Alternating Updates for Efficient Transformers

Cenk Baykal, Dylan Cutler, Nishanth Dikkala,
Nikhil Ghosh*, Rina Panigrahy, Xin Wang

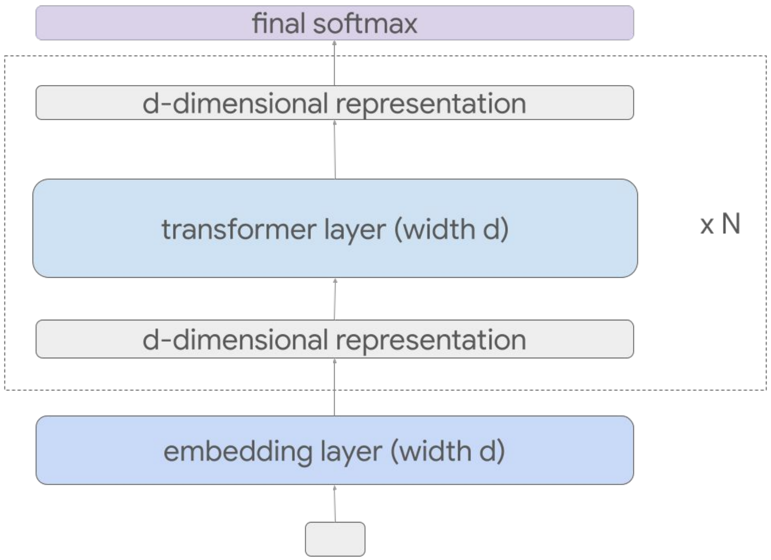
Google Research

*UC Berkeley; work done as an intern at Google Research

Transformers

Form the backbone of all state-of-the-art language models
- Bard, ChatGPT, LLaMA

What are they really doing?
- N layers, each layer with attention and FFN modules
- Iterative refinement of a *token representation vector*



A (decoder-only) transformer iteratively refines d-dimensional token representations across N layers

Transformers

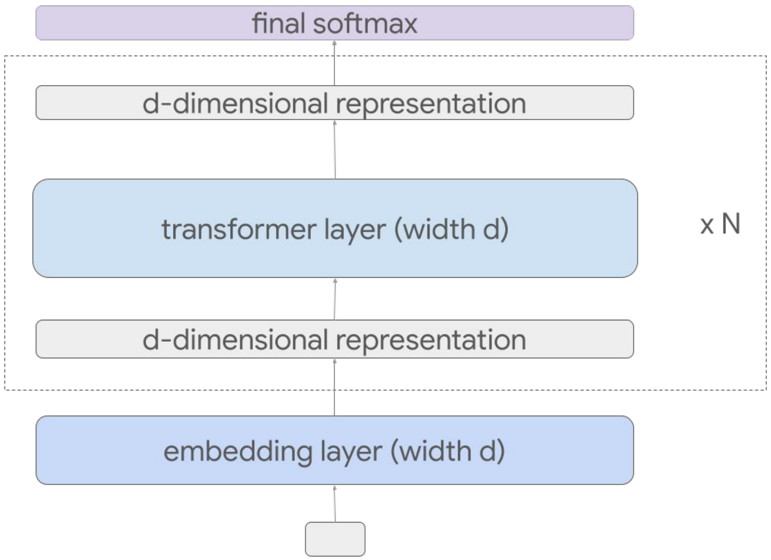
Form the backbone of all state-of-the-art language models

- Bard, ChatGPT, LLaMA

What are they really doing?

- N layers, each layer with attention and FFN modules
- Iterative refinement of a *token representation vector*

Number of parameters in each layer scales with the representation vector dimension (*representation width)*



A (decoder-only) transformer iteratively refines d-dimensional token representations across N layers

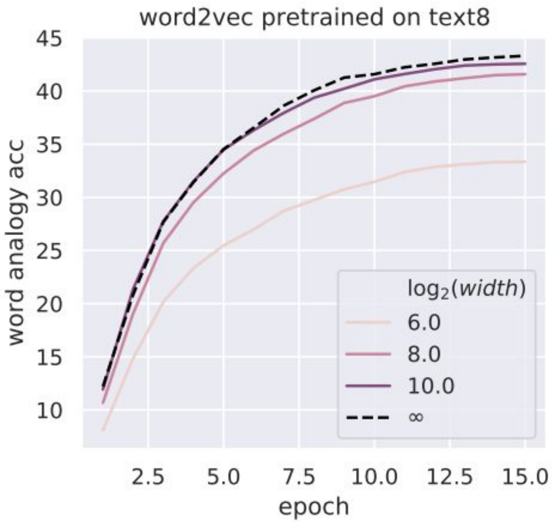
Increasing capacity by increasing representation dimension (d)

Improves performance

- Allows packing more information into representation vector
- Scaling laws (more parameters)
- Enables learning more complicated functions

… at the cost of quadratic increase in parameters and compute

- FFNs in each layer contain O(d^2) parameters



Performance for a word2vec model with varying representation dimension ([figure from Yang et al., 2022]).

Increasing capacity by increasing representation dimension (d)

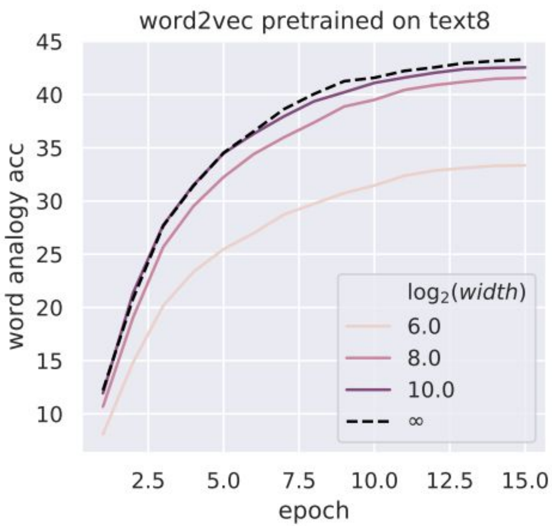
Improves performance

- Allows packing more information into representation vector
- Scaling laws (more parameters)
- Enables learning more complicated functions

… at the cost of quadratic increase in parameters and compute

- FFNs in each layer contain O(d^2) parameters

Can we get the benefits of increased representation dimension without the full computational cost?



Performance for a word2vec model with varying representation dimension (figure from Yang et al., 2022).

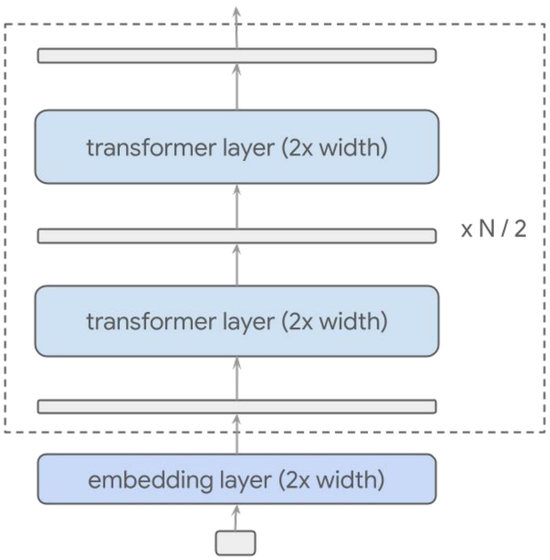
Alternating Updates (AltUp)

Increase representation width, but keep *transformer layer constant*

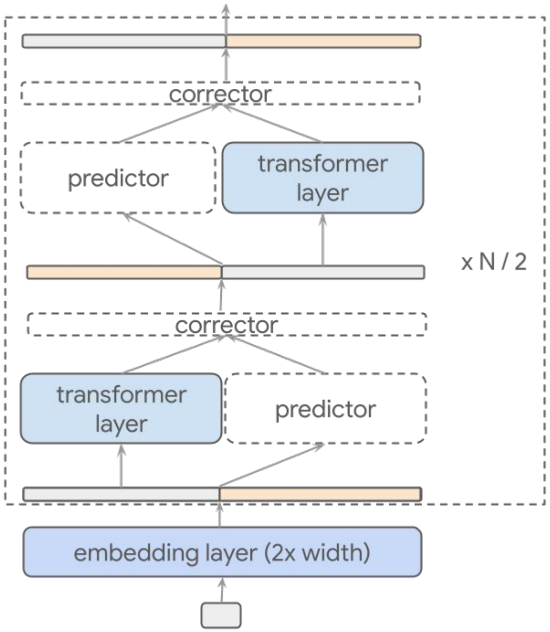
Activate a *sub-block* of the representation in each layer

Use *lightweight* predict-and-correct to update inactivated blocks

Generalizes to K >= 2 blocks



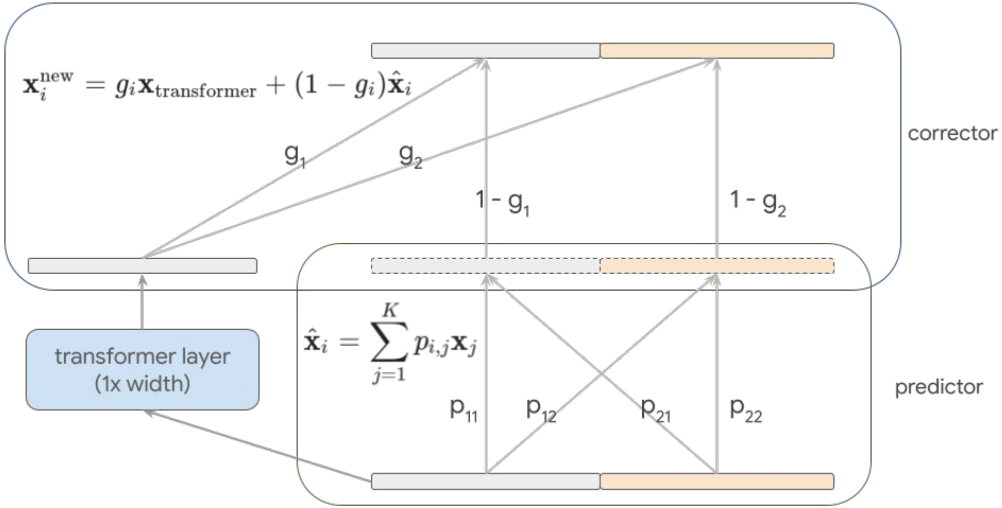
(a) 2x wider model



(b) 2x wider model with AltUp

Predict-and-Correct

Two lightweight components with a total of $K^2 + K$ trainable parameters per layer*



An example of the predict-and-correct step with $K = 2$. All depicted parameters in the predictor and corrector are trainable scalars.

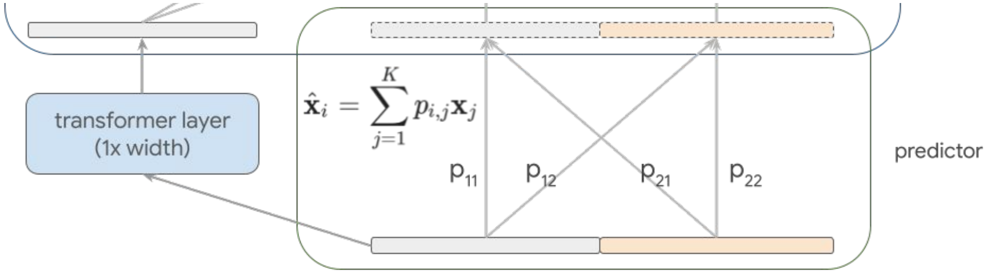
*: $K=2$ by default, so $K^2 + K = \mathbf{6}$ additional parameters per layer

Predict-and-Correct

Two lightweight components with a total of $K^2 + K$ trainable parameters per layer*

Prediction: $O(K^2 * d)$ time

- Each block is a weighted mixture of blocks
- Enables information passing across blocks



An example of the predict-and-correct step with $K = 2$. All depicted parameters in the predictor and corrector are trainable scalars.

*: $K=2$ by default, so $K^2 + K =$ **6** additional parameters per layer

Predict-and-Correct

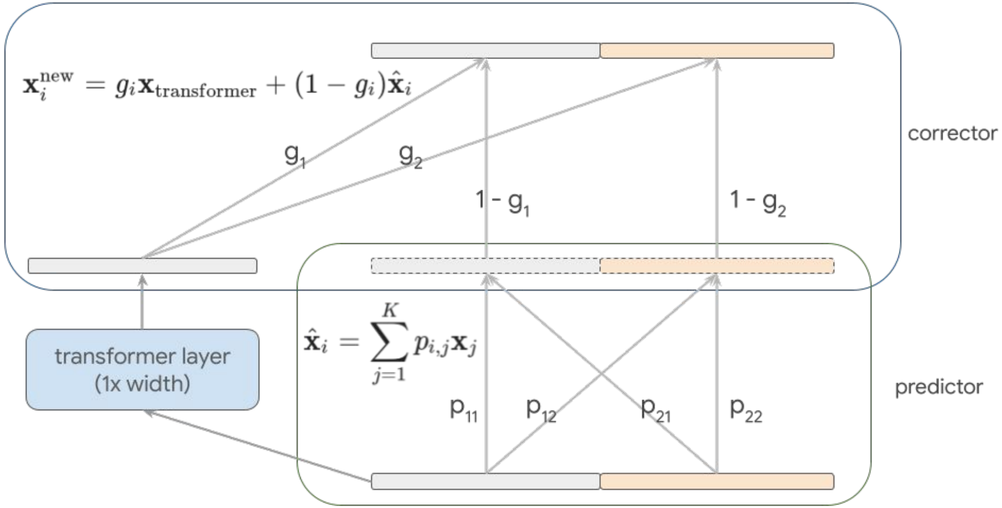
Two lightweight components with a total of K^2 + K trainable parameters per layer*

Prediction: O(K^2 * d) time

- Each block is a weighted mixture of blocks
- Enables information passing across blocks

Correction: O(K * d) time

- Update blocks based on observed output of the activated block

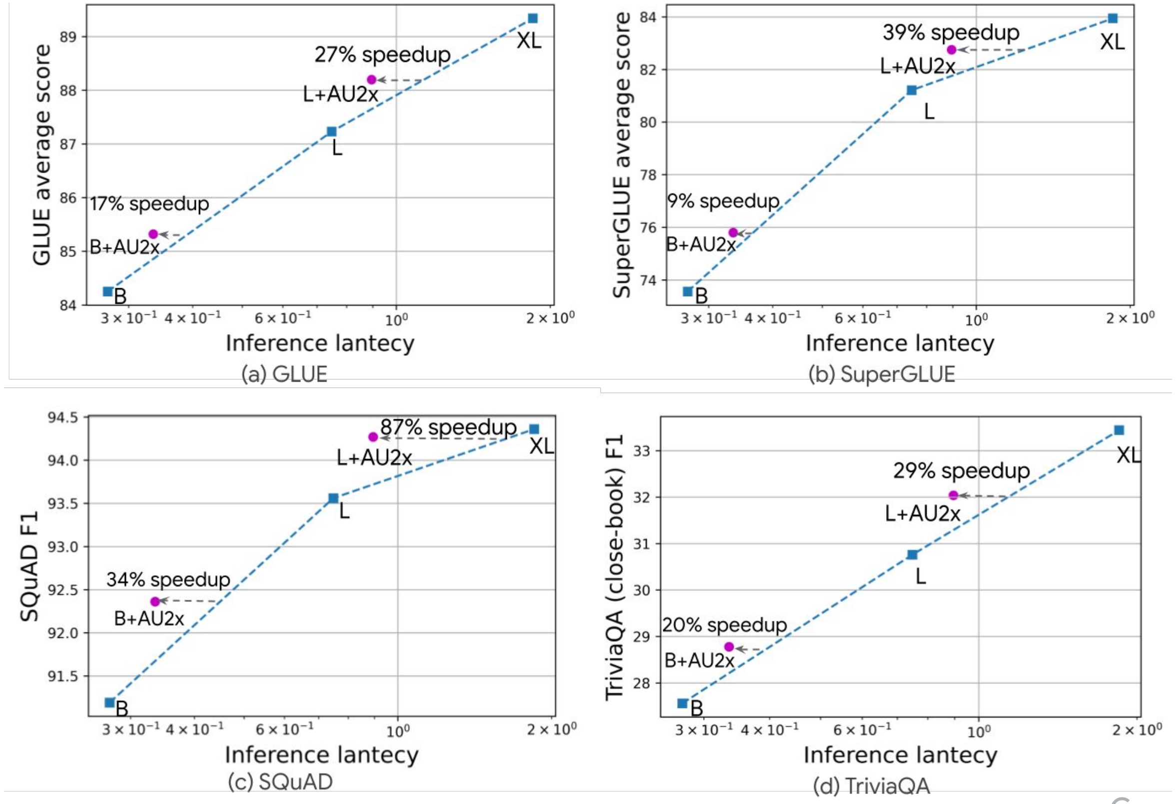


An example of the predict-and-correct step with K = 2. All depicted parameters in the predictor and corrector are trainable scalars.

*: K=2 by default, so K^2 + K = **6** additional parameters per layer

Evaluations on T5: up to 87% speedup relative to dense baselines

We set K = 2 by default.
No hyperparameter tuning required!

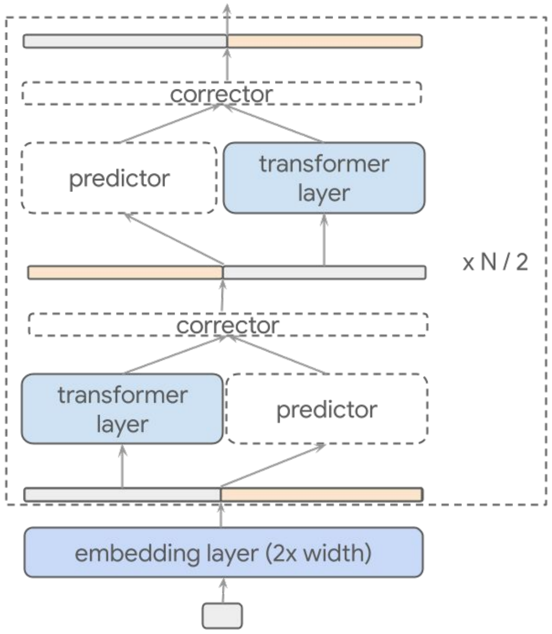


(a) GLUE
(b) SuperGLUE
(c) SQuAD
(d) TriviaQA

AltUp: a conditional computation technique

Conditional computation along the representation dimension

- Same underpinning as Mixture of Experts (MoE) models
- Shifts parameters from backbone to embedding table
- *Orthogonal* to existing conditional computation approaches ⇒ *Synergistic combination*



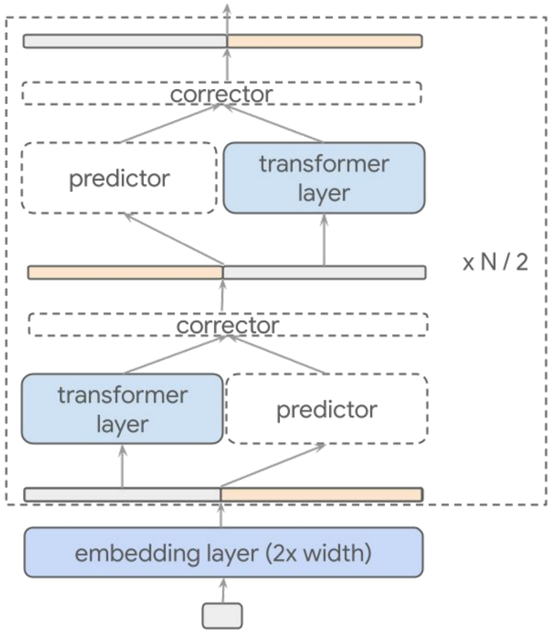
(b) 2x wider model with AltUp

Synergistic combination with existing approaches

Conditional computation along the representation dimension

- Same underpinning as Mixture of Experts (MoE) models
- Shifts parameters from backbone to embedding table
- *Orthogonal* to existing conditional computation approaches ⇒ *Synergistic combination*

Method	**T5 Small**	**T5 Base**	**T5 Large**
Baseline	59.10	63.35	65.58
MoE [60]	59.42	63.62	65.71
AltUp (K=2)	59.67	63.97	65.73
AltUp (K=2) + MoE	**59.91**	**64.13**	**65.95**



(b) 2x wider model with AltUp

Scaling with expansion factor K

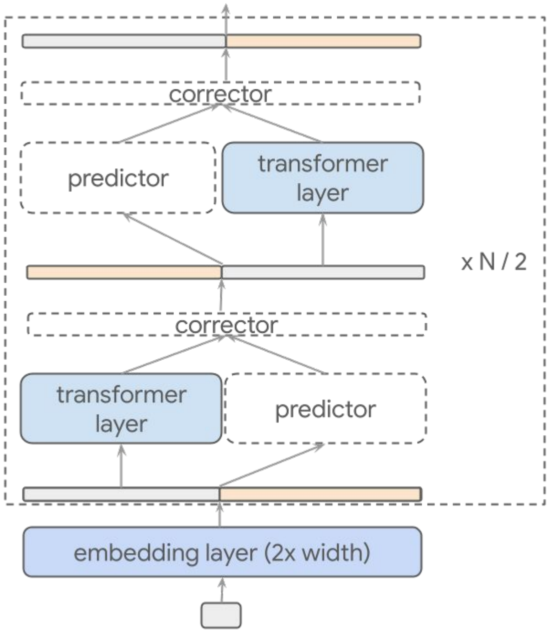
AltUp with expansion factor K on model with vocab size V

- adds O(V(K-1)d) embedding table parameters
- final linear + softmax computation: O(Vd) → O(VKd)

not significant for large models with many layers and moderate-sized vocabularies

What if the vocabulary is relatively very large?

Use Recycled-AltUp!



(b) 2x wider model with AltUp

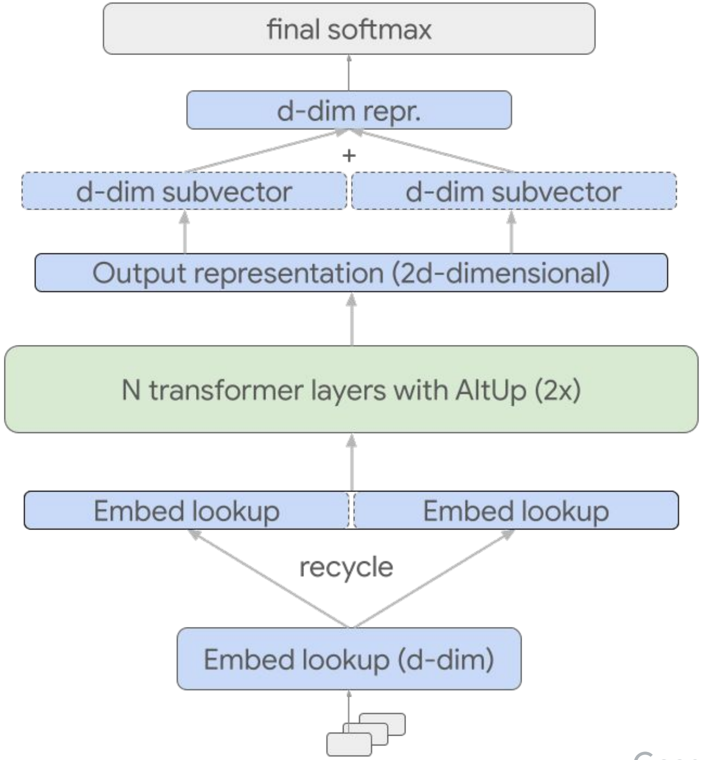
Recycled-AltUp: Lightweight extension of AltUp

Recycles a single embedding lookup (no expansion of table)
Projects down efficiently before final softmax

⇒ embedding table parameters:
O(Vd) ⟶ O(Vd) **(constant)**

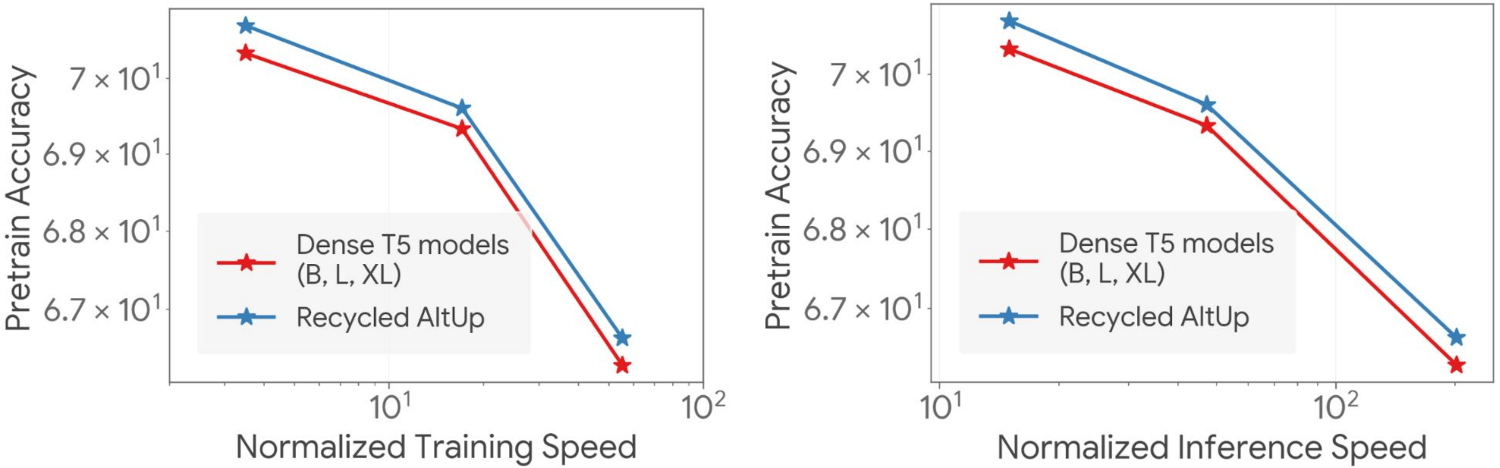
⇒ final linear + softmax computation:
O(Vd) ⟶ O(Vd) **(constant)**

Ideal for models with relatively large vocabularies



Recycled-AltUp Evaluations

O(100) **total** parameters added
Improved performance at the cost of virtually no slowdown



Alternating Updates for Efficient Transformers

Cenk Baykal, Dylan Cutler, Nishanth Dikkala,
Nikhil Ghosh*, Rina Panigrahy, Xin Wang

Google Research

*UC Berkeley; work done as an intern at Google Research