“Matrix MoE is the best Fast Feed-Forward”

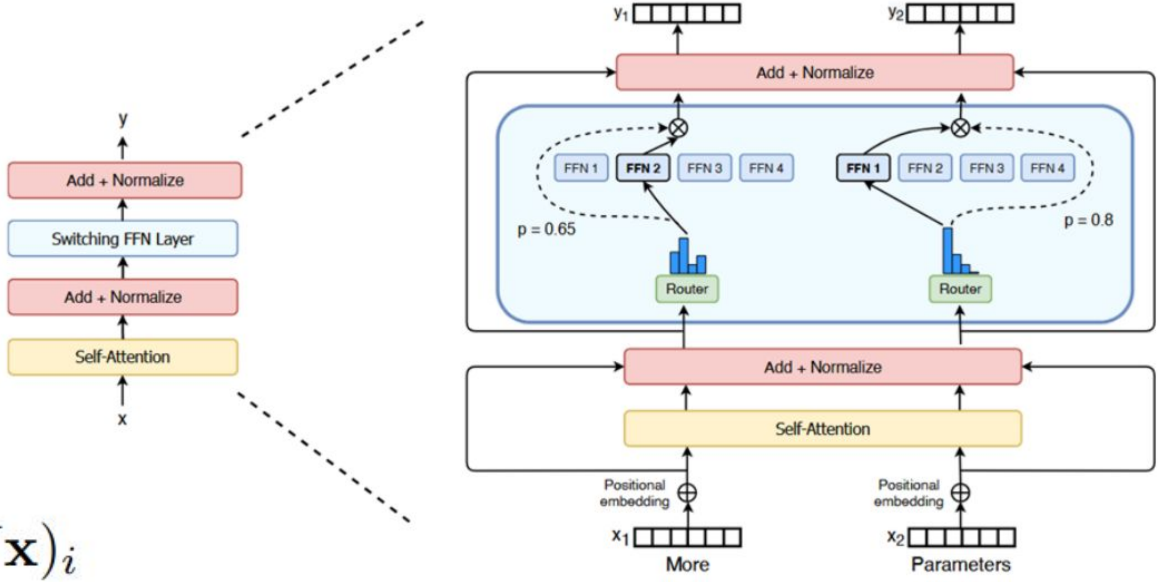
Vyacheslav Chaunin, Nickolay Mikhailovskiy

Higher IT School, Tomsk State University, NTR Labs

MathAI 2025

Mixture of Experts

- Smaller sub-models
- Routing mechanism

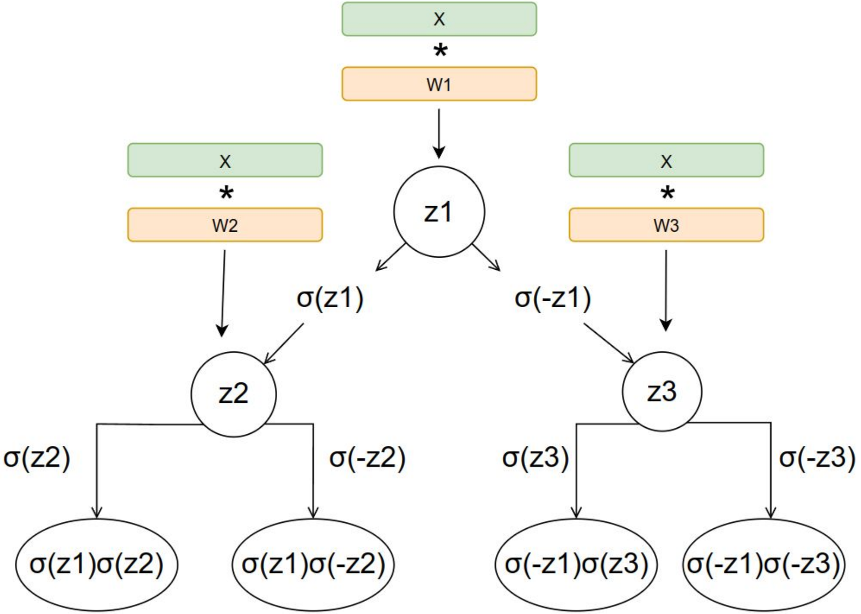


$$R(i|\mathbf{x}) = \text{Softmax}(W\mathbf{x})_i$$

$$\mathbf{y} = \sum_{i \in \text{top-}k(R)} R(i|\mathbf{x}) f_i(\mathbf{x})$$

Fast Feed-Forward

- Binary tree of routers
- $2^d - 1$ router parameter vectors
- 2^d experts



$$P(2i|\mathbf{x}, i) = \sigma(z_i) = \sigma(\mathbf{w}_i^\top \mathbf{x})$$

$$P(2i+1|\mathbf{x}, i) = \sigma(-z_i) = 1 - \sigma(\mathbf{w}_i^\top \mathbf{x})$$

FFF Training

The output of a layer is a weighted average across all experts

$$\mathbf{y} = \sum R(i|\mathbf{x}) f_i(\mathbf{x})$$

$$R(i|\mathbf{x}) = \prod_{j \in \mathrm{Path}(i)} P(j|\mathbf{x}, \lfloor \frac{j}{2} \rfloor)$$

The ***Path(i)*** contains indices of all router nodes that are parents of leaf ***i***

FFF Reformulation

$$\log R(i|\mathbf{x}) = \sum_{j \in \mathrm{Path}(i)} a((-1)^j z_{\lfloor j/2 \rfloor})$$

$$a(x) = \log \sigma(x) = -\mathrm{Softplus}(-x)$$

FFF Reformulation

$$\mathbf{z} = (z_1, \ldots, z_{2^d-1}) = W\mathbf{x}$$

$$R(i|\mathbf{x}) = \mathrm{Softmax}(Ta(S\mathbf{z}))_i$$

$$T \in \mathbb{R}^{2^d \times 2(2^d-1)} \qquad T_{ij} = 1 \text{ if } j \in \mathrm{Path}(i) \text{ and } 0 \text{ otherwise}$$

$$S \in \mathbb{R}^{2(2^d-1) \times 2^d-1} \qquad S_{ij} = \begin{cases} 1 & \text{if } 2i-1=j \\ -1 & \text{if } 2i=j \\ 0 & \text{otherwise} \end{cases}$$

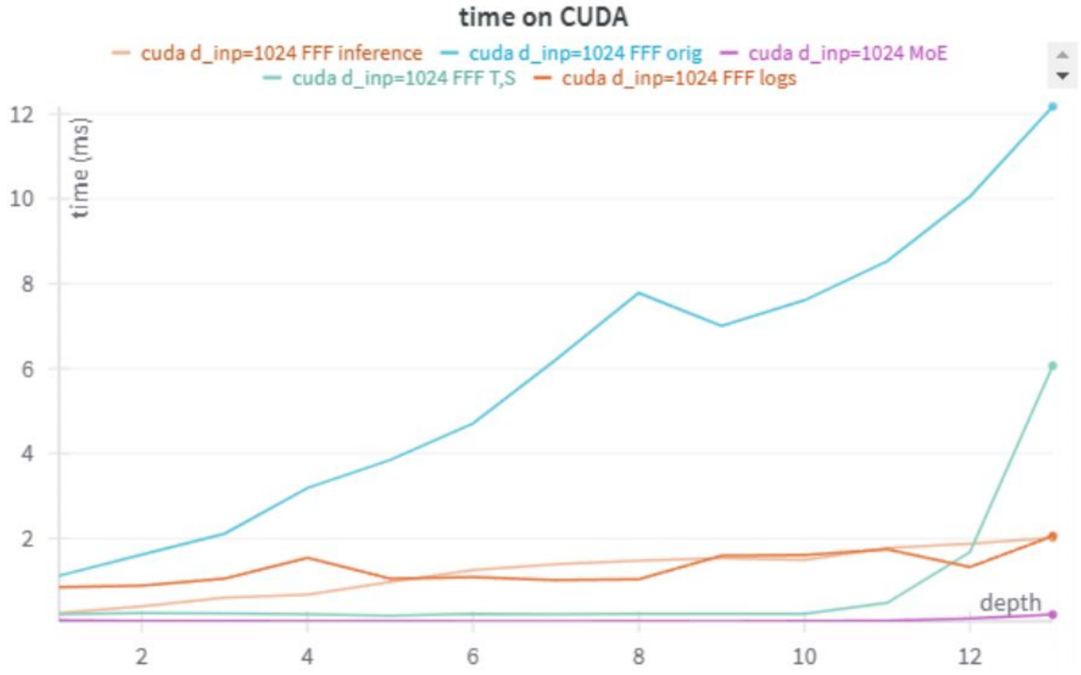
Example of FFF layer with d=2

$$T = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix} \quad S = \begin{pmatrix} 1 & 0 & 0 \\ -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & -1 \end{pmatrix}$$

Benchmarks

1024 input dimensions

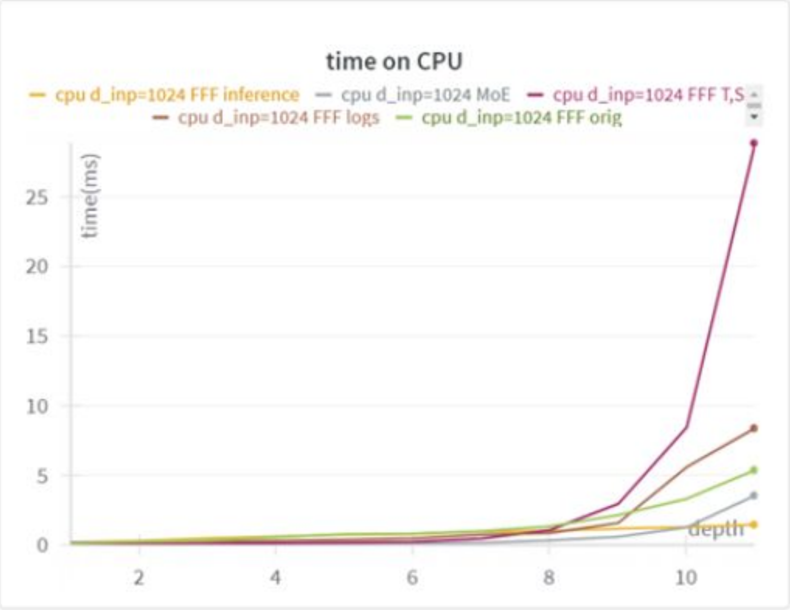
Method	GPU		CPU	
	Up to 8	Up to 13	Up to 8	Up to 13
MoE	**45.63**	**57.84**	**7.27**	**4.55**
FFF T,S	11.89	8.93	3.55	0.88
FFF Inference	4.22	4.52	0.90	1.09
FFF Logs	2.58	3.22	1.27	1.07
FFF Orig	1.00	1.00	1.00	1.00

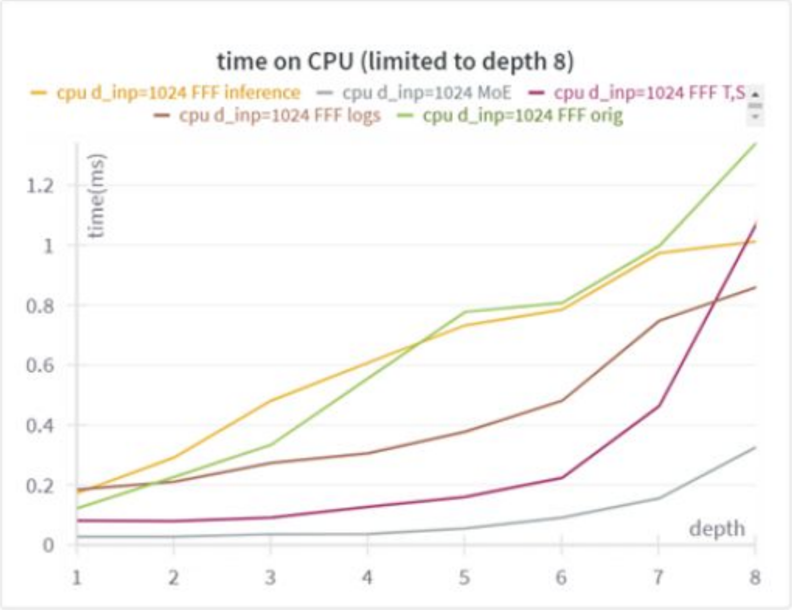


Benchmarks

1024 input dimensions

Method	GPU		CPU	
	Up to 8	Up to 13	Up to 8	Up to 13
MoE	**45.63**	**57.84**	**7.27**	**4.55**
FFF T,S	11.89	8.93	3.55	0.88
FFF Inference	4.22	4.52	0.90	1.09
FFF Logs	2.58	3.22	1.27	1.07
FFF Orig	1.00	1.00	1.00	1.00





Experiments with Activation function

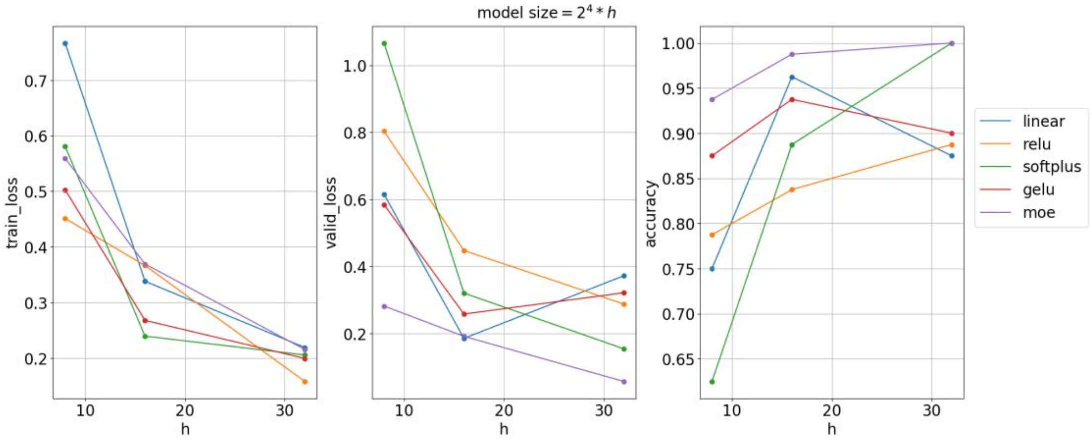
- Candidate functions:
  - Softplus
  - Linear
  - ReLU
  - GELU
- Setups:
  - Clusters of Gaussians
  - CIFAR-10
  - Cramming BERT

$$R(i|\mathbf{x}) = \mathrm{Softmax}(Ta(S\mathbf{z}))_i$$

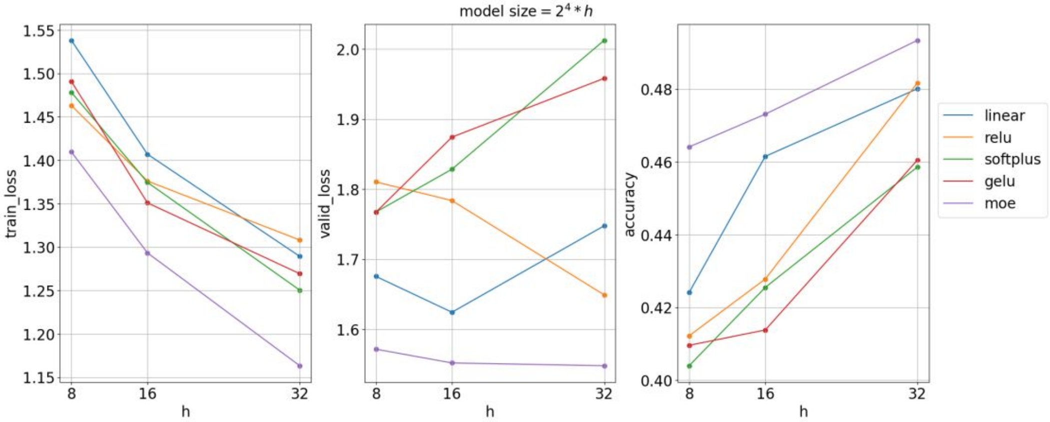
Synthetic and CIFAR-10 setup

- Single FFF/MoE routing layer
- Experts are 2-layer MLPs of varying width from [8, 16, 32]
- Batch size 16 (64)
- Adam optimizer with one-cycle schedule and max lr 8e-4
- Dropout 0.2
- 15 (20) epochs

Experiments



Synthetic dataset



CIFAR-10

Results

CIFAR-10 accuracy across layer depths

Activation	d=2.0	d=3.0	d=4.0	d=5.0
Linear	**0.470**	0.461	**0.455**	**0.447**
ReLU	0.464	**0.468**	0.441	0.416
Softplus	0.458	0.461	0.429	0.427
GELU	0.461	0.456	0.428	0.408
MOE	**0.472**	**0.483**	**0.477**	**0.470**

Cramming BERT results

Activation	STSB	SST2	RTE	MRPC	COLA	Average GLUE
Linear	**0.564**	0.859	0.484	0.706	0.129	0.57
Softplus	0.439	**0.860**	0.509	0.708	0.112	0.54
ReLU	0.560	0.849	**0.560**	0.716	0.117	**0.58**
GELU	0.506	0.855	0.545	**0.723**	**0.165**	**0.58**

Cramming BERT https://arxiv.org/abs/2212.14034

Conclusions

- FFF can be practically used only with large number of experts, i.e. in very fragmented models
- For training it’s more optimal to vectorize the process
- For many models using linear activation might improve the results

Thank you for attention!