Making Hard Problems Easier with Custom Data Distributions and Loss Regularization: A Case Study in Modular Arithmetic

AUTHORS
Eshika Saxena[1]*
Alberto Alfarano[1]*
Emily Wenger[2]†
Kristin Lauter[1]†

AFFILIATIONS
[1]Fundamental AI Research (FAIR)
[2]Duke University

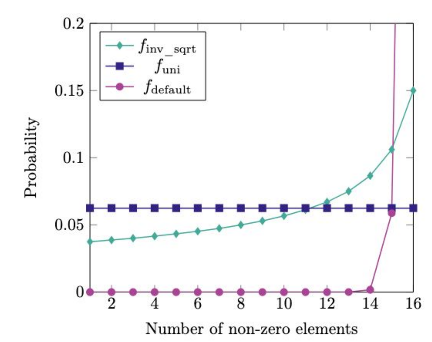
MODULAR ARITHMETIC

ML Models Struggle on Modular Arithmetic

- Task: Given N elements $[x_1, x_2 ... x_N], x_i \in \mathbb{Z}_q$, compute $s = \sum_{i=1}^{N} x_i \mod q$
- Existing methods struggle to do modular arithmetic with high N and q, which is useful for applications of ML in cryptanalysis

Key Improvements to ML Modular Arithmetic

- **Augmenting training data with easy examples**



Varying the number of non-zero elements in each example by sampling from these distributions

- **Loss regularization to avoid model collapse**

$$\ell = \alpha\left(x'^2 + y'^2 + \frac{1}{x'^2 + y'^2}\right) + \left((x - x')^2 + (y - y')^2\right), \quad \alpha = 10^{-4}$$

Key Results and Findings

- High accuracy on modular addition across a range of # terms and modulus
  - Sparse examples are critical for learning
  - KL divergence of train and test sets impacts accuracy

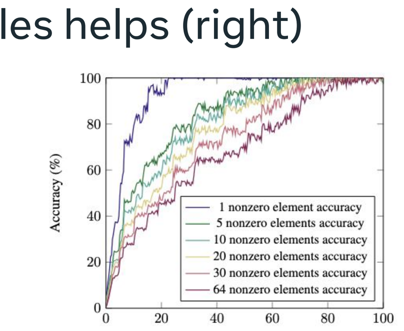
# Terms (N)	Mod (q)	MSE	$\tau = 0.5\%$ Accuracy	$\tau = 1\%$ Accuracy
16	257	0.00	99.8%	100.0%
	3329	0.00	99.7%	100.0%
	42899	0.00	99.7%	100.0%
	974269	0.00	99.7%	100.0%
32	257	0.00	99.5%	100.0%
	3329	0.00	99.4%	100.0%
	42899	0.00	99.4%	100.0%
	974269	0.00	99.5%	100.0%
64	257	0.01	98.9%	99.4%
	3329	0.01	97.4%	99.4%
	42899	0.01	97.4%	99.4%
	974269	0.01	98.2%	99.4%
128	257	0.04	96.1%	98.2%
	3329	0.04	92.9%	98.0%
	42899	0.05	94.1%	97.9%
	974269	0.04	93.3%	97.4%

Best performance of models on different N and q

# Terms (N)	Mod (q)	Training Data f	$\tau = 0.5\%$ Accuracy	KL divergence
16	257	$f_{default}$	1.2%	0.0
		f_{inv_sqrt}	**99.8%**	25.2
		f_{uni}	99.7%	35.4
32	257	$f_{default}$	1.3%	0.0
		f_{inv_sqrt}	**99.5%**	49.8
		f_{uni}	98.9%	71.5
64	257	$f_{default}$	1.3%	0.0
		f_{inv_sqrt}	**98.9%**	98.1
		f_{uni}	95.3%	144.0
128	257	$f_{default}$	1.3%	0.0
		f_{inv_sqrt}	**96.1%**	193.7
		f_{uni}	92.7%	289.5

Data distribution makes a big difference in learning

- Models learn easy examples before hard ones (left) and repeating examples helps (right)



Accuracy on sparser examples increases first and in order of sparsity

	$\tau = 0.5\%$ Accuracy			
# data repeats	$N = 16$	$N = 32$	$N = 64$	$N = 128$
1000	99.3%	97.7%	95.7%	86.1%
100	**99.8%**	99.2%	98.4%	92.7%
10	**99.8%**	**99.5%**	**98.9%**	**96.1%**
1	99.7%	99.2%	97.2%	91.5%

Some repetition (but not too much) improves accuracy

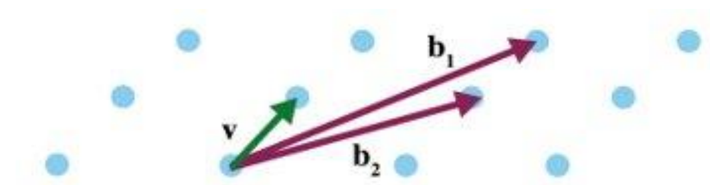
- The distribution is more important than the value of the filler input
- Our data distribution is more consistent than curriculum learning
- Loss regularization prevents model collapse when the task is hard

TL;DR Our **data distribution and loss regularization** methods improve transformer performance on modular arithmetic tasks, unlocking improvements in cryptography applications and other well-studied ML problems.

APPLICATION: CRYPTANALYSIS

Lattice Cryptography

- **Lattice cryptosystems** are believed to be quantum and classically secure



The dots form a lattice Λ generated by taking Z-linear combinations of {b1, b2}. v is the shortest vector in the lattice.

- Lattice cryptosystems based on the **Learning with Errors (LWE)** problem are frontrunners for post-quantum standardization by NIST

LWE Encryption

$$(\mathbf{A} \cdot \mathbf{s} + \mathbf{e}) \bmod q = \mathbf{b}$$

$\mathbf{A} \in \mathbb{Z}_q^{m \times n}$ is a uniformly random $m \times n$ matrix
$\boldsymbol{s}$ = secret, $\boldsymbol{s} \in \mathbb{Z}_q^n$
$\boldsymbol{e}$ = error, $\boldsymbol{e} \in \mathbb{Z}_q^n$, $\boldsymbol{e} \sim N(\mu, \sigma)$
q = modulus, typically a large prime

- LWE-based cryptosystems may be quantum-resistant, but **are they secure against classical attacks?**

Effect of Loss Regularization in LWE Setting

- We apply our loss regularization to LWE (without the error)
- Adding regularization term encourages the model to avoid the origin, leading to **faster and better convergence**

			Recovery %	
# Terms (N)	Hamming weight	Mod (q)	Custom Loss $\alpha = 10^{-2}$	MSE Loss $\alpha = 0$
64	6	257	**15%**	0%
		3329	**20%**	0%
		42899	**15%**	0%
		974269	**15%**	0%
128	5	257	**10%**	0%
		3329	**15%**	0%
		42899	**15%**	0%
		974269	**10%**	0%
256	4	257	**10%**	0%
		3329	**15%**	0%
		42899	**15%**	0%
		974269	**15%**	0%

Model performs better when trained with our custom loss on the LW(ithout)E problem

References

Saxena, E., Alfarano, A., Wenger, E., and Lauter, K. E. Making hard problems easier with custom data distributions and loss regularization: A case study in modular arithmetic. In Forty-second International Conference on Machine Learning, 2025.

Full Paper
Code Repo

BEYOND MODULAR ADDITION

Other Modular Arithmetic Tasks

- Asymmetric functions: class $h : \mathbb{Z}_q^N \to \mathbb{Z}_q$ where $h_{j,k} = \left(\sum_{i=1}^{N} a_i^j\right)^2 + a_1^k$

Function	% Accuracy
$h_{j=1,k=1} = (a_1 + \ldots + a_N)^2 + a_1^1 \mod q$	95.1%
$h_{j=1,k=3} = (a_1 + \ldots + a_N)^2 + a_1^3 \mod q$	96.2%
$h_{j=2,k=1} = (a_1^2 + \ldots + a_N^2)^2 + a_1^1 \mod q$	95.5%

Models can learn other modular arithmetic functions with our methods

- Modular multiplication (left) and scalar product (right)

# Terms (N)	Mod (q)	$\tau = 1\%$ Accuracy
16	97	100%
	257	98%
	3329	3%
32	97	100%
	257	75%
	3329	3%
64	97	100%
	257	65%
	3329	3%

# Terms (N)	Mod (q)	$\tau = 1\%$ Accuracy
2	97	100%
	257	100%
	3329	3%
4	97	100%
	257	30%
	3329	3%
8	97	78%
	257	2%
	3329	3%

Our methods extend to modular multiplication and scalar product, but accuracy declines with higher N and q

Synthetic Tasks

- **Copy task**: given a vector of size N, output an exact copy of the vector
- **Associative recall task**: given N keys and N values sampled from two distinct vocabularies, retrieve the correct value of one key
- **Parity task**: given a binary vector of size N, output the parity of the vector
- **Selective copy**: given a vector of size N with T = 16 non-zero elements and N - T elements equal to zero, output a copy of the vector, discarding all elements equal to zero.

		% Accuracy		
Task	# max_length	$f_{default}$	f_{inv_sqrt}	f_{uni}
Copy	32	**100.0%**	**100.0%**	**100.0%**
	64	**100.0%**	**100.0%**	**100.0%**
	128	94.3%	**100.0%**	**100.0%**
	256	81.4%	**98.1%**	97.4%
Associative recall	8	32.5%	**100.0%**	**100.0%**
	16	6.6%	**100.0%**	**100.0%**
	32	3.4%	**100.0%**	**100.0%**
	64	1.8%	**100.0%**	1.8%
Parity	32	50.3%	**100.0%**	**100.0%**
	64	50.6%	99.8%	**100.0%**
	128	50.0%	**99.7%**	50.2%
	256	50.2%	**99.4%**	50.2%
Selective copy	32	**100.0%**	**100.0%**	**100.0%**
	64	**100.0%**	**100.0%**	**100.0%**
	128	83.4%	**100.0%**	**100.0%**
	256	57.2%	**100.0%**	99.3%

Training data with more diverse problem lengths yields better accuracy across different tasks

Conclusion

- Two key methods that help ML models learn modular addition and can be applied to the Learning with Errors (LWE) problem in cryptography to recover **2x harder secrets than prior work**
- Methods generalize and improve learning outcomes on other tasks
- Future Work:
  - Improve performance as number of terms N scales
  - Transferring techniques to other settings like real-world cryptanalysis