/√fan-in	η·base-fan-in/fan-in
	u-μP	1	1/√fan-in	η
Output	μP	αout/σinit	σinit/fan-in	η
	u-μP	αout	1/fan-in	η
Residual	μP	√(base-depth/depth)	—	√(base-depth/depth)
	u-μP	1/√depth	—	1/√depth

✗ **Removed**

- σinit HPs
- Base shape HPs
- Complex αW weight associations

✓ **Simplified**

- Unit initializations (BW = 1)
- Unit parameter multipliers (AW)
- Cleaner LR scaling rules

+ **Added**

- Operation-associated α HPs
- 1/√fan-out embedding LR rule
- Unit scaling for all operations

THE u-μP SOLUTION

Out-of-the-Box FP8 Training

Simple mixed-precision scheme leveraging Unit Scaling principles

Why μP Struggles with Low Precision

μP only ensures activations are **of order Θ(1)**, ignoring constant factors. This is insufficient for relia ble low-precision training.

The stricter condition:

unit scale

across ALL tensors at initialization

This provides a way to leverage μP's rules to make low-precision training work reliably.

Critical Tensors

Most scales stabilize, but certain tensors exhibit scale growth:

- ⚠ Inputs to attention dense projection
- ⚠ Inputs to final FFN matmul
- ⚠ Decoder head weight (kept in higher precision)

Proposed Mixed Precision Scheme

✓ Non-Critical Matmuls

Cast to FP8 in forward and backward:

Input	Weight	Gradient
E4M3	E4M3	E5M2

Query, key, value, FFN input layers

✗ Critical Tensors

Remain in higher precision:

- Attention dense projection
- Final FFN matmul
- Embedding layer
- Residual additions
- Nonlinear functions

Performance

~70%

of matmul computations in transformer block perform natively in FP8

Assumes standard architecture (e.g., Llama). Dynamic per-tensor scaling can still be applied to critical tensors i f desired.

THE u-μP SOLUTION

A Principled Approach to Hyperparameters

Four ideal criteria for HP selection

Four Ideal Criteria

1 **Minimal Cardinality**

Use **as few HPs as possible**

2 **Maximal Expressivity**

Ability to express any model defined using per-tensor αW, σW, ηW HPs

3 **Minimal Interdependency**

Optimal value of each HP should **not depend on other HPs**

4 **Interpretability**

Clear explanation of what an HP's value 'means'

Three-Step Derivation

1 **Drop σW HPs**

Permute under abc-symmetry with **θ = σW**, leaving just **αW, ηW**. Weights begin with unit sca le.

2 **Re-parametrize αW HPs**

Several **αW** HPs combine linearly—use single HP associated with **operations** instead of weight s (e.g., **αattn-softmax**).

3 **Single Global η**

Use single global **η** and group **α** HPs across layers. Per-tensor **ηW** HPs show minimal benefit a fter tuning global **η**.

Benefits

Reduced HP Count	Less Interdependence
Clear Interpretation	Simpler Search

THE u-μP SOLUTION

The u-μP Hyperparameter Set

Comparison of HP schemes (Table 3)

HP Type	SP	μP	u-μP
Basic HPs (Bold - Commonly Swept)			
Learning Rate (η)			
Init Scale (σinit)	—		—
Embedding Mult (αemb)	—	—	
Embedding LR (ηemb)	—	—	
Extended HPs (Non-Bold)			
Base Width	—		—
Base Depth	—		—
αattn	—		—
αffn-act	—	—	
αattn-softmax	—	—	
αres	—	—	
αres-attn-ratio	—	—	

THE u-μP SOLUTION

A New Embedding Learning Rate Rule

Fixing poor transfer of embedding LR across width

The Problem

Though theoretical transfer properties proved for μP, **not all HPs have shown empirical transfer**. For extended μP transformer HPs, poor transfer across width for **embedding LR multiplier $\hat{\eta}_{emb}$**.

Standard μP rule:

$$c_{emb} = 1$$

(constant in width)

Poor multiplier transfer suggests rule is **mis-specified**.

The Solution

More effective rule keeps optimal **$\hat{\eta}_{emb}$** value same regardless of width:

Proposed u-μP rule:

$$c_{emb} = 1/\sqrt{\text{fan-out}}$$

(scales with width)

This keeps optimal **$\hat{\eta}_{emb}$** value constant across width, enabling better transfer.

Performance Comparison (Figure 3)

✗ Constant Rule (μP)

Leads to **diminishing returns** as width increases. Larger models show minimal improvement.

✓ Proposed Rule (u-μP)

Continues to work well at scale. **2048-width u-μP model matches 4096-width μP model** performance.

Key Factor

This improved embedding LR rule is **key factor** in u-μP's improved performance over μP.

Note: We offer no theoretical justification for our rule, which we leave to further work.

THE u-µP SOLUTION

Hyperparameter Search Strategy

From random search to independent search

Standard µP Approach

Standard approach for µTransfer is **random search over all HPs simultaneously**. Sweeping individual HPs separately is challenging due to dependencies.

Challenges:

- HPs cannot be swept independently
- Requires hundreds of random runs
- High computational cost

This is due to **HP interdependencies** shown in Figure 4, making search space hard to explore.

u-µP Independent Search

u-µP's HPs admit **independent search strategy** due to reduced interdependencies (criterion 3).

1 **Sweep LR**
1D line search for optimal learning rate (9 runs)

2 **Sweep Other HPs**
1D sweeps of other HPs in parallel (can use binary search)

3 **Combine Results**
Combine best settings and re-evaluate (6 runs)

Key Finding (Figure 1a)

9 runs

LR sweep phase alone reaches **near-optimal loss**

When other HPs fixed at 1 (unit-scaled inputs), suggesting **unit scale at initialization is close to ideal** f or learning.

"For u-µP the LR-sweep phase of independent search alone is sufficient to reach near-optimal loss (totaling 9 r uns)."

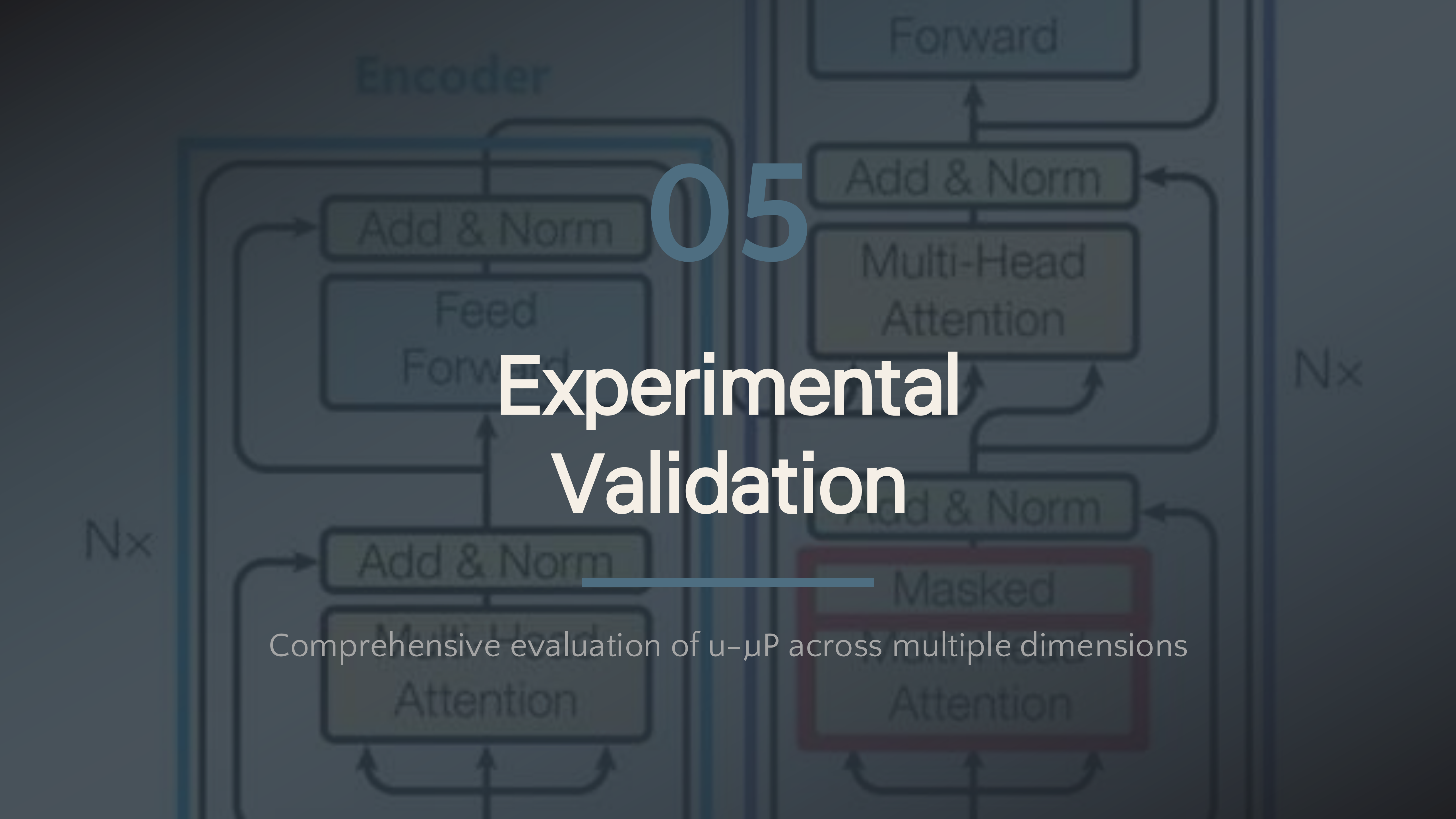
Comparison Results

µP

Still requires non-LR HPs to be swept for reasonable loss. Fixing HPs at 1 results in arbitrarily-scaled inputs → worse training.

u-µP

Lower dependence enables combining individually-swept HPs. Random search unnecessary → much more eff icient.



05

Experimental Validation

Comprehensive evaluation of u-μP across multiple dimensions

EXPERIMENTAL VALIDATION

Experimental Setup

Configuration and training details (Table 5)

Architecture & Data

Architecture	Llama
Dataset	WikiText-103
Seq Length	256
Vocab Size	32K

Model Scale

Width	64 → 4096
Depth	4 → 64
Parameters	19.5M → 1.07B

Training Settings

Batch Size	8192
Training Steps	8192
LR Schedule	Cosine to 10%
Warmup	2000 steps

Stability Fixes Applied

- ✓ Non-parametric RMSNorm
- ✓ Independent AdamW
- ✓ Under-fitting regime

Optimizer

Optimizer	AdamW
β1	0.9
β2	0.999
ε	1e-8
Weight Decay	2ˆ-13 (independent)

Notes

- Settings applied to both μP and u-μP for fair comparison
- Large-scale runs use SlimPajama dataset (300B tokens)
- Up to 1.07B parameter models for comprehensive evaluati on

EXPERIMENTAL VALIDATION

Quantifying Hyperparameter Interdependence

Measuring HP dependencies using transfer error

Transfer Error Metric

Empirical measure of HP dependency (Algorithm 1):

For pair of HPs:

1. 1. One HP 'fixed', other for 'transfer'
2. 2. Take best transfer HP for each non-optimal fixed HP
3. 3. Use with optimal fixed HP
4. 4. Measure difference from minimum loss

Higher transfer error = **greater dependency**

Results (Figure 4)

μP	u-μP
0.03	0.005

6× Improvement

in reducing HP interdependence

Visual Comparison (Figures 14-15)

μP: Strong Coupling

Diagonal structure in grids shows dependence:

· ($\hat{\eta}$emb, σinit) pair

· (η, αattn) pair

Optima depend on each other

u-μP: Reduced Coupling

Less diagonal structure, more independent optima

HPs can be swept more separately

Implications

Lower Risk
Reduced risk of small transfer errors leading to large degradations

Separable Sweeping
Enables more separable HP search, making random search unnecessary

EXPERIMENTAL VALIDATION

Hyperparameter Search Results

Independent search vs. random search (Figure 1a)

Key Findings

1 **u-μP: LR Sweep is Sufficient**

LR-sweep phase of independent search **alone reaches near-optimal loss** (9 r uns total). Other HPs fixed at 1 = unit-scaled inputs.

2 **Unit Scale is Near-Optimal**

Unit scale at initialization is **close to ideal scaling** for effective learning. Not as serted a priori—empirical finding.

3 **μP Requires Full Sweeps**

μP **still requires non-LR HPs** to be swept. Fixing HPs at 1 = arbitrarily-scaled i nputs → worse training.

HP Dependencies Impact

✗ **μP: Combined Mults Fail**

"Combined mults" phase causes **loss spike for μP** due to HP dependencies (Figure 4). HPs cannot be swept independently and used together.

✓ **u-μP: Independent Works**

Lower dependence means **this can be done for u-μP**, making random search unnecessar y. More efficient and reliable.

Practical Impact

u-μP enables **dramatically more efficient HP search**. Instead of hundreds of rando m search runs, practitioners can find near-optimal HPs with **just 9 LR sweep runs**, making large-scale model development significantly more accessible.

EXPERIMENTAL VALIDATION

Hyperparameter Transfer Results

Comprehensive transfer evaluation across multiple axes (Figures 1b, 5)

Width Transfer

Optimal LR is constant across width under u-μP

No diminishing returns at larger widths

Key Result:

2048-width u-μP model matches 4096-width μP model

Other Axes Transfer

Training Steps
Small drift in optimal LR

Batch Size
Small drift in optimal LR

Depth
Larger drift, least reliable for transfer

Proxy Model Recommendations

★ **Width: Most Reliable**
Primary axis for proxy model differences

✓ **Steps & Batch: Good**
Moderate changes permissible

⚠ **Depth: Least Reliable**
Only modest changes recommended

Non-LR HP Transfer

u-μP
Non-LR HPs have **approximately constant optima** across width

μP
η̂emb and σinit have poor transfer due to scaling issues

Optimal Values

Optimal values for non-LR HPs are all **close to 1**. In practice, **dropping these HPs entirely** is potentially viable for similar models and training setups.

EXPERIMENTAL VALIDATION

FP8 Training Validation

Out-of-the-box low-precision training at scale

RMS Analysis (Figure 6)

RMS captures the larger of mean and scale, testing whether tensors likely to suffer ov er/underflow.

µP

Gradients and weights with **low RMS**, at risk of FP8 underflow

u-µP

Tensors have RMS that **starts close to 1** and remains within E4M3 range

Key Observation:

u-µP maintains proper scale throughout training, enabling reliable FP8 computatio n.

Proof-of-Concept

8k-step training (Figure 1c):

Only **small degradation** versus FP32

At this scale, critical tensors can still be cast to FP8 using E5M2, gradients can even us e E4M3.

Large-Scale Training (Figure 7, Table 4)

Setup

- 7B parameter models
- 300B SlimPajama tokens
- FP8 mixed-precision scheme

Results

All FP8 runs **converge with no significant loss degradation** vs BF16

Downstream Performance (7B)

Benchmark	SP	u-µP	u-µP FP8
MMLU	29.6	29.0	**31.2**
HellaSwag	52.4	53.4	**53.4**
OpenBook QA	27.8	31.6	**29.6**
PIQA	76.5	77.1	**77.6**
TriviaQA	22.2	23.4	**21.3**

FP8 model mostly on par with BF16 models

EXPERIMENTAL VALIDATION

Summary of Experimental Findings

Key takeaways from comprehensive evaluation

Five Key Results

1 Reduced HP Interdependence

u-μP reduces transfer error by **6× (0.005 vs 0.03)**, enabling independent search strategy.

2 Efficient HP Search

Independent search with **LR sweep alone reaches near-optimal loss**, making random search unnecessary.

3 Superior HP Transfer

u-μP provides **better HP transfer across width** without diminishing returns, unlike μP.

4 FP8 Training Works

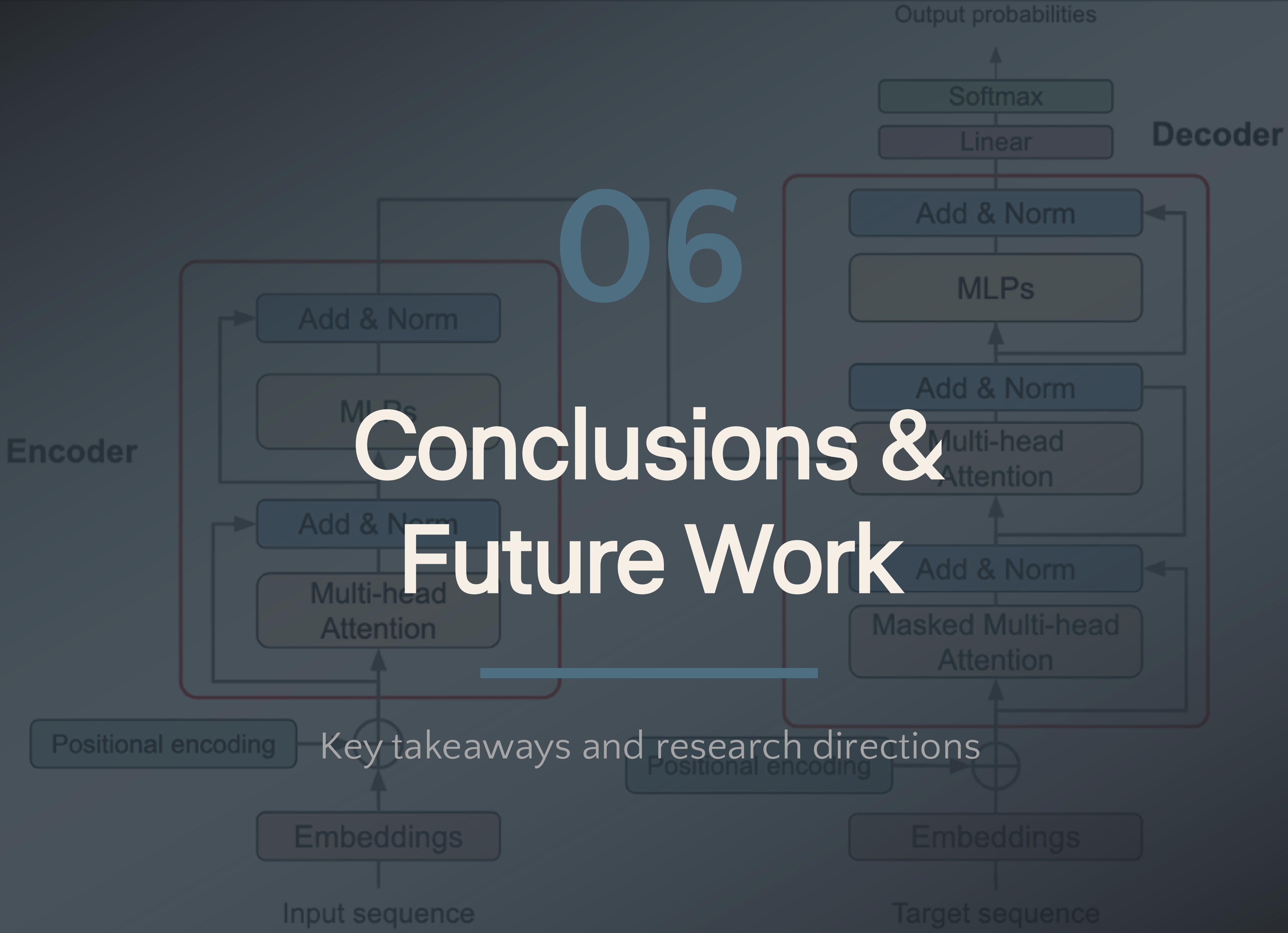
FP8 training works **out-of-the-box** with simple casting on matmul inputs, achieving competitive performance at 7B scale.

5 Optimal HPs Near 1

Optimal non-LR HP values **cluster around 1**, suggesting defaults are reasonable for similar models.

🏆 Overall Impact

u-μP successfully bridges gap between μP theory and practice, providing **principled, stable, and efficient** approach to large-scale neural network training. Combines theoretical rigor with practical usability, enabling **reliable HP transfer, simple FP8 training, and dramatically more efficient HP search**.



06

Conclusions & Future Work

Key takeaways and research directions

CONCLUSIONS

Conclusions

u-μP as a bridge between theory and practice

u-μP in Context

u-μP is an **improved version of μP that satisfies Unit Scaling**. Through careful anal ysis guided by first principles, we identified:

Interpretable HP Set
Minimal interdependencies, facilitating efficient independent sweeping

Robust FP8 Scheme
Stability properties + Unit Scaling enable simple mixed precision

✓ **Practical Validation**
Works in realistic large-scale training scenarios

Evidence for Unit Scaling

u-μP provides **further evidence** that the principle of Unit Scaling is beneficial for model design:

- ✓ Provides **near-optimal scaling** when HPs fixed at 1
- ✓ Enables **out-of-the-box FP8 training**
- ✓ Simplifies **HP search and transfer**

Bridging Theory and Practice

u-μP addresses the **gap between μP's extensive theory and its effective use**:

Before Complex, unstable	**After** Simple, stable
Before Poor transfer	**After** Reliable transfer

FUTURE DIRECTIONS

Limitations and Future Work

Open questions and research opportunities

Current Limitations

Theoretical Gaps

Some choices like the **modified embedding LR rule** are only justified by empiric al observations, lacking theoretical explanation.

No Training Guarantees

Neither μP nor Unit Scaling **guarantee well-behaved network quantities** over t he course of training, particularly **scale growth in critical layers**.

Architecture Limitations

u-μP designed and validated on **Llama-style architectures**. Applicability to oth er architectures needs exploration.

Future Research Directions

1. Theoretical Analysis
Provide theoretical justification for embedding LR rule and other empirical findings

2. Scale Growth Understanding
Better understand and mitigate scale growth in critical layers during training

3. Lower Precision Formats
Push low-precision training further with u-μP as starting point

4. Architecture Generalization
Extend u-μP principles to other architectures beyond Llama-style models

5. Integration with Other Techniques
Combine with QK-norm, z-loss, zero-initialization for even more stable training

Theory → Practice

Bridging the gap between μP theory and practical implementat ion

Stability → Efficiency

Achieving both training stability and computational efficiency

Research → Production

Enabling practical deployment in real-world training scenarios

PRACTICAL GUIDE

Practical Implementation Guide

Recipe for applying u-μP to your models

Prerequisites

Before applying u-μP, apply these stability fixes:

1. Remove trainable parameters from normalization layers
2. Use the independent form of AdamW
3. Ensure training is in the under-fitting regime

Implementation Steps

1 Replace Operations

Use unit-scaled versions with static scales (Appendix B)

2 Parameter Initialization

Initialize parameters with unit variance

3 Apply LR Scaling

Use appropriate LR scaling rules (Table 2)

HP Search Process

Choose HP Set

From Table 3:

· **Basic:** η, α_{emb}, η_{emb} (can be effective)

· **Extended:** Add $\alpha_{ffn\text{-}act}$, $\alpha_{attn\text{-}softmax}$, α_{res}, etc.

Proxy Model Config

Width most reliable for transfer. Steps and batch size also good. Depth least reliable.

Independent Search

Follow process in Section 4.5: LR sweep → individual HP sweeps → combine

FP8 Training

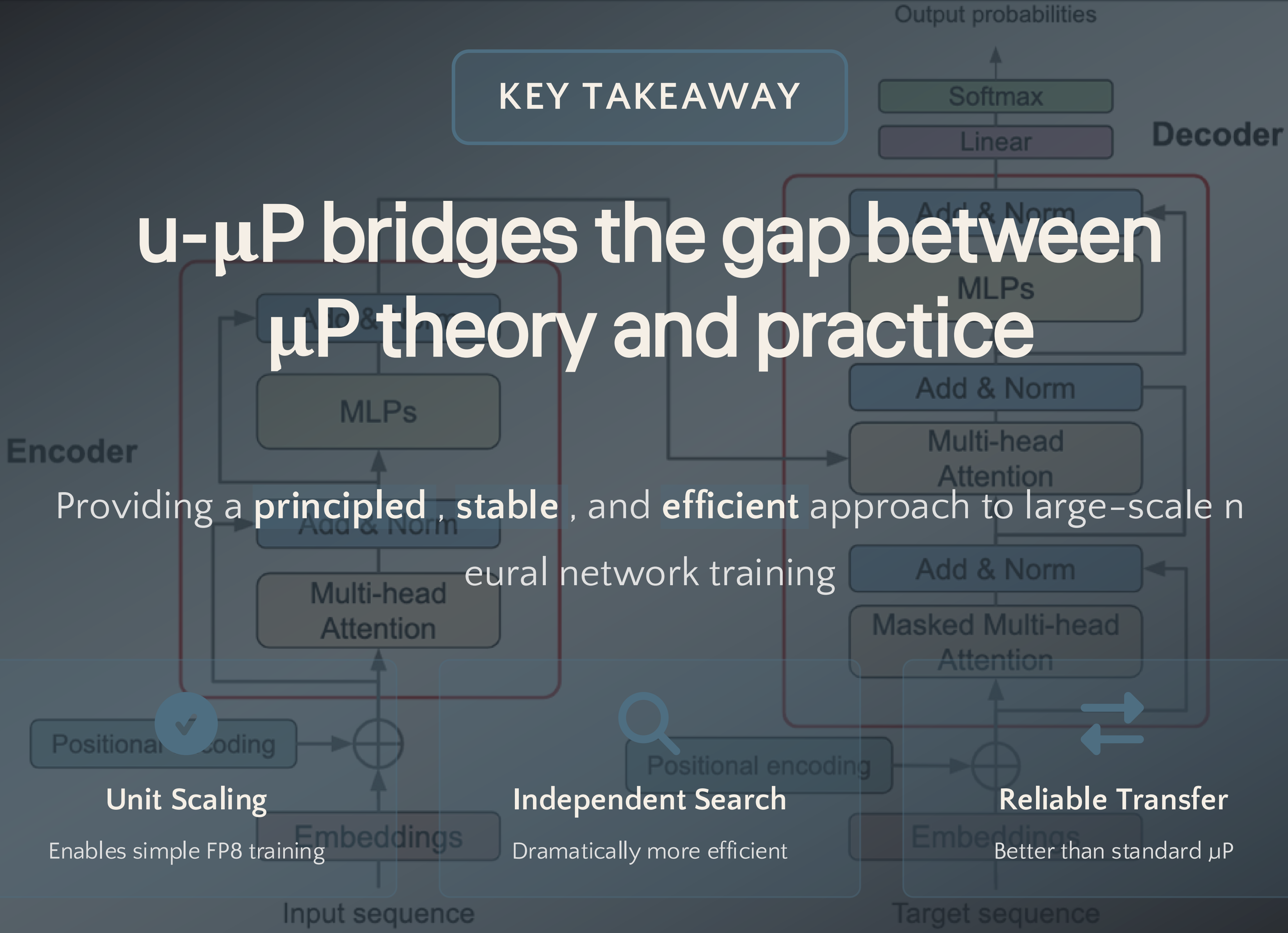
Train target model in FP8 by placing casts on matmul inputs:

Large-scale notes:

Two operations may require lightweight dynamic re-scaling due to scale drift over time

Library Support

All functionality provided in **Unit Scaling library** (Appendix D) to avoid users having to implement th
emselves.



KEY TAKEAWAY

u-μP bridges the gap between μP theory and practice

Providing a **principled** , **stable** , and **efficient** approach to large-scale n
eural network training

Unit Scaling

Enables simple FP8 training

Independent Search

Dramatically more efficient

Reliable Transfer

Better than standard μP