The Devil is in the Detail: Simple Tricks Improve Systematic Generalization of Transformers

Róbert Csordás
robert@idsia.ch

Kazuki Irie
kazuki@idsia.ch

Jürgen Schmidhuber
juergen@idsia.ch

IDSIA — Scuola universitaria professionale della Svizzera italiana SUPSI — Università della Svizzera italiana

Systematic generalization

- Ability to perform well well on **systematically different** inputs, governed by the same rules

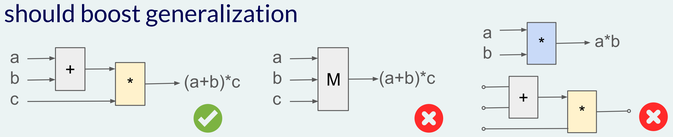
- Existing methods
  1. Neural networks with supervised learning - usually fail
  2. Meta-learning: helps a bit, but far from ideal
  3. Neuro-symbolic hybrids: work well, but task specific

Hypotheses

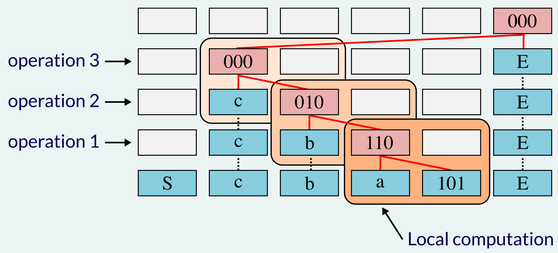
- The transformer achitecture looks well-suited for the algorithmic tasks usually used to test systematic generalization (but they are typically reported to fail in such tasks)
- The default configurations used are sub-optimal: they are typically just taken from the standard machine translation task
- There are existing augmentations of Transformers relevant for systematic generalization which are underexplored!

Architectural changes for sys. gen.

1. Decomposing the problem to elementary, reusable components should boost generalization



   - In Transformers, the output of an operation is available to only to the successive layers. Since operations should be composable in any order, layers should be shared.
   - This should help systematicity
2. Long compositions are often made of multiple local compositions
   - Should have a bias toward local computation



The EOS decision problem

- Described by Newman et al. (2020)
- The performance of neural networks is better if trained without the EOS token, even compared to oracle length-evaluation
- Universal Transformers with relative positional encoding generalize well without any tricks

	ℓ (length cutoff)	22	24	25	26	27	28	30	32	33	36	40
Reference	+EOS	0.00	0.05	0.04	0.00	0.09	0.00	0.09	0.35	0.00	*0.00*	0.00
	+EOS+Oracle	*0.53*	*0.51*	*0.69*	*0.76*	*0.74*	*0.57*	*0.78*	*0.66*	*0.77*	***1.00***	*0.97*
	−EOS+Oracle	***0.58***	***0.54***	*0.67*	*0.82*	*0.88*	*0.85*	*0.89*	*0.82*	***1.00***	***1.00***	***1.00***
Ours (+EOS)	Trafo	0.00	0.04	0.19	0.29	0.30	0.08	0.24	0.36	0.00	0.00	0.00
	+ Relative PE	0.20	0.12	0.31	0.61	**1.00**	**1.00**	**1.00**	0.94	**1.00**	**1.00**	**1.00**
	Universal Trafo	0.02	0.05	0.14	0.21	0.26	0.00	0.06	0.35	0.00	0.00	0.00
	+ Relative PE	0.20	0.12	**0.71**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**

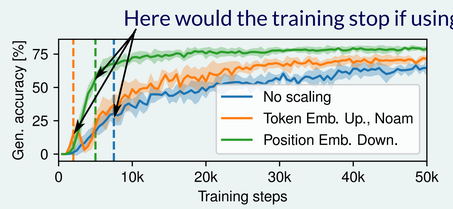
Problems with model selection

- IID validation accuracy have weak or no signal for determining OOD test accuracy

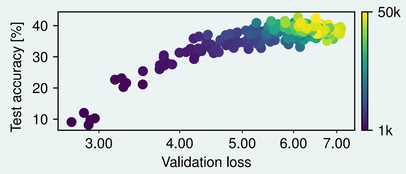
	Transformer	Uni. Transformer	Rel. Transformer	Rel. Uni. Transformer
SCAN (length cutoff=26)	**1.00 ± 0.00** (0.30)	**1.00 ± 0.00** (0.21)	**1.00 ± 0.00** (0.72)	**1.00 ± 0.00** (1.00)
COGS	**1.00 ± 0.00** (0.80)	**1.00 ± 0.00** (0.78)	**1.00 ± 0.00** (0.81)	**1.00 ± 0.00** (0.77)
Math: add_or_sub	**1.00 ± 0.00** (0.89)	**1.00 ± 0.00** (0.94)	**1.00 ± 0.00** (0.91)	**1.00 ± 0.00** (0.97)
Math: place_value	0.80 ± 0.45 (0.12)	**1.00 ± 0.00** (0.20)	-	**1.00 ± 0.00** (0.75)

IID accuracy, (OOD accuracy in parenthesis)

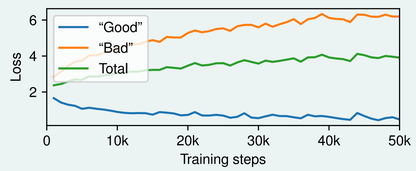
- A particularly interesting case is early stopping on COGS



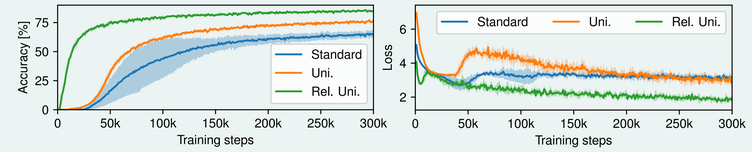
- Validation loss and test accuracy can grow together



- But why?
  - Decompose the train set to "*good*" samples which are at least once correctly classified and "*bad*" ones that are not
  - Loss of "*bad*" samples grows faster than it improves for "*good*" ones



- Validate on OOD *accuracy*, not on loss



Effect of embedding scaling

- Different ways to combine token and positional embeddings
  1. Token Embedding Upscaling (TEU) - Vaswani et al. (2017) Xavier initialization for word embeddings, scale them up
  $$\boldsymbol{H}_i = \sqrt{d_{\text{model}}}\boldsymbol{E}_{w_i} + \boldsymbol{P}_i$$
  2. No scaling, initialize word embeddings to N(0,1)
  $$\boldsymbol{H}_i = \boldsymbol{E}_{w_i} + \boldsymbol{P}_i$$
  3. **Position Embedding Downscaling (PED)**. Kaiming initialization for word embeddings.
  $$\boldsymbol{H}_i = \boldsymbol{E}_{w_i} + \frac{1}{\sqrt{d_{\text{model}}}}\boldsymbol{P}_i$$

		IID Validation	Gen. Test
COGS	TEU	**1.00 ± 0.00**	0.78 ± 0.03
	No scaling	**1.00 ± 0.00**	0.62 ± 0.06
	PED	**1.00 ± 0.00**	**0.80 ± 0.00**
PCFG	TEU	0.92 ± 0.07	0.47 ± 0.27
	No scaling	**0.97 ± 0.01**	0.63 ± 0.02
	PED	0.96 ± 0.01	**0.65 ± 0.03**

Putting them together

- Use layer sharing (Universal Transformers)
- Use relative positional encodings
- Use OOD validation set
- Be careful with early stopping
- Embedding scaling is important

Results

- Revisiting details which are often overlooked in the standard IID tasks helped a lot!
- We obtain very large improvements over existing baselines!

	Trafo	Uni. Trafo	Rel. Trafo	Rel. Uni. Trafo	Prior Work
SCAN (length cutoff=26)	0.30 ± 0.02	0.21 ± 0.01	0.72 ± 0.21	**1.00 ± 0.00**	0.00[1]
CFQ Output length	0.57 ± 0.00	0.77 ± 0.02	0.64 ± 0.06	**0.81 ± 0.01**	∼ 0.66[2]
CFQ MCD 1	**0.40 ± 0.01**	0.39 ± 0.03	0.39 ± 0.01	0.39 ± 0.04	0.37 ± 0.02[3]
CFQ MCD 2	0.10 ± 0.01	0.09 ± 0.02	0.09 ± 0.01	**0.10 ± 0.02**	0.08 ± 0.02[3]
CFQ MCD 3	0.11 ± 0.00	0.11 ± 0.01	0.11 ± 0.01	**0.11 ± 0.03**	0.11 ± 0.00[3]
CFQ MCD mean	0.20 ± 0.14	0.20 ± 0.14	0.20 ± 0.14	**0.20 ± 0.14**	0.19 ± 0.01[2]
PCFG Productivity split	0.65 ± 0.03	0.78 ± 0.01	-	**0.85 ± 0.01**	0.50 ± 0.02[4]
PCFG Systematicity split	0.87 ± 0.01	0.93 ± 0.01	0.89 ± 0.02	**0.96 ± 0.01**	0.72 ± 0.00[4]
COGS	0.80 ± 0.00	0.78 ± 0.03	**0.81 ± 0.01**	0.77 ± 0.01	0.35 ± 0.06[5]
Math: add_or_sub	0.89 ± 0.01	0.94 ± 0.01	0.91 ± 0.03	**0.97 ± 0.01**	∼ 0.91[6]*
Math: place_value	0.12 ± 0.07	0.20 ± 0.02	-	**0.75 ± 0.10**	∼ 0.69[6]*