# Systematic Generalization with Edge Transformers

**Anonymous Author(s)**
Affiliation
Address
email

## Abstract

Recent research suggests that systematic generalization in language understanding remains a challenge for state-of-the-art neural models such as Transformers and Graph Neural Networks. To tackle this challenge, we propose the Edge Transformer, a new model that combines inspiration from Transformers and rule-based symbolic AI. The first key idea in Edge Transformers is to associate vector states with every edge, that is, with every pair of input nodes as opposed to just every node as is done in the Transformer model. The second major innovation is a *triangular attention* mechanism that updates edge representations in a way that is inspired by the concept of term unification from logic programming. We evaluate Edge Transformer on compositional generalization benchmarks in relational reasoning and dependency parsing. In both settings, the Edge Transformer outperforms Relation-aware Transformer and classical Transformer baselines.

## 1 Introduction

Transformers (Vaswani et al., 2017) have become ubiquitous in natural language processing and deep learning in general (e.g., Devlin et al., 2018; Raffel et al., 2020; Carion et al., 2020). Nevertheless, systematic (or compositional) generalization benchmarks remain challenging for instances of this class of models, including even large, pretrained ones (Keysers et al., 2020; Tsarkov et al., 2020; Gontier et al., 2020; Furrer et al., 2020). Similarly, despite their increasing application to a variety of reasoning and inference problems, Graph Neural Networks' (GNN) (Gori et al., 2005; Scarselli et al., 2009; Veličković et al., 2018) ability to generalize systematically has also been recently questioned (Sinha et al., 2019, 2020). These challenges remain a concern for building systems capable of robust language understanding and able to reason from available knowledge.

In this work, inspired by symbolic AI, we seek to equip Transformers with additional representational and computational capacity to better capture these kinds of information processing. Our intuition is that classical Transformers can be interpreted as a kind of inferential system similar to a continuous relaxation of a subset of the Prolog logic programming language. In particular, as we discuss in the next section, Transformers can be seen as implementing a kind of reasoning over *properties* of entities—for example, RED$(\cdot)$ or TABLE$(\cdot)$—where any *relations* required for such reasoning—for example PART-OF$(\cdot, \cdot)$ or GRANDMOTHER$(\cdot, \cdot)$—are inferred on the fly with the self-attention mechanism.

Building on this intuition, in this work we propose *Edge Transformers*, a generalization of the Transformer model that uses a novel *triangular attention* mechanism that is inspired by a much more general family of inferential rules.

To achieve this, we endow the Edge Transformer with a 3D tensor state such that every *edge*, that is, every pair of input nodes, contains a vector that represents relations between the two nodes. The updates of each edge are computed using all adjacent edges in a way that is directly inspired by term unification in logic programming. While the use of edge features or even dynamic edge states can

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

be found in the Transformer and GNN literatures (Shaw et al., 2018; Gilmer et al., 2017; Gong and Cheng, 2019), to the best of our knowledge such triangular updates are novel to our approach.

We evaluate the Edge Transformers on two recently proposed compositional generalization challenges. First, on the graph version of the CLUTRR relational reasoning challenge (Sinha et al., 2019), our model displays stronger systematic generalization than Graph Attention Networks (Veličković et al., 2018) and Relation-aware Transformers (Shaw et al., 2018). Second, on a dependency parsing version (Anonymous, 2021) of the Compositional Freebase Questions benchmark (Keysers et al., 2020) our model achieves a higher parsing accuracy than that of classical Transformers and BiLSTMs.

## 2 Intuitions

A transformer operates over a set of $n$ entities—such as subword tokens in a sentence—which we represent as *nodes* in a graph like those displayed on the left-hand side of Figure 1. Each node is associated with a $d$-dimensional *node-state* $X(i)$ for $i \in n$, which can be thought of as a representation of the node, metadata about the node, or output associated with applying some predicate or function to the node—in other words, a *properties* of the node. In transformers, computation proceeds by layer and each node is sequentially associated with $l$ node-states, with each such property/node-state updated from node states at the preceding layer via the attention mechanism.

Adopting a Prolog-like notation, we could write the fundamental inferential process implemented by the transformer architecture as

$$X^{l+1}(1) \vdash_{\mathcal{A}} X^l(1), X^l(2), \ldots, X^l(n). \tag{1}$$

In logic programming, the turnstile symbol $\vdash_{\mathcal{A}}$ means that whenever the right-hand side of the expression is true, I can deduce the left-hand side. In our interpretation of Transformers, the inference rules expressed by $\vdash_{\mathcal{A}}$ are learned via the attention mechanism $\mathcal{A}$, as is the meaning of each property $X^l(\cdot)$. Classical transformers can therefore be interpreted as an architecture specialized to learning how entity properties—for example, $\textsc{solid}(\cdot)$—can be inferred from the properties of other entities. To this end a transformer would have to first extract *relations* between entities, for example to infer $\textsc{solid}(1)$ from $\textsc{solid}(2)$ using the rule $\textsc{solid}(x) \vdash_{\mathcal{A}} \textsc{part-of}(x, y), \textsc{solid}(y)$ the attention mechanism would need to reconstruct the relation $\textsc{part-of}(1, 2)$ from the properties of the nodes 1 and 2.

Despite the power of this mechanism, it still has notable limits. In particular, for many practical reasoning and NLU tasks systems must learn dependencies between relations, such as family relations: $\textsc{mother}(x, y)$ and $\textsc{mother}(y, z)$ implies $\textsc{grandmother}(x, z)$. Such a general reasoning problem might be expressed in Prolog-like notation as follows.

$$X^{l+1}(1, 2) \vdash_{\mathcal{A}} X^l(1, 1), X^l(1, 2), \ldots, X^l(2, 1), X^l(2, 2), \ldots. \tag{2}$$

It is of course possible for classical transformers to capture such reasoning patterns. But for them to do so, each transformer state $X^{l+1}(1)$ must encode **both** the features or properties of the relation itself, that is, $\textsc{mother}(\cdot, \cdot)$ versus $\textsc{grandmother}(\cdot, \cdot)$ versus $\textsc{part-of}(\cdot)$—as well as all target nodes $x$ with which node 1 stands in the relation, that is $\textsc{mother}(1, x)$, $\textsc{part-of}(1, x)$). In other words, to encode relations a classical transformer must use its state representations for two distinct purposes. This mixing of concerns places a high burden on the learning of transformer representations, and we hypothesize that the resulting inductive bias hinders the ability of this class of models to generalize systematically.

To address this problem we propose that the fundamental object represented by the model should not be a state associated with a node but, rather, states associated with edges, like those displayed on the right of Figure 1. Such edge representations in turn are updated based on an attention mechanism which attends to other edges. This mechanism is inspired by the process of term unification in logic programming. Critical to an inference rule such as $\textsc{grandmother}(x, y) \vdash_{\mathcal{A}} \textsc{mother}(x, z)\textsc{mother}(z, y)$ is the fact that the two terms on the right hand side of this rule share a variable $z$. Successfully inferring $\textsc{grandmother}(x, y)$ which involve finding a binding for $z$ which

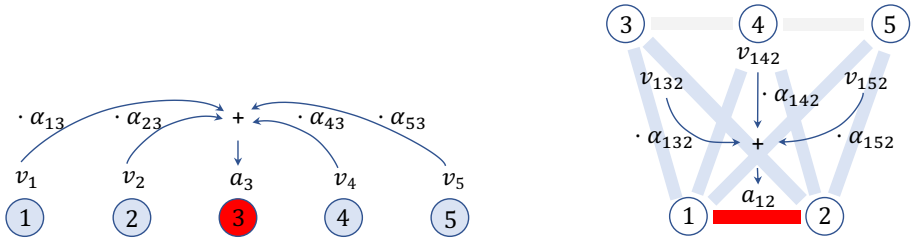


Figure 1: Left: transformer self-attention computes an update $a_3$ for the node 3 by mixing value vectors $v_i$ associated with each node $i$ with attention weights $\alpha_i$. Right: triangular attention computes an update $a_{12}$ for the edge $(1, 2)$ by mixing value vectors $v_{1l2}$ computed for each triangle $(1, l, 2)$ with attention weights $\alpha_{1l2}$. Values $v_3$, $v_{112}$, $v_{122}$ and their contributions are not shown for simplicity.

satisfies both $\textsc{mother}(z, y)$ and $\textsc{mother}(x, z)$. We believe that this is a fundamental aspect of relational reasoning and to capture this we make use of a form of attention which we call *triangular attention* and which we further describe in the next section.

## 3 Edge Transformers

An edge transformer operates on a complete graph with $n$ nodes and $n^2$ directed edges (we allow self-connections) labeled with $d$-dimensional feature vectors. An edge transformer state therefore is a 3-dimensional tensor $X \in \mathbb{R}^{n,n,d}$ that consist of edges states $X(i, j)$ corresponding to every edge $(i, j)$, $i, j \in [1, n]$. We will also write $x_{ij}$ for each edge state. An edge transformer layer computes

$$X' = \mathrm{FFN}(\textsc{ln}(X + \mathrm{TriangularAttention}(\textsc{ln}(X)))). \tag{3}$$

Here, $\mathrm{FFN}(X)$ is a fully-connected feed-forward network with one hidden layer of $d$ units that is applied independently to each edge state $x_{ij}$, $\textsc{ln}$ stands for the layer normalization mechanism (Ba et al., 2016) which rescales activations of each feature $f \in [1, d]$ across its occurrences in edge states $x_{ij}$. Equation 3 follows the familiar structure of transformer (Vaswani et al., 2017) computations with two important differences: (a) It uses a 3D-tensor state instead of a matrix state; and (b) it makes use of a novel triangular attention mechanism that we describe below.

For a single edge state $x_{ij}$ a single-head triangular attention update outputs a vector $a_{ij}$ that is computed as follows:

$$a_{ij} = W^o \sum_{l \in [1,n]} \alpha_{ilj} v_{ilj}, \tag{4}$$

$$\alpha_{ilj} = \underset{l \in [1,n]}{\mathrm{softmax}}\, q_{il} k_{lj} / \sqrt{d}, \tag{5}$$

$$q_{il} = W^q x_{il}, \tag{6}$$

$$k_{lj} = W^k x_{lj}, \tag{7}$$

$$v_{ilj} = V^1 x_{il} \odot V^2 x_{lj}, \tag{8}$$

where $\odot$ stands for elementwise multiplication, and $W^q, W^k, W^o, V^1, V^2 \in \mathbb{R}^{d,d}$ are matrices that are used to produce key, query, output and value vectors $k_{il}$, $q_{lj}$, $a_{ij}$ and $v_{ilj}$ respectively. Here and in the rest of the paper we omit bias terms for clarity.

Informally, updates associated with an edge $(i, j)$ proceed by aggregating information across all pairs of edges that share a node $l$, that is, all pairs $(i, l)$ and $(l, j)$. The updated edge value, $a_{ij}$, is an attention-weighted mixture of contributions from each such pair of edges. Figure 1 visualizes some of the key differences between transformer and edge transformer computations. Note that edges $(i, j)$, $(i, l)$, and $(l, j)$ form a triangle in the figure—hence the name of our attention mechanism.

We define a multi-head generalization of the above mechanism in a way similar to how multi-head attention is implemented in transformers. Specifically, each head $h \in [1, m]$ will perform Edge

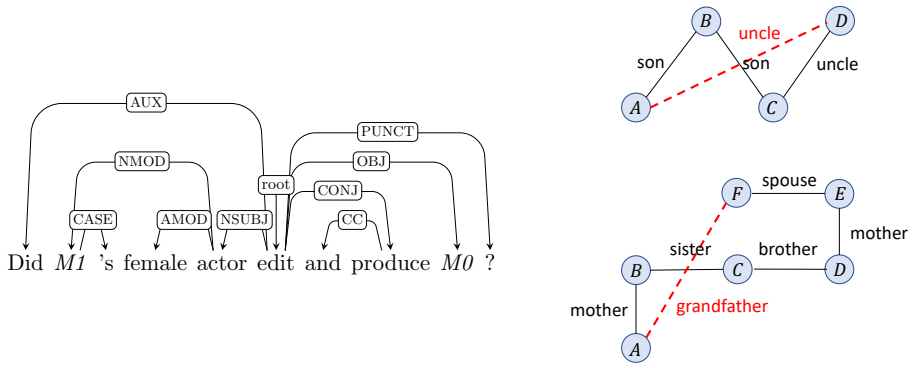


Figure 2: Illustrations of the benchmarks that we use to evaluate the Edge Transformer. **Left:** Dependency parsing annotations for Compositional Freebase Questions (figure reproduced with the permission of Anonymous (2021)). **Right:** Relational reasoning scenarios for the CLUTRR benchmark, with the relation length of 3 (top) and 6 (bottom).

Attention using the smaller corresponding matrices $W^q, W^k, V^1, V^2 \in \mathbb{R}^{d,d/m}$, and the joint output $a_{ij}$ will be computed by multiplying a concatenation of the heads' output by $W^o$:

$$a_{ij} = W^o \left[a_{ij}^1; \ldots; a_{ij}^m\right]. \tag{9}$$

In practice a convenient way to implement an efficient batched version of edge attention is using the Einstein summation operation which is readily available in modern deep learning frameworks. We present a compact implementation pseudo-code in supplementary material.

**Tied & Untied Edge Transformer** Edge Transformer layers can be stacked with or without tying weights across layers. By default we tie the weights in a way similar to universal transformers (Dehghani et al., 2019). We will refer to the version with weights untied as Untied Edge Transformers.

**Input & Output** The Edge Transformer's initial state $X^0$ is defined differently for different applications. When the input is a labeled graph with (possibly null) edge labels, the initial state $x_{ij}^0$ is produced by embedding the labels with a trainable embedding layer. When the input is a sequence, such as for example a sequence of words or tokens $t_i$, $x_{ij}^0$ is computed by adding contributions from tokens $i$ and $j$ as follows:

$$x_{ij}^0 = U^l(e(t_i) + \mathrm{pos}_i) + U^r(e(t_j) + \mathrm{pos}_j), \tag{10}$$

where $U^l, U^r \in \mathbb{R}^{d,d}$ are matrices, $e$ is a trainable $d$-dimensional embedding layer, $\mathrm{pos}_i$ is a sine-wave-based positional embeddings of the position $i$.

To produce a graph-structured output with edge transformers, a linear layer with cross-entropy loss can be used for every edge $(i, j)$. For classification problems with a single label graph the labels can be associated with one of the edges, or edge representations can be pooled.

## 4 Experiments

In our experiments we compare systematic generalization capabilities of Edge Transformers to that of Transformers (Vaswani et al., 2017), Relation-aware Transformers (Dehghani et al., 2019) and Graph Attention Networks (Veličković et al., 2018). We focus on two synthetic benchmarks with carefully controlled train-test splits, Compositional Language Understanding and Text-based Relational Reasoning (CLUTRR), proposed by Sinha et al. (2019) and Compositional Freebase Questions (CFQ) proposed by Keysers et al. (2020). Systematic generalization refers to the ability to recombine known primitive units in novel combinations. Both benchmarks match test and train in terms of primitive atoms used and rules of combination, but test on novel combinations of atoms not seen in train. These benchmarks are thus appropriate to assess if the Edge Transformer has a stronger inductive bias to generalize systematically than existing models.

Table 1: Hyperparameter settings for the Edge Transformer and for the baselines. $L$ is the number of layers, $d$ is the number of units, $h$ is the number of heads, $B$ is the batch size, $\rho$ is the learning rate.

| Model | Task | $L$ | $d$ | $h$ | $B$ | $\rho$ |
|---|---|---|---|---|---|---|
| Edge Transformer | CLUTRR | 8 | 200 | 4 | 120 | $5 \cdot 10^{-4}$ |
| Relation-aware Transformer | CLUTRR | 5 | 200 | 4 | 160 | $1 \cdot 10^{-3}$ |
| Edge Transformer | CFQ | 6 | 256 | 8 | $10^4$ words | $1 \cdot 10^{-3}$ |
| Transformer | CFQ | 7 | 256 | 4 | $5 \cdot 10^3$ words | $5 \cdot 10^{-4}$ |

The experiments were performed using a cluster of 6 GPUs (2 24GB, 2 12GB, 2 11GB). A single training run on CFQ or CLUTRR requires less than 30 minutes on a single GPU. Hyperparameters were selected using grid search. For both the Edge Transformer and (Relation-aware) Transformer models, hyperparameter search was the same, with the exception of number of layers and number of heads. The number of units were varied between 200-400; batch size between 40-160 (for CLUTRR) and $5 \cdot 10^3 - 1 \cdot 10^4$ words (for CFQ); learning rate between $1 \cdot 10^{-4} - 2 \cdot 10^{-3}$. For Transformer models, heads were varied between 2-8 and layers were varied from 4-10. For Edge Transformer models, layers were varied from 4-8 and heads between 1-8.

### 4.1 Relational Reasoning on Graphs

In the first round of experiments we use the CLUTRR benchmark proposed by Sinha et al. (2019). CLUTRR evaluates model's ability to infer new family relations from known ones. For example, if one knows that A is a son of B, C is a son of B and C has an uncle called D, one can infer that D must also be the uncle of A (see Figure 2, top-right). The authors propose to train models on scenarios that require a small number (2, 3 or 4) of supporting relations to infer the target relation and test on scenarios where more (up to 10) supporting relations are required. We will refer to the number of inferential steps required to prove the target relation as the *relation length* $k$. We train models on the two CLUTRR training sets from Sinha et al. (2019); the first contains 10K examples of lengths $k = 2, 3$ and the second contains 15K examples of lengths $k = 2, 3, 4$.

We use the noiseless (only the required facts are included) graph-based version of the challenge, where the input is represented as a labeled graph, and the model's task is to label a missing edge in this graph. We create the initial state $X^0$ by embedding edge labels. We apply a linear layer to the final representation $x_{ij}^L$ of the query edge $(i, j)$ in order to produce logits for the cross-entropy loss. Our main baseline is a Relation-aware Transformer (Shaw et al., 2018), a Transformer variant that can be conditioned on an arbitrary graph. We also include the Graph Attention Networks (GAT, Veličković et al. (2018)) results from the original CLUTRR paper for comparison.

Figure 3 displays the results. One can see the Edge Transformer beats both baselines with a wide margin, in both $k = 2, 3$ and $k = 2, 3, 4$ training settings.

### 4.2 Dependency Parsing

In our second set of experiments we use the dependency parsing version of the CFQ challenge (Keysers et al., 2020) recently proposed by Anonymous (2021). The original CFQ challenge measures compositional generalization in semantic parsing—the task of translating natural language questions into corresponding knowledge graph queries. We use the dependency parsing task because dependency parses are graph structured and, thus, they are easy to predict with Edge Transformers (see Figure 2, left).

The CFQ benchmark was constructed by generating test/train splits which minimize the difference between the test and train distributions over primitive units, like words, while maximizing the *compound divergence* (CD)—the dissimilarity between test and train distributions over larger structures, like phrases. As a result, at test time, a model has to generalize to novel syntactic constructions, such as "Was NP NP?" ("Was Alice the sister of Bob?") that were not seen during training. Keysers et al. (2020) released three different maximum compound divergence (MCD) splits—MCD1, MCD2, and MCD3—which we report performance on below.

Table 2: Dependency parsing accuracy on Compositional Freebase Queries (CFQ). EM is Exact Match accuracy, LAS is Labeled Attachment Score. BiLSTM results are from Anonymous (2021). Transformer and Edge Transformer results are averaged over 20 runs.

| Model | MCD1 | | MCD2 | | MCD3 | |
|---|---|---|---|---|---|---|
| | EM | LAS | EM | LAS | EM | LAS |
| BiLSTM | 96.7 $\pm$ 0.1 | **99.7** $\pm$ 0.5 | 70.3 $\pm$ 1.4 | 92.8 $\pm$ 0.3 | 56.3 $\pm$ 0.8 | 90.2 $\pm$ 0.6 |
| Transformer | 83.0 $\pm$ 0.8 | 98.1 $\pm$ 0.1 | 70.7 $\pm$ 1.1 | 93.2 $\pm$ 0.2 | 57.3 $\pm$ 1.1 | 90.8 $\pm$ 0.2 |
| Edge Transformer | **97.6** $\pm$ 0.3 | 99.6 $\pm$ 0.1 | **83.7** $\pm$ 0.6 | **95.9** $\pm$ 0.1 | **81.1** $\pm$ 0.7 | **95.2** $\pm$ 0.2 |

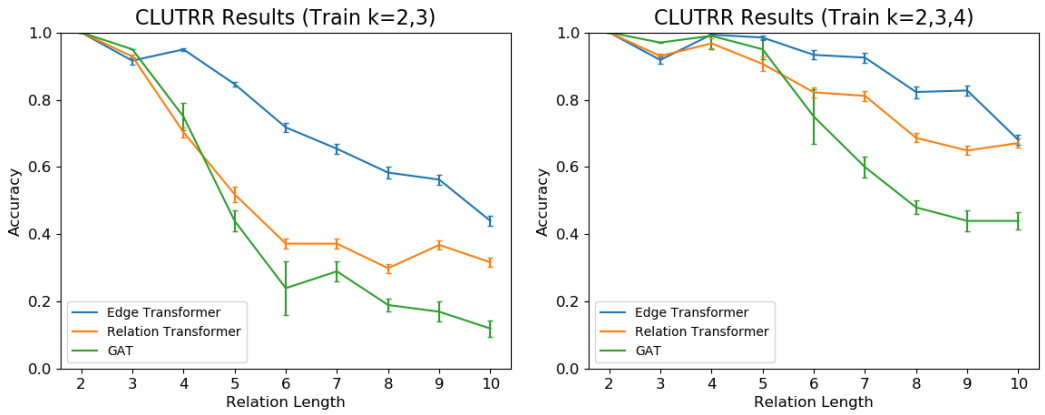


Figure 3: Relation prediction accuracy on CLUTRR for test relations of lengths $k \in [2, 10]$. GAT results from Sinha et al. (2019). Edge and Relation-aware Transformer results are averaged over 20 runs.

We use the Stanza framework for depending parsing (Qi et al., 2020) in which we replace the default BiLSTM model with either Transformer or Edge Transformer variants. The models are trained to label each pair of positions $i$ and $j$ with a dependency type label or a null label. Their predictions are then refined by Stanza's spanning-tree-based parsing algorithm. The model's initial state $X^0$ is initialized from token and positional embeddings as described by Equation 10. The training set contains approximately 100K questions.

We report the Labeled Attachment Score (LAS) that assesses partial correctness (F1) of the predicted dependency trees as well as the exact match accuracy (EM) that only takes into account completely predicted trees. As can be seen in Table 2, Edge Transformer enjoys a sizable performance advantage (12-25% EM accuracy) over both baseline models on the MCD2 and MCD3 splits. On MCD1 split the BiLSTM baseline matches the performance of Edge Transformer, while the classical Transformer lags behind. As discussed in Anonymous (2021), MCD1 is the least challenging split for dependency parsing, exhibiting performance which compares favorably with models trained on a random split and which is near perfect for all models. Since all models are near ceiling performance on this split, it may simply be too easy to see the advantage of one model over another.

### 4.3 Lesion Experiments

To validate our intuitions about why Edge Transformers perform better, we experiment with several variations of the model. First, we train Edge Transformers with untied weights, that is using a separate set of weights for each layer, as is typically done with classical Transformers. Second, we simplify the Edge Transformer computations by computing the value $v_{ilj}$ for each triangle $(i, l, j)$ using only the edge $(i, l)$ instead of using both edges $(i, l)$ and $(l, j)$. This is equivalent to replacing Equation 8 with $v_{ilj} = V^1 x_{il}$. We will refer to this condition as *lesion values*. Third and finally, we use a key vector $k_{ij} = W^k x_{ij}$ instead of $k_{lj} = W^k x_{lj}$ for computing attention scores in Equation 5. We call this condition *lesion attention*.

Table 3: Results of lesion experiments on CLUTRR. "Base" stands for the Edge Transformer as described in Section 3. See Section 4 for details on lesioned model variants. Results are averaged over 10 runs.

| | Model | Test k | | | | |
|---|---|---|---|---|---|---|
| | | 6 | 7 | 8 | 9 | 10 |
| Train on k=2,3 | Base | $71.8 \pm 1.4$ | $65.4 \pm 1.5$ | $58.3 \pm 1.8$ | $56.2 \pm 1.4$ | $44.0 \pm 1.5$ |
| | Untied weights | $70.8 \pm 1.3$ | $65.6 \pm 1.0$ | $57.1 \pm 1.5$ | $53.9 \pm 1.3$ | $43.6 \pm 0.9$ |
| | Lesion values | $27.2 \pm 1.5$ | $25.5 \pm 1.6$ | $17.5 \pm 1.3$ | $18.7 \pm 1.3$ | $15.3 \pm 1.4$ |
| | Lesion attention | $54.9 \pm 2.4$ | $47.8 \pm 2.8$ | $34.1 \pm 2.4$ | $33.3 \pm 2.7$ | $31.0 \pm 2.5$ |
| Train on k=2,3,4 | Base | $93.3 \pm 0.6$ | $92.5 \pm 0.8$ | $82.2 \pm 1.8$ | $82.7 \pm 2.5$ | $68.0 \pm 4.1$ |
| | Untied weights | $91.2 \pm 0.8$ | $92.0 \pm 0.7$ | $79.6 \pm 1.4$ | $75.8 \pm 2.1$ | $61.6 \pm 4.3$ |
| | Lesion values | $83.7 \pm 2.1$ | $73.7 \pm 3.0$ | $62.7 \pm 3.9$ | $49.3 \pm 4.9$ | $59.3 \pm 4.7$ |
| | Lesion attention | $93.9 \pm 0.6$ | $88.8 \pm 1.6$ | $81.8 \pm 2.3$ | $78.7 \pm 3.7$ | $68.2 \pm 4.0$ |

Table 4: Results of lesion experiments on Compositional Freebase Questions. "Base" stands for the Edge Transformer as described in Section 3. See Section 4 for details on lesioned model variants. Results are averaged over 10 runs. EM is Exact Match accuracy, LAS is Labeled Attachment Score.

| Model | MCD3 | |
|---|---|---|
| | EM | LAS |
| Base | $81.1 \pm 0.7$ | $95.2 \pm 0.2$ |
| Untied Weights | $80.5 \pm 0.9$ | $94.8 \pm 0.2$ |
| Lesion values | $67.9 \pm 3.8$ | $93.0 \pm 0.5$ |
| Lesion attention | $77.6 \pm 2.0$ | $94.5 \pm 0.3$ |

We compare our base model to the above variants on the CLUTRR task and the MCD3 split of the CFQ task (the most difficult split). The results for are reported in Tables 3 and 4 respectively. Lesioning values consistently leads to worse performance, confirming the importance of the unification-inspired triangular updates described in Equations 5-8 and illustrated in Figure 1. The negative impact of the lesioning attention is smaller than that of lesioning values, with the biggest performance drop on the CLUTRR $k = 2, 3$ settings. Lastly, untying weights has a limited impact on performance, however there is some evidence of deterioration for the CLUTRR setting where we train on $k = 2, 3, 4$.

## 5 Related Work

Most variations of the original Transformer architecture by Vaswani et al. (2017) aim to improve upon its quadratic memory and computation requirements and, thereby, handle larger inputs. To this end, researchers have introduced recurrency (Dai et al., 2019) and experimented with sparse connectivity (Ainslie et al., 2020) and kernel-based approximations of self-attention (Choromanski et al., 2021). Most relevant to this work is the use of relative positional embeddings in Relation-aware Transformers proposed by (Shaw et al., 2018) and refined by (Dai et al., 2019). This approach can be seen as introducing edge representations to Transformers. Unlike the edge states in the Edge Transformer, these edge representations are static; that is, they do not depend on any kind of external input other than the respective edge labels.

Graph Neural Networks (GNNs; Gori et al., 2005; Scarselli et al., 2009; Kipf and Welling, 2017), and especially Graph Attention Networks (Veličković et al., 2018) are closely related to Transformers, especially relation-aware variants. Unlike Transformers, GNNs operate on sparse graphs. While some GNN variants use edge features (Gilmer et al., 2017; Chen et al., 2019) or even compute dynamic edge states (Gong and Cheng, 2019), to the best of our knowledge no GNN architecture updates edge states based on the states of other edges as we do in Equations 5-8.

Research on systematic (or compositional) generalization studies the ability of neural networks to handle inputs consisting of familiar training atoms that are combined in novel ways. Such research is

typically conducted using carefully controlled train-test splits (Lake and Baroni, 2018; Hupkes et al., 2019; Sinha et al., 2019; Bahdanau et al., 2019; Keysers et al., 2020; Kim and Linzen, 2020). Various techniques have been used to improve generalization on some of the these benchmarks, including (but not limited to) data augmentation (Andreas, 2019; Akyürek et al., 2020), program synthesis (Nye et al., 2020), meta-learning (Lake, 2019), and task-specific model design (Guo et al., 2020). Most relevant to our work is the Neural Theorem Proving (NTP) approach proposed by (Rocktäschel and Riedel, 2017) and further developed by (Minervini et al., 2020). NTP can be described as a differentiable version of Prolog backward-chaining inference with vector embeddings for literals and predicate names. Unlike NTP, Edge Transformer does not make explicit rules; we merely provide the networks with an architecture that facilitates the learning of rule-like computations by the attention mechanism. Edge Transformers remain closer to general purpose neural models such as Transformers in spirit and, as demonstrated by our CFQ experiments, can be successfully applied in the contexts where the input is not originally formulated as a knowledge graph.

Lastly, the Inside-Outside autoencoder proposed by Drozdov et al. (2019) for unsupervised constituency parsing is the most similar to our work at the technical level. The Inside-Outside approach also features vector states for every pair of input nodes. However, in contrast to the Edge Transformer, the attention is restricted by the span structure of the input.

## 6 Discussion

We present the Edge Transformer, an instantiation of the idea that a neural model may benefit from an extended 3D-tensor state that more naturally accommodates relational reasoning. Our experimental results are encouraging, with clear performance margins in favor of the Edge Transformer over competitive baselines on two compositional generalization benchmarks. Our lesion experiments confirm the core intuitions underlying the model design.

Edge Transformers can be seen as belonging to the line of work on neuro-symbolic architectures (Mao et al., 2018; Rocktäschel and Riedel, 2017; Evans and Grefenstette, 2018, to name but a few). Compared to most work in the field, Edge Transformers are decidedly more neural: no logical gates or rule templates can be found in our architecture. We believe this is a promising direction for further exploration, as it allows the model to retain the key advantages of neural architectures: their generality, interchangeability and ability to learn from data.

Edge Transformer computations and memory usage scale as $O(n^3)$ with respect to the input size $n$. While this makes scaling to larger inputs challenging, in our experience modern GPU hardware allows training Edge Transformers with inputs of size up to a hundred entities. One way to use Edge Transformer in applications with larger inputs is to create Edge Transformer nodes only for the most salient or important components of an input. For example, in the language modelling context these could be just the heads of noun phrases and/or just the named entities. Future work could build and pretrain on raw text hybrid models that combine both Transformer- and Edge Transformer-based processing, the latter performed only for a small subset of nodes.

# References


Ainslie, J., Ontanon, S., Alberti, C., Cvicek, V., Fisher, Z., Pham, P., Ravula, A., Sanghai, S., Wang, Q., and Yang, L. (2020). ETC: Encoding Long and Structured Inputs in Transformers. *arXiv:2004.08483 [cs, stat]*. arXiv: 2004.08483.

Akyürek, E., Akyürek, A. F., and Andreas, J. (2020). Learning to Recombine and Resample Data For Compositional Generalization.

Andreas, J. (2019). Good-Enough Compositional Data Augmentation. *arXiv:1904.09545 [cs]*. arXiv: 1904.09545.

Anonymous, A. (2021). *Under Review*.

Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer Normalization. *arXiv:1607.06450 [cs, stat]*. arXiv: 1607.06450.

Bahdanau, D., Murty, S., Noukhovitch, M., Nguyen, T. H., Vries, H. d., and Courville, A. (2019). Systematic Generalization: What Is Required and Can It Be Learned? In *International Conference on Learning Representations, ICLR 2019*.

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., and Zagoruyko, S. (2020). End-to-End Object Detection with Transformers. *arXiv:2005.12872 [cs]*. arXiv: 2005.12872.

Chen, P., Liu, W., Hsieh, C.-Y., Chen, G., and Zhang, S. (2019). Utilizing Edge Features in Graph Neural Networks via Variational Information Maximization. *arXiv:1906.05488 [cs, stat]*. arXiv: 1906.05488.

Choromanski, K., Likhosherstov, V., Dohan, D., Song, X., Gane, A., Sarlos, T., Hawkins, P., Davis, J., Mohiuddin, A., Kaiser, L., Belanger, D., Colwell, L., and Weller, A. (2021). Rethinking Attention with Performers. *arXiv:2009.14794 [cs, stat]*. arXiv: 2009.14794.

Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q. V., and Salakhutdinov, R. (2019). Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context. *arXiv:1901.02860 [cs, stat]*. arXiv: 1901.02860.

Dehghani, M., Gouws, S., Vinyals, O., Uszkoreit, J., and Kaiser, L. (2019). Universal Transformers. *arXiv:1807.03819 [cs, stat]*. arXiv: 1807.03819.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics, NAACL 2019*.

Drozdov, A., Verga, P., Chen, Y.-P., Iyyer, M., and McCallum, A. (2019). Unsupervised Labeled Parsing with Deep Inside-Outside Recursive Autoencoders. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1507–1512, Hong Kong, China. Association for Computational Linguistics.

Evans, R. and Grefenstette, E. (2018). Learning Explanatory Rules from Noisy Data. *arXiv:1711.04574 [cs, math]*. arXiv: 1711.04574.

Furrer, D., van Zee, M., Scales, N., and Schärli, N. (2020). Compositional Generalization in Semantic Parsing: Pre-training vs. Specialized Architectures. *arXiv:2007.08970 [cs]*. arXiv: 2007.08970.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. (2017). Neural Message Passing for Quantum Chemistry. *arXiv:1704.01212 [cs]*. arXiv: 1704.01212.

Gong, L. and Cheng, Q. (2019). Exploiting Edge Features in Graph Neural Networks. *arXiv:1809.02709 [cs, stat]*. arXiv: 1809.02709.

Gontier, N., Sinha, K., Reddy, S., and Pal, C. (2020). Measuring Systematic Generalization in Neural Proof Generation with Transformers. *arXiv:2009.14786 [cs, stat]*. arXiv: 2009.14786.

Gori, M., Monfardini, G., and Scarselli, F. (2005). A new model for learning in graph domains. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pages 729–734 vol. 2. ISSN: 2161-4407.

Guo, Y., Lin, Z., Lou, J.-G., and Zhang, D. (2020). Hierarchical Poset Decoding for Compositional Generalization in Language. *arXiv:2010.07792 [cs]*. arXiv: 2010.07792.

Hupkes, D., Dankers, V., Mul, M., and Bruni, E. (2019). The compositionality of neural networks: integrating symbolism and connectionism. *arXiv:1908.08351 [cs, stat]*. arXiv: 1908.08351.

Keysers, D., Schärli, N., Scales, N., Buisman, H., Furrer, D., Kashubin, S., Momchev, N., Sinopalnikov, D., Stafiniak, L., Tihon, T., Tsarkov, D., Wang, X., van Zee, M., and Bousquet, O. (2020). Measuring Compositional Generalization: A Comprehensive Method on Realistic Data. In *International Conference on Learning Representations*. arXiv: 1912.09713.

Kim, N. and Linzen, T. (2020). COGS: A Compositional Generalization Challenge Based on Semantic Interpretation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9087–9105, Online. Association for Computational Linguistics.

Kipf, T. N. and Welling, M. (2017). Semi-Supervised Classification with Graph Convolutional Networks. *arXiv:1609.02907 [cs, stat]*. arXiv: 1609.02907.

Lake, B. M. (2019). Compositional generalization through meta sequence-to-sequence learning. arXiv: 1906.05381.

Lake, B. M. and Baroni, M. (2018). Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *Proceedings of the 36th International Conference on Machine Learning*. arXiv: 1711.00350.

Mao, J., Gan, C., Kohli, P., Tenenbaum, J. B., and Wu, J. (2018). The Neuro-Symbolic Concept Learner: Interpreting Scenes, Words, and Sentences From Natural Supervision. In *International Conference on Learning Representations, ICLR 2019*.

Minervini, P., Riedel, S., Stenetorp, P., Grefenstette, E., and Rocktäschel, T. (2020). Learning Reasoning Strategies in End-to-End Differentiable Proving. *arXiv:2007.06477 [cs]*. arXiv: 2007.06477.

Nye, M. I., Solar-Lezama, A., Tenenbaum, J. B., and Lake, B. M. (2020). Learning Compositional Rules via Neural Program Synthesis. *arXiv:2003.05562 [cs]*. arXiv: 2003.05562.

Qi, P., Zhang, Y., Zhang, Y., Bolton, J., and Manning, C. D. (2020). Stanza: A Python Natural Language Processing Toolkit for Many Human Languages. *arXiv:2003.07082 [cs]*. arXiv: 2003.07082.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2020). Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Rocktäschel, T. and Riedel, S. (2017). End-to-end Differentiable Proving. *arXiv preprint arXiv:1705.11040*.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. (2009). The Graph Neural Network Model. *IEEE Transactions on Neural Networks*, 20(1):61–80. Conference Name: IEEE Transactions on Neural Networks.

Shaw, P., Uszkoreit, J., and Vaswani, A. (2018). Self-Attention with Relative Position Representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 464–468, New Orleans, Louisiana. Association for Computational Linguistics.

Sinha, K., Sodhani, S., Dong, J., Pineau, J., and Hamilton, W. L. (2019). CLUTRR: A Diagnostic Benchmark for Inductive Reasoning from Text. *arXiv:1908.06177 [cs, stat]*. arXiv: 1908.06177.

Sinha, K., Sodhani, S., Pineau, J., and Hamilton, W. L. (2020). GraphLog: A Benchmark for Measuring Logical Generalization in Graph Neural Networks.

Tsarkov, D., Tihon, T., Scales, N., Momchev, N., Sinopalnikov, D., and Schärli, N. (2020). *-CFQ: Analyzing the Scalability of Machine Learning on a Compositional Task. *arXiv:2012.08266 [cs]*. arXiv: 2012.08266.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention Is All You Need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017*. arXiv: 1706.03762.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. (2018). Graph Attention Networks. *arXiv:1710.10903 [cs, stat]*. arXiv: 1710.10903.

## Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 6.
   (c) Did you discuss any potential negative societal impacts of your work? [No]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [No]
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]