Universal Transformers

Daniel Hsu

Columbia University

Joint work with Jingwen Liu and Alexandr Andoni

NSF MFAI PI Meeting
November 6, 2026

Hooray, computing!

- Many applications of computing:
  - Mathematical computations
  - Playing games
  - Rendering graphics
  - Etc.
- Do we need a different computer for each such application?



Math computer



Game computer



Graphics computer

Universality in computing

- Some of our favorite models of computation have universal instances
- [Turing, 1936] There is a Turing Machine UTM s.t.:
  for any Turing Machine M,

$$\mathrm{UTM}(\langle \mathrm{M} \rangle, x) = \mathrm{M}(x), \qquad \forall \text{ inputs } x$$

- [Valiant, 1976] Pick any size S. There is a Boolean circuit UC of size $O(n + S \log S)$ s.t.:
  for any Boolean circuit C of size $\leq S$,

$$\mathrm{UC}(x, \langle \mathrm{C} \rangle) = \mathrm{C}(x), \qquad \forall x \in \{0,1\}^n$$

Hooray, machine learning!

- Many applications of machine learning (ML):
  - Image classification
  - Speech recognition
  - Machine translation
  - Etc.
- Do we need a different ML model for each such application?



Image ML model



Speech ML model

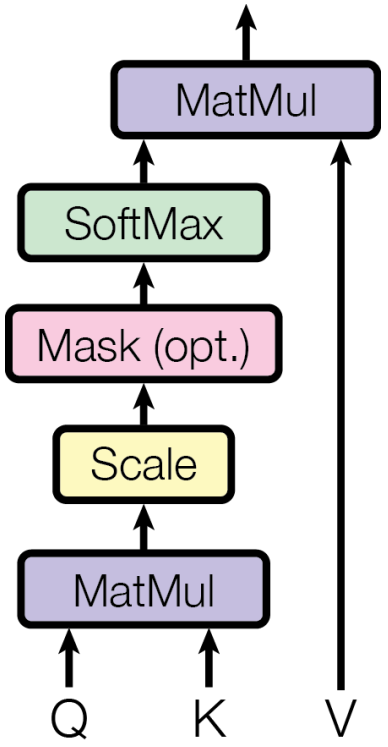


Language ML model

Transformer architecture vs. transformers

- "Transformer": composition of (sums of) "self-attention heads"
  - **This talk**: a "transformer" refers to an instantiation of the "transformer architecture" that specifies parameter values for all self-attention heads

$$Q := X\ W_Q$$
$$K := X\ W_K$$
$$V := X\ W_V$$



[Vaswani et al, 2017]

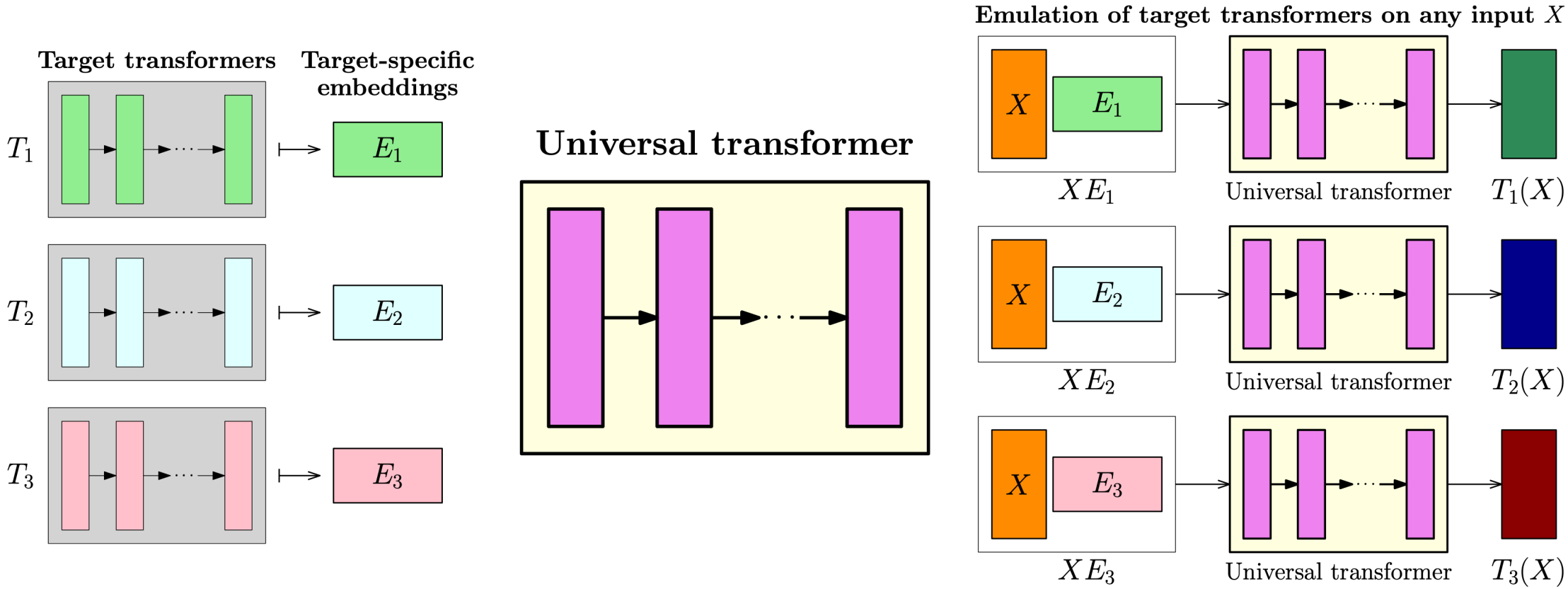
Universal single-headed transformers

- [Liu, Andoni, H.] Pick any depth L and model dim d. There is a single-headed transformer UTF with depth L and model dim $m \asymp Ld$ s.t.:

  for any single-headed transformer TF of depth L & model dim d,

$$\mathrm{UTF}(X\, E_{\mathrm{TF}}) = \mathrm{TF}(X), \qquad \forall X \in \mathbb{R}^{N\times d}\ \forall N \in \mathbb{N}$$

  - Here, $E_{\mathrm{TF}} \in \mathbb{R}^{d\times m}$ is an embedding matrix based on params of TF
  - In fact, UTF is a "looped transformer": shares param matrices across layers


Target transformers
Target-specific embeddings
T_1
E_1
T_2
E_2
T_3
E_3
Universal transformer
Emulation of target transformers on any input X
X
E_1
XE_1
Universal transformer
$T_1(X)$
X
E_2
XE_2
Universal transformer
$T_2(X)$
X
E_3
XE_3
Universal transformer
$T_3(X)$

Analogies

Universal Object	Target Class	Target Description	Simulation Guarantee
UTM	Turing Machines	<TM> (input bits)	UTM(<M>,x) = M(x)
UC (size $O(n + S \log S)$)	Circuits of size $\leq S$	<C> (control bits)	UC(x,<C>) = C(x)
UTF (dim $m \asymp Ld$)	(L, d) Transformers	E_{TF} (embed matrix)	UTF(XE_{TF}) = TF(X)

Universal multi-headed transformers

- [Liu, Andoni, H.] To get universal TF for class of multi-headed transformers with depth L, H heads per layer ($H \geq 2$), model dim d, suffices to use

$$m \asymp H^L d$$

(and depth L and H heads per layer)

  - And this is (almost) best possible m for our "attention-matching" approach

Similar idea: model reprogramming [Chen, 2023]

- Swap-in embedding/unembedding layers to solve new tasks

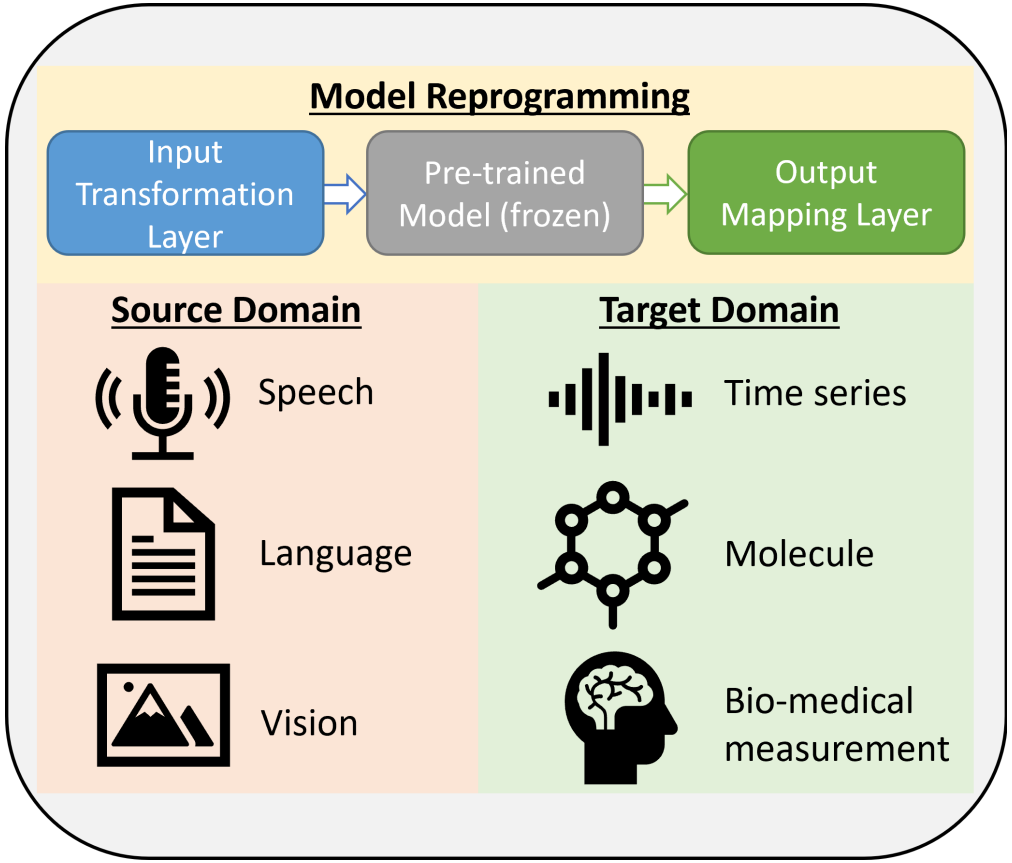
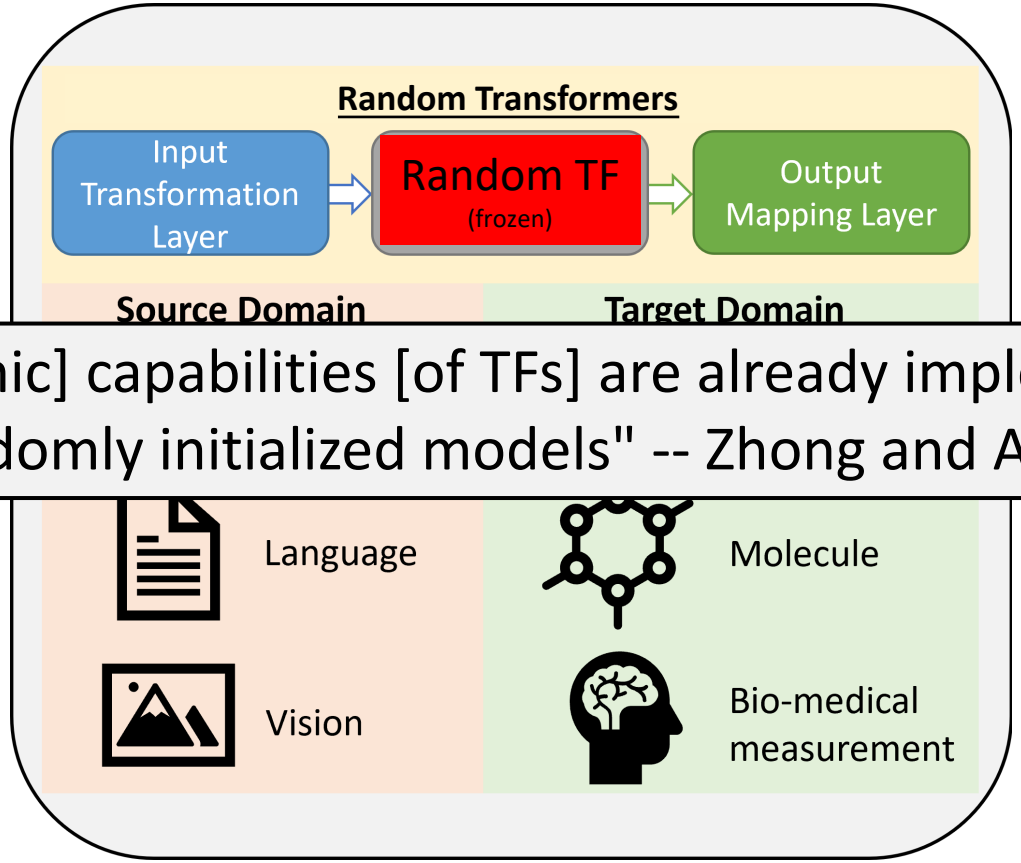


Figure from Pin-Yu Chen, 2023

Random transformers [Zhong and Andreas, 2024]

- Trained only embedding/unembedding layers to solve various tasks



"[Perhaps algorithmic] capabilities [of TFs] are already implemented in some fashion even in randomly initialized models" -- Zhong and Andreas, 2024

Original figure from Pin-Yu Chen, 2023

Random transformers are universal

- [Liu, Andoni, H.] (Untrained/frozen) randomly-initialized transformers (depth L, H heads per layer, model dim m as before) are universal a.s.
  - Mapping $\mathrm{TF} \mapsto E_{\mathrm{TF}}$ depends on the universal random TF's param matrices

- Note: Our "hand-constructed" UTF has sparse $m \times m$ param matrices, whereas a random TF has dense param matrices

Getting embedding matrices for specific tasks

- Q: How do we get the embedding matrix E for a task of interest?
- Example task: *k*-hop induction head [Sanford, H., Telgarsky, 2024]
  - Can represent by depth $L = 2 + \log_2 k$ (single-headed) transformer $\mathrm{TF}_{\mathrm{k-hop}}$
  - [SHT'24] describes param matrices for $\mathrm{TF}_{\mathrm{k-hop}}$;
    now apply the $\mathrm{TF} \mapsto E_{\mathrm{TF}}$ mapping from our theorem to get $E_{\mathrm{TF}_{\mathrm{k-hop}}}$
- Alternative: Use gradient descent to learn E from examples

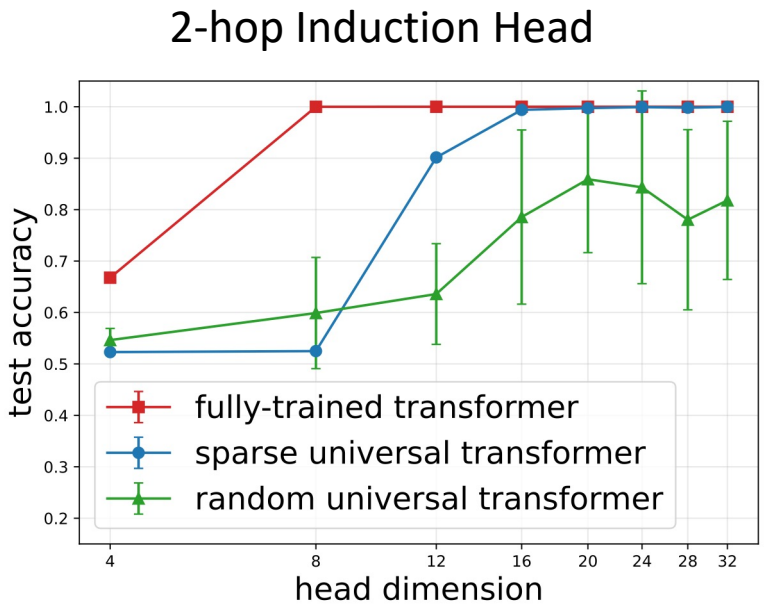
Experiments with learning the embedding matrices

Task	Sparse	Random
Parenthesis Balancing	100.0%	94.4%
1-hop Induction Head	30.2%	29.4%
2-hop Induction Head	33.2%	30.5%
3-hop Induction Head	39.2%	32.6%
4-hop Induction Head	39.4%	35.2%

(a) Attention-only.

Task	Sparse	Random
Parenthesis Balancing	100.0%	100.0%
1-hop Induction Head	100.0%	96.1%
2-hop Induction Head	100.0%	85.7%
3-hop Induction Head	100.0%	92.6%
4-hop Induction Head	100.0%	89.2%

(b) Adding residual and layer normalization.



Takeaways

- Transformers are flexible enough to be "universal"
  - Even without CoT, even with "reasonable" size constraints
- NOT: "Universal Approximation Theorem", "Turing Completeness"
  - These are properties of the transformer architecture
  - UAT/TC => transformer architecture can be used for many things
  - (TC => there is a transformer that implements Python+Pytorch … but $m =$?)
- What's the role of parameters in a transformer?

Open questions

- Optimal embedding dimension m for multi-headed TFs?
- What does this imply about feature learning?
  (On-going work with Amirhesam Abedsoltan & Misha Belkin)
- Universal instances of other ML models?
- Universality in "AI" more generally?

Thanks!

arXiv:2605.31423
To appear @ NeurIPS 2026

Thanks to: NSF
Surbhi Goel, Enric Boix-Adsera, Ezra Edelman, Sanjoy Dasgupta