

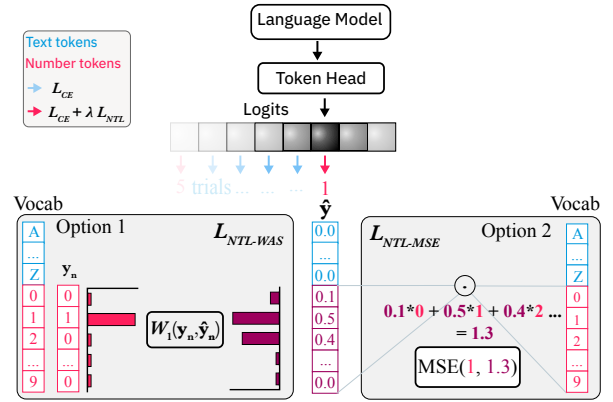
# Regress, Don't Guess – A Regression-like Loss on Number Tokens for Language Models

Jonas Zausinger<sup>\*12</sup> Lars Pennig<sup>\*1234</sup> Anamarija Kozina<sup>12</sup> Sean Sdahl<sup>12</sup> Julian Sikora<sup>12</sup>  
 Adrian Dendorfer<sup>12</sup> Timofey Kuznetsov<sup>12</sup> Mohamad Hagog<sup>15</sup> Nina Wiedemann<sup>6</sup> Kacper Chlodny<sup>12</sup>  
 Vincent Limbach<sup>12</sup> Anna Ketteler<sup>12</sup> Thorben Prein<sup>12</sup> Vishwa Mohan Singh<sup>15</sup> Michael Danziger<sup>7</sup>  
 Jannis Born<sup>7</sup>

## Abstract

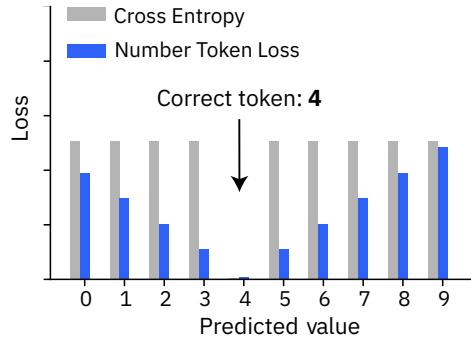
While language models have exceptional capabilities at text generation, they lack a natural inductive bias for emitting numbers and thus struggle in tasks involving quantitative reasoning, especially arithmetic. One fundamental limitation is the nature of the cross-entropy (CE) loss, which assumes a nominal scale and thus cannot convey proximity between generated number tokens. In response, we here present a regression-like loss that operates purely on token level. Our proposed Number Token Loss (NTL) comes in two flavors and minimizes either the  $L_p$  norm or the Wasserstein distance between the *numerical values* of the real and predicted number tokens. NTL can easily be added to any language model and extend the CE objective during training without runtime overhead. We evaluate the proposed scheme on various mathematical datasets and find that it consistently improves performance in math-related tasks. In a direct comparison on a regression task, we find that NTL can match the performance of a regression head, despite operating on token level. Finally, we scale NTL up to 3B parameter models and observe improved performance, demonstrating its potential for seamless integration into LLMs. We hope to inspire LLM developers to improve their pretraining objectives and distribute NTL as a minimalistic and lightweight PyPI package `ntloss`: <https://github.com/ai4sd/number-token-loss>. Development code for full paper reproduction is available separately.

<sup>\*</sup>Equal contribution <sup>1</sup>TUM.ai <sup>2</sup>Technical University of Munich <sup>3</sup>Helmholtz Munich <sup>4</sup>Munich Center for Machine Learning (MCML) <sup>5</sup>LMU Munich <sup>6</sup>ETH Zurich <sup>7</sup>IBM Research Europe. Correspondence to: Jannis Born <jab@zurich.ibm.com>.



(a) The two flavors of the proposed Number Token Loss (NTL).

Loss as a function of predicted number token



(b) Comparison of NTL to CE.

**Figure 1. (a) Number Token Loss (NTL).** The NTL allows computing a regression-like loss directly on a language modeling head. We propose two schemes:  $\mathcal{L}_{\text{NTL-MSE}}$  uses a dot product of number token values and their probabilities;  $\mathcal{L}_{\text{NTL-WAS}}$  uses the Wasserstein-1 distance between label and prediction distributions. **(b)** CE is nominal-scale and thus assigns equal loss to all incorrect predictions whereas NTL increases with distance from ground truth like a regression loss.

## 1. Introduction

As coined by Thawani et al. (2021), numbers in natural texts are *ubiquitous* and *important*, yet systematically *neglected* by language models (LMs). Even worse, while Transformers (Vaswani et al., 2017) were invented for NLP, they have permeated various scientific domains (Jumper et al., 2021; Boiko et al., 2023), where tabular/numerical data is more prevalent than in NLP and often governs task definitions: Molecules are labeled with drug efficacy, chemical reactions with yield, and synthesis procedures are natural text interspersed with quantities and times. Prior art for joint language-number modeling suggested the use of verifiers (Cobbe et al., 2021; Li et al., 2023), chain-of-thought (CoT) reasoning (Zhong et al., 2024; Wei et al., 2022; Lee & Kim, 2023), calculators or code interpreters (Qu et al., 2025; Gao et al., 2023) to yield improved performance in LMs. On a model-level, customized ideas such as reverting numbers (Zhang-Li et al., 2024), right-to-left tokenization (Singh & Strouse, 2024; Lee et al., 2023), or joint prediction of multiple number tokens have been proposed (Bachmann & Nagarajan, 2024). Still, LMs notoriously struggle even with simple arithmetic tasks like three-digit multiplication (Dziri et al., 2024). We argue that all these strategies – reformulating the task, trying to correct answers *a posteriori* with calculators, or using significantly more compute (CoR) – avoid the fundamental, underlying problem: number representation in LMs is poor, for multiple reasons:

1. **Tokenization:** Standard subword tokenization splits numbers into arbitrary tokens, disrupting their structure and losing a significant amount of numerical information (Wallace et al., 2019). Mitigation strategies include scientific notation (Zhang et al., 2020) or digit-level tokenization (Geva et al., 2020), which may also preserve the decimal order of each digit (Born & Manica, 2023).
2. **Embedding:** Canonically, LMs have to recover the structure of numbers from data because number token embeddings are learned like any other. To mitigate that, many flavors of numeracy-preserving word embeddings exist (Sundararaman et al., 2020; Born & Manica, 2023; Golkar et al., 2023), often akin to positional encodings and sometimes adapted for special cases like angular embeddings for modular arithmetics (Stevens et al., 2024; Saxena et al., 2024).
3. **Sequential prediction:** Token-wise decoding fails to account for the greater significance of higher-order digits in a number. Zhang-Li et al. (2024) find that reversing the digit order and human-inspired, step-by-step calculations for addition and multiplication improve accuracy.
4. **Training objective:** Cross-entropy (CE) loss assumes a nominal scale, thus it fails to convey the proximity between numbers, effectively inducing a semi-supervised setting. For example, predicting a 3 instead of a 2 will not generally induce lower loss than a 9.

In this paper, we aim to equip LMs with better inductive biases to handle combinations of textual and numerical data, such as math word problems or scientific datasets. We propose two versions of a regression loss on number tokens that respect numerical proximity and can be effectively combined with standard CE (Figure 1a). The first version of the number token loss (NTL) computes the Mean Squared Error (MSE) between the numerical value of the label and the predicted class probabilities weighted by their numerical token value. The second version computes the Wasserstein distance between the distribution of the predicted number probabilities and the ground truth distribution, which is the one-hot encoding of the label.

In both cases, the NTL has the following key characteristics. **(1) NTL is model agnostic.** It can be applied to any language model (Transformer, Mamba, etc.) in any style (encoder-decoder, decoder-only etc). **(2) NTL is plug-and-play for any LM** because it makes minimal assumptions about the vocabulary. It only requires a map between tokens (strings) and their numerical value (float) and is thus compatible with digit-level as well as multi-digit tokenization<sup>1</sup>. **(3) NTL does not add computational overhead.** The most capable NTL version (NTL-WAS) slows down the loss calculation only by 1% compared to cross-entropy. This difference becomes negligible over the whole training step (see later in Figure 3).

Our empirical analysis further shows that: **(4) NTL consistently improves performance in mathematical tasks** across architectures, compared to cross-entropy alone. **(5) On a real regression task, a LM with NTL performs on-par with a regression head** and improves by 10% over a LM with standard CE. **(6) NTL does not hamper performance on text-only tasks** and **(7) NTL scales well to billion-size models**. Therefore nothing speaks against using it during LLM pretraining.

## 2. Number Token Loss (NTL)

Inevitably, some tokens of any LM correspond to digits or numbers, not to text. Our basic idea is to combine the logits of these *number tokens* with their numerical values to compute a loss. Such a loss can effectively augment cross-entropy by taking numerical proximity of number tokens into account. Note that outside NLP, some systematic studies of "regression by classification" exist (Shah & Aamodt, 2023; Stewart et al., 2023), yet a principled approach that facilitates seamless integration into LLMs lacks entirely.

<sup>1</sup>digit-level, e.g.: LLaMA2 (Touvron et al., 2023), DeepSeek-V2 (DeepSeek-AI et al., 2024) and PaLM (Chowdhery et al., 2022); multi-digit, e.g. LLaMA3 (Grattafiori et al., 2024) or T5 (Raffel et al., 2020).

**NTL with  $\mathcal{L}_p$  norm (e.g., NTL-MSE)** This loss maps the predicted token probabilities to a real-valued output by calculating the dot product of the probabilities and their corresponding numerical token value (cf. Figure 1a). This transformation enables the application of standard regression losses, such as the MSE, to compare the predicted output against the numerical value of the ground truth token. Assume a model  $f(\cdot)$ , input tokens  $\mathbf{x}_{\leq i}$  ( $i \leq N$ ) and a vocabulary  $V$ . Let  $\hat{\mathbf{y}}_i := f(\mathbf{x}_{\leq i})$  be the predicted probability distribution,  $\mathbf{y}_i$  the one-hot label with  $y_i$  as the numerical value of the ground truth token and  $\mathcal{V} : V \rightarrow \mathbb{R}$  a map to convert tokens (strings) to their numerical values (floats), with the index range  $s \dots t$  representing the number tokens. Then we compute NTL-MSE:

$$\mathcal{L}_{\text{NTL-MSE}} = \frac{1}{N} \sum_i (y_i - \hat{\mathbf{y}}_i^{s:t} \circ \mathcal{V}^{s:t})^2 \quad (1)$$

Notably, instead of a nominal-scale loss like CE, the NTL-MSE effectively conveys proximity between numbers. For example, if the label is 4 and the LM predicts 5 instead of 9, the loss will be lower, matching our intuitive expectation. This is in contrast to the CE which gives constant loss regardless of the proximity of the number due to its nominal nature (cf. Figure 1b). By changing the  $p$ -order in NTL-MSE, different  $L_p$ -norm losses can be obtained (e.g., NTL-MAE). Huber loss is also compatible. While NTL- $L_p$  describes a family of losses, we note that NTL-MSE has been proposed already in concurrent work by Lukasik et al. (2025) where it is called RAFT and has shown improved performance of LLMs on pure regression tasks with up to five number digits. However, NTL- $L_p$  has a non-unique minimum and can thus return spuriously low loss for incorrect predictions. Consider e.g., a sample with label 4 where 50% of the mass is on 0 and 50% on token 8, then NTL will be zero (see Figure 2). While such cases are rare due to the softmax emphasizing logit differences, combining NTL with CE loss helps correct spurious cases. But to address the non-unique minima more systematically we propose a second, refined version.

**NTL with Wasserstein-1 Distance (NTL-WAS)** To measure the similarity of a predicted probability distribution of number tokens and a ground truth distribution, we take inspiration from Optimal Transport (OT) and leverage the discrete Wasserstein-1 distance, generally defined as:

$$W_c(P, Q) = \min_{\gamma \in \Gamma(P, Q)} \sum_i \sum_j \gamma_{ij} c(u_i - v_j) \quad (2)$$

where  $u_i$  and  $v_j$  are points,  $P$  and  $Q$  their associated probabilities respectively, and  $c$  a function specifying the transport costs.  $\gamma$  is a coupling between  $P$  and  $Q$ , where  $\gamma_{ij}$  specifies the mass to be transported from  $u_i$  to  $v_j$ . We apply OT to measure the difference between the distributions  $\hat{\mathbf{y}}_i$  and  $\mathbf{y}_i$

by defining the cost function  $c$  as the Euclidean distance between the numerical token values:

$$W_{\text{LM}}(P, Q) = \min_{\gamma \in \Gamma(\mathbf{y}_i^{s:t}, \hat{\mathbf{y}}_i^{s:t})} \sum_{j=s}^t \sum_{k=s}^t \gamma_{jk} \|\mathcal{V}^j - \mathcal{V}^k\| \quad (3)$$

**Note on Cost Function:** Equation 3 is very flexible – by changing the cost function  $c$ , NTL can be applied even when the number tokens lie in arbitrary, non-Euclidean spaces. The cost function does not need to adhere to the definition of a distance metric and can instead be specified as a pairwise matrix between number tokens. NTL could thus find utility also in exotic cases such as modular arithmetic (Charton, 2023; Kera et al., 2024; Gromov, 2023). In this paper, we mainly assume standard Euclidean distance due to its general applicability. A practical example where a non-euclidean cost is useful are multi-digit tokenizers which often contain individual tokens for very large number (Section 4.5 contains explicit results on non-euclidean cost).

Computing the quantity in Equation 3 requires solving a linear program and is thus not generally differentiable. Approximations could be made for the general case via entropic regularization (Cuturi, 2013), but our application of  $W_c$  entails two special cases that enable its efficient and differentiable computation: First, if  $\mathbf{y}_i$  is one-hot,  $\mathcal{L}_{\text{NTL-WAS}}$  coincides with the weighted sum of the absolute differences from the label and the numerical token values:

$$\mathcal{L}_{\text{NTL-WAS}} = \frac{1}{N} \sum_{i=1}^N \sum_{j=s}^t \hat{\mathbf{y}}_i^j |y_i - \mathcal{V}^j| \quad (4)$$

Second, if the number token indices  $s \dots t$  are sorted by numerical value  $\mathcal{V}^{s:t}$  and the values are equidistantly spaced, the Wasserstein-1-distance can be computed using the cumulative distribution function  $\text{CDF}(\cdot)$ :

$$\mathcal{L}_{\text{NTL-WAS-CDF}} = \frac{1}{N} \sum_{i=1}^N |\text{CDF}(\mathbf{y}_i^{s:t}) - \text{CDF}(\hat{\mathbf{y}}_i^{s:t})| \quad (5)$$

The advantage of  $\mathcal{L}_{\text{NTL-WAS-CDF}}$  is that it does not require the label  $\mathbf{y}_i$  to be one-hot – instead it supports arbitrary target distributions obtained e.g., via label smoothing on number tokens (Wang et al., 2025). In practice, unless mentioned otherwise, we compute  $\mathcal{L}_{\text{NTL-WAS}}$  via Equation 4 because it is 230x faster than Equation 5 and does not require sorting or equidistant spacing and is thus applicable beyond digit-level tokenization. As one can see in Figure 1b,  $\mathcal{L}_{\text{NTL-WAS}}$  not only conveys proximity between numbers correctly but also eliminates the problem of non-unique minima in  $\mathcal{L}_{\text{NTL-MSE}}$  (Figure 2). Both versions of the NTL are scaled with  $\lambda$  (default 0.3) and added to the standard CE loss:

$$\mathcal{L} = \mathcal{L}_{\text{CE}} + \lambda \mathcal{L}_{\text{NTL}} \quad (6)$$

Note that both versions of the NTL yield zero loss for all non-numerical tokens (see Appendix A.2 for pseudo-code).

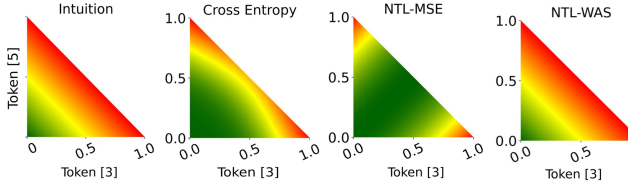


Figure 2. The heatmap plot shows the respective loss for a given combination of the class probabilities for tokens 3 and 5, where the ground truth is token 4. The behavior of the NTL-WAS is closest to the intuitive desired behavior of the loss function, while the NTL-MSE does not have a unique minimum.

### 3. Experiment Setup

#### 3.1. Backbone T5 and Model Variants

We use a T5 backbone (Raffel et al., 2020) (Appendix A.3) for most of our experiments, unless stated otherwise, due to its flexible encoder-decoder architecture and its success in various natural language processing tasks. We built upon the T5 implementation and language modeling trainer from transformers (Wolf et al., 2020). For the multitask mathematics experiments in Section 4.1 we use T5-Base with 220M parameters. For the scaling experiment in Section 4.7 we use T5-3B. For ablation studies and other experiments, we use T5-Small with 60M parameters. To illustrate the model-agnostic nature of NTL, we evaluate it on both GPT-2 and IBM Granite model variants. We always initialize the models with their pre-trained weights. Building up on prior work (Geva et al., 2020), we adapt the tokenization scheme to tokenize all numbers on the digit level, while leaving all other tokens unchanged. Although both NTL versions are compatible with any tokenizer, single-digit tokenization improves performance for both CE and NTL on mathematical tasks in our experiments (see subsection 4.5).

#### 3.2. Baselines

We compare to three related methods.

First, the **Regression Transformer** (Born & Manica, 2023) which tokenizes numbers on digit level (considering both the position and value of each digit) and leverages a fixed number embedding that preserves relative proximity of the numbers (details in Appendix A.4).

Second, **xVal** (Golkar et al., 2023) which encodes real numbers using a single [NUM] token multiplied by their numerical value. For decoding, a number/regression head predicts the value while the token head outputs the sequence, replacing [NUM] during inference. However, this scheme is incompatible with T5 (see Appendix A.5) and problematic for beam search decoding. We thus use the xVal encoder and masked language modeling in our experiments.

Third, **Gaussian Cross Entropy** (or Gaussian label smooth-

ing; Wang et al. (2025)) which addresses the limitations of standard CE for continuous-valued tokens. Instead of one-hot encoded labels, a Gaussian distribution is centered on the true numeric label. In this scheme, nearby numeric tokens receive partial probability mass, reflecting the continuous nature of numerical values. Given a numeric label  $y_i$ , its one-hot vector  $\mathbf{y}_i$  is replaced with a Gaussian distribution:

$$\tilde{\mathbf{y}}_i^j = \frac{1}{\sqrt{2\pi}\sigma^2} \exp\left(-\frac{(\mathcal{V}_i^j - y_i)^2}{2\sigma^2}\right), \quad \forall j \in [s..t] \quad (7)$$

where  $\sigma$  governs the smoothing width. The objective remains the cross-entropy, but now between the modified labels  $\tilde{\mathbf{y}}_i$  and the predicted distribution  $\hat{\mathbf{y}}_i$ :

$$\mathcal{L}_{\text{GCE}} = -\frac{1}{N} \sum_{i=1}^N \sum_{j=s}^t \tilde{\mathbf{y}}_i^j \log \hat{\mathbf{y}}_i^j, \quad (8)$$

The obvious disadvantage is that the smoothing is entirely fabricated: there is no real uncertainty in the label. While this technically provides incorrect labels, it ensures that the inherent similarity among nearby numeric tokens is captured and continuity in numeric space is preserved. Wang et al. (2025) finds that this does not alter classification performance on strictly categorical tokens.

### 4. Experimental Results

Across all experiments, we always optimize NTL jointly with CE (cf. Equation 6; default  $\lambda = 0.3$ ). For brevity we refer to this as just NTL-MSE or NTL-WAS.

#### 4.1. NTL Improves Performance in Arithmetics

To test the mathematical capabilities of our models, we use more than 25 million samples from the mathematical Q&A dataset from DeepMind (Saxton et al., 2019). The dataset comes with two sets of tests: interpolation tests, one for each type of question occurring in the training set, and extrapolation tests, which measure generalization along various axes of difficulty beyond that seen during training. We provide more information about the dataset in Appendix A.6.1.

##### 4.1.1. MULTITASK MATHEMATICS DATASET

We evaluated all models on the two test sets of this dataset and report the accuracy (the proportion of exactly predicted numbers), as well as the Mean Absolute Error (MAE) and the R<sup>2</sup>-score. Since the dataset is skewed with some very high values, we perform a  $\log_{10}$  transformation on the predicted and ground truth numbers before calculating MAE and R<sup>2</sup>-score. All models were trained for  $\sim 1\text{M}$  steps with a batch size of 32. More details on the models’ training hyperparameters can be found in Appendix A.6.

The results in Table 1 and Figure A1 show that vanilla T5 clearly benefits from both NTL variants. In fact, the accu-



Table 1. Evaluation on Mathematics Dataset

Table 1. (a) Interpolated Test Data

Model	Loss	Accuracy	MAE	R <sup>2</sup>
T5	CE	0.64	0.13	0.97
T5	NTL-MSE	0.72	0.11	0.97
T5	NTL-WAS	<b>0.75</b>	<b>0.10</b>	<b>0.98</b>
T5	NTL-MAE	0.74	<b>0.10</b>	0.97
RT	CE	0.71	0.11	0.97
xVal	MSE	0.10	0.26	0.97

Table 1. (b) Extrapolated Test Data

Model	Loss	Accuracy	MAE	R <sup>2</sup>
T5	CE	0.367	0.785	<b>0.913</b>
T5	NTL-MSE	0.428	0.779	0.909
T5	NTL-WAS	<b>0.432</b>	<b>0.744</b>	<b>0.913</b>
T5	NTL-MAE	0.427	0.792	0.906
RT	CE	0.404	0.987	0.738
xVal	MSE	0.058	0.826	0.819

racy increases by more than 10% for NTL-WAS compared to vanilla T5 in the interpolation tasks. The NTL-WAS was found to have the best performance across all three metrics and both interpolation and extrapolation tasks. Moreover, NTL consistently outperforms competing methods such as the RT and xVal. This confirms our hypothesis that number representation in LMs can be effectively improved through a minor, architecture-agnostic loss modification.

Since xVal is equipped with a regression head, numbers are never predicted exactly, so the accuracy value for xVal refers to the predictions rounded to two decimal places. The poor performance of xVal can likely be explained by the extensive range of numbers in the used dataset. The effective number range of xVal is limited to  $[-5, 5]$  due to the combination of its scaling of the number token embeddings and the pre-layer-norm in the backbone (Xiong et al., 2020; Golkar et al., 2023). To take this into account, we scale our dataset for xVal with  $\log(1 + x)$ . However, this means that large numbers can no longer be adequately distinguished by the model, as their embeddings become very similar. For a direct comparison without any modifications to the xVal processing, we repeated the 3-digit multiplication experiment from Golkar et al. (2023). Again, our method beats xVal. Details can be found in Appendix A.5.

#### 4.1.2. ABLATION STUDIES

In order to investigate the impact of variations in loss functions, we conducted extensive experiments testing nine different configurations: standard cross entropy loss, NTL-MSE and NTL-WAS (both with  $\lambda \in \{0.3, 0.8, 2\}$ ), NTL-

MAE and NTL-Huber. Training of these models was done on a subtask of the original dataset (100k samples), consisting of basic integer arithmetic Q&A pairs, where the input is a short natural language question containing an arithmetic expression, and the output is a single integer. A comparison of their performances on the interpolation and extrapolation test sets is shown in Table 2.

Table 2. Evaluation of NTL Loss Variants

Table 2. (a) Interpolated Test Data

Loss	$\lambda$	Accuracy	MAE	R <sup>2</sup>
CE		0.34 $\pm$ 0.01	2.15 $\pm$ 0.08	0.95 $\pm$ 0.01
NTL-MSE	0.15	0.44 $\pm$ 0.02	0.92 $\pm$ 0.05	0.99 $\pm$ 0.00
	0.3	0.41 $\pm$ 0.01	1.02 $\pm$ 0.06	0.99 $\pm$ 0.00
	0.8	0.37 $\pm$ 0.02	1.29 $\pm$ 0.11	0.99 $\pm$ 0.00
	2.0	0.33 $\pm$ 0.01	1.67 $\pm$ 0.20	0.97 $\pm$ 0.01
NTL-WAS	0.15	0.44 $\pm$ 0.02	0.93 $\pm$ 0.01	0.99 $\pm$ 0.00
	0.3	0.43 $\pm$ 0.05	0.91 $\pm$ 0.06	0.99 $\pm$ 0.00
	0.8	0.43 $\pm$ 0.04	0.94 $\pm$ 0.07	0.99 $\pm$ 0.00
	2.0	0.41 $\pm$ 0.06	1.01 $\pm$ 0.08	0.99 $\pm$ 0.00
NTL-Huber	0.3	0.44 $\pm$ 0.02	<b>0.89</b> $\pm$ 0.03	<b>1.00</b> $\pm$ 0.00
NTL-MAE	0.3	<b>0.45</b> $\pm$ 0.02	<b>0.89</b> $\pm$ 0.07	0.99 $\pm$ 0.00

Table 2. (b) Extrapolated Test Data

Loss	$\lambda$	Accuracy	MAE	R <sup>2</sup>
CE		0.05 $\pm$ 0.00	61.91 $\pm$ 1.31	0.61 $\pm$ 0.01
NTL-MSE	0.15	0.09 $\pm$ 0.01	<b>57.13</b> $\pm$ 1.51	<b>0.68</b> $\pm$ 0.01
	0.3	0.09 $\pm$ 0.01	60.99 $\pm$ 1.35	0.65 $\pm$ 0.01
	0.8	0.08 $\pm$ 0.01	58.35 $\pm$ 1.06	<b>0.68</b> $\pm$ 0.01
	2.0	0.07 $\pm$ 0.01	59.48 $\pm$ 2.72	0.66 $\pm$ 0.02
NTL-WAS	0.15	0.09 $\pm$ 0.00	57.31 $\pm$ 0.57	<b>0.68</b> $\pm$ 0.00
	0.3	<b>0.10</b> $\pm$ 0.01	58.18 $\pm$ 1.89	<b>0.68</b> $\pm$ 0.02
	0.8	<b>0.10</b> $\pm$ 0.02	59.46 $\pm$ 1.38	0.66 $\pm$ 0.01
	2.0	<b>0.10</b> $\pm$ 0.01	62.29 $\pm$ 1.04	0.64 $\pm$ 0.01
NTL-Huber	0.3	<b>0.10</b> $\pm$ 0.01	58.81 $\pm$ 1.67	0.67 $\pm$ 0.01
NTL-MAE	0.3	<b>0.10</b> $\pm$ 0.01	57.99 $\pm$ 1.48	0.67 $\pm$ 0.01

First, it is evident that all flavors of NTL are consistently better than a standard CE loss. Mean and standard deviation were calculated over 4 different training runs. Details and additional metrics can be found in Appendix A.6.2. The ablation studies in Table 2 support our finding that both variants of the NTL improve arithmetic performance compared to using the CE loss alone. Additionally, NTL-WAS mostly outperforms NTL-MSE across both interpolation and extrapolation test sets. Among the tested weights for the NTL,  $\lambda = 0.3$  yields the best performance for both loss

variants. Interestingly, we find that NTL-Huber achieves competitive and NTL-MAE sometimes even superior results to NTL-WAS. Across all experiments, the  $R^2$  achieves high values, seemingly in contrast to the accuracy. A closer investigation of the error distribution revealed that this can be attributed to many predictions being numerically very close to the ground truth, despite not being strictly accurate. These smaller errors have a very limited impact on  $R^2$ , while still affecting accuracy.

#### 4.1.3. GAUSSIAN CROSS ENTROPY

The Gaussian Cross Entropy (GCE) proposed by Wang et al. (2025) constitutes an alternative, yet complimentary means to convey proximity between number tokens. We argue that it is less principled because it essentially blurs the labels and assigns mass to technically incorrect tokens. Yet, we assessed its performance by replacing the vanilla CE objective with a GCE objective. We found  $\sigma = 0.5$  to yield the best results (Table A6).

The experiments on both the interpolation and extrapolation sets suggest that while both NTL and GCE improve model performance, NTL is generally preferable, as it leads to larger improvements. Furthermore, we investigated the

Table 3. **Gaussian Cross Entropy (GCE)**. Standalone and combinatory effect of GCE and NTL. Means and standard deviations across four runs.

GCE	NTL	Accuracy	MAE	R2
Interpolation				
✗	✗	0.34	2.15	0.95
✗	✓	0.43	0.91	<b>0.99</b>
✓	✗	0.42	0.95	<b>0.99</b>
✓	✓	<b>0.48</b>	<b>0.76</b>	<b>0.99</b>
Extrapolation				
✗	✗	0.05	61.92	0.61
✗	✓	<b>0.10</b>	<b>58.18</b>	<b>0.68</b>
✓	✗	<b>0.10</b>	58.55	0.65
✓	✓	<b>0.10</b>	66.67	0.59

performance of using NTL and GCE in combination. Since the labels are no longer one-hot encodings in this setting, we employed the NTL-WAS-CDF formulation (Equation 5) which allows for training with arbitrary target distributions. On the interpolation set, the combination of NTL and GCE yields the best results. These experiments suggest that NTL and GCE can be mutually beneficial for certain tasks.

## 4.2. Training with NTL Is Efficient

To assess potential computational overhead, we performed a comprehensive benchmarking across four distinct loss configurations: CE loss serving as the baseline, CE with NTL-MSE, CE with NTL-Wasserstein implemented according to Equation 5 (NTL-WAS-CDF), and CE with its optimized variant (NTL-WAS, Equation 4). We evaluated these loss functions under two different scenarios to quantify both the standalone computational overhead of the loss calculation and its impact within a complete training step, including the forward pass, backpropagation and optimizer step. We standardized the number of tokens in the input for both forward passes and training steps, along with maintaining a consistent proportion of number tokens across all experiments, although their positions were randomized. Each loss function was evaluated through 100 iterations on a GPU. The results in Figure 3 demonstrate that NTL-MSE and NTL-WAS introduce only marginal computational overhead across both benchmark scenarios, as each benchmark includes the runtime of the CE loss combined with the runtime of the respective NTL variant.

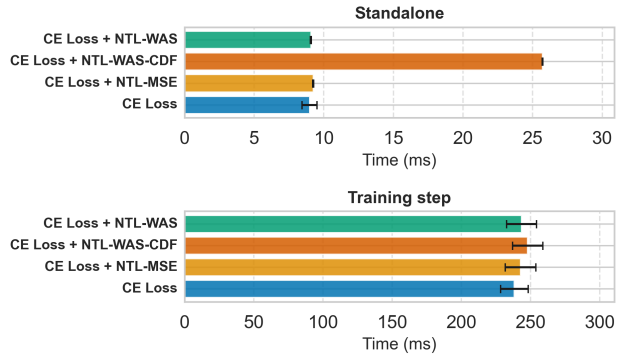


Figure 3. **Runtime Comparison**. Benchmarking of the four loss functions in two configurations: standalone execution (top) and a complete training step (bottom), with a mean number token proportion of 80 percent, highlighting computational overheads.

Notably, computing NTL-WAS alone is around 125x faster than computing CE alone. This is achieved because only a tiny fraction of tokens in the vocab are number tokens. When assessing only the loss computation time, augmenting the CE loss with NTL introduces a computational overhead ranging from a 1% (NTL-WAS) over 2.9% (NTL-MSE) and up to a substantial 286% for the less efficient NTL-WAS-CDF. However, during the full training step, the relative overhead becomes negligible as the overall computational effort consisting of the forward pass, backpropagation and optimizer step becomes dominant. For instance, the relative overhead of NTL-WAS-CDF compared to the CE decreases to 4.4%.

### 4.3. NTL Can Match Regression Models

Ideally, LMs should be able to solve even tasks that are focused solely on predicting numerical values, such as estimating a property of a molecule. Such tasks are often approached with regression heads on LMs which allow to use appropriate loss functions such as the MSE. To test the ability of NTL to solve a regression task solely token-based, we train on the rJokes dataset (Weller & Seppi, 2020) that poses the challenge to predict the level of humor in a joke. A train-test split is provided with the dataset.

In Table 4, we show the results of the T5 model with CE, NTL-WAS and combined with a regression head, as well as the results from the rJokes benchmark (Weller & Seppi, 2020). We evaluated all flavors of the T5 model over three independent runs to ensure robust metrics. NTL significantly improves over standard CE and performs on-par with the regression-head model. This is remarkable considering that it operates on a token-level and can still be utilized for non-numeric tasks, unlike models that rely on a dedicated regression head. Even though BERT, RoBERTA, and XLNet with a regression head outperform T5 trained with NTL-WAS, the performance gap is not significant considering the difference in the number of parameters. We attribute this to the fact that these models have nearly twice as many parameters as our T5-small model. Details and additional metrics can be found in Appendix A.6.3.

Table 4. **NTL Performance on a Regression Task.** NTL performs on-par with a regression-head model. Mean and standard deviations across five runs. BERT, RoBERTA and XLNet values are taken from Weller & Seppi (2020).

Model	Loss	Reg. Head	RMSE	Pearson
T5	CE	✗	2.01 $\pm$ 0.01	0.41 $\pm$ 0.00
T5	NTL-WAS	✗	1.81 $\pm$ 0.01	0.44 $\pm$ 0.00
T5	MSE	✓	1.82 $\pm$ 0.01	0.45 $\pm$ 0.00
BERT	MSE	✓	1.62	0.47
RoBERTA	MSE	✓	1.61	0.47
XLNet	MSE	✓	1.74	0.46

### 4.4. NTL Is Model-Agnostic

As a loss function, NTL can be applied to train arbitrary models, including LMs beyond Transformers like Mamba. The experiments described so far were conducted with the T5 model, which is based on an encoder-decoder Transformer architecture. In this section, we demonstrate the effectiveness of training the decoder-only models GPT-2 (Radford et al., 2019) and IBM Granite (IBM Research, 2024) with NTL. We test model sizes from 125M to 2B.

To this end, we constructed an arithmetic multiplication task, much akin to the length generalization task (Jelassi

et al., 2023). This task consists of multiplying two numbers with  $k$  and  $l$  digits, with  $k, l \in [1, \dots, 5]$  in training and  $k, l \in [1, \dots, 6]$  in testing. We report the mean absolute percentage error (MAPE) separately for unseen interpolation samples, which involve multiplying numbers with up to  $5 \times 5$  digits, and extrapolation samples, which involve multiplications with at least one six-digit factor. The results presented in Table 5 demonstrate the effectiveness of training decoder-only models with NTL, as it consistently improves performance on the multiplication task in terms of MAPE.

Model	Size	Interpolation		Extrapolation	
		CE	NTL	CE	NTL
GPT2 Small	125M	0.55	<b>0.49</b>	1.11	<b>1.00</b>
GPT2 Medium	350M	0.43	<b>0.42</b>	0.82	0.82
GPT2 Large	774M	0.39	<b>0.37</b>	0.76	<b>0.75</b>
GPT2 XL	1.5B	0.43	<b>0.40</b>	0.83	<b>0.82</b>
Granite 3.2	2B	0.35	<b>0.21</b>	0.60	<b>0.42</b>
Granite 3.1 MOE	1B	0.28	<b>0.15</b>	0.68	<b>0.23</b>

Table 5. **NTL Performance for Decoder-only Models.** MAPE for training GPT-2 and IBM Granite variants with CE and NTL on the multiplication task. All values are error percentages.

The positive impact of NTL is evident across all model scales. Notably, the improvements from NTL are more pronounced in the extrapolation setting than in interpolation, indicating that training with NTL enhances a model’s ability to generalize beyond the training distribution. This is particularly relevant due to the poor extrapolation capabilities in mathematical tasks (Razeghi et al., 2022). Some models, such as Granite 3.1, benefit particularly from training with NTL, showing the largest performance gains among the evaluated models (full evaluation in Appendix Figure A2).

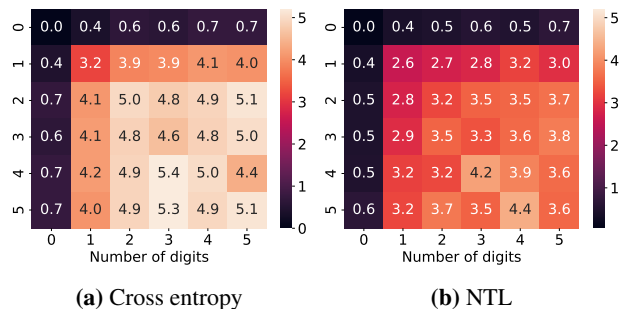


Figure 4. **Sample Efficiency.** Number of epochs needed to reach a MAPE below 0.5 on the multiplication task (GPT2 Small). NTL particularly improves sample efficiency for samples with longer factors.

Furthermore, we analyzed the sample efficiency of training with NTL on the multiplication task compared to using CE only. As revealed by Figure 4, GPT2 Small trained with

NTL requires significantly fewer epochs to achieve a MAPE below 0.5. This effect is more pronounced for challenging samples consisting of larger factors.

Similarly, we observe that the average number output distributions are more sharply centered on the correct token when using NTL (see Appendix Figure A4).

#### 4.5. NTL Is Effective for Different Tokenizations

Our standard implementation of NTL-WAS for the T5 model relies on a custom single-digit tokenizer derived from the standard T5 tokenizer, which is based on `SentencePiece` and includes some tokens consisting of multiple-digits. We conducted additional ablation studies to disentangle the effect of the custom tokenizer and the loss function. To this end, we performed further experiments on the rJokes and the mathematical Q&A datasets, in which we also evaluated the effect of using the single-digit tokenizer without the NTL-WAS and a more general version of NTL-WAS that supports multi-digit tokens. The results on the rJokes dataset in Table 6 confirm the effectiveness the NTL-WAS loss across both tokenization schemes, with the combination of single-digit tokenization and NTL-WAS performing best.

Table 6. **Tokenizer Comparison on rJokes Dataset.** T5-small used across all experiments. Mean and standard deviation across four runs.

Loss	Custom Tokenizer	RMSE	Pearson
CE	✗	2.01 $\pm$ 0.01	0.41 $\pm$ 0.00
NTL-WAS	✗	1.97 $\pm$ 0.01	0.41 $\pm$ 0.00
CE	✓	2.02 $\pm$ 0.01	0.41 $\pm$ 0.00
NTL-WAS	✓	<b>1.81<math>\pm</math>0.01</b>	0.44 $\pm$ 0.00
MSE	✗	1.82 $\pm$ 0.01	<b>0.45<math>\pm</math>0.00</b>

A similar trend is observed in the extrapolation test set of the mathematical Q&A task in Table 7, where the NTL-WAS improves accuracy for both tokenization strategies. In the interpolation test set in Table A5, the NTL-WAS

Table 7. **Extrapolation Performance for Different Models and Tokenizers**

Loss	Custom Tokenizer	Accuracy (Extrapolate)	Pearson (Extrapolate)
CE	✗	0.05 $\pm$ 0.00	0.81 $\pm$ 0.01
NTL-WAS	✗	0.06 $\pm$ 0.00	0.76 $\pm$ 0.01
CE	✓	0.09 $\pm$ 0.01	0.87 $\pm$ 0.01
NTL-WAS	✓	<b>0.10<math>\pm</math>0.01</b>	<b>0.88<math>\pm</math>0.01</b>

enhances performance for the standard tokenizer, while the

single-digit tokenizer performs slightly better without NTL-WAS. Since the extrapolation task and the rJokes task more closely resemble real-world scenarios, we argue that these experiments further reinforce the overall benefits of the NTL, particularly when combined with single-digit tokenization.

Furthermore, we observed a practical issue with models that tokenize numbers into multiple digits: some tokens have large numerical values, which disproportionately affect the loss, especially when the number tokens are irregularly spaced. For digit-level tokenization and NTL with euclidean distance (as used thus far) the loss for the farthest incorrect number token cannot exceed 9x of the loss for the nearest incorrect number token<sup>2</sup>. But a single, large number token in the vocabulary, say 1001, can distort this ratio to 1000x and may even destabilize training. NTL can intrinsically mitigate such edge cases because it is not limited to using Euclidean distances between numbers (cf. Equation 2). This can be accomplished through an optional rescaling factor that squashes the distances between tokens. Note that, with digit-level tokenization, the implicit squashing factor is 9 while setting it to 1 effectively recovers standard cross-entropy. Thus squashing allows to smoothly interpolate between vanilla NTL and CE. On the integer multiplication dataset (Jelassi et al., 2023), we found that GPT2-Large with a squashing factor of 3 improves over CE but performs worse than NTL as measured by MAPE (Table A7). Overall, we recommend to enforce digit-level tokenization as this ensures NTL is well-behaved.

#### 4.6. NTL Does Not Hamper Text Learning

To ensure that training with NTL does not introduce drawbacks for normal text generation tasks, we conduct experiments on the MultiRC dataset (Khashabi et al., 2018), a question-answering dataset testing reading comprehension. We reformat the dataset to train the model to generate the answer based on the question, rather than evaluating it as a multiple-choice task. To this end, we preprocess the dataset by extracting passages along with their corresponding questions and concatenating them to construct the input. The answer field consists of all correct answers concatenated with a delimiter. The NTL was only applied to the number tokens in the answers, which account for only 1.5% of all answer tokens – a proportion we believe to be roughly representative of real-world text datasets. In Table 8 the mean and standard deviation over three runs evaluated on the validation set are shown. Token accuracy as well as BLEU and ROUGE-1 score remain unaffected by the addition of the NTL and remain on the same level as when only using CE. These results demonstrate that training with NTL does not hamper performance on text comprehension in a task where

<sup>2</sup>For a ground truth of 0, the nearest/farthest incorrect tokens are 1 and 9.



Table 8. NTL on a Standard Language Modeling Task. NTL does not impact performance on a pure textual task.

Loss	Token Acc.	BLEU	ROUGE-1
CE	0.36 $\pm$ 0.00	0.13 $\pm$ 0.00	0.32 $\pm$ 0.00
NTL-WAS	0.36 $\pm$ 0.00	0.13 $\pm$ 0.00	0.32 $\pm$ 0.00

its benefits for numerical data are limited or absent. These findings underline that NTL can be integrated as a standard enhancement in LM training, without compromising general text understanding and generation.

#### 4.7. NTL Scales Well to LLM-Size

To demonstrate the potential to integrate NTL into LLMs, we train the 3B parameter version of the T5 model on the GSM8k dataset (Cobbe et al., 2021), a benchmark for mathematical reasoning based on school math problems. This dataset poses a significant challenge for standard LLMs; for instance, 6B-parameter models initially achieved only 22% accuracy after fine-tuning (Cobbe et al., 2021). In our experiments, the T5-3B model trained on GSM8K with CE loss attains a top-1-accuracy of 13.5% on GSM8K (see Table 9). Training with the NTL loss improves accuracy to 17.7%, matching the performance of LLMs that are more suitable for mathematical reasoning, such as Gemma-2B (Team et al., 2024), and surpassing larger models such as LLaMA-2 7B (14.6%). A qualitative example of a GSM8K task and predicted solutions is provided in Appendix Example 1.

Table 9. Results with T5-3B on GSM8k

Loss	Accuracy	Pearson
CE	13.5%	0.67
NTL-WAS	<b>17.7%</b>	<b>0.72</b>

Notably, specialized training approaches can significantly boost performance. For example, tailored fine-tuning strategies have enabled even a 1.3B model to reach 81.5% accuracy (Liu et al., 2023). Future work should explore how NTL can further enhance such specialized models.

## 5. Conclusion

We introduced the Number Token Loss (NTL) for Language Models as a novel approach to enhance their intrinsic ability to handle numerical data by considering the numerical proximity between tokens. By augmenting the standard cross entropy loss with NTL, we provide a simple yet effective method that integrates seamlessly into existing architectures without requiring additional computational overhead. Our

experiments unambiguously demonstrate the effectiveness of the NTL for encoder-decoder and decoder-only architectures. Experiments on several datasets related to mathematics, arithmetics and reasoning showed significant improvements in numerical reasoning, especially in models without specialized numerical embeddings. Furthermore, our results on regression datasets revealed that training with NTL allows a LM with a token-head to compete with models designed specifically for regression tasks. We also verified that NTL does not compromise capabilities on standard text-only tasks like reading comprehension. Finally, the scalability of NTL was demonstrated by successfully applying it to a 3B-parameter model on the challenging GSM8k dataset, opening the door for integration into LLMs.

From the two proposed NTL flavors (NTL-WAS and NTL- $\mathcal{L}_p$ ) the NTL-WAS is a preferable loss since it avoids the undesired local minima of NTL- $\mathcal{L}_p$  caused by errors canceling out. However, unlike in NTL-WAS, the dot product inside NTL- $\mathcal{L}_p$  yields a quantity in the original number space which can be useful to compute more complex losses incorporating multiple tokens. For example, a current limitation of NTL is to treat each digit-level token as equally important, thereby ignoring the higher relevance of digits with higher decimal places. This limitation could be addressed by computing the loss at the number level rather than at the digit level, e.g., by naive scaling by decimal place or by leveraging the dot-product idea of NTL- $\mathcal{L}_p$  to calculate a number-level prediction. Furthermore, future research should focus on exploring synergies with additional arithmetic-specific training adaptations. Particularly exciting will be the fine-tuning of mid- and large-scale open-source models, such as LLaMA or Gemma, to further assess its scalability and generalization potential. Overall, this approach offers a practical solution for enhancing language models in numerically rich domains, contributing towards more accurate and reliable applications of LMs in mathematics and science.

## Further Resources

### Code

1. **Usage:** NTL ships via PyPI. For minimalistic integration in your work use [github.com/AI4SD/number-token-loss](https://github.com/AI4SD/number-token-loss)
2. **Reproduction:** Full development code for reproduction of all experiments: [github.com/TUM-AI/number-token-loss](https://github.com/TUM-AI/number-token-loss)

### Learn more

1. **Landing page:** [Project overview site](#)
2. **Demo:** [Streamlit app on HuggingFace spaces](#)
3. **Video:** [5min paper walkthrough on YouTube](#)

## Impact Statement

The introduction of the NTL enhances the ability of LMs to handle numerical data and can improve performance in tasks involving quantitative reasoning and arithmetic. This advancement has potential implications across fields like scientific research, finance, and engineering, where precise numerical outputs are essential for data-driven decision-making and automated analysis. However, this capability also raises potential risks, including misuse of generated numerical data or biased predictions in sensitive areas, such as healthcare or economics. As with any AI methods, careful implementation and human oversight are necessary to ensure model transparency and accuracy.

## References

- Bachmann, G. and Nagarajan, V. The pitfalls of next-token prediction, 2024. URL <https://arxiv.org/abs/2403.06963>.
- Boiko, D. A., MacKnight, R., Kline, B., and Gomes, G. Autonomous chemical research with large language models. *Nature*, 624(7992):570–578, 2023.
- Born, J. and Manica, M. Regression transformer enables concurrent sequence regression and generation for molecular language modelling. *Nature Machine Intelligence*, 5(4):432–444, 2023.
- Charton, F. Learning the greatest common divisor: explaining transformer predictions. In *The 3rd Workshop on Mathematical Reasoning and AI at NeurIPS’23*, 2023.
- Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., Schuh, P., Shi, K., Tsvyashchenko, S., Maynez, J., Rao, A., Barnes, P., Tay, Y., Shazeer, N., Prabhakaran, V., Reif, E., Du, N., Hutchinson, B., Pope, R., Bradbury, J., Austin, J., Isard, M., Gur-Ari, G., Yin, P., Duke, T., Levskaya, A., Ghemawat, S., Dev, S., Michalewski, H., Garcia, X., Misra, V., Robinson, K., Fedus, L., Zhou, D., Ippolito, D., Luan, D., Lim, H., Zoph, B., Spiridonov, A., Sepassi, R., Dohan, D., Agrawal, S., Omernick, M., Dai, A. M., Pillai, T. S., Pellat, M., Lewkowycz, A., Moreira, E., Child, R., Polozov, O., Lee, K., Zhou, Z., Wang, X., Saeta, B., Diaz, M., Firat, O., Catasta, M., Wei, J., Meier-Hellstern, K., Eck, D., Dean, J., Petrov, S., and Fiedel, N. Palm: Scaling language modeling with pathways, 2022. URL <https://arxiv.org/abs/2204.02311>.
- Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.
- Cuturi, M. Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in neural information processing systems*, 26, 2013.
- Dai, A. M. and Le, Q. V. Semi-supervised sequence learning. *Advances in neural information processing systems*, 28, 2015.
- DeepSeek-AI, Liu, A., Feng, B., Wang, B., Wang, B., Liu, B., Zhao, C., Deng, C., Ruan, C., Dai, D., Guo, D., Yang, D., Chen, D., Ji, D., Li, E., Lin, F., Luo, F., Hao, G., Chen, G., Li, G., Zhang, H., Xu, H., Yang, H., Zhang, H., Ding, H., Xin, H., Gao, H., and et al. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model, 2024.

- Dziri, N., Lu, X., Sclar, M., Li, X. L., Jiang, L., Lin, B. Y., Welleck, S., West, P., Bhagavatula, C., Le Bras, R., et al. Faith and fate: Limits of transformers on compositionality. *Advances in Neural Information Processing Systems*, 36, 2024.
- Gao, L., Madaan, A., Zhou, S., Alon, U., Liu, P., Yang, Y., Callan, J., and Neubig, G. Pal: Program-aided language models. In *International Conference on Machine Learning*, pp. 10764–10799. PMLR, 2023.
- Geva, M., Gupta, A., and Berant, J. Injecting numerical reasoning skills into language models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2020.
- Golkar, S., Pettee, M., Eickenberg, M., Bietti, A., Cranmer, M., Krawezik, G., Lanusse, F., McCabe, M., Ohana, R., Parker, L., et al. xval: A continuous number encoding for large language models. *arXiv preprint arXiv:2310.02989*, 2023.
- Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Vaughan, A., Yang, A., Fan, A., and et al. The Llama 3 herd of models, 2024.
- Gromov, A. Grokking modular arithmetic. *arXiv preprint arXiv:2301.02679*, 2023.
- IBM Research. Granite 3.0 language models, 2024.
- Jelassi, S., d’Ascoli, S., Domingo-Enrich, C., Wu, Y., Li, Y., and Charton, F. Length generalization in arithmetic transformers, 2023. URL <https://arxiv.org/abs/2306.15400>.
- Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnoy, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Židek, A., Potapenko, A., et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.
- Kenton, J. D. M.-W. C. and Toutanova, L. K. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of naacL-HLT*, volume 1, pp. 2, 2019.
- Kera, H., Ishihara, Y., Kambe, Y., Vaccon, T., and Yokoyama, K. Learning to compute gröbner bases. *Advances in Neural Information Processing Systems*, 37: 33141–33187, 2024.
- Khashabi, D., Chaturvedi, S., Roth, M., Upadhyay, S., and Roth, D. Looking beyond the surface: A challenge set for reading comprehension over multiple sentences. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 252–262, 2018.
- Lee, N., Sreenivasan, K., Lee, J. D., Lee, K., and Papailiopoulos, D. Teaching arithmetic to small transformers, 2023. URL <https://arxiv.org/abs/2307.03381>.
- Lee, S. and Kim, G. Recursion of thought: A divide-and-conquer approach to multi-context reasoning with language models. *arXiv preprint arXiv:2306.06891*, 2023.
- Li, Y., Lin, Z., Zhang, S., Fu, Q., Chen, B., Lou, J.-G., and Chen, W. Making language models better reasoners with step-aware verifier. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 5315–5333, 2023.
- Liu, B., Bubeck, S., Eldan, R., Kulkarni, J., Li, Y., Nguyen, A., Ward, R., and Zhang, Y. Tinygsm: achieving 80% on gsm8k with small language models. *arXiv preprint arXiv:2312.09241*, 2023.
- Lukasik, M., Meng, Z., Narasimhan, H., Chang, Y.-W., Menon, A. K., Yu, F., and Kumar, S. Better autoregressive regression with llms via regression-aware fine-tuning. In *The Thirteenth International Conference on Learning Representations*, 2025.
- Qu, C., Dai, S., Wei, X., Cai, H., Wang, S., Yin, D., Xu, J., and Wen, J.-R. Tool learning with large language models: A survey. *Frontiers of Computer Science*, 19(8):198343, 2025.
- Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. Language models are unsupervised multitask learners. Technical report, OpenAI, 2019. Technical report.
- Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21 (140):1–67, 2020.
- Razeghi, Y., IV, R. L. L., Gardner, M., and Singh, S. Impact of pretraining term frequencies on few-shot numerical reasoning. In Goldberg, Y., Kozareva, Z., and Zhang, Y. (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pp. 840–854. Association for Computational Linguistics, 2022. doi: 10.18653/V1/2022.FINDINGS-EMNLP.59. URL <https://doi.org/10.18653/v1/2022.findings-emnlp.59>.

- Saxena, E., Alfarano, A., Wenger, E., and Lauter, K. Teaching transformers modular arithmetic at scale, 2024. URL <https://arxiv.org/abs/2410.03569>.
- Saxton, D., Grefenstette, E., Hill, F., and Kohli, P. Analysing mathematical reasoning abilities of neural models, 2019. URL <https://arxiv.org/abs/1904.01557>.
- Shah, D. and Aamodt, T. M. Learning label encodings for deep regression. In *The Eleventh International Conference on Learning Representations*, 2023. URL [https://openreview.net/forum?id=k60XE\\_b0Ix6](https://openreview.net/forum?id=k60XE_b0Ix6).
- Singh, A. K. and Strouse, D. Tokenization counts: the impact of tokenization on arithmetic in frontier llms, 2024. URL <https://arxiv.org/abs/2402.14903>.
- Stevens, S., Wenger, E., Li, C., Nolte, N., Saxena, E., Char-ton, F., and Lauter, K. Salsa fresca: Angular embeddings and pre-training for ml attacks on learning with errors, 2024. URL <https://arxiv.org/abs/2402.01082>.
- Stewart, L., Bach, F., Berthet, Q., and Vert, J.-P. Regression as classification: Influence of task formulation on neural network features. In *International Conference on Artificial Intelligence and Statistics*, pp. 11563–11582. PMLR, 2023.
- Sundararaman, D., Si, S., Subramanian, V., Wang, G., Hazarika, D., and Carin, L. Methods for numeracy-preserving word embeddings. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 4742–4753, 2020.
- Team, G., Mesnard, T., Hardin, C., Dadashi, R., Bhupatiraju, S., Pathak, S., Sifre, L., Rivière, M., Kale, M. S., Love, J., et al. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024.
- Thawani, A., Pujara, J., Ilievski, F., and Szekely, P. Representing numbers in nlp: a survey and a vision. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 644–656, 2021.
- Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., Bikel, D., Blecher, L., Ferrer, C. C., Chen, M., Cucurull, G., Esiobu, D., Fernandes, J., Fu, J., Fu, W., Fuller, B., Gao, C., Goswami, V., Goyal, N., Hartshorn, A., Hosseini, S., Hou, R., Inan, H., Kardaş, M., Kerkez, V., Khabsa, M., Kloumann, I., Korenev, A., Koura, P. S., Lachaux, M.-A., Lavril, T., Lee, J., Liskovich, D., Lu, Y., Mao, Y., Martinet, X., Mihaylov, T., Mishra, P., Molybog, I., Nie, Y., Poulton, A., Reizenstein, J., Rungta, R., Saladi, K., Schelten, A., Silva, R., Smith, E. M., Subramanian, R., Tan, X. E., Tang, B., Taylor, R., Williams,
- A., Kuan, J. X., Xu, P., Yan, Z., Zarov, I., Zhang, Y., Fan, A., Kambadur, M., Narang, S., Rodriguez, A., Stojnic, R., Edunov, S., and Scialom, T. Llama 2: Open foundation and fine-tuned chat models, 2023. URL <https://arxiv.org/abs/2307.09288>.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., et al. Attention is all you need. *Advances in neural information processing systems*, 30(1): 261–272, 2017.
- Wallace, E., Wang, Y., Li, S., Singh, S., and Gardner, M. Do NLP models know numbers? probing numeracy in embeddings, 2019.
- Wang, J., Qin, R., Wang, M., Fang, M., Zhang, Y., Zhu, Y., Su, Q., Gou, Q., Shen, C., Zhang, O., Wu, Z., Jiang, D., Zhang, X., Zhao, H., Ge, J., Wu, Z., Kang, Y., Hsieh, C.-Y., and Hou, T. Token-mol 1.0: tokenized drug design with large language models. *Nature Communications*, 16 (1):4416, 2025. doi: 10.1038/s41467-025-59628-y.
- Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q. V., Zhou, D., et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.
- Weller, O. and Seppi, K. The rjokes dataset: a large scale humor collection. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pp. 6136–6141, 2020.
- Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., et al. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pp. 38–45, 2020.
- Xiong, R., Yang, Y., He, D., Zheng, K., Zheng, S., Xing, C., Zhang, H., Lan, Y., Wang, L., and Liu, T. On layer normalization in the transformer architecture. In *International Conference on Machine Learning*, pp. 10524–10533. PMLR, 2020.
- Zhang, X., Ramachandran, D., Tenney, I., Elazar, Y., and Roth, D. Do language embeddings capture scales? In Cohn, T., He, Y., and Liu, Y. (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 4889–4896, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.439. URL <https://aclanthology.org/2020.findings-emnlp.439>.



Zhang-Li, D., Lin, N., Yu, J., Zhang, Z., Yao, Z., Zhang, X., Hou, L., Zhang, J., and Li, J. Reverse that number! decoding order matters in arithmetic learning. *arXiv preprint arXiv:2403.05845*, 2024.

Zhong, Q., Wang, K., Xu, Z., Liu, J., Ding, L., Du, B., and Tao, D. Achieving  $> 97\%$  on GSM8K: Deeply understanding the problems makes LLMs perfect reasoners. *arXiv preprint arXiv:2404.14963*, 2024.

## A. Appendix

### A.1. Statement on code

The code for this paper is available at <https://github.com/tum-ai/number-token-loss>.

### A.2. Algorithm for the Number Token Loss

**Algorithm A1** Pseudo-code to compute NTL-MSE

---

```

1: Initialize: n_vocab  $\leftarrow \left[ \begin{cases} \text{int}(\text{vocab}[i]) & \text{if } \text{vocab}[i] \in \mathbb{R} \\ \text{NaN} & \text{otherwise} \end{cases} \right]_{i=1}^V$ 
2:
3: function FORWARD(logits  $\in \mathbb{R}^{B \times T \times V}$ , labels  $\in \mathbb{R}^{B \times T}$ ) : Float
4:   ntl  $\leftarrow 0$ 
5:   n_logits  $\leftarrow \text{logits}[:, :, \neg \text{n\_vocab.isnan}()]$  ▷ Ignore non-number tokens
6:   n_probs  $\leftarrow \text{Softmax}(\text{logits})$ 
7:    $\hat{y} \leftarrow \sum_i \text{n\_probs}[:, :, i] \cdot \text{n\_vocab}$  ▷  $\hat{y}$  is  $B \times T$ 
8:    $y \leftarrow \text{n\_vocab}[\text{labels}]$  ▷  $y$  is  $B \times T$ 
9:   ntl  $\leftarrow \text{MSE}(y, \hat{y})$  ▷ Can be any regression loss
10:  return ntl
11: end function

```

---

**Algorithm A2** Pseudo-code to compute NTL-WAS

---

```

1: Initialize: n_vocab  $\leftarrow \left[ \begin{cases} \text{int}(\text{vocab}[i]) & \text{if } \text{vocab}[i] \in \mathbb{R} \\ \text{NaN} & \text{otherwise} \end{cases} \right]_{i=1}^V$ 
2: if order_numbers is True then
3:   Sort the numbers in n_vocab by their numerical values
4: end if
5:
6: function FORWARD(logits  $\in \mathbb{R}^{B \times T \times V}$ , labels  $\in \mathbb{N}^{B \times T}$ ) : Float
7:   n_logits  $\leftarrow \text{logits}[:, :, \neg \text{n\_vocab.isnan}()]$  ▷ Ignore non-number tokens
8:   n_probs  $\leftarrow \text{Softmax}(\text{logits})$ 
9:    $y \leftarrow \text{n\_vocab}[\text{labels}]$  ▷ Retrieve true numerical values
10:  if using CDF version of NTL-WAS then
11:     $y\_distr[b, t] \leftarrow \text{one\_hot}(y[b, t], \text{num\_classes}=\text{len}(\text{n\_vocab}))$  ▷ One hot encode y
12:     $\text{wasserstein\_distance}[b, t] = \sum_{v=1}^V |\text{CDF}(\text{n\_probs}[b, t])[v] - \text{CDF}(y\_distr[b, t])[v]|$ 
13:  end if
14:  if using absolute difference version of NTL-WAS then
15:     $\text{wasserstein\_distance}[b, t] \leftarrow \frac{1}{V} \sum_v \text{n\_probs}[b, t, v] \cdot |\text{n\_vocab}[v] - y|$ 
16:  end if
17:  ntl  $\leftarrow \text{Mean}(\text{wasserstein\_distance}[\neg y.\text{isnan}()])$ 
18:  return ntl
19: end function

```

---

### A.3. T5 Architecture

The T5 model is built upon the Transformer architecture (Vaswani et al., 2017), consisting of stacked self-attention and feed-forward layers in both the encoder and decoder. The encoder processes the input tokens to create contextualized representations, while the decoder generates the output tokens autoregressively, attending to both the encoder’s outputs and the previously generated tokens. The model can be trained with both Masked Language Modelling (MLM) (Kenton & Toutanova, 2019) and Causal/Auto-Regressive Language Modelling (CLM) (Dai & Le, 2015), whereby we chose to use CLM.

#### A.4. Regression Transformer

The Regression Transformer (Born & Manica, 2023) preserves the inherent structure of numbers by inducing information on relative proximity through numerical encodings that are set deterministically for all tokens. For every combination of a decimal place and digit value, a corresponding numerical token is added to the vocabulary. For instance, the number 11.4 is tokenized to [1\_1, 1\_0, 4\_-1].

Non-numeric tokens are set to zero vectors. The numerical encodings are designed so that their pairwise distances are symmetric and monotonically decreasing with the float value. The final encoding of the input tokens is obtained by summing over numerical and regular word encodings. The Regression Transformer numerical encodings NE at dimension  $j$  for numerical token  $t_{v,p}$  with value  $v$  and decimal place  $p$  can be determined by

$$\text{NE}_{\text{Float}}(v, p, j) = (-1)^j \cdot \frac{v \cdot 10^p}{j + 1}. \quad (9)$$

#### A.5. Challenges with Integrating xVal in Transformer Models like T5

In transformer models like T5, integrating numerical encoding schemes like xVal presents challenges. xVal multiplies the [NUM] token embedding  $X$  by the number value  $a$ . In T5, however, a per-sample pre-layer normalization is applied immediately after the embedding, which effectively removes the scaling by  $a$ . Specifically:

$$\frac{aX - E[aX]}{\sigma(aX)} = \frac{aX - aE(X)}{\sqrt{a^2E(X^2) - a^2E(X)^2}} = \frac{X - E(X)}{\sigma(X)} \quad (10)$$

Hence, under T5’s architecture, all numbers collapse to the same embedding, making xVal incompatible with T5.

Even in the original xVal architecture, the range of values xVal can process meaningfully is limited by the layer normalization that follows the positional embedding step. Please see Golkar et al. (2023) under “Implicit normalization via layer-norm” for more information. Therefore, xVal normalizes each value to  $[-5, 5]$  prior to training to mitigate this issue.

We argue that this approach cannot be applied in practice, since in real texts the range of numbers is not known in advance, and thus a simple min-max normalisation to  $[-5, 5]$  prior to training or inference is not really practical.

Therefore we opted for a simpler approach in our experiments: Applying a signed  $\log(1 + x)$  transformation to all numeric inputs. This avoids the overhead of parsing and re-scaling each number to  $[-5, 5]$  prior to training, but it also has the drawback that large numbers are squashed in the embedding space, making fine-grained distinctions difficult for the model.

For a direct comparison without any modifications to the xVal processing, we repeated the 3-digit multiplication experiment from Golkar et al. (2023). The results can be seen in Table A1.

Encoding	R <sup>2</sup> Value
P10	0.9989
P1000	0.9997
B1999	0.9998
FP15	0.7119
xVal	0.9986
T5 CE	0.999934
T5 NTL-WAS	<b>0.999997</b>
T5 regression head	0.999891

Table A1. R<sup>2</sup> scores for various number encoding methods on the 3-digit multiplication experiment from Golkar et al. (2023).

#### A.6. Experiments

For all trainings, we use transformers (Wolf et al., 2020) 4.42.4. We train with a batch size of 32, a learning rate of 1e-4 and a weight decay of 0.01. All models were trained on single graphics processing units (GPUs) of type NVIDIA RTX A6000, NVIDIA A100 or NVIDIA A40.

### A.6.1. MULTITASK MATHEMATICS DATASET

To test the mathematical capabilities of the methods, we use a subset of the mathematical question-answer dataset from DeepMind (Saxton et al., 2019). The dataset was generated synthetically and therefore contains limited linguistic variability, but is sufficient for our purposes to compare the mathematical capabilities of the different methods.

The dataset contains different modules and difficulty levels. For training and testing the models, we chose all difficulty levels but excluded modules where the answer contains complex fractions or variables. This allows us to focus on purely numeric answers to simplify the evaluation of the model and still leaves us with a large enough dataset of  $\sim 26$  million samples.

For training, validation, and interpolation tests, we selected the following modules from the DeepMind mathematical question-answer dataset:

- algebra\_\_linear\_1d.txt
- algebra\_\_linear\_1d\_composed.txt
- algebra\_\_linear\_2d.txt
- algebra\_\_linear\_2d\_composed.txt
- algebra\_\_sequence\_next\_term.txt
- arithmetic\_\_add\_or\_sub.txt
- arithmetic\_\_add\_sub\_multiple.txt
- arithmetic\_\_mul.txt
- numbers\_\_div\_remainder.txt
- numbers\_\_div\_remainder\_composed.txt
- numbers\_\_place\_value.txt
- numbers\_\_round\_number.txt
- numbers\_\_round\_number\_composed.txt

For extrapolation tests, we selected the following modules:

- arithmetic\_\_add\_or\_sub\_big.txt
- arithmetic\_\_add\_sub\_multiple\_longer.txt
- arithmetic\_\_mixed\_longer.txt
- arithmetic\_\_mul\_big.txt
- arithmetic\_\_mul\_div\_multiple\_longer.txt
- numbers\_\_place\_value\_big.txt
- numbers\_\_round\_number\_big.txt

This resulted in a training dataset of 25,986,948 samples, a validation dataset of 13,026 samples, an interpolation test set of 130,000 samples, and an extrapolation test set of 70,000 samples.

We train each model for 1050000 iterations. For these experiments, we used the T5-base architecture (220M parameters). For the Number Token Loss, we trained with the hyperparameter  $\lambda$  set to 0.3. The results can be seen in Table 1 and Figure A1.

### A.6.2. ABLATION STUDIES AND NTL-WAS-CDF WITH GAUSSIAN SMOOTHING

For training and validation, we selected a subset of the DeepMind mathematical Q&A dataset, in all three difficulty levels: arithmetic\_\_add\_sub\_multiple.txt. Similarly, interpolation and extrapolation tests were done on arithmetic\_\_add\_sub\_multiple.txt and arithmetic\_\_add\_sub\_multiple\_longer.txt, respectively. This resulted in a training set of 100,000 examples, a validation set of 3,000 examples, and two test sets of 10,000 examples each.



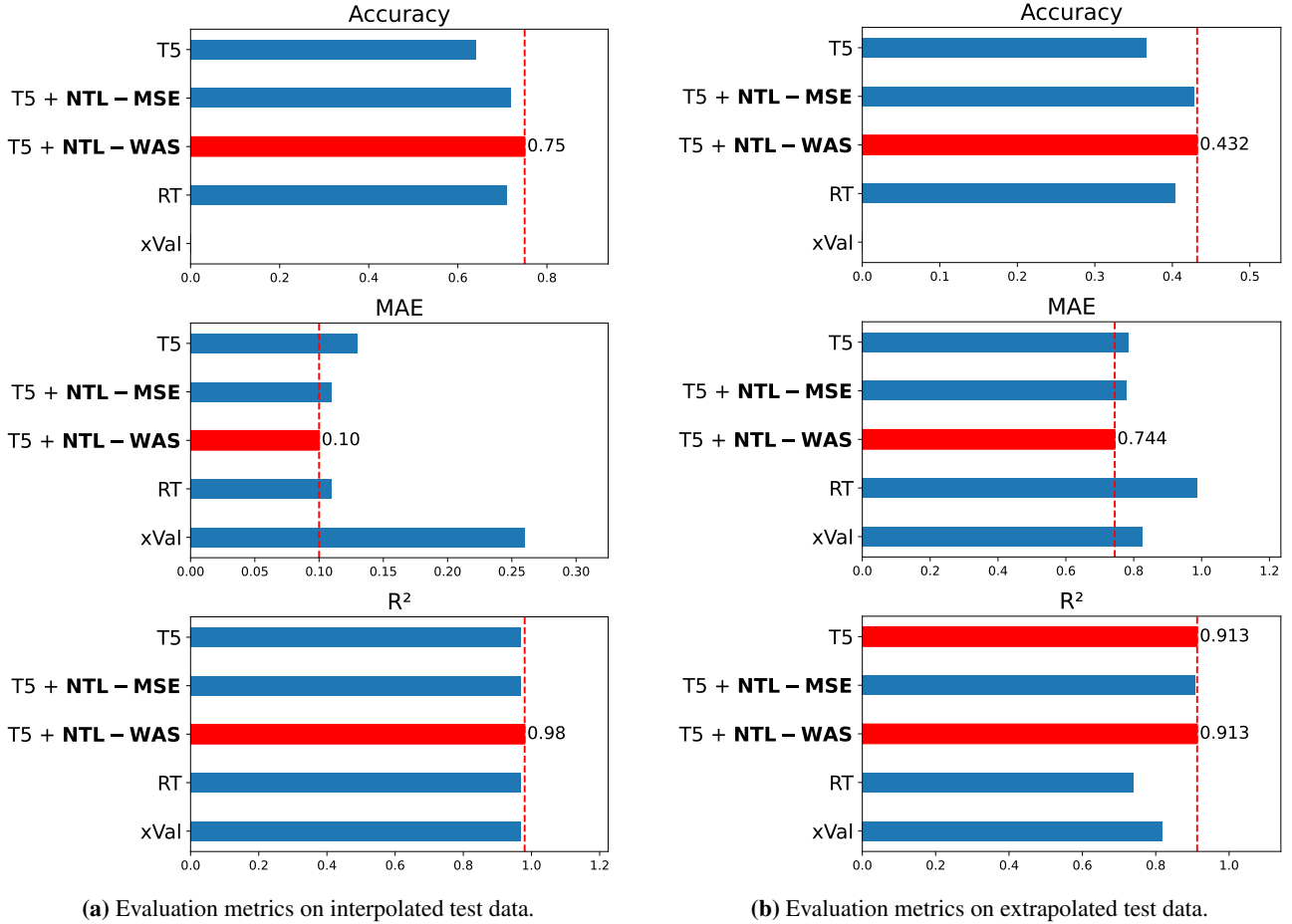


Figure A1. Comparison of evaluation metrics on interpolated and extrapolated test data.

For these experiments, we used the T5-small architecture (60M parameters). The mean and standard deviation were taken over 4 different runs for the ablation studies and 3 different runs for the studies on GCE. We trained each run for 2.5M iterations.

In addition to accuracy, mean absolute error and  $R^2$  given in Table 2 and Table 3, the achieved Pearson correlation and Spearman’s rank correlation coefficient are shown in Table A2.

#### A.6.3. NTL FOR REGRESSION AND NORMAL NLP TASK

We trained and evaluated the NTL on two tasks: a regression task using the rJokes dataset (Weller & Seppi, 2020) and a test generation task using the transformed MultiRC dataset (Khashabi et al., 2018). In both cases, each model was trained for 2.5M iterations. The mean and standard deviation were taken over 3 different runs. When using T5 with NTL-WAS, we set the hyperparameter  $\lambda$  to 2.0.

In addition to RMSE and Pearson correlation given in Table 6, mean squared error, accuracy and Spearman’s rank correlation coefficient are shown in Table A3.

#### A.6.4. NTL IS EFFECTIVE FOR DIFFERENT TOKENIZATIONS

To evaluate the effect of the tokenizer we trained T5-small again for 2.5M iterations. For the NTL-WAS loss we set the hyperparameter  $\lambda$  to 2.0. A comparison of the achieved number accuracies and Pearson correlation of the models is shown in Table A5, Table 7 and Table A3.

Table A2. Pearson and Spearman Correlation Coefficients on the Interpolation and Extrapolation Test Sets

Loss	$\lambda$	Pearson		Spearman	
		Interpolation	Extrapolation	Interpolation	Extrapolation
CE		$0.98_{\pm 0.00}$	$0.81_{\pm 0.01}$	$0.98_{\pm 0.00}$	$0.86_{\pm 0.01}$
NTL-MSE	0.3	$1.00_{\pm 0.00}$	$0.86_{\pm 0.00}$	$1.00_{\pm 0.00}$	$0.89_{\pm 0.00}$
	0.8	$0.99_{\pm 0.00}$	$0.87_{\pm 0.00}$	$1.00_{\pm 0.00}$	$0.90_{\pm 0.00}$
	2.0	$0.99_{\pm 0.00}$	$0.86_{\pm 0.01}$	$0.99_{\pm 0.00}$	$0.89_{\pm 0.01}$
NTL-WAS	0.3	$1.00_{\pm 0.00}$	$0.88_{\pm 0.01}$	$1.00_{\pm 0.00}$	$0.91_{\pm 0.01}$
	0.8	$1.00_{\pm 0.00}$	$0.86_{\pm 0.00}$	$1.00_{\pm 0.00}$	$0.89_{\pm 0.00}$
	2.0	$0.99_{\pm 0.00}$	$0.85_{\pm 0.01}$	$1.00_{\pm 0.00}$	$0.88_{\pm 0.01}$
NTL-MAE	0.3	$1.00_{\pm 0.00}$	$0.87_{\pm 0.01}$	$1.00_{\pm 0.00}$	$0.90_{\pm 0.01}$
NTL-Huber	0.3	$1.00_{\pm 0.00}$	$0.87_{\pm 0.01}$	$1.00_{\pm 0.00}$	$0.90_{\pm 0.01}$
GS + CE		$0.49_{\pm 0.38}$	$0.20_{\pm 0.22}$	$1.00_{\pm 0.00}$	$0.90_{\pm 0.01}$
GS + NTL-WAS	0.3	$1.00_{\pm 0.00}$	$0.87_{\pm 0.01}$	$1.00_{\pm 0.00}$	$0.90_{\pm 0.01}$

 Table A3. Full Evaluation on the rJokes Dataset. Evaluation metrics (mean  $\pm$  std).

Model	MSE	Number accuracy	Pearson	Spearman
Standard T5 (CE)	$4.03 \pm 0.03$	$0.30 \pm 0.00$	$0.41 \pm 0.00$	$0.35 \pm 0.00$
T5 + NTL	<b><math>3.27 \pm 0.04</math></b>	$0.30 \pm 0.00$	$0.44 \pm 0.00$	<b><math>0.40 \pm 0.00</math></b>
T5 + Regression Head (MSE)	$3.32 \pm 0.04$	$0.00 \pm 0.00$	<b><math>0.45 \pm 0.00</math></b>	<b><math>0.40 \pm 0.00</math></b>
T5 + NTL + Standard tokenizer	$3.88 \pm 0.02$	$0.30 \pm 0.00$	$0.41 \pm 0.00$	$0.35 \pm 0.00$
Standard T5 (CE) + Custom Tokenizer	$4.09 \pm 0.04$	$0.30 \pm 0.00$	$0.41 \pm 0.00$	$0.37 \pm 0.00$

#### A.6.5. ERROR ANALYSIS

we conducted a detailed error analysis on the GSM8K dataset to examine predictions for numbers ending with specific digits (0–9), comparing CE and NTL. The error histograms shown in Figure A3 reveal a consistent pattern across all digit groups: NTL error distributions are narrower and concentrated around zero, confirming improved numerical reasoning compared to CE.

We further analyzed model predictions for numbers near digit boundaries, focusing particularly on numbers ending with 0 or 9 as well as powers of 10. Table A4 reveals how often predictions are overestimations, underestimations and exact matches. The results show, that NTL achieves a more balanced error distribution. The exact match accuracy is consistently higher, but particularly so for samples ending with the 9 token, implying that NTL leads to a better handling on those digit boundaries.

Table A4. **Error Rates.** Error Rates by Digit Boundary Sample Type on GSM8K.

Sample Type	Metric	CE (%)	NTL (%)
Ends with 0	Overestimation Rate	<b>28.4</b>	29.4
	Underestimation Rate	56.5	<b>51.1</b>
	Exact Match Rate	15.0	<b>19.4</b>
Power of 10	Overestimation Rate	54.4	<b>49.1</b>
	Underestimation Rate	24.6	24.6
	Exact Match Rate	21.1	<b>26.3</b>
Ends with 9	Overestimation Rate	<b>33.3</b>	38.8
	Underestimation Rate	46.7	<b>32.6</b>
	Exact Match Rate	20.0	<b>28.6</b>

 Table A5. **Interpolation Performance for Different Models and Tokenizers**

Loss	Custom Tokenizer	Accuracy (Interpolate)	Pearson (Interpolate)
CE	✗	0.34 $\pm$ 0.01	0.98 $\pm$ 0.00
NTL-WAS	✗	0.39 $\pm$ 0.00	0.96 $\pm$ 0.00
CE	✓	<b>0.45<math>\pm</math>0.01</b>	<b>1.00<math>\pm</math>0.00</b>
NTL-WAS	✓	0.43 $\pm$ 0.05	<b>1.00<math>\pm</math>0.00</b>

 Table A6. **Gaussian Cross Entropy (GCE).** Standalone and combinatory effect of GCE and NTL.

GCE	$\sigma$	NTL	Accuracy	MAE	R2
Interpolation test set					
✗	-	✗	0.34	2.15	0.95
✗	-	✓	0.43	0.91	<b>0.99</b>
✓	1	✗	0.19	4.34	0.95
✓	1	✓	0.34	3.14	0.96
✓	0.5	✗	0.42	0.95	<b>0.99</b>
✓	0.5	✓	<b>0.48</b>	<b>0.76</b>	<b>0.99</b>
Extrapolation test set					
✗	-	✗	0.05	61.92	0.61
✗	-	✓	<b>0.10</b>	<b>58.18</b>	<b>0.68</b>
✓	1	✗	0.03	126.74	0.08
✓	1	✓	0.06	111.09	0.25
✓	0.5	✗	<b>0.10</b>	58.55	0.65
✓	0.5	✓	<b>0.10</b>	66.97	0.59

Loss	MAPE
CE	0.502%
NTL-Squash-2	0.491%
NTL	<b>0.485%</b>

 Table A7. **Squashing NTL Loss.** MAPE on the multiplication task using the GPT-2 model trained with a squashing NTL variant.

Integer multiplication: Train on up to 5x5 digits, test on up to 6x6

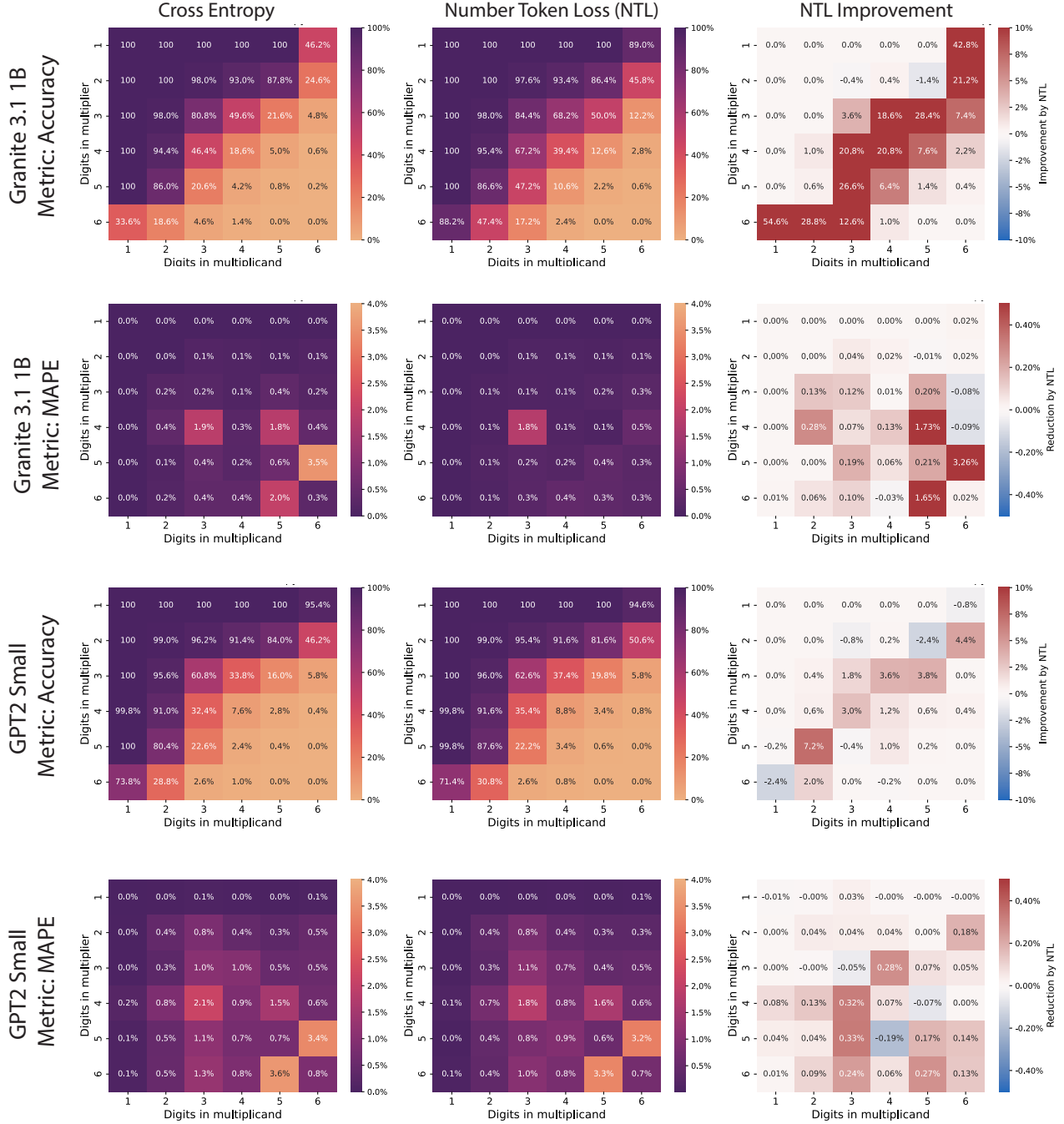


Figure A2. Performance improvement on the multiplication task using NTL, evaluated on GPT-2 and Granite 3.1.



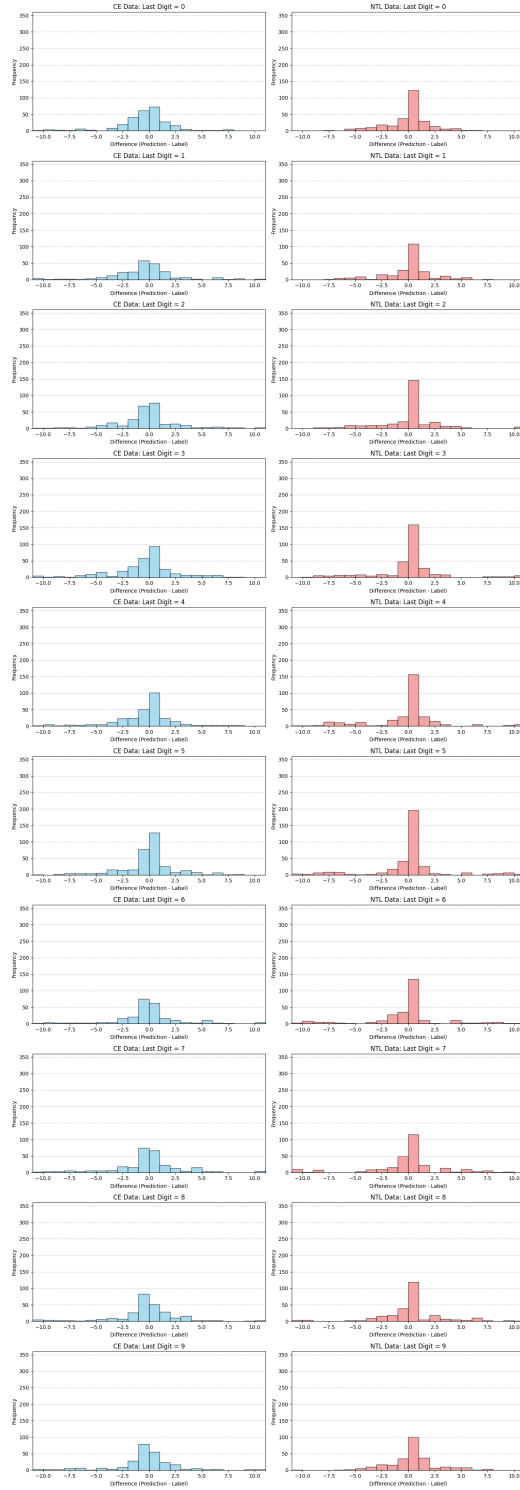


Figure A3. Error histograms for all number tokens for the GSM8k task.

### Average Number Token Distributions in Early Training

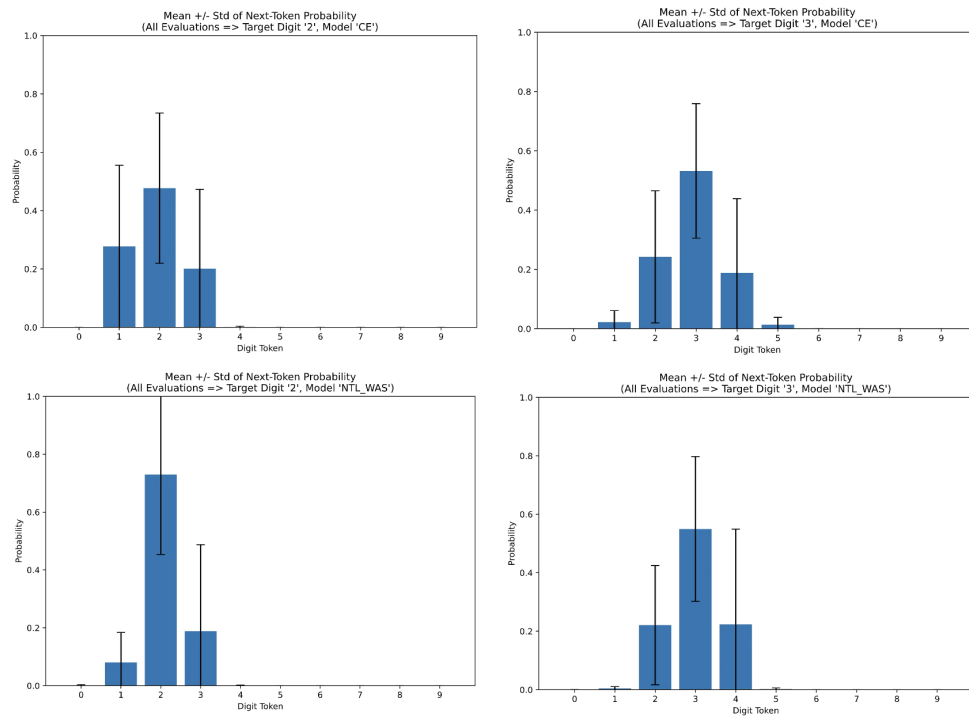


Figure A4. Average number token distributions for simple arithmetic tasks with target token 2 and 3 after training on the arithmetic task for 50k steps.

**Example 1: GSM8K sample – NTL vs. CE**

*Question:*

Roberta wants to have a dinner party centered around soufflés. Each savory souffle calls for 8 eggs each and the dessert ones call for 6 eggs each. She wants to make 3 savory soufflés and 5 dessert soufflés for the party. How many eggs will she need?

*Correct answer:*

The savory soufflés need 8 eggs each and she wants to make 3 of these so she needs  $8 \times 3 = 24$  eggs The dessert soufflés need 6 eggs each and she wants to make 5 of these so she needs  $6 \times 5 = 30$  eggs For the dinner party she will need  $24 + 30 = 54$  eggs in total ##### 54

*T5-CE:*

Roberta wants to make 3 savory soufflés that are 8 eggs each for a total of  $3 \times 8 = 24$  eggs. She wants to make 5 dessert soufflés that are 6 eggs each for a total of  $5 \times 6 = 30$  eggs. In total, she will need  $24 + 30 = 62$  eggs. ##### 62

*T5-NTL:*

Roberta wants to make 3 savory soufflés that use 8 eggs each so that's  $3 \times 8 = 24$  eggs She wants to make 5 dessert soufflés that use 6 eggs each so that's  $5 \times 6 = 30$  eggs All total, she needs  $24 + 30 = 54$  eggs ##### 54