Dependency Parsing is More Parameter-Efficient with Normalization

Paolo Gajo
University of Bologna
paolo.gajo2@unibo.it

Domenic Rosati
Dalhousie University
Domenic.Rosati@Dal.Ca

Hassan Sajjad
Dalhousie University
HSajjad@dal.ca

Alberto Barrón-Cedeño
University of Bologna
a.barron@unibo.it

Abstract

Dependency parsing is the task of inferring natural language structure, often approached by modeling word interactions via attention through biaffine scoring. This mechanism works like self-attention in Transformers, where scores are calculated for every pair of words in a sentence. However, unlike Transformer attention, biaffine scoring does not use normalization prior to taking the softmax of the scores. In this paper, we provide theoretical evidence and empirical results revealing that a lack of normalization necessarily results in overparameterized parser models, where the extra parameters compensate for the sharp softmax outputs produced by high variance inputs to the biaffine scoring function. We argue that biaffine scoring can be made substantially more efficient by performing score normalization. We conduct experiments on six datasets for semantic and syntactic dependency parsing using a one-hop parser. We train N-layer stacked BiLSTMs and evaluate the parser's performance with and without normalizing biaffine scores. Normalizing allows us to achieve state-of-the-art performance with fewer samples and trainable parameters. Code: `https://anonymous.4open.science/r/EfficientSDP-70C1`

1 Introduction

Dependency parsing (DP) consists in classifying node labels t_i (words), edges e_{ij} (relations), and edge labels r_{ij} (relation types) of a dependency graph [25]. A popular model for this task, introduced by Dozat and Manning [7], entails modeling word interactions as a fully connected graph via biaffine attention. Despite its simplicity, a number of models using this biaffine transformation [7, 8, 11, 12, 3] require more parameters than necessary, due to what we identify as a lack of normalization of its outputs. We propose that this overparameterization is caused by the high variance of the outputs, which extra parameters help mitigate. After showing that variance can be reduced through normalization, we demonstrate that similar or better performance can be obtained for DP with fewer parameters and training samples.

In this task, we consider a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ comprising a set of nodes $\mathcal{V}$, connected by a set of edges $\mathcal{E}$, with each edge $e_{ij} \in \mathcal{E}$ connecting pairs of nodes $(v_i, v_j) \in \mathcal{V}$. In latent graph inference for dependency graphs, a sentence of $|\mathcal{V}|$ words is modeled as a $|\mathcal{V}| \times |\mathcal{V}|$ graph via biaffine scoring [7]:

$$XWX^\top = X(W_Q W_K^\top)X^\top = XW_Q(XW_K)^\top = QK^\top, \quad X \in \mathbb{R}^{|\mathcal{V}| \times d}.$$

Observe that this is equivalent to the unnormalized scores used in [31]'s self-attention: $QK^\top/\sqrt{d_k}$ where Attention$(Q, K, V) = $ Softmax $\left(QK^\top/\sqrt{d_k}\right) V$ and d_k is the output size of the keys.

Preprint. Under review.

The main difference resides in the fact that Transformer attention scores are scaled by $a = 1/\sqrt{d_k}$. As explained in [31], this scaling is done because high variance inputs to the softmax function will result in large values dominating the output ($e^{x \gg 1}$) and small values decaying ($e^{x \ll 1}$). The downstream effect is exploding and vanishing gradients. To see how this works, observe that the score s between any query and key vector is the result of their dot product $q_i k_i^\top$. Now *if* we assume that these vectors are zero mean and unit variance, then the variance is $\text{Var}(s) = d_k$. Finally, we get our normalization factor, which ensures that each entry in the score matrix has a standard deviation of 1, since $\text{Std} = \sqrt{d_k}$. Consequently, lower input variance will result in more stable outputs.

We observe that there is no consistency in the literature on the use of this scaling term for DP. Most works do not use this scaling term [7, 8, 11, 3, 8, 12], with the exception of [30] who use [31]'s self-attention out of the box. In this paper, we carry out a series of experiments on semantic and syntactic DP tasks, using a wide range of architecture ablations. We extend the work of Bhatt et al. [3], which represents the state of the art on dependency graph parsing in NLP.

Following the state of the art, we train stacked BiLSTMs and a biaffine classifier to infer the latent dependency graph connecting the words of a sentence. We find that, when not normalizing the scores produced by the biaffine transformation, model performance drops in terms of micro-averaged F_1-measure and attachment score [20]. In particular, increasing the amount of layers produces a normalization effect by reducing the variance of the output scores. Using score normalization, we find that in some cases similar or better performance can be obtained by reducing the amount of trained BiLSTM parameters by as much as 85%.

Our contributions are thus three-fold. (*i*) We show the impact of normalizing the output of the biaffine scorer in relation to the architectural changes in a DP model. (*ii*) We show DP models can obtain better performance with substantially fewer trained parameters. (*iii*) We provide a new method that substantially improves scores and obtains state-of-the-art performance for semantic [17, 36] and syntactic [19, 38] dependency parsing.

The rest of the paper is structured as follows. Section 2 presents an overview of the literature concerning DP, with specific focus on how to infer the adjacency matrix of a dependency graph. Section 3 explains why layer depth can compensate for normalization. Section 4 provides an overview of the datasets, models, evaluation, and other experimental details. Section 5 illustrates how normalization mitigates overparameterization. Finally, Section 6 provides avenues for future research, such as applications of our work to more complicated graph neural network-based approaches.

2 Background

In general, DP is an NLP task concerned with inferring the dependency graph of a sentence [25]. Depending on the nature of the nodes and relations of the graph, it can e.g., take on the name of semantic (SemDP) [8, 30, 3, 12] or syntactic (SynDP) [7, 38, 11]. In this work, we tackle both tasks, but focus more specifically on SemDP, which can be thought of as comprising two sub-tasks: Named Entity Recognition (NER) [18] and Relation Extraction (RE) [3, 12]. They can be either approached in a pipeline as separate objectives [5, 37, 40, 43], or by jointly training a single model on both sub-tasks [3, 4, 12, 39]. In the context of deep learning models trained end-to-end, three main paradigms are used: encoder-based models [3, 7, 8, 11, 12, 32, 33], decoder-only Transformers (LLMs) [4, 34, 41], and seq2seq encoder-decoder Transformers [14, 16, 21, 39].

In this work, we focus on encoder-style models, since they currently achieve the best performance on DP tasks [30, 12, 3] and are much more parameter efficient than LLM-based solutions. These models approach entity (node) prediction analogously to NER, while edges and relations are handled via MLP projection of node-pair features [13, 22, 23, 44], or via attention-based edge and relation inference [3, 8, 11, 12, 30]. Current state-of-the-art models, exemplified by [3], are based on the biaffine dependency parser introduced by [7], which uses stacked BiLSTM layers on top of embeddings from a pre-trained language model.

3 Layer depth can compensate for normalization

In order to reduce the number of parameters used for biaffine scoring, we observe that the softmax function is sensitive to inputs with high variance: large values dominate the output ($e^{x \gg 1}$) and small

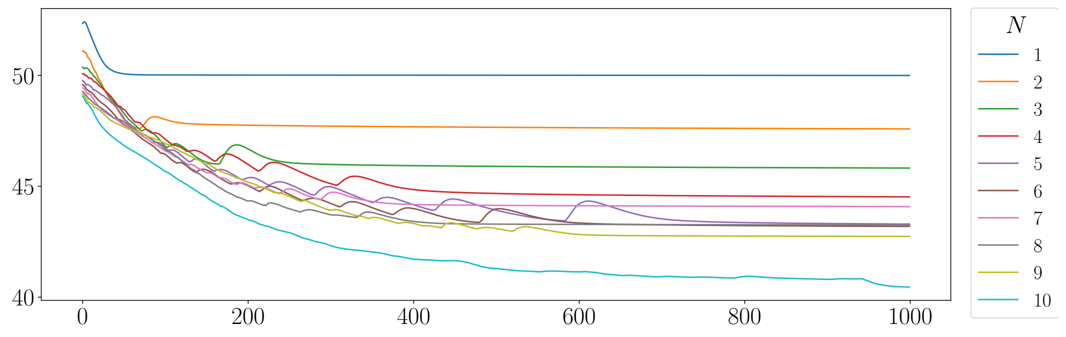


Figure 1: Effective rank $\rho(W)$ reduces over training epochs as we increase N of BiLSTM layers.

values decay ($e^{x \ll 1}$). This causes a drop in the downstream task performance, due to some values dominating the probability outputs in the score matrix, as well as exploding and vanishing gradients. Typically, contemporary architectures based on [7] do not employ normalization, but are still able to perform well on DP tasks.

We explain this discrepancy using insights from the theory of implicit regularization [1], which states that, as layer depth increases, the effective rank of the weight matrices is reduced during gradient descent. Our claim is that a reduction in the rank of the weight matrices causes a corresponding decrease in the variance of the input to the softmax function.

Result 1 (Singular Value Dynamics under Gradient Descent [1])**.** *Minimization of a loss function $\mathcal{L}(W)$ with gradient descent using weight matrices* $\mathbf{W}$ *(assuming a small learning rate η and initialization close to the origin). An N-layer linear neural network leads the singular values of* $\mathbf{W}$ *to evolve in the number of iterations t by:*

$$\sigma_r(t+1) \leftarrow \sigma_r t - \eta \cdot \langle \nabla \mathcal{L}(W(t)), \mathbf{u}_r(t)\mathbf{v_r}^\top(t) \rangle \cdot N \cdot \sigma_r(t)^{2-2/N}$$

This implies that, as the number of layers increases, the rank of the weight matrices decreases, since the smallest singular values decay.

While Result 1 is stated for deep linear neural networks for the matrix factorization task, it has been validated empirically as applying to deep non-linear neural networks [27, 42]. We validate this finding for the biaffine scoring setting in Figure 1, which shows the inverse proportionality between the effective rank $\rho(W)$ [29] of a BiLSTM's weights and the number of its layers.

Claim 1 (Rank variance inequality)**.** *Let $Y_r \in \mathbb{R}^m$ be a random vector that is the outcome of a linear transformation* $\mathbf{y} = \mathbf{A}_r\mathbf{x}$*, where the vector* $\mathbf{x} \in \mathbb{R}^n$ *is drawn from the random vector X and* $\mathbf{A}_r \in \mathbb{R}^{m \times n}$ *denotes a rank r approximation of a fixed matrix* $\mathbf{A}$*. The variance of Y_r is smaller than the variance of Y_{r+1} as measured by the Frobenius norm $||Var(Y_r)||_F \leqslant ||Var(Y_{r+1})||_F$.*

Proof. The variance of a random vector X is represented by the positive semi-definite (PSD) covariance matrix $\mathbf{K_{xx}} \in \mathbb{R}^{n \times n}$. Applying the variance of X to the linear transformation, we get $\text{Var}(Y) = \mathbf{A_r K_{xx} A_r}^\top \in \mathbb{R}^{m \times m}$.

Since $\mathbf{K_{xx}}$ is PSD, so is each term in $\mathbf{x_i}^\top \mathbf{K_{xx} x_i}$, and so the matrix $\mathbf{A_r K_{xx} A_r}^\top$ is also PSD for each rank-r approximation $\mathbf{A}_r = \sum_{i=1}^r \sigma_i \mathbf{u}_i \mathbf{v}_i^\top$.

This means that increasing the rank from $r-1$ to r adds a new PSD component Δ_r:

$$\mathbf{A_r K_{xx} A_r}^\top = \mathbf{A_{r-1} K_{xx} A_{r-1}}^\top + \Delta_r$$

Table 1: Samples per partition and entity/relation classes for the datasets used in this paper.

Data (Train / Dev / Test)		Entities	Relations
ADE 2,563 / 854 / 300	[10]	disease, drug	adverseEffect
CoNLL04 922 / 231 / 288	[28]	organization, person, location	kill, locatedIn, workFor, orgBasedIn, liveIn
SciERC 1,366 / 187 / 397	[17]	generic, material, method, metric, otherSciTerm, task	usedFor, featureOf, hyponymOf, evaluateFor, partOf, compare, conjunction
ERFGC 242 / 29 / 29	[36]	food, tool, duration, quantity, actionByChef, discontAction, actionByFood, actionByTool, foodState, toolState	agent, target, indirectObject, toolComplement, foodComplement, foodEq, foodPartOf, foodSet, toolEq, toolPartOf, actionEq, timingHeadVerb, other
enEWT (UD 2.2) 10,098 / 1,431 / 1,427	[36]	xPOS tags	UD relations
SciDTB 2,567 / 814 / 817	[38]	xPOS tags	UD relations

Since the Frobenius norm is given by $\sqrt{\sum_i^m \sum_j^n |a_{ij}|^2}$, then we must have:

$$\begin{aligned}||\mathbf{A_r K_{xx} A_r}^\top||_F &= ||\mathbf{A_{r-1} K_{xx} A_{r-1}}^\top + \Delta_r||_F \\ &\geqslant ||\mathbf{A_{r-1} K_{xx} A_{r-1}}^\top||_F \\ &= ||\mathbf{A_{r-2} K_{xx} A_{r-2}}^\top + \Delta_{r-1}||_F \\ &\dots \\ &\geqslant ||\mathbf{A_1 K_{xx} A_1}^\top||_F\end{aligned}$$

Which shows that, as the rank decreases, so does the variance of the linear transformation.

□

The implication of Claim 1 is that the pre-softmax score matrix from a shallow network without normalization will have a higher variance than a deeper network because of Result 1. Based on this finding, we propose to remove BiLSTM layers and use normalization —rather than having greater layer depth— in order to develop a more parameter-efficient method.

4 Experimental setting

4.1 Data

To train and evaluate our models, we use four SemDP datasets,[1] along with the 2.2 version of the Universal Dependencies English EWT treebank (enEWT) [19] and SciDTB [38].

As regards SemDP, **ADE** [10] is a medical-domain dataset comprising reports of drug adverse-effect reactions. Each disease is associated to a drug which caused it through the only type of relation present, i.e. "adverseEffect." **CoNLL04** [28] contains news texts and is annotated with the classic entities $e_i \in \{\text{per, org, loc}\}$ and relations $r_j \in \{\text{workFor, kill, orgBasedIn, liveIn, locIn}\}$. ADE and CoNLL04 are characterized by relations which are not complex enough to form connected graphs of considerable size. **SciERC** [17] is a dataset compiled from sentences extracted from literature in the

[1]We neglect ACE04 and ACE05, two popular datasets for relation extraction, due to their prohibitive cost.

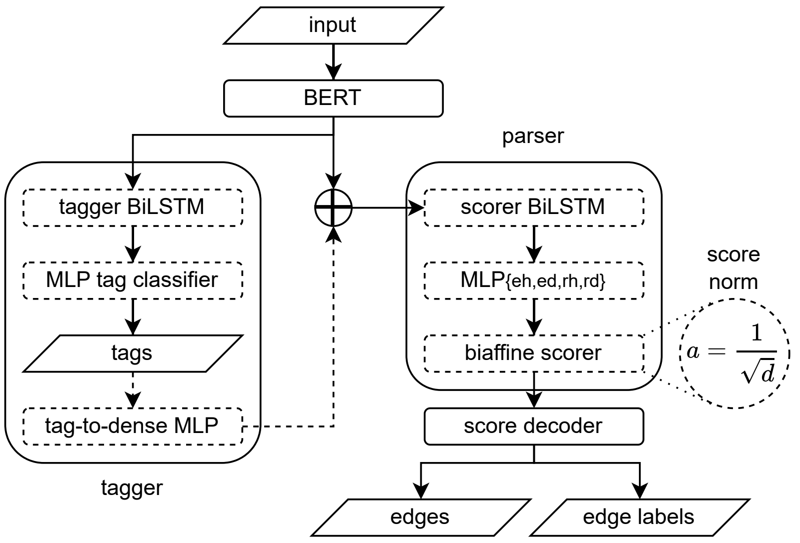


Figure 2: Dependency parsing diagram. Dashed components are the targets of ablation experiments.

domain of artificial intelligence. It is arguably the most challenging dataset used in this study, since most of the entities in the validation and testing partitions are not assigned an entity class. This makes it hard to train tag embeddings that can help to infer edges. **ERFGC** [36] is a dataset comprising "flow graphs" parsed from culinary recipes. The semantic dependency graphs of these recipes are directed and acyclic, with a single root. Note that, differently from [3], and as advised directly by [36] in personal correspondence, we ignore the "-" edge labels present in the corpus.

As regards SynDP, we use **enEWT** [19] to compare directly against [11]'s results. While many works evaluate SynDP on the Penn Treebank [24] it is closed access and prohibitively expensive. Instead, we use **SciDTB** [38], a discourse analysis dataset comprising 798 abstracts extracted from the ACL Anthology. It was processed for the syntax dependency parsing task using Stanza [26]. From these two datasets, we consider the xPOS tags (e.g. noun, preposition, determiner) and the syntactic dependencies (e.g. sentence root, object, adverbial modifier) to respectively be the entities and relations of the dependency graphs to infer.

For ADE, CoNLL04, and SciERC we use the splits provided in [4].[2] ERFGC is not available online; we obtained it by contacting the authors of [36]. Table 1 summarizes the statistics of the datasets used in this work and reports their entity and relation class annotations.

4.2 Model

We adopt the architecture of [3] in our work, schematized in Figure 2. It can be subdivided into four main components: encoder, tagger, parser, and decoder. The input is tokenized and passed through a BERT-like encoder, where token representations are averaged into $|\mathcal{V}|$ word-level features $\mathbf{x}_i \in \mathbb{R}^{d_f}$.[3] Optionally, additional features can be obtained by predicting the entity classes of each word with a tagger, composed by a single-layer BiLSTM ϕ, followed by a classifier:

$$\begin{aligned}\mathbf{h}_i^{tag} &= \phi(\mathbf{x}_i), \quad \mathbf{h}_i^{tag} \in \mathbb{R}^{d_h}\\ \mathbf{y}_i^{tag} &= \text{Softmax}(\text{MLP}^{tag}(\mathbf{h}_i^{tag})), \quad \mathbf{y}_i^{tag} \in \mathbb{R}^{|T|}\end{aligned}$$

where T is the set of word tag classes. The tagger's predictions are then converted into one-hot vectors and projected into dense representations by another MLP, such that $\mathbf{e}_i^{tag} = \text{MLP}^{emb}(\mathbf{1}_T(\mathbf{y}_i^{tag}))$. These new tag embeddings are concatenated with the original BERT output and sent to the parser.

[2] `https://drive.google.com/drive/folders/1vVKJIUzK4hIipfdEGmSOCCoFmUmZwOQV`

[3] Using token-level representations resulted in much lower performance in preliminary experiments.

In the parser, an optional k-layered BiLSTM ψ produces new representations $\mathbf{h}_i = \psi(\mathbf{e}_i^{tag} \oplus \mathbf{x}_i)$, which are then projected into four different representations:

$$\begin{aligned} \mathbf{e}_i^h &= \text{MLP}^{(edge-head)}(\mathbf{h}_i), \ \mathbf{e}_i^d = \text{MLP}^{(edge-dept)}(\mathbf{h}_i) \\ \mathbf{r}_i^h &= \text{MLP}^{(rel-dept)}(\mathbf{h}_i), \quad \mathbf{r}_i^d = \text{MLP}^{(rel-head)}(\mathbf{h}_i) \end{aligned}$$

The edge scores s_i^{edge} and relation scores s_i^{rel} are then calculated with the biaffine function f:

$$f(\mathbf{x}_1, \mathbf{x}_2; W) = \mathbf{x}_1^\top W \mathbf{x}_2 + \mathbf{x}_1^\top \mathbf{b}$$

$$\begin{aligned} s_i^{edge} &= f^{(edge)}(\mathbf{e}_i^h, \mathbf{e}_i^d; W_e), \quad W_e \in \mathbb{R}^{d \times 1 \times d} \\ s_i^{rel} &= f^{(rel)}(\mathbf{r}_i^h, \mathbf{r}_i^d; W_r), \quad W_r \in \mathbb{R}^{d \times |R| \times d} \end{aligned}$$

where R is the set of relation classes, i.e. the possible labels applied to an edge.

Finally, in the decoder, the edge scores are used in conjunction with the relation representations $\mathbf{r}_i^h$ and $\mathbf{r}_i^d$ to obtain the final predictions. During training, we do greedy decoding, while during inference, we use Chu-Liu/Edmonds' maximum spanning tree (MST) algorithm [9] to ensure the predictions are well-formed trees. This is especially useful with big dependency graphs, since greedy decoding is more likely to produce invalid trees as size increases. When doing greedy decoding, an edge index (i.e. an adjacency matrix) $a_i = \arg\max_j s_{ij}^{edge}$ is produced by taking the argmax of the attention scores s_i^{edge} across the last dimension. The edge index is then used to select which head relation representations r_i^h to use to calculate the relation scores $s_i^{rel} = f(\mathbf{r}_i^h, \mathbf{r}_i^d; W), \quad W \in \mathbb{R}^{d \times |R| \times d}$. The relations are then predicted as $r_i = \arg\max_j s_{ij}^{rel}$. When using MST decoding, edge and relation scores are combined into a single energy matrix where each entry represents the score of a specific head-dependent pair with its most likely relation type. This energy matrix is then used in the MST algorithm, producing trees with a single root and no cycles. Prior to energy calculation, edge scores and relation scores are scaled so that low values are squished and high values are increased, making the log softmax produce a hard adjacency matrix.

The model is trained end-to-end jointly on the entity, edge, and relation classification objectives:

$$\begin{aligned} \mathcal{L}_{tag} &= -\frac{1}{|\mathcal{V}|} \sum_{i=1}^{|\mathcal{V}|} \sum_{t=1}^{|T|} y_{i,t}^{tag} \log p\big(y_{i,t}^{tag}\big) \\ \mathcal{L}_{edge} &= -\sum_{i,j=1}^{|\mathcal{V}|} \log p\big(y_{i,j}^{edge} = 1\big) \\ \mathcal{L}_{rel} &= -\sum_{i=1}^{|\mathcal{V}|} \sum_{j=1}^{|\mathcal{V}|} \mathbb{1}\big(y_{i,j}^{edge} = 1\big) \sum_{\ell=1}^{|R|} y_{i,j,\ell}^{rel} \log p\big(y_{i,j,\ell}^{rel}\big) \\ \mathcal{L} &= \lambda_1 \mathcal{L}_{tag} + \lambda_2 \big(\mathcal{L}_{edge} + \mathcal{L}_{rel}\big) \end{aligned}$$

Losses are calculated based on the gold tags, edges, and relations. We set $\lambda_1 = 0.1$ and $\lambda_2 = 1$ as hyperparameters because the tagging task is much simpler than predicting the edges, since the same top performance is always achieved regardless of any other selected architecture hyperparameters. Following the usual approach for SynDP [7, 11, 12], when training on enEWT and SciDTB we use an oracle, the gold tags, and do not predict the POS tags ourselves. Since in this case we only focus on training the edge and relation classification tasks, we set $\lambda_1 = 0$.

4.3 Hyperparameters

We experiment with a range of hyperparameters for the encoder, tagger, and parser, as listed in Table 2. We use BERT$_{base}$ [6] as our pre-trained encoder, which we keep frozen throughout the whole training

Table 2: Hyperparameter ranges tested in our experiments. ψ = Parser BiLSTM. f = biaffine layer.

Component	Hyperparameter	Values
Encoder	Freeze BERT	$\nabla_{\text{BERT}} \in \{\checkmark, \times\}$
Tagger	Tagger BiLSTM ϕ	$\phi \in \{\checkmark, \times\}$
	Concat. tag embeds.	$\mathbf{e}_i^{tag} \in \{\checkmark, \times\}$
Parser	ψ num. layers	$L_\psi \in \{0, 1, 2, 3\}$
	ψ hidden dim.	$h_\psi \in \{100, 200, 300, 400\}$
	MLP$^{(edge)}$ output dim.	$d_{\text{MLP}} \in \{100, 300, 500\}$
	ψ LayerNorm	$\texttt{LN}_\psi \in \{\checkmark, \times\}$
	MLP$^{(edge)}$ and $f^{(edge)}$ init.	$I_{\text{par}} \in \{\mathcal{U}, \mathcal{N}\}$
	Score scaling	$a \in \{1, \frac{1}{\sqrt{d}}\}$

run in our main setting. As regards the tagger, we set $L_\phi = 1$ and $h_\phi = 100$, as in [3], with weights initialized with a Xavier uniform distribution. In Appendix B.1, we ablate the ϕ BiLSTM and the use of the tag embeddings $\mathbf{e}_i^{tag}$ to assess their impact on overall performance. Finally, with relation to the parser, for all hyperparameter combinations of $\{L_\psi, h_\psi, d_{\text{MLP}}\}$, we run our experiments by initializing its weights with a Xavier uniform distribution ($I_{par} = \mathcal{U}$) and no LayerNorm $\texttt{LN}_\psi$. In Appendices B.2 to B.5, we conduct ablations over the parser hyperparameters indicated in Table 2.

We train our models for $2k$ steps and evaluate on the development partition of each dataset every 100 steps for ADE, CoNLL04, SciERC, and ERFGC. For enEWT and SciDTB, we train for $5k$ steps with $1k$-step validation intervals to make our results more comparable with the start of the art [11]. We apply early stopping at 30% of the total steps without improvement and choose the best model based on top performance on the development split. In Appendix B.6, we extend the training to $10k$ steps and fully fine-tune a variety of small and large pre-trained language models to show time-wise test performance trends more clearly. We set the learning rate at $\eta = 1 \times 10^{-3}$ when the encoder is kept frozen. In all settings, including ablations, we use AdamW [15] as the optimizer and a batch size of 8.

We use [3]'s original architecture as our baseline. It uses a frozen BERT$_{base}$ model as encoder and trains all of the components showed in Figure 2. Following the best results obtained by [7], they use three BiLSTM layers in the parser with a hidden size of 400, while the four MLPs following the stacked BiLSTM have an output size of 500 for the edge representations and 100 for the relations.

4.4 Evaluation

Following [3, 8, 12], we measure tagging and parsing performance on ADE, CoNLL04, SciERC, and ERFGC in terms of micro-averaged F$_1$-measure. In addition, for enEWT and SciDTB we use unlabeled (UAS) and labeled (LAS) attachment score [20]. To corroborate the validity of our results, we train and evaluate each setting, including ablations, with five random seeds. We report mean and standard deviation for the F$_1$, UAS, and LAS metrics, averaged over the five runs. For brevity, the performance on the tagging and unlabeled edge prediction tasks are reported in Appendix A. To test the significance of our results, we use the one-tailed Wilcoxon signed-rank test [35].

5 Results and discussion

Table 3 shows the results of our experiments, with the first row using the same hyperparameters as [3] ($h_\psi = 400, d_{\text{MLP}} = 500$). We also use these hyperparameters for UD and SciDTB, since our ablation study only concerns SemDP due to SynDP being less challenging. The lower half uses the best combinations for ADE (200, 100), CoNLL04 (400, 300), SciERC (300, 300), and ERFGC (400, 300), chosen based on mean performance across these four datasets.

Overall, ***normalizing biaffine scores provides an evident performance boost at all BiLSTM depths.*** In particular, for ERFGC we beat the state-of-the-art performance achieved by [3]. Most of the performance gain is obtained by adding the first layer, with additional ones yielding diminishing returns. In other words, the performance boost provided by score normalization is highest in the absence of the implicit normalization provided by extra parameters ($L_\psi > 0$). In general, the top

Table 3: Micro-F_1 (SemDP) and LAS (SynDP) for the labeled edge prediction task. Best in bold.

Model	a	L_ψ	**ADE**	**CoNLL04**	**SciERC**	**ERFGC**	**enEWT**	**SciDTB**
[3]	1	3	$0.653_{\pm 0.018}$	$0.566_{\pm 0.019}$	$0.257_{\pm 0.024}$	$0.701_{\pm 0.009}$	$0.804_{\pm 0.006}$	$0.915_{\pm 0.002}$
Ours	1	0	$0.541_{\pm 0.021}$	$0.399_{\pm 0.024}$	$0.147_{\pm 0.049}$	$0.548_{\pm 0.010}$	$0.559_{\pm 0.005}$	$0.729_{\pm 0.004}$
		1	$0.657_{\pm 0.011}$	$0.556_{\pm 0.021}$	$0.282_{\pm 0.009}$	$0.676_{\pm 0.010}$	$0.771_{\pm 0.006}$	$0.892_{\pm 0.002}$
		2	$0.667_{\pm 0.011}$	$0.573_{\pm 0.025}$	$0.273_{\pm 0.010}$	$0.694_{\pm 0.010}$	$0.796_{\pm 0.006}$	$0.910_{\pm 0.002}$
		3	$0.662_{\pm 0.027}$	$0.562_{\pm 0.021}$	$0.299_{\pm 0.023}$	$0.705_{\pm 0.011}$	$0.804_{\pm 0.006}$	$0.915_{\pm 0.002}$
	$\frac{1}{\sqrt{d}}$	0	$0.567_{\pm 0.014}$	$0.438_{\pm 0.033}$	$0.181_{\pm 0.027}$	$0.612_{\pm 0.008}$	$0.646_{\pm 0.002}$	$0.796_{\pm 0.002}$
		1	$0.668_{\pm 0.017}$	$0.597_{\pm 0.015}$	$0.299_{\pm 0.019}$	$0.692_{\pm 0.009}$	$0.789_{\pm 0.003}$	$0.904_{\pm 0.002}$
		2	$0.676_{\pm 0.019}$	$0.596_{\pm 0.014}$	$0.312_{\pm 0.011}$	$0.699_{\pm 0.009}$	$0.805_{\pm 0.003}$	$0.916_{\pm 0.002}$
		3	$\mathbf{0.686}_{\pm 0.025}$	$\mathbf{0.602}_{\pm 0.017}$	$\mathbf{0.320}_{\pm 0.013}$	$\mathbf{0.708}_{\pm 0.008}$	$\mathbf{0.807}_{\pm 0.005}$	$\mathbf{0.919}_{\pm 0.001}$

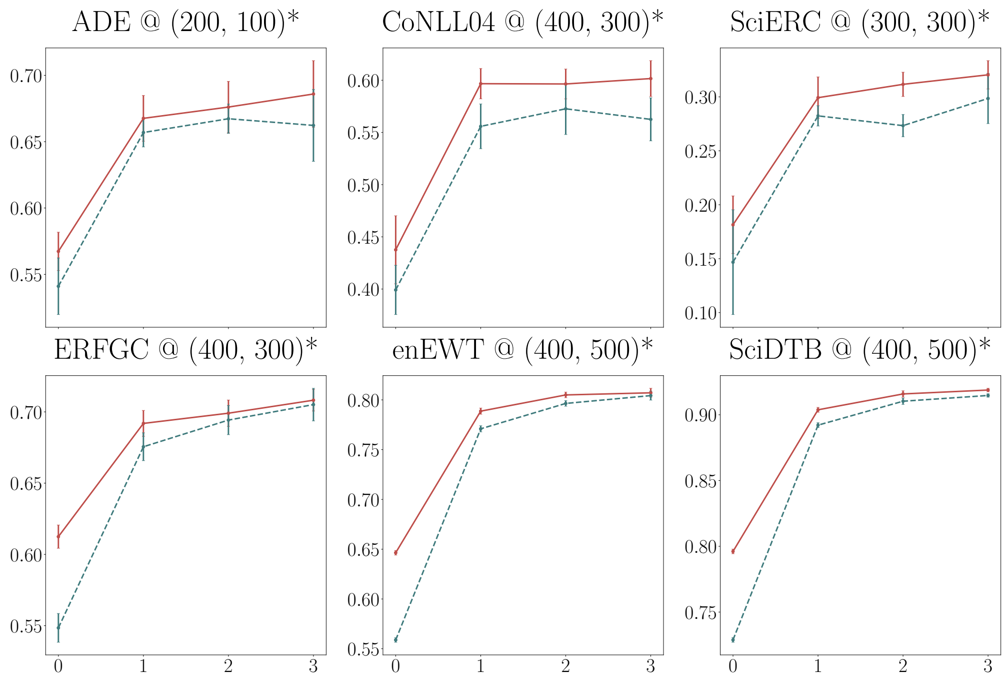


Figure 3: Micro-F_1 (SemDP) and LAS (SynDP) vs $L_\psi \in \{0, 1, 2, 3\}$ at $2k$ training steps. Red = norm; blue = raw. *Performance increase with normalization is statistically significant ($p < 0.01$).

performance obtained by using three BiLSTM layers and no score normalization can be matched or surpassed ***with a single BiLSTM layer, when using score normalization.*** Taking into account the lower values for h_ψ and d_{MLP}, this represents a reduction in trained parameters of up to 85%.

As laid out in Section 3, score variance tends to decay with deeper BiLSTM stacks. This in turn produces a converging trend as L_ψ increases. When looking at Figure 3, this is especially evident for ERFGC, enEWT, and SciDTB, for which the beneficial effect of score normalization shrinks smoothly with higher values of L_ψ. Although the same cannot be said for ADE, CoNLL04, and SciERC, the performance boost is still statistically significant across all layer depths ($p < 0.01$).

As regards SciERC, normalizing scores without any BiLSTM layers ($L_\psi = 0$) produces a 23% increase in performance with a strong reduction in standard deviation. SciERC arguably has the hardest dependency graphs to parse, due to the little overlap between training and testing entities, which makes it difficult to leverage tag embeddings. In addition, it is characterized by complex semantic dependencies. Therefore, we reckon normalizing biaffine scores to be especially important when dealing with hard tasks. In particular, normalizing scores helps mitigate the higher variance of the model's predictions for this challenging dataset. Conversely, the lack of score normalization

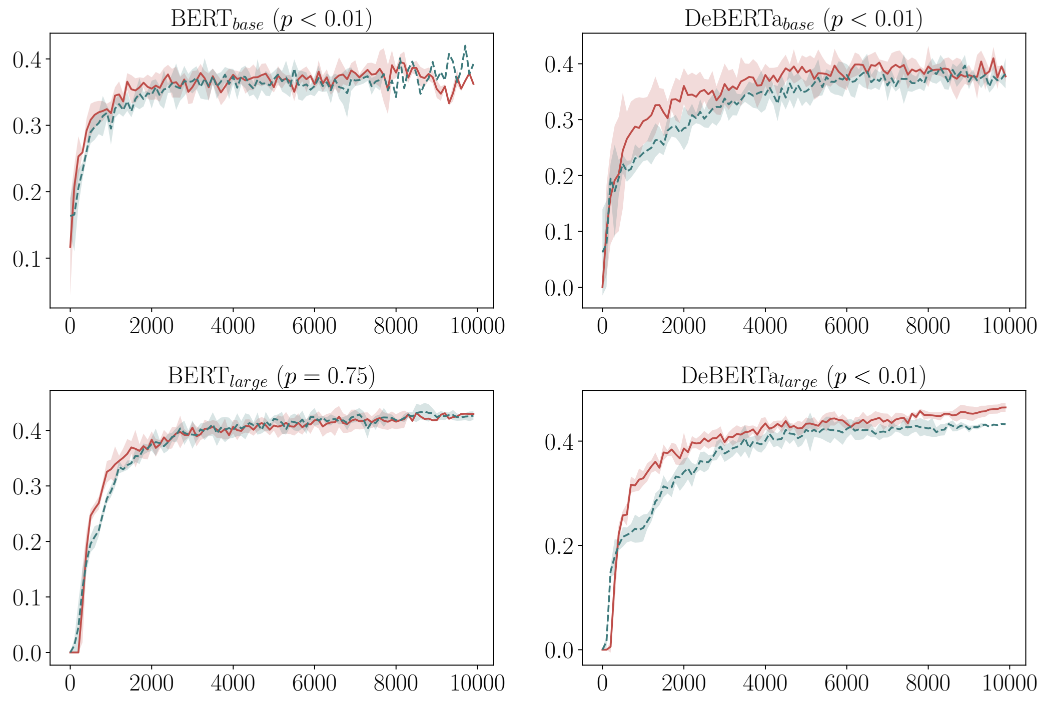


Figure 4: Test performance on SciERC (micro-averaged F$_1$-measure vs number of training steps). The p-values indicate greater performance with normalization (one-tailed Wilcoxon signed-rank test).

exacerbates already uncertain predictions. For SciERC, this makes the trend observed in Figure 3 more unstable. Indeed, the performance first drops at $L_\psi = 2$ and then increases once again at $L_\psi = 3$, which does not happen for the other datasets.

Compared to existing approaches which do not scale the biaffine scores, fewer parameters can thus be trained to obtain similar performance. Therefore, our results indicate that parsers using raw biaffine scores are likely overparameterized, compared to the performance they could achieve by using score normalization. In addition, score normalization increases sample efficiency by accelerating convergence. Figure 4 displays the performance on SciERC's test set for four models during full fine-tuning. As the figure shows, the speedup in convergence is statistically significant when normalizing scores. This shows how score normalization can be beneficial even when fully fine-tuning models with ~10^8 parameters, given a hard task which forces the model to make uncertain predictions.

6 Conclusions

In this work, we explored the effect of scaling the scores produced by biaffine transformations when predicting the edges of a dependency graph. We have demonstrated, both theoretically and empirically, that the score variance produced by a lack of score scaling hurts model performance when predicting edges and relations. In addition, our theoretical work and experiments highlight a strong relationship between the number of trained layers and their intrinsic normalization effect.

Departing from a state-of-the-art architecture for semantic dependency parsing, we were able to improve its performance on both semantic and syntactic dependency parsing on six datasets. On ERFGC, a dataset of directed acyclic semantic dependency graphs compiled from culinary recipes, our approach allowed us to beat the state-of-the-art performance achieved by Bhatt et al. [3]. Moreover, our results showed that a single BiLSTM layer can be sufficient to match or surpass the results of state-of-the-art architectures with a decrease in trained parameters of up to 85%. In the case of SciERC, a challenging dataset for semantic dependency parsing, we found that the performance boost was particularly great when only training the biaffine scorer, without any BiLSTM layers. Moreover, for this challenging dataset, we find that scaling the predictions of the biaffine scorer can accelerate convergence speed even when fully fine-tuning models in the 100 to $400M$ parameter range. In addition, for three of the datasets we also observed that the performance obtained with normalized and raw scores converged smoothly as the number of trained layers increased. This supported our claim that stacking BiLSTM layers mainly serves the purpose of producing an implicit regularization, and that this effect can be obtained by normalizing the scores, without any extra parameters.

Due to the marginal performance increase reported for GNN-based dependency parsers [11], iterative adjacency refinement methods in DP might suffer from limitations akin to the lack of normalization which we will explore in future work. This will also help us verify the actual effectiveness of triaffine two-hop and k-hop methods in general. Furthermore, we plan to extend our work to large scale graph inference tasks which have been limited by the lack of parameter efficient methods, such as long form discourse parsing tasks. With regard to score scaling, we also wish to verify whether the positive effects of this normalization can carry over to other tasks. Finally, further explorations in model efficiency are warranted, given our findings, e.g., via pruning of model parameters.

References

[1] Sanjeev Arora, Nadav Cohen, Wei Hu, and Yuping Luo. Implicit Regularization in Deep Matrix Factorization. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d' Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[2] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[3] Dhaivat J. Bhatt, Seyed Ahmad Abdollahpouri Hosseini, Federico Fancellu, and Afsaneh Fazly. End-to-end Parsing of Procedural Text into Flow Graphs. In Nicoletta Calzolari, Min-Yen Kan, Veronique Hoste, Alessandro Lenci, Sakriani Sakti, and Nianwen Xue, editors, *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 5833–5842, Torino, Italia, May 2024. ELRA and ICCL.

[4] Zhen Bi, Jing Chen, Yinuo Jiang, Feiyu Xiong, Wei Guo, Huajun Chen, and Ningyu Zhang. CodeKGC: Code Language Model for Generative Knowledge Graph Construction, January 2024. arXiv:2304.09048 [cs].

[5] Yee Seng Chan and Dan Roth. Exploiting syntactico-semantic structures for relation extraction. In *Proceedings of the 49th annual meeting of the association for computational linguistics: human language technologies*, pages 551–560, 2011.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[7] Timothy Dozat and Christopher D. Manning. Deep Biaffine Attention for Neural Dependency Parsing. In *International Conference on Learning Representations*. arXiv, March 2017. arXiv:1611.01734 [cs].

[8] Timothy Dozat and Christopher D. Manning. Simpler but More Accurate Semantic Dependency Parsing. In Iryna Gurevych and Yusuke Miyao, editors, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 484–490, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[9] Jack Edmonds. Optimum branchings. *Journal of Research of the National Bureau of Standards Section B Mathematics and Mathematical Physics*, 71B(4):233, October 1967.

[10] Harsha Gurulingappa, Abdul Mateen Rajput, Angus Roberts, Juliane Fluck, Martin Hofmann-Apitius, and Luca Toldo. Development of a benchmark corpus to support the automatic extraction of drug-related adverse effects from medical case reports. *Journal of Biomedical Informatics*, 45(5):885–892, 2012. Text Mining and Natural Language Processing in Pharmacogenomics.

[11] Tao Ji, Yuanbin Wu, and Man Lan. Graph-based Dependency Parsing with Graph Neural Networks. In Anna Korhonen, David Traum, and Lluís Màrquez, editors, *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2475–2485, Florence, Italy, July 2019. Association for Computational Linguistics.

[12] Shu Jiang, Zuchao Li, Hai Zhao, and Weiping Ding. Entity-Relation Extraction as Full Shallow Semantic Dependency Parsing. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 32:1088–1099, 2024.

[13] Eliyahu Kiperwasser and Yoav Goldberg. Simple and Accurate Dependency Parsing Using Bidirectional LSTM Feature Representations. *Transactions of the Association for Computational Linguistics*, 4:313–327, July 2016. _eprint: https://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00101/1567410/tacl_a_00101.pdf.

[14] Tianyu Liu, Yuchen Eleanor Jiang, Nicholas Monath, Ryan Cotterell, and Mrinmaya Sachan. Autoregressive structured prediction with language models. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang, editors, *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 993–1005, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics.

[15] Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization, January 2019. arXiv:1711.05101 [cs, math].

[16] Yaojie Lu, Qing Liu, Dai Dai, Xinyan Xiao, Hongyu Lin, Xianpei Han, Le Sun, and Hua Wu. Unified structure generation for universal information extraction. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5755–5772, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[17] Yi Luan, Luheng He, Mari Ostendorf, and Hannaneh Hajishirzi. Multi-task identification of entities, relations, and coreference for scientific knowledge graph construction. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3219–3232, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.

[18] David Nadeau and Satoshi Sekine. A survey of named entity recognition and classification. In Satoshi Sekine and Elisabete Ranchhod, editors, *Recognition, classification and use*, pages 3–28. John Benjamins Publishing Company, 2009.

[19] Joakim Nivre, Mitchell Abrams, Željko Agić, Lars Ahrenberg, and Lene Antonsen. Universal dependencies 2.2, 2018. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

[20] Joakim Nivre and Chiao-Ting Fang. Universal dependency evaluation. In *Proceedings of the NoDaLiDa 2017 Workshop on Universal Dependencies (UDW 2017)*, pages 86–95, 2017.

[21] Giovanni Paolini, Ben Athiwaratkun, Jason Krone, Jie Ma, Alessandro Achille, RISHITA ANUBHAI, Cicero Nogueira dos Santos, Bing Xiang, and Stefano Soatto. Structured prediction as translation between augmented natural languages. In *International Conference on Learning Representations*, 2021.

[22] Wenzhe Pei, Tao Ge, and Baobao Chang. An Effective Neural Network Model for Graph-based Dependency Parsing. In Chengqing Zong and Michael Strube, editors, *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 313–322, Beijing, China, July 2015. Association for Computational Linguistics.

[23] Hao Peng, Sam Thomson, and Noah A. Smith. Deep multitask learning for semantic dependency parsing. In Regina Barzilay and Min-Yen Kan, editors, *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2037–2048, Vancouver, Canada, July 2017. Association for Computational Linguistics.

[24] Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind Joshi, and Bonnie Webber. The Penn Discourse TreeBank 2.0. In Nicoletta Calzolari, Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odijk, Stelios Piperidis, and Daniel Tapias, editors, *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco, May 2008. European Language Resources Association (ELRA).

[25] Peng Qi, Timothy Dozat, Yuhao Zhang, and Christopher D Manning. Universal dependency parsing from scratch. *arXiv preprint arXiv:1901.10457*, 2019.

[26] Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. Stanza: A Python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, 2020.

[27] Noam Razin and Nadav Cohen. Implicit regularization in deep learning may not be explainable by norms. *Advances in neural information processing systems*, 33:21174–21187, 2020.

[28] Dan Roth and Wen-Tau Yih. A linear programming formulation for global inference in natural language tasks. In *Proceedings of the Eighth Conference on Computational Natural Language Learning (CoNLL-2004) at HLT-NAACL 2004*, pages 1–8, Boston, Massachusetts, USA, May 6 - May 7 2004. Association for Computational Linguistics.

[29] Olivier Roy and Martin Vetterli. The effective rank: A measure of effective dimensionality. In *2007 15th European signal processing conference*, pages 606–610. IEEE, 2007.

[30] Wei Tang, Benfeng Xu, Yuyue Zhao, Zhendong Mao, Yifeng Liu, Yong Liao, and Haiyong Xie. UniRel: Unified representation and interaction for joint relational triple extraction. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang, editors, *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 7087–7099, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics.

[31] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[32] David Wadden, Ulme Wennberg, Yi Luan, and Hannaneh Hajishirzi. Entity, relation, and event extraction with contextualized span representations. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *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 5784–5789, Hong Kong, China, November 2019. Association for Computational Linguistics.

[33] Jue Wang and Wei Lu. Two are better than one: Joint entity and relation extraction with table-sequence encoders. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1706–1721, Online, November 2020. Association for Computational Linguistics.

[34] Xiang Wei, Xingyu Cui, Ning Cheng, Xiaobin Wang, Xin Zhang, Shen Huang, Pengjun Xie, Jinan Xu, Yufeng Chen, Meishan Zhang, Yong Jiang, and Wenjuan Han. ChatIE: Zero-Shot Information Extraction via Chatting with ChatGPT, May 2024. arXiv:2302.10205 [cs].

[35] Robert F Woolson. Wilcoxon signed-rank test. *Encyclopedia of biostatistics*, 8, 2005.

[36] Yoko Yamakata, Shinsuke Mori, and John Carroll. English Recipe Flow Graph Corpus. In Nicoletta Calzolari, Frédéric Béchet, Philippe Blache, Khalid Choukri, Christopher Cieri, Thierry Declerck, Sara Goggi, Hitoshi Isahara, Bente Maegaard, Joseph Mariani, Hélène Mazo, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 5187–5194, Marseille, France, May 2020. European Language Resources Association.

[37] Zhiheng Yan, Chong Zhang, Jinlan Fu, Qi Zhang, and Zhongyu Wei. A partition filter network for joint entity and relation extraction. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih, editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 185–197, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

[38] An Yang and Sujian Li. SciDTB: Discourse dependency TreeBank for scientific abstracts. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 444–449, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[39] Urchade Zaratiana, Nadi Tomeh, Pierre Holat, and Thierry Charnois. An autoregressive text-to-graph framework for joint entity and relation extraction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19477–19487, 2024. Issue: 17.

[40] Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. Kernel methods for relation extraction. *Journal of machine learning research*, 3(Feb):1083–1106, 2003.

[41] Bowen Zhang and Harold Soh. Extract, Define, Canonicalize: An LLM-based Framework for Knowledge Graph Construction, April 2024. arXiv:2404.03868 [cs].

[42] Dan Zhao. Combining explicit and implicit regularization for efficient learning in deep networks. *Advances in Neural Information Processing Systems*, 35:3024–3038, 2022.

[43] Zexuan Zhong and Danqi Chen. A frustratingly easy approach for entity and relation extraction. In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tur, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou, editors, *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 50–61, Online, June 2021. Association for Computational Linguistics.

[44] Hao Zhu, Yankai Lin, Zhiyuan Liu, Jie Fu, Tat-Seng Chua, and Maosong Sun. Graph neural networks with generated parameters for relation extraction. In Anna Korhonen, David Traum, and Lluís Màrquez, editors, *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1331–1339, Florence, Italy, July 2019. Association for Computational Linguistics.

Table 4: Micro-averaged test F_1 performance on all tasks (ϕ = ✓, $\mathbf{e}_i^{tag}$ = ✓). Best in bold.

Metric	a	L_ψ	ADE	CoNLL04	SciERC	ERFGC	enEWT	SciDTB
(h_ψ, d_{MLP}) =			(200, 100)	(400, 300)	(300, 300)	(400, 300)	(400, 500)	(400, 500)
rels.	1	0	0.541 $\pm$0.021	0.399 $\pm$0.024	0.147 $\pm$0.049	0.548 $\pm$0.010	0.559 $\pm$0.005	0.729 $\pm$0.004
		1	0.657 $\pm$0.011	0.556 $\pm$0.021	0.282 $\pm$0.009	0.676 $\pm$0.010	0.771 $\pm$0.006	0.892 $\pm$0.002
		2	0.667 $\pm$0.011	0.573 $\pm$0.025	0.273 $\pm$0.010	0.694 $\pm$0.010	0.796 $\pm$0.006	0.910 $\pm$0.002
		3	0.662 $\pm$0.027	0.562 $\pm$0.021	0.299 $\pm$0.023	0.705 $\pm$0.011	0.804 $\pm$0.006	0.915 $\pm$0.002
	$\frac{1}{\sqrt{d}}$	0	0.567 $\pm$0.014	0.438 $\pm$0.033	0.181 $\pm$0.027	0.612 $\pm$0.008	0.646 $\pm$0.002	0.796 $\pm$0.002
		1	0.668 $\pm$0.017	0.597 $\pm$0.015	0.299 $\pm$0.019	0.692 $\pm$0.009	0.789 $\pm$0.003	0.904 $\pm$0.002
		2	0.676 $\pm$0.019	0.596 $\pm$0.014	0.312 $\pm$0.011	0.699 $\pm$0.009	0.805 $\pm$0.003	0.916 $\pm$0.002
		3	**0.686** $\pm$0.025	**0.602** $\pm$0.017	**0.320** $\pm$0.013	**0.708** $\pm$0.008	**0.807** $\pm$0.005	**0.919** $\pm$0.001
edges	1	0	0.536 $\pm$0.009	0.415 $\pm$0.020	0.173 $\pm$0.050	0.601 $\pm$0.013	0.589 $\pm$0.006	0.745 $\pm$0.005
		1	0.652 $\pm$0.009	0.566 $\pm$0.022	0.321 $\pm$0.009	0.744 $\pm$0.013	0.793 $\pm$0.006	0.901 $\pm$0.002
		2	0.657 $\pm$0.021	0.586 $\pm$0.017	0.322 $\pm$0.017	0.769 $\pm$0.014	0.819 $\pm$0.006	0.919 $\pm$0.002
		3	0.649 $\pm$0.027	0.578 $\pm$0.011	0.355 $\pm$0.028	0.782 $\pm$0.007	0.827 $\pm$0.006	0.924 $\pm$0.002
	$\frac{1}{\sqrt{d}}$	0	0.549 $\pm$0.014	0.453 $\pm$0.032	0.203 $\pm$0.030	0.674 $\pm$0.007	0.682 $\pm$0.003	0.815 $\pm$0.001
		1	0.654 $\pm$0.020	0.598 $\pm$0.021	0.351 $\pm$0.019	0.762 $\pm$0.009	0.810 $\pm$0.003	0.913 $\pm$0.002
		2	0.660 $\pm$0.015	0.600 $\pm$0.008	0.367 $\pm$0.015	0.775 $\pm$0.008	0.827 $\pm$0.003	0.925 $\pm$0.002
		3	**0.666** $\pm$0.028	**0.610** $\pm$0.022	**0.377** $\pm$0.021	**0.786** $\pm$0.004	**0.829** $\pm$0.004	**0.928** $\pm$0.002
tags	1	0	0.615 $\pm$0.115	0.285 $\pm$0.150	0.014 $\pm$0.015	0.662 $\pm$0.073		
		1	0.665 $\pm$0.147	0.671 $\pm$0.018	0.049 $\pm$0.014	0.794 $\pm$0.057		
		2	0.746 $\pm$0.011	0.648 $\pm$0.074	0.060 $\pm$0.012	0.838 $\pm$0.016		
		3	**0.768** $\pm$0.022	0.669 $\pm$0.033	**0.062** $\pm$0.005	0.853 $\pm$0.013		
	$\frac{1}{\sqrt{d}}$	0	0.604 $\pm$0.134	0.464 $\pm$0.115	0.046 $\pm$0.005	0.735 $\pm$0.066		
		1	0.734 $\pm$0.048	0.702 $\pm$0.031	0.058 $\pm$0.014	0.857 $\pm$0.010		
		2	0.724 $\pm$0.064	0.688 $\pm$0.080	0.060 $\pm$0.007	**0.864** $\pm$0.018		
		3	0.762 $\pm$0.017	**0.690** $\pm$0.062	0.057 $\pm$0.012	0.850 $\pm$0.022		

A Full Results

Table 4 reports the results for the best hyperparameter combinations for all three tasks of predicting tags, edges, and relations of the semantic dependency graphs. Note that the first part of Table 4 is identical to the bottom of Table 3. Since we already discussed the performance for labeled edges in Section 5, we forgo discussing them again in this appendix.

As regards unlabeled edges, using score normalization improves mean performance for all datasets, similarly to relations. This is expected, as unlabeled edges directly influence the prediction of the edge labels, together with entity class prediction. As we have already observed for relations, the performance boost provided by normalization is particularly great for SciERC at $L_\psi = 0$.

It is also interesting to notice that for CoNLL04, the performance for the unlabeled edge prediction task is only slightly higher than that of the relations. In fact, as regards ADE, the opposite is true, with the labeled performance being seemingly higher. Due to the high variances, it would therefore seem that for these two datasets there is essentially no difference in difficulty between the labeled and unlabeled edge prediction tasks. In the case of ADE, this makes perfect sense, since there is only one possible relation. For CoNLL04, the amount of relation types is rather limited as well; therefore, it is sensible for the performance to also be similar in this case. Indeed, when looking at SciERC and ERFGC, the performance gap between labeled and unlabeled tasks is considerable. This is a strong indication that CoNLL04 is thus found somewhere in the middle, where the number of possible relations only slightly affects labeled performance.

As far as enEWT and SciDTB are concerned, the performance of edges (UAS) and relations (LAS) is also rather similar. Since in this case the predictions involve syntactic rather than semantic relations, the similar performance hints at relations being easier to predict in SynDP than in SemDP.

Table 5: Tagger ablation results: performance for the best hyperparameters (h_ψ, d_{MLP}). Best in bold, second-best underlined.

a	L_ψ	ϕ	$\mathbf{e}_i^{tag}$	**ADE**	**CoNLL04**	**SciERC**	**ERFGC**	**Mean**
$(h_\psi, d_{\text{MLP}}) =$				(200, 100)	(400, 300)	(300, 300)	(400, 300)	
1	0	■	■	0.541 ± 0.021	0.399 ± 0.024	0.147 ± 0.049	0.548 ± 0.010	0.515
	1	■	■	0.657 ± 0.011	0.556 ± 0.021	0.282 ± 0.009	0.676 ± 0.010	
	2	■	■	0.667 ± 0.011	0.573 ± 0.025	0.273 ± 0.010	0.694 ± 0.010	
	3	■	■	0.662 ± 0.027	0.562 ± 0.021	0.299 ± 0.023	0.705 ± 0.011	
$\frac{1}{\sqrt{d}}$	0	■	■	0.567 ± 0.014	0.438 ± 0.033	0.181 ± 0.027	0.612 ± 0.008	**0.541**
	1	■	■	0.668 ± 0.017	$\underline{0.597}$ ± 0.015	0.299 ± 0.019	0.692 ± 0.009	
	2	■	■	0.676 ± 0.019	0.596 ± 0.014	0.312 ± 0.011	0.699 ± 0.009	
	3	■	■	**0.686** ± 0.025	**0.602** ± 0.017	0.320 ± 0.013	**0.708** ± 0.008	
1	0	■	□	0.543 ± 0.013	0.402 ± 0.023	0.162 ± 0.019	0.554 ± 0.008	0.517
	1	■	□	0.674 ± 0.011	0.527 ± 0.026	0.275 ± 0.013	0.677 ± 0.005	
	2	■	□	0.672 ± 0.019	0.575 ± 0.009	0.293 ± 0.012	0.689 ± 0.011	
	3	■	□	0.657 ± 0.027	0.583 ± 0.011	0.301 ± 0.012	0.697 ± 0.007	
$\frac{1}{\sqrt{d}}$	0	■	□	0.563 ± 0.013	0.443 ± 0.019	0.188 ± 0.006	0.612 ± 0.003	0.534
	1	■	□	0.653 ± 0.023	0.565 ± 0.027	0.299 ± 0.020	0.687 ± 0.006	
	2	■	□	0.672 ± 0.017	0.592 ± 0.020	0.307 ± 0.008	0.702 ± 0.005	
	3	■	□	0.661 ± 0.022	0.593 ± 0.017	0.305 ± 0.014	**0.708** ± 0.007	
1	0	□	■	0.550 ± 0.026	0.387 ± 0.030	0.157 ± 0.012	0.553 ± 0.006	0.514
	1	□	■	0.667 ± 0.022	0.532 ± 0.036	0.273 ± 0.013	0.673 ± 0.006	
	2	□	■	0.665 ± 0.021	0.565 ± 0.024	0.278 ± 0.027	0.694 ± 0.013	
	3	□	■	0.676 ± 0.021	0.558 ± 0.051	0.288 ± 0.012	$\underline{0.706}$ ± 0.006	
$\frac{1}{\sqrt{d}}$	0	□	■	0.578 ± 0.022	0.437 ± 0.030	0.187 ± 0.014	0.611 ± 0.008	0.534
	1	□	■	0.651 ± 0.032	0.559 ± 0.029	0.301 ± 0.007	0.684 ± 0.003	
	2	□	■	0.673 ± 0.029	0.582 ± 0.017	0.310 ± 0.014	0.701 ± 0.008	
	3	□	■	0.659 ± 0.019	0.586 ± 0.026	**0.324** ± 0.019	**0.708** ± 0.012	
1	0	□	□	0.547 ± 0.019	0.402 ± 0.033	0.156 ± 0.029	0.549 ± 0.015	0.516
	1	□	□	0.651 ± 0.017	0.552 ± 0.012	0.288 ± 0.008	0.680 ± 0.007	
	2	□	□	0.664 ± 0.031	0.566 ± 0.012	0.288 ± 0.017	0.693 ± 0.011	
	3	□	□	0.663 ± 0.008	0.561 ± 0.014	0.291 ± 0.012	0.703 ± 0.007	
$\frac{1}{\sqrt{d}}$	0	□	□	0.566 ± 0.014	0.431 ± 0.015	0.182 ± 0.023	0.606 ± 0.012	0.534
	1	□	□	0.655 ± 0.015	0.567 ± 0.010	0.286 ± 0.018	0.678 ± 0.013	
	2	□	□	0.678 ± 0.014	0.591 ± 0.026	0.307 ± 0.007	0.702 ± 0.007	
	3	□	□	$\underline{0.684}$ ± 0.022	0.595 ± 0.017	$\underline{0.321}$ ± 0.016	0.698 ± 0.015	

In the tagging task, very high standard deviation can be observed. In the case of ADE and CoNLL04, this could be caused by our choice of $\lambda_1 = 0.1$ being too low a value. Regarding SciERC, as already mentioned, most of the entities in the validation and test set are not found in the training set. This means it is not possible to train the model to recognize them, which results in the abysmal tagging performance reported in Table 4. This makes it challenging to train good tag embeddings which can help inform the edge and relation prediction tasks. Surprisingly, we notice that for CoNLL04 and ERFGC the tagging performance is also seemingly higher with score normalization. However, the overlap of the standard deviations is too high to be able to make any claims on the matter.

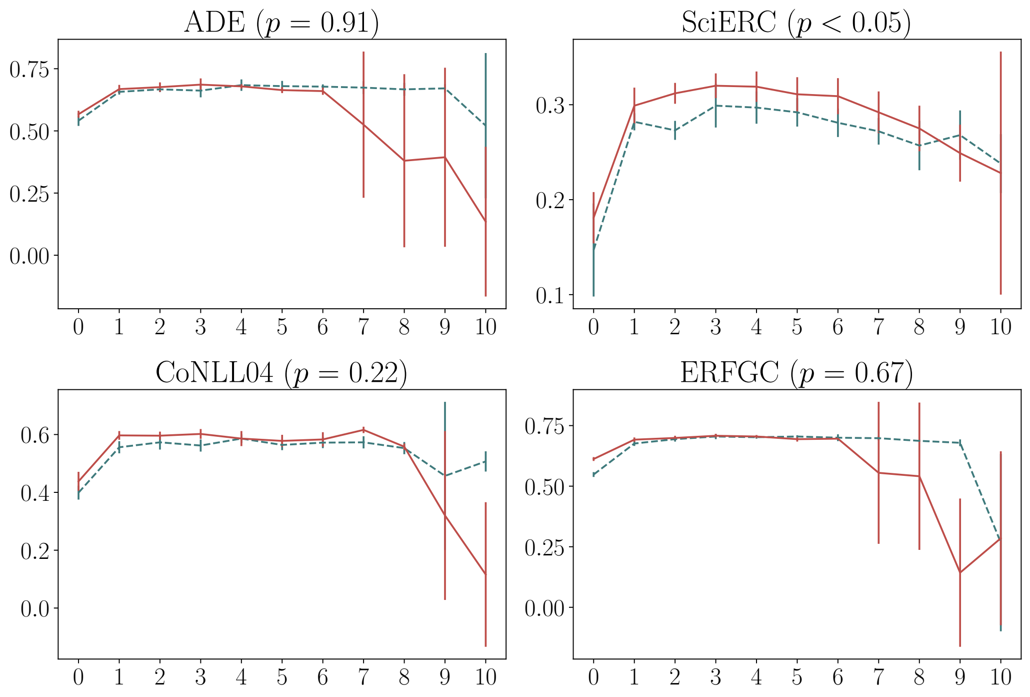


Figure 5: Performance in terms of F_1-measure vs $L_\psi \in \{0, \ldots, 10\}$ on ADE, CoNLL04, SciERC, and ERFGC at $2k$ training steps. As hyperparameters, we use the best values (h_ψ, d_{MLP}) with $\phi = \checkmark$ and $\mathbf{e}_i^{tag} = \checkmark$. Red = norm; blue = raw. The p-values indicate that the performance gap between using normalized and raw scores is statistically significant (one-tailed Wilcoxon signed-rank test).

B Ablations

B.1 Tagger

Table 5 reports the results for the best values of h_ψ and d_{MLP} (chosen as described in Section 4), ablating over $\phi \in \{\checkmark, \times\}$ and $\mathbf{e}_i^{tag} \in \{\checkmark, \times\}$. Note that the first quarter of the table is equivalent to Table 3.

Using both the tagger BiLSTM and tag embeddings on average produces the best results. This is sensible, since a better tagger produces better tag embeddings, which in turn help inform the edge and relation classification tasks. The model in this case obtains the best performance on ADE, CoNLL04, and ERFGC. Its top performance for SciERC (0.320) is also close to the overall peak performance (0.324), obtained without using the tagger BiLSTM. On average, the mean performance is considerably higher when combining the positive contributions of the BiLSTM and the tag embeddings.

B.2 Additional BiLSTM layers

As shown in Figure 5, performance is initially greater when normalizing scores, with $L_\psi \in \{0, 1, 2, 3, 4, 5\}$. In this range, an increasing amount of layers reduces the amount of benefit obtained by using normalization. However, at $L_\psi > 5$, for ADE, CoNNL04, and ERFGC, performance drops suddenly and rapidly when using biaffine score normalization. In the case of SciERC, the drop is more gentle, with the performance for the normalized and raw biaffine scores eventually converging, albeit with very high standard deviation at $L_\psi = 10$.

As observed in Section 5, adding trainable layers seemingly allows the parameters to scale the variance of the scores to compensate for the lack of explicit normalization. As reported in Appendix B.6, a similar behavior can be observed with fully fine-tuned models, where normalization yields diminishing

Table 6: Performance in terms of F_1-measure when ablating over different output dimension for the parser's MLPs ($\phi = \checkmark$, $\mathbf{e}_i^{tag} = \checkmark$). Best in bold.

a	d_{MLP}	**ADE**	**CoNLL04**	**SciERC**	**ERFGC**
$(L_\psi, h_\psi) =$		(3, 200)	(3, 400)	(3, 300)	(3, 400)
1	100	0.662 ± 0.027	0.579 ± 0.030	0.302 ± 0.018	0.709 ± 0.008
	300	0.658 ± 0.018	0.562 ± 0.021	0.299 ± 0.023	0.705 ± 0.011
	500	0.657 ± 0.018	0.566 ± 0.019	0.281 ± 0.016	0.701 ± 0.009
$\frac{1}{\sqrt{d}}$	100	0.686 ± 0.025	0.586 ± 0.013	0.318 ± 0.017	**0.712** ± 0.008
	300	0.680 ± 0.018	**0.602** ± 0.017	**0.320** ± 0.013	0.708 ± 0.008
	500	**0.685** ± 0.019	0.600 ± 0.015	0.311 ± 0.009	0.707 ± 0.004

Table 7: Performance in terms of F_1-measure when ablating over different hidden sizes for the parser's stacked BiLSTMs ($\phi = \checkmark$, $\mathbf{e}_i^{tag} = \checkmark$). Best in bold.

a	h_ψ	**ADE**	**CoNLL04**	**SciERC**	**ERFGC**
$(L_\psi, d_{\text{MLP}}) =$		(3, 100)	(3, 300)	(3, 300)	(3, 300)
1	100	0.682 ± 0.021	0.580 ± 0.031	0.291 ± 0.021	0.693 ± 0.011
	200	0.662 ± 0.027	0.582 ± 0.006	0.286 ± 0.032	0.703 ± 0.007
	300	0.674 ± 0.029	0.585 ± 0.026	0.299 ± 0.023	0.703 ± 0.006
	400	0.663 ± 0.032	0.562 ± 0.021	0.289 ± 0.038	0.705 ± 0.011
$\frac{1}{\sqrt{d}}$	100	0.685 ± 0.020	0.600 ± 0.018	0.302 ± 0.013	0.698 ± 0.008
	200	**0.686** ± 0.025	**0.610** ± 0.018	0.314 ± 0.019	0.702 ± 0.008
	300	0.678 ± 0.012	0.599 ± 0.012	**0.320** ± 0.013	**0.711** ± 0.005
	400	0.674 ± 0.011	0.602 ± 0.017	0.306 ± 0.016	0.708 ± 0.008

benefits the more parameters we train. This behavior points to a trade-off between the use of our technique and the amount of stacked BiLSTM layers, at a given amount of training steps.

B.3 MLP output dimension

As shown in Table 6, without normalization, increasing the output dimension of the two MLPs responsible for projecting edge representation projections leads to a decrease in performance. This is in contrast with the behavior observed when using normalization, where performance is essentially independent from the output dimension. This is in line with our claim that higher variance in the edge scores causes lower performance. Normalizing the scores assuages the variance, which makes performance stable even at high output dimensions. Once again, these results show the relationship between score variance and performance, with equivalent performance being achievable with fewer parameters.

B.4 BiLSTM hidden size

As Table 7 shows, the hidden size of the BiLSTMs does not have a visible effect on performance. This is especially the case when applying score normalization, which produces smaller standard deviations. As a result, not only do models perform better with biaffine score normalization, but performance is also less dependent on the hidden size of the BiLSTM encoders. This supports our claim that score normalization is a useful technique to obtain the same performance with lower parameter count, since the models tend to perform comparably, despite the lower BiLSTM hidden sizes.

Table 8: Ablation over the best hyperparameters (h_ψ, d_{MLP}) and $\phi = \checkmark$ and $\mathbf{e}_i^{tag} = \checkmark$, using LayerNorm layers $\texttt{LN}_\psi \in \{\checkmark, \times\}$ and Xavier uniform/normal initialization $I_{par} \in \{\mathcal{U}, \mathcal{N}\}$ for the parser. The first quarter of the table is equivalent to Table 3. Best in bold, second-best underlined.

I_{par}	$\texttt{LN}_\psi$	a	L_ψ	**ADE**	**CoNLL04**	**SciERC**	**ERFGC**	**Mean**
$(h_\psi, d_{\text{MLP}}) =$				(200, 100)	(400, 300)	(300, 300)	(400, 300)	
$\mathcal{U}$	☐	1	0	0.541 ± 0.021	0.399 ± 0.024	0.147 ± 0.049	0.548 ± 0.010	0.515
	☐		3	0.662 ± 0.027	0.562 ± 0.021	0.299 ± 0.023	0.705 ± 0.011	
	☐		1	0.657 ± 0.011	0.556 ± 0.021	0.282 ± 0.009	0.676 ± 0.010	
	☐		2	0.667 ± 0.011	0.573 ± 0.025	0.273 ± 0.010	0.694 ± 0.010	
	☐	$\frac{1}{\sqrt{d}}$	0	0.567 ± 0.014	0.438 ± 0.033	0.181 ± 0.027	0.612 ± 0.008	**0.541**
	☐		1	0.668 ± 0.017	0.597 ± 0.015	0.299 ± 0.019	0.692 ± 0.009	
	☐		2	0.676 ± 0.019	0.596 ± 0.014	0.312 ± 0.011	0.699 ± 0.009	
	☐		3	0.686 ± 0.025	0.602 ± 0.017	**0.320** ± 0.013	0.708 ± 0.008	
	■	1	0	0.545 ± 0.018	0.399 ± 0.024	0.151 ± 0.014	0.548 ± 0.010	0.468
	■		1	0.680 ± 0.014	0.554 ± 0.042	0.265 ± 0.028	0.686 ± 0.007	
	■		2	0.679 ± 0.011	0.578 ± 0.033	0.260 ± 0.020	**0.715** ± 0.012	
	■		3	0.680 ± 0.013	0.117 ± 0.261	0.060 ± 0.133	0.569 ± 0.318	
	■	$\frac{1}{\sqrt{d}}$	0	0.567 ± 0.014	0.433 ± 0.029	0.181 ± 0.027	0.614 ± 0.004	0.503
	■		1	0.664 ± 0.017	0.564 ± 0.037	0.282 ± 0.029	0.686 ± 0.004	
	■		2	**0.697** ± 0.022	**0.623** ± 0.019	0.300 ± 0.042	0.702 ± 0.011	
	■		3	0.675 ± 0.019	0.239 ± 0.327	0.111 ± 0.152	0.703 ± 0.006	
$\mathcal{N}$	☐	1	0	0.545 ± 0.017	0.415 ± 0.014	0.155 ± 0.019	0.558 ± 0.009	0.520
	☐		1	0.667 ± 0.014	0.543 ± 0.017	0.275 ± 0.020	0.680 ± 0.015	
	☐		2	0.674 ± 0.025	0.576 ± 0.023	0.272 ± 0.014	0.699 ± 0.006	
	☐		3	0.672 ± 0.028	0.580 ± 0.022	0.297 ± 0.019	0.705 ± 0.006	
	☐	$\frac{1}{\sqrt{d}}$	0	0.570 ± 0.013	0.454 ± 0.006	0.181 ± 0.023	0.613 ± 0.007	$\underline{0.538}$
	☐		1	0.671 ± 0.015	0.578 ± 0.031	0.299 ± 0.030	0.685 ± 0.010	
	☐		2	0.671 ± 0.031	0.593 ± 0.016	0.301 ± 0.019	0.704 ± 0.007	
	☐		3	0.681 ± 0.024	0.590 ± 0.014	$\underline{0.315}$ ± 0.009	0.702 ± 0.011	
	■	1	0	0.545 ± 0.017	0.415 ± 0.014	0.155 ± 0.019	0.558 ± 0.009	0.469
	■		1	0.679 ± 0.012	0.582 ± 0.012	0.259 ± 0.010	0.680 ± 0.016	
	■		2	0.668 ± 0.015	0.580 ± 0.041	0.284 ± 0.024	$\underline{0.711}$ ± 0.005	
	■		3	0.679 ± 0.018	0.000 ± 0.000	0.000 ± 0.000	$\underline{0.711}$ ± 0.009	
	■	$\frac{1}{\sqrt{d}}$	0	0.570 ± 0.013	0.454 ± 0.006	0.181 ± 0.023	0.613 ± 0.007	0.512
	■		1	0.675 ± 0.019	0.578 ± 0.022	0.280 ± 0.022	0.682 ± 0.006	
	■		2	$\underline{0.687}$ ± 0.021	$\underline{0.609}$ ± 0.012	0.308 ± 0.012	0.702 ± 0.011	
	■		3	0.678 ± 0.009	0.476 ± 0.266	0.000 ± 0.000	0.701 ± 0.006	

B.5 LayerNorm and parameter initialization

In this section, we analyze the effects of using a Xavier normal distribution ($I_{par} = \mathcal{N}$)[4] for the weights of the two projections MLP$^{(edge-head)}$ and MLP$^{(edge-dept)}$, along with the edge biaffine layer $f^{(edge)}(\,\cdot\,; W_e)$. We also apply a LayerNorm function $\texttt{LN}_\psi$ [2] for each of the BiLSTM layers.

As reported in Table 8, on average the best results are obtained with the base setting, i.e. with $I_{par} = \mathcal{U}$ and $\texttt{LN}_\psi = \times$. Using LayerNorm can provide a performance boost for some datasets, especially with uniform initialization. However, it makes performance unstable for some. As a matter of fact, when using LayerNorm with three BiLSTM layers on CoNLL04 and SciERC, the model often breaks and is not able to converge. Based on average performance, then, the best models are obtained by scaling biaffine scores ($a = 1/\sqrt{d}$) and initializing the parser with a uniform weight distribution, without any LayerNorm in between the stacked BiLSTM layers.

[4]https://docs.pytorch.org/docs/stable/nn.init.html#torch.nn.init.xavier_normal_

Table 9: Performance for the fully fine-tuned BERT and DeBERTa models ($h_\psi = 400$, $h_{out} = 500$, $\phi = ✓$ and $\mathbf{e}_i^{tag} = ✓$).

Model	a	**ADE**	**CoNLL04**	**SciERC**	**ERFGC**	**Mean**
BERT$_{base}$	1	0.748 ± 0.028	0.613 ± 0.019	0.414 ± 0.011	0.726 ± 0.006	0.321
	$\frac{1}{\sqrt{d}}$	0.731 ± 0.025	0.629 ± 0.022	0.412 ± 0.016	0.726 ± 0.006	0.321
BERT$_{large}$	1	0.748 ± 0.016	0.700 ± 0.019	0.473 ± 0.011	0.750 ± 0.006	0.340
	$\frac{1}{\sqrt{d}}$	0.777 ± 0.011	0.697 ± 0.014	0.446 ± 0.025	0.748 ± 0.010	0.341
DeBERTa$_{base}$	1	0.754 ± 0.013	0.674 ± 0.014	0.429 ± 0.015	0.751 ± 0.002	0.332
	$\frac{1}{\sqrt{d}}$	0.761 ± 0.021	0.700 ± 0.013	0.425 ± 0.019	0.751 ± 0.010	0.338
DeBERTa$_{large}$	1	0.786 ± 0.010	0.739 ± 0.016	0.478 ± 0.019	0.763 ± 0.006	0.352
	$\frac{1}{\sqrt{d}}$	0.794 ± 0.015	0.747 ± 0.013	0.476 ± 0.010	0.763 ± 0.007	0.353

B.6 Full fine-tuning and sample efficiency

In this section, we present the results obtained when fine-tuning BERT$_{base}$, DeBERTa$_{base}$, BERT$_{large}$, and DeBERTa$_{large}$ over $10k$ steps. In our previous settings, we only used a frozen BERT$_{base}$, whose last hidden states we fed as input to the tagger and parser. This evaluation allows us to test the effectiveness of our approach in a setting in which the number of learnable parameters is unconstrained. In this experiment, we use different learning rates for the base models ($\eta = 1 \times 10^{-4}$) and the large models ($\eta = 3 \times 10^{-5}$) and we apply gradient norm clipping with $\|\nabla\|_{max} = 1.0$. In the case of the large models, we also use a cosine schedule with warm-up over 6% of the steps. We adopt these measures because during early trials we experienced sudden mid-run gradient explosions. Note that, in this case, we do not use any downstream BiLSTMs and only vary whether we use biaffine score normalization in the parser.

Table 9 reports the performance in terms of micro-averaged F$_1$-measure for the best models, picked based on top validation performance, evaluated every 100 steps. As the table shows, normalizing the scores generally does not increase top performance when fully fine-tuning these models. This is in line with our theoretical results, since tuning all of their ~$100 - 400M$ parameters can compensate for the lack of biaffine score normalization.

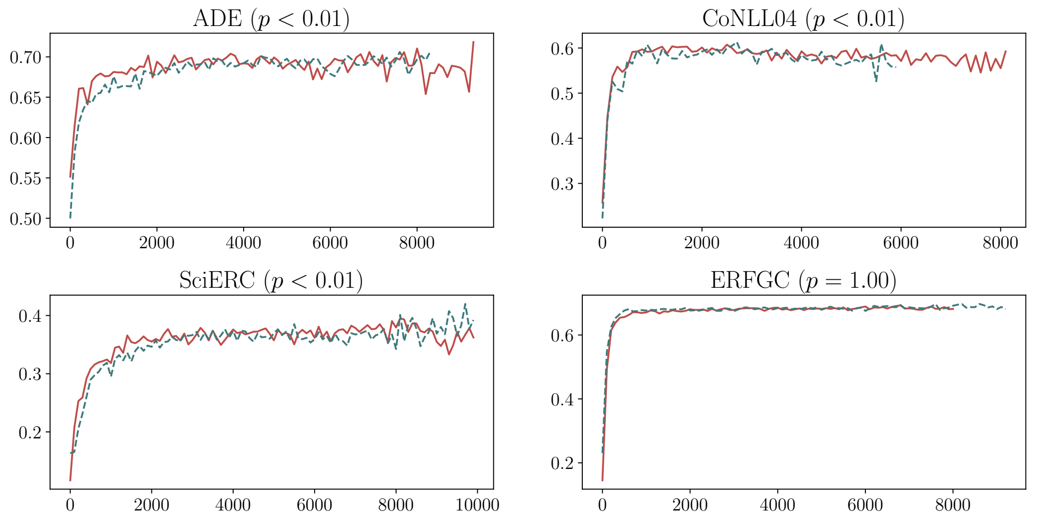
In order to have a stronger indication of whether score normalization also works in this setting, we study the performance on the test set versus the amount of training steps. Figure 6 and Figure 7 respectively visualize the performance of the base and large models on the test set in terms of micro-averaged F$_1$-measure over $10k$ training steps. We still use early stopping, which is why some of the series are cut short before reaching $10k$ steps.

As Figure 6 shows, a one-tailed Wilcoxon signed-rank test finds the difference in performance throughout the training to be significantly higher when normalizing scores. Since the average test performance of the models is similar with and without normalization, the gap between the two curves being statistically significant indicates faster convergence with normalization.

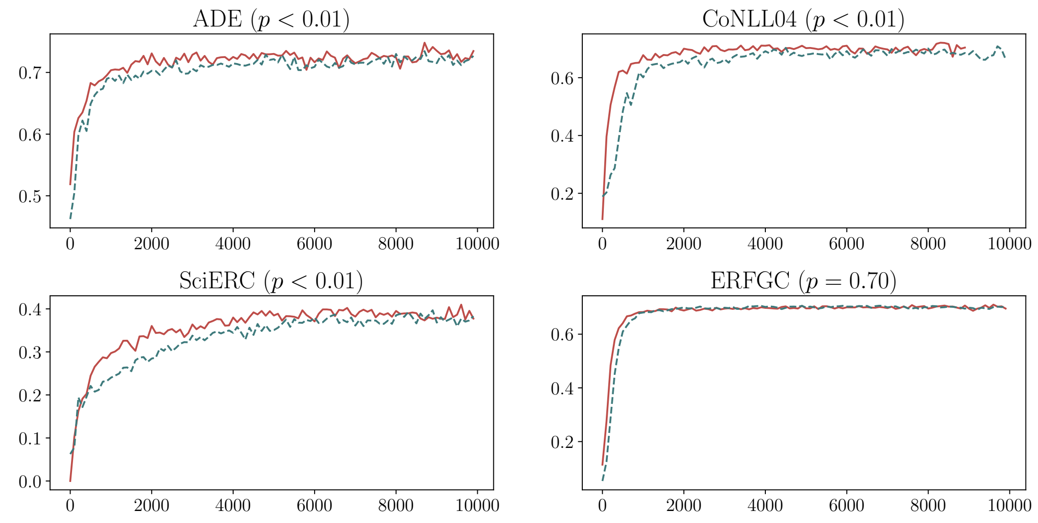
For both BERT$_{base}$ and DeBERTa$_{base}$, the difference in convergence speed and performance is statistically significant on ADE, CoNLL04, and SciERC. The same is true for BERT$_{large}$ only on ADE. However, the effect is statistically significant with DeBERTa$_{large}$ for all datasets. This indicates our score normalization approach can produce positive effects in terms of sample efficiency also when fine-tuning models with hundreds of millions of parameters.

C Computational resources

We ran all of our experiments on a cluster of NVIDIA H100 (96GB of VRAM) and NVIDIA L40 (48GB of VRAM) GPUs, one run per single GPU. When freezing the BERT$_{base}$ encoder and only training the BiLSTMs and the classifiers, each training and evaluation run took ~5-7 minutes, depending on the number of BiLSTM layers. For the main setting, finding the best hyperparameters involved training and testing 6,120 models, for a total of ~600 GPU hours. When carrying out the



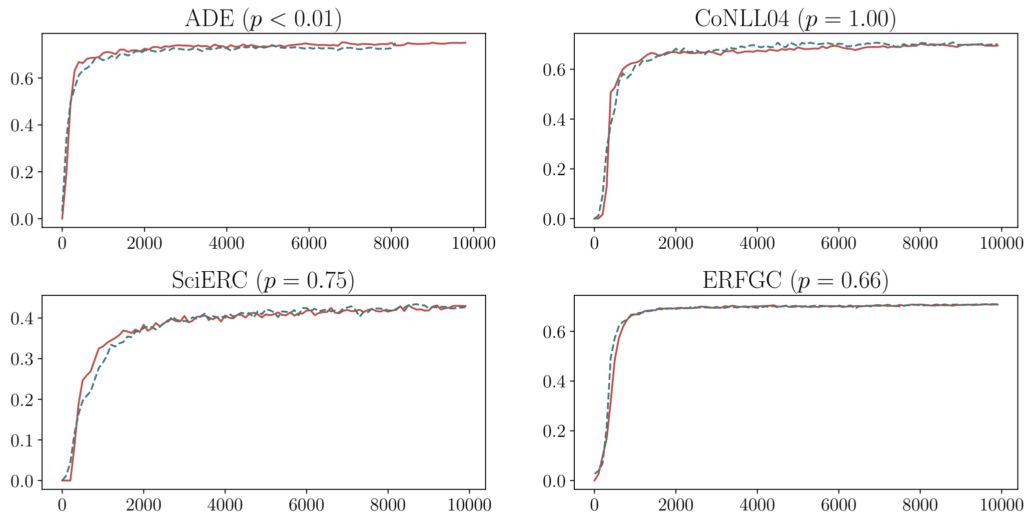
(a) BERT$_{base}$



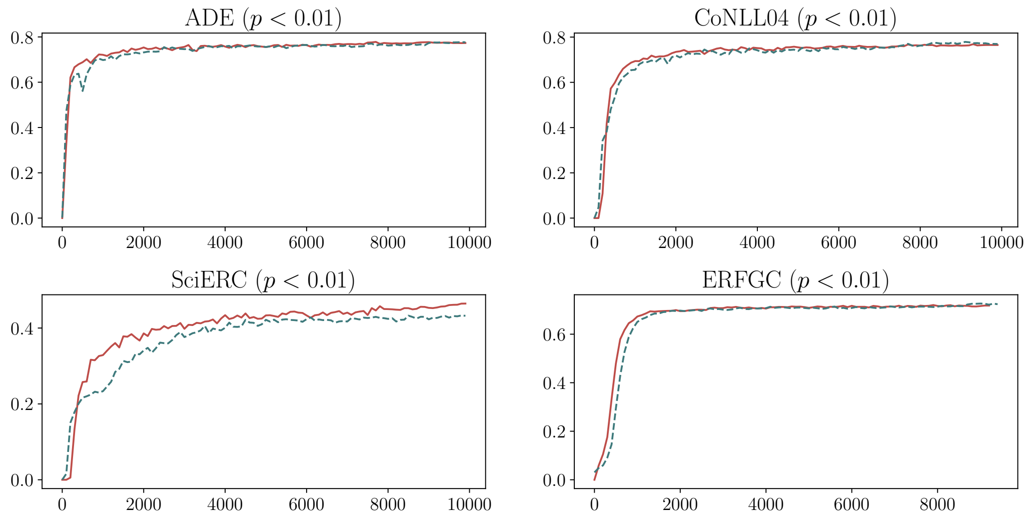
(b) DeBERTa$_{base}$

Figure 6: Performance in terms of F$_1$-measure vs the number of training steps for the base models on the SemDP datasets. Red = norm; blue = raw. The p-values refer to the performance being greater with score normalization (one-tailed Wilcoxon signed-rank test).

full fine-tuning ablation (Appendix B.6), training and evaluation took ~1 hour for each of the 40 base models and ~2-3 hours for each of the 40 large models, for an additional ~140 GPU hours.



(a) BERT$_{large}$



(b) DeBERTa$_{large}$

Figure 7: Performance in terms of F$_1$-measure vs the number of training steps for the large models on the SemDP datasets. Red = norm; blue = raw. The p-values refer to the performance being greater with score normalization (one-tailed Wilcoxon signed-rank test).

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We believe our abstract and introduction faithfully report the claims made in the rest of the paper, i.e. we show theoretically and empirically that normalizing biaffine scores for dependency parsing in NLP provides higher parsing performance.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: In our paper, we discuss that we limit ourselves to the task of dependency parsing in NLP. In addition, in the conclusions we mention that our approach does not use k-hop parsers, for example with GNNs or triaffine transformations.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: In Section 3 we provide a lengthy introduction to Result 1, with additional information right after it, which provides more context.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: All of the hyperparameters and methodology are provided to the reader and the experiments can be easily reproduced through the provided GitHub repository.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The code is provided. The repository includes a README.md file which guides the reader through its usage. The data for ADE, CoNLL04, and SciERC can be downloaded the link provided in Section 4.1. As regards ERFGC, [36] provide access to the corpus upon request. UD 2.2 enEWT and SciDTB are openly available.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Yes, we report information about the data we use in Section 4.1. Training and evaluation information is provided in Sections 4.2 to 4.4.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Yes, all of our tables and diagrams provide standard deviations and error bars, calculated from model training runs carried out with five different seeds.

   Guidelines:

   - The answer NA means that the paper does not include experiments.

- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Appendix C provides all relevant information with relation to the hardware used, its characteristics, and the time necessary to reproduce the experiments.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We believe our work does NOT raise any ethics concerns with regard to any of the guidelines listed within the NeurIPS Code of Ethics. We have no human participants, we use publicly available data, and we believe our work does not have any broader potentially unintended societal impact.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [NA]

Justification: As mentioned at point #9, we do not believe that the Code of Ethics is relevant for our work, reason for which we do not discuss any societal impacts in relation with our study.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: We do not release any new data and/or pre-trained language models.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We explicitly cite all of the used datasets and models and discuss their availability.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.

- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: We do not release any new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: Our work does not involve any human subjects in any manner.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [Yes]

    Justification: Our work does not involve the participation of any human subjects in any manner.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: We only use LLMs for the construction of pandas tables and matplotlib visualizations.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.