When Chosen Wisely, More Data Is What You Need: A Universal Sample-Efficient Strategy For Data Augmentation

Anonymous ACL submission

Abstract

Data Augmentation (DA) is vital in deep learning to improve the generalizability of neural networks. Most of existing DA techniques in NLP naively add a certain number of augmented samples without paying attention to the quality and added computational cost of these samples. Furthermore, state-of-the-art DA techniques in the literature usually learn to generate or re-weight augmented samples more specific to the main task; however, these learning-based DA techniques are not sample-efficient and they are computationally expensive. In this work, we propose a universal DA technique, called *Glitter*, for NLP which aims at efficiency and performance at the same time. In other words, Glitter can be applied to any existing DA technique to improve its training efficiency and sample efficiency and maintain its competitive performance. We evaluate Glitter on several downstream tasks such as the GLUE benchmark, SQuAD, and HellaSwag in a variety of scenarios including general single network, consistency training, self-distillation and knowledge distillation (KD) setups.

1 Introduction

Data Augmentation (DA) is shown to be effective in improving generalization of deep neural networks (Xie et al., 2019; DeVries and Taylor, 2017) and in increasing the number of training samples especially in low resource data regimes (Zhang et al., 2017; Sennrich et al., 2016). The undeniable importance of data in deep learning (Sambasivan et al., 2021; Rogers, 2021) and the costly process of data annotation has propelled researchers into leveraging DA in a broad range of applications from computer vision (Cubuk et al., 2018; Wang et al., 2020) to natural language processing (NLP) including machine translation (Sennrich et al., 2016; Shen et al., 2020), language understanding (Shen et al., 2020; Qu et al., 2020; Du et al., 2021; Kamalloo et al., 2021), and question answering (Alberti et al., 2019; Longpre et al., 2019; Shakeri et al., 2020). In NLP, however, the discrete nature of text poses additional challenges in DA mainly because generating semantically viable text from another text is still formidable (Feng et al., 2021).

DA methods in NLP can be broadly categorized into *task-aware* and *task-agnostic* methods. Task-agnostic DA methods essentially generate augmented text regardless of the task at hand and often do not warrant additional training or fine-tuning. They can be based on some hand-crafted heuristics (Zhang et al., 2015; Wei and Zou, 2019), back-translation (Sennrich et al., 2016; Edunov et al., 2018), or token replacement from a pre-trained language model (Kobayashi, 2018; Wu et al., 2019; Ng et al., 2020). Although employing task-agnostic methods is straightforward, these methods do not take into account any task-specific information, and thus, their performance is usually limited. On the other hand, task-aware DA methods are capable of generating augmented samples, conditioned on the downstream task objective (Hu et al., 2019; Rashid et al., 2021; Xie et al., 2019). These methods adapt augmented examples specifically for a task—i.e., producing augmented examples either partly or fully take place during training. Notwithstanding their advantage over task-agnostic methods, they often incur additional training costs, resulting in prohibitively slow and computationally expensive training.

We stipulate the central problems surrounding DA techniques in NLP are as follows: First, DA methods are mostly not sample efficient—i.e. they generate and add arbitrary number of augmented samples to the training data and naively incorporate all of them into training without investigating how many of augmented samples are actually needed. Second, although more effective, task-aware methods are notoriously time-consuming to train. This is especially problematic in large-scale datasets such as SQuAD (Rajpurkar et al., 2016)

and MNLI (Williams et al., 2018). Third, most DA methods are not universal—i.e. they work in a particular setup for instance only with a single-network (Xie et al., 2019), or only with a teacher-student KD (Rashid et al., 2021). Overall, the importance of both sample efficiency and training efficiency for DA has been often overlooked in the literature.

Motivated by the above problems, in this work, we introduce a universal DA method, Glitter ✨[1], which can be plugged into any DA method to make them sample efficient, task-aware, while retaining performance (if not improving). Methodology of our work involves the minimax approach in which first a pool of augmented samples is generated and then a subset of them with maximum expected loss is selected to be involved in minimizing the corresponding task objective. Our key contributions in this paper can be summarized as follows:

1. Glitter is a universal solution which can be effortlessly applied to most existing DA techniques to improve upon training and sample efficiency while maintaining (or even boosting) their performance.

2. We show how Glitter can be adapted to a variety of scenarios such as general single network, consistency training, self-distillation and knowledge transfer (Teacher-Student) setups.

3. We empirically demonstrate that Glitter exhibits superior efficiency and performance over state-of-the-art DA methods on several downstream tasks including GLUE, SQuAD, and HellaSwag as well as on out-of-domain datasets.

2 Related Work

2.1 Task-agnostic DA in NLP

Contextual augmentation techniques (Kobayashi, 2018; Wu et al., 2019) use contextual language models for DA. Kobayashi (2018) propose bidirectional language models for word substitution conditioned on the label of their input text. SSMBA (Ng et al., 2020) and TinyBERT (Jiao et al., 2019) perturb the input by masking some parts, and then use a pre-trained BERT model to substitute those masked words of the input text to generate augmented samples. Back-Translation Sennrich et al. (2016) augments data using two consecutive translation models: the first model to translate input samples from their current language to an arbitrary target language; then, a second model to translate the result back to the original language. Mixed-up (Guo et al., 2019) generates augmented samples based on interpolating word embedding and sentence embedding vectors. Shen et al. (2020) introduce a set of *cut-off* techniques (i.e. zeroing out some parts of the embedding matrix) at token level, feature level and span level of the input data. Then, they define a particular loss function to enforce consistency of predictions for these set of perturbed inputs. EDA (Wei and Zou, 2019) consists of some simple augmentation operations such as replacing synonyms, deleting, inserting and swapping random words.

2.2 Task-aware DA in NLP

While most of DA techniques consider augmented samples uniformly in the training process, (Yi et al., 2021) propose a re-weighting technique to take the importance of different augmented samples into account. Although re-weighting is shown to be effective in improving the performance of DA techniques, it is not sample efficient. Wu et al. (2019) introduced c-BERT which is a masked input augmentation technique obtained by applying a label-aware fine-tuning to a pre-trained BERT model on the task labeled data. Qu et al. (2020), in CODA, combine different label preserving transformations to produce high quality augmented samples. One effective case in point is obtained by applying adversarial training over back-translation transformations with a consistency regularized loss. Unsupervised DA (UDA) (Xie et al., 2019) uses off-the-shelf DA techniques without modifying them and then adds an auxiliary *consistency loss* to the training objective. However, UDA is not sample efficient and it is designed for single network augmentation setup only (i.e. it is not clear how we can deploy it in scenarios such as the KD setup). Hu et al. (2019) proposed a reinforcement learning-based technique in which the reward function is defined according to whether the data generated by the augmenter network preserves the label of the original training sample.

[1] Inspired by "All that is gold does not glitter" J.R.R. Tolkien

2.3 DA for KD

KD (Hinton et al., 2015; Buciluǎ et al., 2006) is a promising model compression technique aiming at transferring the knowledge of an already trained network (so-called teacher) to a smaller or same-size student network. It is shown that DA can boost KD's performance in NLP significantly; for example, TinyBERT (Jiao et al., 2019) uses a task-agnostic DA technique for its task specific fine-tuning. However, (Kamalloo et al., 2021) and (Rashid et al., 2021) showed that DA module can be tailored for KD. For example, (Rashid et al., 2021) in MATE-KD, tune a separate masked language model in order to generate maximum divergence augmented samples. (Kamalloo et al., 2021; Du et al., 2021) use a KNN retrieval-based technique to extract their augmented samples form a large sentence bank.

Our evaluation strategy is to take the best models of each category and show that our universal Glitter can be applied to them to improve their original performance and at the same time make them sample efficient.

3 Methodology

In this section, we introduce our task-aware DA method, Glitter ✨, that aims at using an efficient number of augmented samples without sacrificing performance. Our proposed strategy is agnostic to DA methods—i.e., it can be plugged into any DA method with any training strategy to accomplish sample efficiency.

Existing learning-based DA methods train a separate DA model and adapt its output for a particular objective function that is entirely task-dependent:

$$\begin{aligned} \phi^* &\leftarrow \min_{\phi} \ell_{DA}(M(\Omega(x;\phi);\theta)) \\ x'^* &= \Omega(x;\phi^*) \end{aligned} \tag{1}$$

where $\ell_{DA}()$ is a loss function, geared towards the task objective, $\Omega(;\phi)$ is the DA model with trainable parameters ϕ, and $M(;\theta)$ refers to the original model, parameterized by θ.

In contrast to learning-based DA, we propose to generate many augmented candidates using any arbitrary DA method prior training, and select most suitable candidates during training. This procedure does not introduce additional trainable parameters into training, and more importantly, is capable of automatically ignoring unnecessary augmented examples. Let $(x_i, y_i)_{i=1}^N \in \{(\mathcal{X}, \mathcal{Y})\}$ represent training data such that a pair $x_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$ are an input example and its corresponding label. Suppose a pool of K augmented examples, $X'(i) = \{x'_k(i)\}_{k=1}^K$, are sampled from some DA model for each training example $(x_i, y_i) \in (\mathcal{X}, \mathcal{Y})$. Note that Glitter imposes no restrictions on how to augment training data; augmented samples can be generated via a single or even multiple DA models.

Sample Selection. Given a pool of augmented samples, our approach is to select the best candidates among them according to particular defined criteria. Inspired by the minimax approach (Farnia and Tse, 2016; Volpi et al., 2018), our selection mechanism is based on finding top k_1 (out of K) worst-case augmented samples from the X' set. Minimizing the main model loss function on these worst-case augmented samples will help improving generalization of the model (Volpi et al., 2018). In order to rank augmented samples, we evaluate $X'(i)$ based on a distance function with respect to the corresponding original training sample, x_i, within the model's latent space:

$$\begin{aligned} X'^*(i) &\leftarrow \text{top}_{k_1}\Big(\ell_{\text{eval}}\big(M(x_i;\theta), M(X'(i);\theta)\big)\Big) \\ X'^*(i) &= \{x'^*_j(i)\}_{j=1}^{k_1} \subset X'(i) \end{aligned} \tag{2}$$

where $\text{top}_{k_1}()$ denotes returns top-k_1 indices based on the scores returned by ℓ_{eval}, $X'^*(i)$ is the set of k_1 selected augmented samples for x_i; $\ell_{eval}()$ is the evaluation loss which is determined via the task objective.

Updating the Model Parameters. After obtaining the top k_1 augmented samples, we group them with the original training samples, $\{x_i\} \cup X'^*(i)$, and subsequently, update the model parameters only based on this selected set of augmented samples on the original loss:

$$\begin{aligned} \mathcal{L}(\theta) &= \sum_{i=1}^{N} \ell_{\text{task}}\Big(M(x_i;\theta), M(X'^*(i);\theta), y_i\Big) \\ \theta_t &\leftarrow \theta_{t-1} - \lambda \nabla_\theta(\mathcal{L}(\theta))|_{\theta_{t-1}} \end{aligned} \tag{3}$$

where N is the number of training samples, λ is the learning rate, and $\ell_{\text{task}}()$ is the final task loss—e.g., cross entropy (ce) for classification—that is computed over both original data and selected augmented data. In the remainder of this section, we

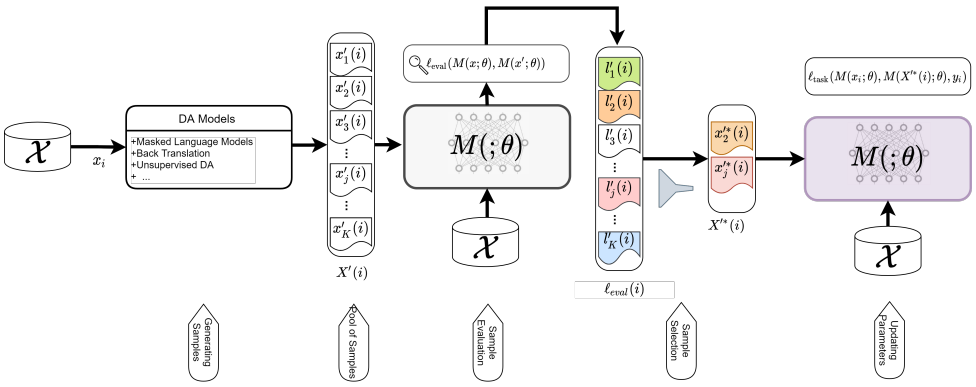


Figure 1: Illustration of Glitter (from left to right): first, generating augmented samples from different DA techniques; second, forming a pool of samples $X'(i)$; third, evaluating the augmented samples using the $\ell_{eval}()$ loss; fourth, filtering the top k_1 samples based on their corresponding $\ell_{eval}()$; fifth, updating the parameters of the model by minimizing the task loss $\ell_{\text{task}}(:\theta)$.

discuss how Glitter can be applied to various scenarios including general DA for single networks, and DA for the teacher-student (KD) setups.

3.1 General DA for Single Networks

We consider three potential setups for the single network scenario: (1) General single network, (2) Self-distillation, and (3) Consistency training.

General Single Network. In this setup, augmented samples are exploited in a semi-supervised manner where we can evaluate them based on the divergence of their predicted output $M(x'_k(i);\theta) = p(y|x'_k(i);\theta)$ from the ground-truth label or the prediction of the original corresponding training sample $M(x_i;\theta) = p(y|x_i;\theta)$ using the cross entropy loss, ℓ_{ce}:

$$\ell_{\text{eval}} = \ell_{ce}\big(y_i, M(x'_k(i);\theta)\big) \quad \text{or} \quad \ell_{\text{eval}} = \ell_{ce}\big(M(x_i;\theta), M(x'_k(i);\theta)\big). \tag{4}$$

Then, for the final task loss, ℓ_{task} we can deploy a standard cross entropy loss over both training samples and their corresponding selected augmented samples:

$$\ell_{\text{task}} = \ell_{ce}\big(y_i, M(x_i;\theta)\big) + \frac{1}{k_1}\sum_{x\in X'^*(i)} \ell_{ce}\big(y_i, M(x;\theta)\big). \tag{5}$$

Consistency Training (CT; Xie et al. 2019) In this configuration, we can employ the same ℓ_{eval} introduced in Eq. (4). As a result, our method naturally selects top k_1 most inconsistent augmented samples for each training sample. Then, the network is optimized to make predictions for input augmented samples that are consistent with predictions of their corresponding original training samples:

$$\ell^{\text{CT}}_{\text{task}} = \ell_{ce}\big(y_i, M(x_i;\theta_t)\big) + \frac{1}{k_1}\sum_{x\in X'^*(i)} \ell_{ce}\big(M(x_i;\theta_{t-1}), M(x;\theta_t)\big). \tag{6}$$

As stated by Xie et al. (2019), the second term in Eq. (6) leverages the previous prediction of the network for each training example.

Self-Distillation (Self-KD) In Self-KD, we first train a model, and then, use it $(M(;\theta^*))$ as a teacher to train an identical model but initialized from scratch using KD (Furlanello et al., 2018). We follow §3.2 to adjust ℓ_{eval} and ℓ_{task}.

3.2 DA for Teacher-Student (KD)

In this setup, we have a teacher model, $T(;\psi^*)$ with parameters ψ that is already trained on the training data, along with a student model, $M(;\theta)$, which we aim to train. The selection criteria for augmented samples is to maximize the divergence between the teacher and the student:

$$\ell^{\text{KD}}_{\text{eval}} = \ell_{KL}\Big(T\big(x'_k(i);\psi^*\big), M\big(x'_k(i);\theta\big)\Big) \tag{7}$$

where ℓ_{KL} refers to the KL divergence. After selecting the maximum divergence augmented samples, then we calculate the KD loss as following:

$$\ell^{\text{KD}}_{\text{task}} = \alpha\, \ell_{ce}\big(y_i, M(x_i;\theta)\big) + (1-\alpha)\times \frac{1}{k_1+1}\sum_{x\in\{x_i\}\cup X'^*(i)} \ell_{KL}\big(T(x;\psi^*), M(x;\theta)\big) \tag{8}$$

where α is a hyperparameter.

4 Experiments

4.1 Setup

To incorporate unlabelled augmented data into training, we adopt CT (Xie et al., 2019) and KD (Hinton et al., 2015). To this end, we conduct experiments under two settings:

Standalone where we train a single model on the augmented data. In this setting, we seek to answer two questions: (1) How much is DA capable of improving the model generalization? (2) Is our proposal on sample efficiency helpful? For this purpose, we fine-tune RoBERTa$_{\text{base}}$ (Liu et al., 2019) using CT and Self-KD on augmented data.

Distilled where we distill DistilRoBERTa (Sanh et al., 2019) (student) from RoBERTa$_{\text{Large}}$ (Liu et al., 2019) (teacher) using the augmented data. Note that DA comes into play only during distilling the student model and the teacher is trained on the original data in advance. Our goal here is to investigate whether DA is an effective means in knowledge transfer to curb the capacity gap (Cho and Hariharan, 2019) between a large model and a small one.

In both settings, we take the best performing model on the development set and evaluate it on the test set (depicted by *Test*). Additionally, for the standalone model setting, we also report results on the development set when models are trained only for 5 epochs (depicted by *Dev*), similar to CoDA (Qu et al., 2021), to make a comparison with baselines. Our *Dev* results are an average of 10 runs with different seeds. The implementation details and hyperparameters are provided in §A.

4.1.1 DA Methods

We leverage three widely used textual augmentation methods:

1. **EDA** (Wei and Zou, 2019)[2]: We randomly replace 5% of the tokens with their synonyms and randomly delete up to 10%.

2. **Back-Translation (BT)** (Sennrich et al., 2016): We use fairseq (Ott et al., 2019) to translate sentences into German and then back into English. We do nucleus sampling (Holtzman et al., 2020) with $p = 0.9$ for both translations. We find that $p = 0.6$ works better on sentiment classification.

3. **Mask-and-reconstruct (MR)** (Yi et al., 2021; Ng et al., 2020): We randomly mask 15% of the tokens and construct a new sentence by sampling from a pre-trained BERT$_{\text{Large}}$ for masked tokens. We adopt top-k sampling with $k = 20$ to select new tokens. For MNLI, we obtain better results with top-10 sampling.

For each augmentation method, we generate 12 augmented examples per training instance for all datasets, except for large datasets—i.e., MNLI, QQP, and SQuAD—where the number of augmented examples are 8 per train example.

4.1.2 Baselines

Because our settings—i.e., standalone and distilled—are different in nature, we compare Glitter with different baselines for each setting. For both, Vanilla-DA that takes all augmented data into account without reservation is the first baseline.

The baselines for the standalone setting are: CoDA (Qu et al., 2021), MMEL (Yi et al., 2021), and HiddenCut (Chen et al., 2021). And for distilled, we consider MATE-KD (Rashid et al., 2021) a distillation strategy that generates suitable augmented data on-the-fly during training to maximize the loss.

4.2 GLUE

The GLUE benchmark (Wang et al., 2019) is a well-known suite of nine[3] tasks that aim at evaluating natural language understanding models. We present test results in the distilled mode in Table 1. Glitter consistently outperforms Vanilla-DA, while it is faster to train. Specifically, Glitter achieves parity with Vanilla-DA for EDA in terms of the overall average score, while scoring +0.2% and +0.4% higher for BT and MR, respectively. We observe that only in few cases Vanilla-DA negligibly outperforms Glitter—e.g., on MRPC, and STS-B for BT. Nonetheless, Glitter *8x/1x* trains 50% faster than Vanilla-DA *8x* on average, and 30% faster for *8x/2x*. Also, Glitter surpasses MATE-KD by +0.2% in the overall score. Unlike Glitter, MATE-KD introduces additional parameters to the model during

[2] https://github.com/makcedward/nlpaug

[3] We excluded WNLI since our DA methods are not essentially designed for this task.

Method	CoLA Mcc	SST Acc	MRPC Acc/F_1	STS-B P/S	QQP Acc/F_1	MNLI-m/mm Acc	QNLI Acc	RTE Acc	Avg.
RoB_{Large} (teacher)	63.8	96.8	90.6	92.4	81.5	90.3/89.8	94.8	88.3	87.3
$BERT_{Large}$ ♣	60.5	94.9	87.4	87.1	80.7	86.7/85.9	92.7	70.1	82.5
DistilRoB	55.2	93.9	85.9	86.0	80.3	84.0/83.1	90.6	73.6	81.1
KD	54.9	94.0	86.8	87.3	80.5	85.1/83.7	91.9	73.5	81.7
Task-Aware DA									
MATE-KD ♣	56.0	94.9	**90.2**	**88.0**	**81.2**	85.5/84.8	92.1	**75.0**	82.8
EDA (Wei and Zou, 2019)									
Vanilla-DA (8x)	55.5	94.8	87.6	86.1	80.7	85.3/84.7	92.0	72.8	81.8
Glitter	54.5	**95.1**	87.5	86.5	80.4	85.4/84.8	92.1	73.2	81.8
	8x/2x	*8x/1x*	*8x/2x*	*8x/2x*	*8x/2x*	*8x/2x*	*8x/2x*	*8x/1x*	
Back-Translation									
Vanilla-DA (8x)	53.4	**95.1**	88.5	87.5	80.9	85.9/**85.9**	92.2	73.5	82.1
Glitter	54.9	**95.1**	88.4	87.3	80.9	86.2/85.3	92.2	73.7	82.3
	8x/2x	*8x/1x*	*8x/1x*	*8x/2x*	*8x/2x*	*8x/2x*	*8x/2x*	*8x/2x*	
Mask-and-reconstruct									
Vanilla-DA (8x)	58.8	94.5	88.7	87.0	80.9	85.8/84.9	91.8	74.0	82.6
Glitter	**59.2**	**95.1**	89.2	87.6	81.0	**86.6**/84.8	**92.4**	74.1	**83.0**
	8x/1x	*8x/1x*	*8x/2x*	*8x/1x*	*8x/2x*	*8x/2x*	*8x/2x*	*8x/2x*	

Table 1: Test results of the distilled experiment on GLUE. (♣) denotes results are taken verbatim from: $BERT_{Large}$ (Devlin et al., 2019), and MATE-KD (Rashid et al., 2021). **Bold** and underlined numbers indicate the best and the second best results across the DA methods.

Method	CoLA Mcc	SST Acc	MRPC Acc/F_1	STS-B P/S	QQP Acc/F_1	MNLI-m Acc	QNLI Acc	RTE Acc	Avg.
RoBERTa	61.9	95.4	88.6	89.3	80.4	87.6	93.0	81.6	84.7
Self-KD	61.7	95.7	89.0	89.0	80.8	**88.3**	93.0	81.7	84.9
+ Vanilla-DA	61.5	**96.1**	88.9	**89.7**	81.0	88.0	92.9	81.1	84.9
	8x	*8x*	*8x*	*8x*	*8x*	*8x*	*8x*	*12x*	
+ Glitter	62.5	96.0	**89.8**	89.5	**81.1**	88.1	**93.5**	**82.3**	**85.4**
	8x/1x	*8x/2x*	*8x/2x*	*8x/2x*	*8x/2x*	*8x/2x*	*8x/2x*	*12x/1x*	
CT + Vanilla-DA	59.4	95.6	89.0	85.8	80.3	82.5	92.0	80.2	83.1
	8x	*8x*	*8x*	*10x*	*8x*	*8x*	*8x*	*10x*	
CT + Glitter	**62.7**	95.8	89.2	87.9	80.9	84.1	92.9	81.8	84.4
	8x/1x	*8x/1x*	*8x/1x*	*10x/1x*	*8x/2x*	*8x/2x*	*8x/2x*	*10x/1x*	

Table 2: Test result of the standalone experiments on GLUE using $RoBERTa_{base}$.

training and also, it trains drastically slower because it generates augmented examples on-the-fly. Moreover, Table 1 illustrates that MR yields the best test results across the three DA methods except for SST that BT leads to better results. Based on this observation, we report results on MR augmented data for all GLUE datasets except for SST in the remainder of our experiments.

For the standalone mode, Tables 2 and 3 present the results on test and dev, respectively. Similar to distilled, Glitter outperforms Vanilla-DA by +0.5% for both self-KD and CT. Self-KD yields better results than CT on all GLUE tasks except CoLA. CT falls short on most GLUE tasks, compared to no DA results—i.e., top-2 rows in Table 2. This is why, we only evaluated Glitter with self-KD on the dev data. Glitter achieves superior performance gains, compared to all three baselines on all datasets except QNLI. The key advantage of Glitter is that the training procedure remains intact since our method is data-centric.

4.2.1 Out-of-Domain Generalization

We also evaluate Glitter on OOD datasets. To this end, we test our models, already trained on GLUE tasks, on OOD datasets whose data distribution differs from the original data. In particular, here are our selected OOD datasets:

- SST: IMDb (Maas et al., 2011), IMDb-Cont. (Gardner et al., 2020), and IMDb-CAD (Kaushik et al., 2020), as done in Chen et al. (2021). Although both SST and IMDb datasets are collected on movie reviews, IMDb reviews tend to be substantially longer than SST sentences.

- STS-B: SICK (Marelli et al., 2014), a semantic relatedness dataset, created from image

Method	SST Acc	MRPC F_1	MNLI-m Acc	QNLI Acc	RTE Acc	IMDb-Con. Acc	A-NLI Acc	HANS Acc
RoB♠	94.8	90.2	87.6	92.8	78.7	-	-	-
CoDA♠	95.3	91.7	88.1	93.6	82.0	-	-	-
HiddenCut♠	95.8	92.0	**88.2**	**93.7**	83.4	87.8	**32.8**	71.2
MMEL†	94.6 ± 0.8	91.9 ± 0.4	88.1 ± 0.1	93.2 ± 0.1	85.3 ± 1.0	90.5 ± 0.7	31.4 ± 0.6	74.5 ± 0.6
RoB†	94.3 ± 0.1	91.6 ± 0.5	87.7 ± 0.1	92.8 ± 0.2	84.5 ± 0.8	90.0 ± 0.4	30.8 ± 0.9	73.6 ± 0.7
Self-KD	94.3 ± 0.2	91.5 ± 0.3	87.9 ± 0.1	92.9 ± 0.2	84.0 ± 0.6	90.3 ± 0.5	30.9 ± 0.4	73.5 ± 0.7
+ Vanilla-DA	95.4 ± 0.5	92.0 ± 0.3	**88.2 ± 0.1**	93.4 ± 0.1	84.4 ± 0.7	90.2 ± 0.4	31.3 ± 0.5	73.9 ± 0.4
+ Glitter ✨	**95.7 ± 0.2**	**92.2 ± 0.5**	**88.2 ± 0.1**	93.4 ± 0.1	**85.6 ± 0.7**	**90.6 ± 0.2**	31.8 ± 0.4	**74.6 ± 0.3**

Table 3: Dev results of the standalone experiment on GLUE using $\text{RoBERTa}_{\text{base}}$. (♠) denotes results are taken verbatim from: RoB and CoDA (Qu et al., 2021), and HiddenCut (Chen et al., 2021). (†) indicates the results are obtained from our implementation of MMEL (Yi et al., 2021).

and video captions. SICK and STS-B are collected on roughly identical domains, but from different sources.

- QQP: PAWS_{QQP} (Zhang et al., 2019), analogous to Chen et al. (2021), and MQP (McCreery et al., 2020), a medical question similarity dataset.

- MNLI: SciTail (Khot et al., 2018), collected from school-level science questions, and similar to Chen et al. (2021), A-NLI (Nie et al., 2020), and HANS (McCoy et al., 2019).

- RTE: HANS (McCoy et al., 2019).

Table 9 in §B.1 showcases the OOD results for the distilled mode. Glitter outperforms Vanilla-DA in most cases, and is on par with it for virtually the rest. The only exceptions are IMDb-Cont., MQP, and PAWS_{QQP} where Vanilla-DA outperforms Glitter by almost 1% on average. Also, all models do not generalize well to PAWS_{QQP} and A-NLI because their performance is below a majority-class performance. Moreover, a fine-tuned DistilRoBERTa achieves the best OOD performance on HANS, highlighting that DA is not actually helpful for OOD accuracy on HANS.

Table 3 (the right side) reports the OOD results for standalone models. The complete results are presented in §B.2—i.e., Table 10 on test and Table 11 on dev. Glitter overwhelmingly outperforms all the baselines with a few exceptions. In the dev results, the fine-tuned model with no DA achieves the best OOD generalization on IMDb, and SciTail, while HiddenCut scores the highest on A-NLI with a 1% margin. Similarly, in the test results, Self-KD with no DA outperform Glitter on IMDb, IMDb-CAD, and SciTail.

Method	SQuAD EM/F_1	HellaSwag Acc
$\text{RoB}_{\text{Large}}$	88.9/94.6	85.2
DistilRoB	80.9/87.9	42.9
KD	81.1/88.2	42.5
+ Vanilla-DA *(8x)*	81.8/89.1	41.8
+ Glitter ✨ *(8x/2x)*	**83.6/90.3**	**44.1**

Table 4: Dev results of the distilled experiment on two downstream tasks.

4.3 HellaSwag

HellaSwag (Zellers et al., 2019) is a dataset for situated commonsense reasoning that involves picking the best ending given a context. We augment contexts in HellaSwag using only BT to ensure that the choices remain meaningful for the augmented contexts. Because our standalone results have been consistent with the distilled results, we report our results only in the distilled mode. According to our results demonstrated in Table 4, Glitter comfortably surpasses Vanilla-DA by a +2.3% margin.

4.4 SQuAD

SQuAD (Rajpurkar et al., 2016) is a crowd-sourced reading comprehension benchmark that consists of more than 100K questions, derived from Wikipedia passages. The task objective is to extract an answer span from a given question/passage pair. We augment questions in SQuAD v1.1 using only BT to ensure that the answer can still be found in the given passage for the augmented questions. Analogous to HellaSwag, we report our results only in the distilled mode. As shown in Table 4, Glitter outperformas Vanilla-DA by +1.8% in exact-match accuracy on the development set.

We also evaluate our trained models under distribution shift by testing them on QA datasets from four different domains: Wikipedia, New

York Times, Reddit, and Amazon product reviews (Miller et al., 2020). The OOD results are presented in Table 5. Glitter is consistently superior to Vanilla-DA in all four domains.

5 Ablation Study and Discussion

In this section, we aim to answer the following questions:

- How does training time of Glitter fare against Vanilla-DA varying the number of augmentations?
- Does pre-processing augmented data to dispense with unnecessary examples affect DA performance?
- How many augmented examples are required for Glitter to work? (studied in §D)
- Is our selection strategy based on sorting of ℓ_{eval} in Glitter important? (studied in §D)

For this purpose, we conduct a detailed analysis on 4 GLUE tasks—i.e., SST, MRPC, QNLI, and RTE. We trained models based on Vanilla-DA and Glitter using Self-KD and tested them on the development set (the dev setting).

Runtime Analysis. Throughout our experiments in §4, we compare Glitter with Vanilla-DA when number of augmentations are similar for both methods—i.e., *8x*. A natural question is whether Vanilla-DA with fewer data can achieve parity with Glitter. To this end, we vary augmentation size from $1x$ to $8x$ and train different Vanilla-DA models on each augmented dataset. We measure average training time per epoch for all models. Figure 2 in §C illustrates dev accuracy as training time increases. The training speed of Glitter *8x/2x* is slightly faster than Vanilla-DA *6x* on SST, MRPC, and QNLI and for Glitter *8x/1x*, is faster than Vanilla-DA *4x* on RTE. Glitter is superior of the two on all datasets.

Effect of Pre-processing Augmented Data. We conjecture that Glitter does not need any data engineering on augmented examples to obtain preferable performance gains. However, Vanilla-DA may require some pre-processing by weeding out potentially noisy data to become more effective. To investigate this, we exploit two pre-processing techniques: **(1) Confidence-based filtering:** Augmented examples for which the model's confidence is below a minimum threshold β are discarded, **(2) Label-preserving augmentation (LP):** Augmented examples for which the model predicts a different label than the original example are discarded. The results, reported in Table 6, show no meaningful performance gains by these pre-processing techniques. For Vanilla-DA, minimum confidence threshold of 0.7 performs slightly better as it brings minor improvements on MRPC (+0.3%) and QNLI (+0.1%), but is still lower than Glitter. On the other hand, applying these techniques slightly deteriorates the performance of Glitter in almost all cases. The only improvements are +0.1% on QNLI for LP and β=0.7.

Method	Wiki EM	NYT EM	Reddit EM	Amzn EM
$\text{RoB}_{\text{Large}}$	84.4	85.9	76.6	74.4
DistilRoB	76.6	78.1	66.2	62.9
KD	76.5	78.7	65.7	63.0
+ Vanilla-DA	77.3	79.0	65.9	63.3
+ Glitter	**79.3**	**80.7**	**68.1**	**64.7**

Table 5: OOD results for models trained on SQuAD and tested on QA datasets from four different domains (Miller et al., 2020).

Method	SST Acc	MRPC F_1	QNLI Acc	RTE Acc
Vanilla-DA	95.1	92.2	93.3	84.8
$\beta = 0.7$	95.1	92.5	93.4	84.8
$\beta = 0.9$	95.0	92.2	93.3	83.8
LP	94.8	92.4	93.3	84.8
Glitter	95.8	92.8	93.4	85.9
$\beta = 0.7$	95.0	91.5	93.5	85.2
$\beta = 0.9$	95.0	92.5	93.3	84.1
LP	95.1	92.2	93.5	85.9

Table 6: Dev results of self-KD exhibiting the effectiveness of different pre-processing techniques to filter augmented examples on 4 GLUE tasks. β and LP depict a minimum confidence threshold, and label preserving, respectively.

6 Conclusion

In this work, we highlighted the importance of training efficiency and sample efficiency of DA techniques in NLP. We proposed *Glitter* and showed that it is a universal DA technique which can be applied to any DA technique to make them sample efficient and training efficient without requiring to train the main DA model. We evaluated Glitter on different NLU tasks and in different scenarios such as general single network, consistency training, self-distillation and KD setups and demonstrated their promising results.

References

Chris Alberti, Daniel Andor, Emily Pitler, Jacob Devlin, and Michael Collins. 2019. Synthetic QA corpora generation with roundtrip consistency. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6168–6173, Florence, Italy. Association for Computational Linguistics.

Cristian Buciluǎ, Rich Caruana, and Alexandru Niculescu-Mizil. 2006. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 535–541.

Jiaao Chen, Dinghan Shen, Weizhu Chen, and Diyi Yang. 2021. HiddenCut: Simple data augmentation for natural language understanding with better generalizability. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4380–4390, Online. Association for Computational Linguistics.

Jang Hyun Cho and Bharath Hariharan. 2019. On the efficacy of knowledge distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4794–4802.

Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. 2018. Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. 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: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Terrance DeVries and Graham W Taylor. 2017. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*.

Jingfei Du, Edouard Grave, Beliz Gunel, Vishrav Chaudhary, Onur Celebi, Michael Auli, Veselin Stoyanov, and Alexis Conneau. 2021. Self-training improves pre-training for natural language understanding. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5408–5418, Online. Association for Computational Linguistics.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding back-translation at scale. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 489–500, Brussels, Belgium. Association for Computational Linguistics.

Farzan Farnia and David Tse. 2016. A minimax approach to supervised learning. *Advances in Neural Information Processing Systems*, 29:4240–4248.

Steven Feng, Varun Gangal, Jason Wei, Sarath Chandar, Soroush Vosoughi, Teruko Mitamura, and Eduard Hovy. 2021. A survey of data augmentation approaches for NLP. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 968–988, Online. Association for Computational Linguistics.

Tommaso Furlanello, Zachary Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. 2018. Born again neural networks. In *Proceedings of the 35th International Conference on Machine Learning*, pages 1607–1616. PMLR.

Matt Gardner, Yoav Artzi, Victoria Basmov, Jonathan Berant, Ben Bogin, Sihao Chen, Pradeep Dasigi, Dheeru Dua, Yanai Elazar, Ananth Gottumukkala, Nitish Gupta, Hannaneh Hajishirzi, Gabriel Ilharco, Daniel Khashabi, Kevin Lin, Jiangming Liu, Nelson F. Liu, Phoebe Mulcaire, Qiang Ning, Sameer Singh, Noah A. Smith, Sanjay Subramanian, Reut Tsarfaty, Eric Wallace, Ally Zhang, and Ben Zhou. 2020. Evaluating models' local decision boundaries via contrast sets. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1307–1323, Online. Association for Computational Linguistics.

Hongyu Guo, Yongyi Mao, and Richong Zhang. 2019. Augmenting data with mixup for sentence classification: An empirical study. *arXiv preprint arXiv:1905.08941*.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *International Conference on Learning Representations*.

Zhiting Hu, Bowen Tan, Ruslan Salakhutdinov, Tom Mitchell, and Eric P Xing. 2019. Learning data manipulation for augmentation and weighting. *arXiv preprint arXiv:1910.12795*.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2019. Tinybert: Distilling bert for natural language understanding. *arXiv preprint arXiv:1909.10351*.

Ehsan Kamalloo, Mehdi Rezagholizadeh, Peyman Passban, and Ali Ghodsi. 2021. Not far away, not so close: Sample efficient nearest neighbour data augmentation via MiniMax. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3522–3533, Online. Association for Computational Linguistics.

Divyansh Kaushik, Eduard Hovy, and Zachary Lipton. 2020. Learning the difference that makes a difference with counterfactually-augmented data. In *International Conference on Learning Representations*.

Tushar Khot, Ashish Sabharwal, and Peter Clark. 2018. SciTail: A textual entailment dataset from science question answering. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Sosuke Kobayashi. 2018. Contextual augmentation: Data augmentation by words with paradigmatic relations. 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 452–457, New Orleans, Louisiana. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. arXiv:1907.11692.

Shayne Longpre, Yi Lu, Zhucheng Tu, and Chris DuBois. 2019. An exploration of data augmentation and sampling techniques for domain-agnostic question answering. *arXiv preprint arXiv:1912.02145*.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. 2014. A SICK cure for the evaluation of compositional distributional semantic models. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 216–223, Reykjavik, Iceland. European Language Resources Association (ELRA).

Tom McCoy, Ellie Pavlick, and Tal Linzen. 2019. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3428–3448, Florence, Italy. Association for Computational Linguistics.

Clara H. McCreery, Namit Katariya, Anitha Kannan, Manish Chablani, and Xavier Amatriain. 2020. Effective transfer learning for identifying similar questions: Matching user questions to COVID-19 FAQs. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 3458—-3465. Association for Computing Machinery.

John Miller, Karl Krauth, Benjamin Recht, and Ludwig Schmidt. 2020. The effect of natural distribution shift on question answering models. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 6905–6916. PMLR.

Marius Mosbach, Maksym Andriushchenko, and Dietrich Klakow. 2021. On the stability of fine-tuning BERT: Misconceptions, explanations, and strong baselines. In *International Conference on Learning Representations*.

Nathan Ng, Kyunghyun Cho, and Marzyeh Ghassemi. 2020. SSMBA: Self-supervised manifold based data augmentation for improving out-of-domain robustness. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1268–1283, Online. Association for Computational Linguistics.

Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. 2020. Adversarial NLI: A new benchmark for natural language understanding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4885–4901, Online. Association for Computational Linguistics.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53, Minneapolis, Minnesota. Association for Computational Linguistics.

Yanru Qu, Dinghan Shen, Yelong Shen, Sandra Sajeev, Weizhu Chen, and Jiawei Han. 2021. Co{da}: Contrast-enhanced and diversity-promoting data augmentation for natural language understanding. In *International Conference on Learning Representations*.

Yanru Qu, Dinghan Shen, Yelong Shen, Sandra Sajeev, Jiawei Han, and Weizhu Chen. 2020. Coda: Contrast-enhanced and diversity-promoting data augmentation for natural language understanding. *arXiv preprint arXiv:2010.08670*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Ahmad Rashid, Vasileios Lioutas, and Mehdi Rezagholizadeh. 2021. MATE-KD: Masked adversarial TExt, a companion to knowledge distillation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language*

Processing (Volume 1: Long Papers), pages 1062–1071, Online. Association for Computational Linguistics.

Anna Rogers. 2021. Changing the world by changing the data. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2182–2194, Online. Association for Computational Linguistics.

Nithya Sambasivan, Shivani Kapania, Hannah Highfill, Diana Akrong, Praveen Paritosh, and Lora M Aroyo. 2021. "everyone wants to do the model work, not the data work": Data cascades in high-stakes ai. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, pages 1–15. Association for Computing Machinery.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. arXiv:1910.01108.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96, Berlin, Germany. Association for Computational Linguistics.

Siamak Shakeri, Cicero Nogueira dos Santos, Henghui Zhu, Patrick Ng, Feng Nan, Zhiguo Wang, Ramesh Nallapati, and Bing Xiang. 2020. End-to-end synthetic data generation for domain adaptation of question answering systems. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5445–5460, Online. Association for Computational Linguistics.

Dinghan Shen, Mingzhi Zheng, Yelong Shen, Yanru Qu, and Weizhu Chen. 2020. A simple but tough-to-beat data augmentation approach for natural language understanding and generation. *arXiv preprint arXiv:2009.13818*.

Riccardo Volpi, Hongseok Namkoong, Ozan Sener, John Duchi, Vittorio Murino, and Silvio Savarese. 2018. Generalizing to unseen domains via adversarial data augmentation. *arXiv preprint arXiv:1805.12018*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*.

Dongdong Wang, Yandong Li, Liqiang Wang, and Boqing Gong. 2020. Neural networks are more productive teachers than human raters: Active mixup for data-efficient knowledge distillation from a black-box model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1498–1507.

Jason Wei and Kai Zou. 2019. EDA: Easy data augmentation techniques for boosting performance on text classification tasks. 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 6382–6388, Hong Kong, China. Association for Computational Linguistics.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, New Orleans, Louisiana. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Xing Wu, Shangwen Lv, Liangjun Zang, Jizhong Han, and Songlin Hu. 2019. Conditional BERT contextual augmentation. In *International Conference on Computational Science*, pages 84–95. Springer International Publishing.

Qizhe Xie, Zihang Dai, Eduard Hovy, Minh-Thang Luong, and Quoc V Le. 2019. Unsupervised data augmentation for consistency training. *arXiv preprint arXiv:1904.12848*.

Mingyang Yi, Lu Hou, Lifeng Shang, Xin Jiang, Qun Liu, and Zhi-Ming Ma. 2021. Reweighting augmented samples by minimizing the maximal expected loss. In *International Conference on Learning Representations*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, Florence, Italy. Association for Computational Linguistics.

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. 2017. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, volume 28, pages 649–657.

Yuan Zhang, Jason Baldridge, and Luheng He. 2019. PAWS: Paraphrase adversaries from word scrambling. In *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 1298–1308, Minneapolis, Minnesota. Association for Computational Linguistics.

A Implementation Details

A.1 Fine-tuning details

We adopted the publicly available pre-trained RoBERTa (Liu et al., 2019) and DistilRoBERTa (Sanh et al., 2019)—using the Huggingface Transformers library (Wolf et al., 2020) and the Pytorch Lightning library[4].

For the *test* settings, the model is evaluated on the development data once per epoch for small datasets and twice per epoch for large ones—i.e., SST-2, MNLI, QNLI, SQuAD, and HellaSwag. The best performing model is chosen for testing. Our learning rate schedule follows a linear decay scheduler with a warm-up, specified as a ratio of the total number of training steps. Maximum number of epochs is set to 20 for all tasks except SQuAD, following (Mosbach et al., 2021). For large datasets, we early stop with a patience of 10. The learning rate, and the batch size are tuned for each task separately. The details of hyperparameters are summarized in Table 8. We ran $\text{RoBERTa}_{\text{base}}$ experiments with the similar hyperparameters, but with these exceptions: On QNLI, learning rate, batch size, and weight decay are set to 3e-5, 64, and 0.1; warmup ratio is set to 0.06 on QQP.

For *dev* experiments, we follow CoDA (Qu et al., 2021) on the GLUE tasks. Specifically, we train the model for 5 epochs with a batch size of 32, learning rate 1e-5, warmup ratio 0.06, weight decay 0.1, and linear learning rate decay. For SQuAD, and HellaSwag, the hyperparameters are detailed in Table 7.

All experiments were conducted on two Nvidia Tesla V100 GPUs.

Hyperparam.	SQuAD	HellaSwag
Learning rate	1.5e-5	1.5e-5
Batch size	16	32
Max length	512	512
Max epochs	3	20
Warmup ratio	0.06	0.06
Grad. acc. steps	4	1
Weight Decay	0.01	0.01
temp. τ (for KD)	5.0	10.0

Table 7: Hyperparameters of DistilRoBERTa on two downstream tasks.

[4] https://github.com/PyTorchLightning/pytorch-lightning

A.2 Knowledge distillation details

We implemented knowledge distillation by caching the teacher's logits prior training. We performed grid search to find the best softmax temperature τ from $\{5.0, 10.0, 12.0, 20.0, 30.0\}$. The value of τ used in our experiments are reported in Tables 7 and 8 for DistilRoBERTa and $\text{RoBERTa}_{\text{base}}$; with the exception $\tau = 20.0$ on MRPC for $\text{RoBERTa}_{\text{base}}$. Loss weight α, in Eq. (8), is set to 0.5 for all tasks except CoLA in which $\alpha = 0.75$.

B OOD results

B.1 Distilled Mode

OOD results for models trained in the distilled mode are presented in Table 9.

B.2 Standalone Mode

Table 10 presents OOD results for models trained using *test* settings, and Table 11 (complementary to Table 3 in §4.2.1) presents OOD results for dev experiments.

C Runtime Analysis

Due to space constraint, the runtime plots are provided in Figure 2. The discussion of these results are given in §5.

D More Ablation Study

Effect of Augmentation Size in Glitter. We explore how augmentation size affects the performance of Glitter. Throughout our experiments, we fix the augmentation size to *8x*, but now, we reduce augmentation size K to $6x$ and $4x$, while retaining selection size k_1 as before—i.e., 1 for RTE, and 2 for the rest. Our results, shown in Table 12, reveal that when K becomes close to k_1, Glitter's performance declines. Nonetheless, for a sufficiently large augmentation, Glitter starts to shine. For SST, and MRPC, the magic number is $8x$, whereas for QNLI, and RTE, Glitter performs best on $6x$.

Effect of Selection Strategy in Glitter. In this section, our objective is to assess whether our proposed selection algorithm is crucial in Glitter. To this end, we sample random augmented examples at each iteration, namely *Glitter-Rnd*, instead of selecting worst-case examples. As illustrated in Table 12 (the bottom two rows), the performance drops on all datasets—i.e.,0.2% on QNLI, and more than 1% on the rest, confirming the effectiveness of our selection algorithm.

Hyperparam.	**CoLA**	**SST**	**MRPC**	**STS-B**	**QQP**	**MNLI-m/mm**	**QNLI**	**RTE**
Learning rate	1e-5	1e-5	1e-5	1e-5	1e-5	3e-5/1e-5	5e-5*	1e-5
Batch size	32	64	16	32	64	64	128*	32
Max length	128	256	128	128	256	256	256	256
Warmup ratio	0.1	0.06	0.06	0.06	0.1*	0.08/0.06	0.08	0.06
Gradient acc. steps	1	4	1	1	4	4	4	1
Weight Decay	0.1	0.1	0.1	0.1	0.1	0.0/0.1	0.0*	0.1
Softmax temp. τ (for KD)	30.0	20.0	12.0*	12.0	20.0	12.0	12.0	12.0

Table 8: Hyperparameters of DistilRoBERTa on the GLUE benchmark. We used the same configuration for $\text{RoBERTa}_{\text{base}}$ albeit with a few exceptions marked by (*).

Trained On →	*SST*	*SST*	*SST*	*STS*	*QQP*	*QQP*	*MNLI*	*MNLI*	*RTE*
Method	**IMDb**	**IMDb-Con.**	**IMDb-CAD**	**SICK**	**MQP**	**PAWS_{QQP}**	**SciTail**	**A-NLI**	**HANS**
	Acc	Acc	Acc	P/S	Acc/F_1	Acc	Acc	Acc	Acc
$\text{RoB}_{\text{Large}}$	93.7	92.0	94.0	84.3	71.6	43.6	82.0	45.9	81.8
DistilRoB	90.2	87.6	92.5	79.6	67.3	36.3	74.8	27.8	**71.3**
KD	90.6	87.4	93.2	79.9	65.6	33.1	77.3	28.9	70.6
EDA (Wei and Zou, 2019)									
Vanilla-DA	91.8	87.2	92.9	80.0	59.9	**38.0**	75.8	27.3	66.6
Glitter	91.2	87.1	**94.0**	80.0	64.0	36.6	75.6	28.8	65.6
Back-Translation									
Vanilla-DA	92.2	87.9	92.1	80.3	**69.6**	35.0	76.5	27.9	68.0
Glitter	**92.4**	87.9	92.8	**81.2**	68.7	35.2	77.6	**30.4**	70.5
Masked-and-reconstruct									
Vanilla-DA	91.8	**88.8**	92.9	80.4	68.5	33.7	77.4	28.5	69.3
Glitter	92.0	88.0	92.5	80.7	68.8	35.3	**78.2**	29.9	70.9

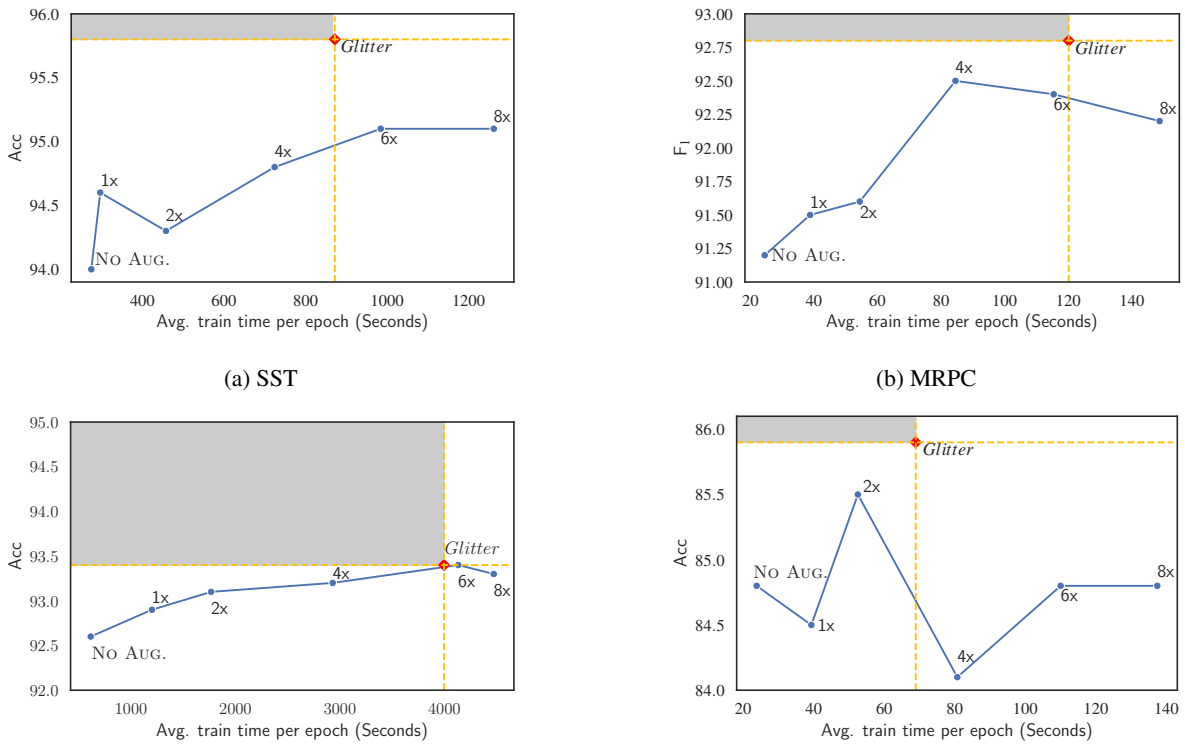
Table 9: OOD results of models whose in-domain test results are reported in Table 1 for the distilled mode. **Bold** numbers indicate the best result across DistilRoB models.

Trained On →	*SST*	*SST*	*SST*	*STS*	*QQP*	*QQP*	*MNLI*	*MNLI*	*RTE*
Method	**IMDb**	**IMDb-Con.**	**IMDb-CAD**	**SICK**	**MQP**	**PAWS_{QQP}**	**SciTail**	**A-NLI**	**HANS**
	Acc	Acc	Acc	P/S	Acc/F_1	Acc	Acc	Acc	Acc
RoB_{Base}	92.2	89.1	94.3	80.6	70.7	38.6	78.5	31.4	78.5
Self-KD	**92.6**	89.1	**95.0**	80.2	70.9	37.6	**79.4**	32.1	79.5
+ Vanilla-DA	91.8	88.8	94.8	81.5	71.4	38.8	78.4	31.5	79.3
+ Glitter	92.0	**89.6**	94.8	**81.7**	**72.1**	**39.4**	79.1	**32.7**	80.1
CT + Vanilla-DA	90.6	88.1	92.1	76.6	70.6	38.3	76.6	30.3	78.4
CT + Glitter	92.2	88.6	93.7	79.4	70.7	38.8	77.0	31.6	**80.2**

Table 10: OOD results of models whose in-domain test results are reported in Table 2 for the standalone experiment. **Bold** numbers indicate the best result.

Trained On →	*SST*	*SST*	*SST*	*MNLI*	*MNLI*	*MNLI*	*RTE*
Method	**IMDb**	**IMDb-Con.**	**IMDb-CAD**	**SciTail**	**A-NLI**	**HANS**	**HANS**
	Acc	Acc	Acc	Acc	Acc	Acc	
RoB_{Base}	**91.9** ± **0.3**	90.0 ± 0.4	94.1 ± 0.4	**80.1** ± **0.4**	31.0 ± 0.6	73.7 ± 0.7	78.3 ± 0.4
HiddenCut♠	-	87.8	90.4	-	**32.8**	71.2*	-
MMEL†	91.6 ± 0.1	90.5 ± 0.7	94.5 ± 0.4	79.7 ± 0.3	31.4 ± 0.6	74.5 ± 0.6	78.3 ± 0.3
Self-KD	**91.9** ± **0.3**	90.3 ± 0.5	94.4 ± 0.4	79.9 ± 0.3	30.9 ± 0.4	73.5 ± 0.7	78.2 ± 0.4
+ Vanilla-DA	91.6 ± 0.4	90.2 ± 0.4	94.3 ± 0.3	79.3 ± 0.4	31.3 ± 0.5	73.9 ± 0.4	77.8 ± 0.3
+ Glitter ✨	91.7 ± 0.2	**90.6** ± **0.2**	**94.8** ± **0.2**	79.4 ± 0.1	31.8 ± 0.4	**74.6** ± **0.3**	**78.4** ± **0.2**

Table 11: OOD results of models with *dev* settings in the standalone mode, same models whose results are reported in Table 3. (♠) denotes results are taken verbatim from: HiddenCut (Chen et al., 2021). (†) indicates the results are obtained from our implementation of MMEL (Yi et al., 2021). **Bold** numbers indicate the best result.



(a) SST

(b) MRPC

(c) QNLI

(d) RTE

Figure 2: Runtime Analysis of DA when training $RoBERTa_{base}$ using self-KD. The red point signifies Glitter and the grey area highlights a desirable area where accuracy is higher while training time is faster, compared to Glitter.

Method	SST Acc	MRPC F_1	QNLI Acc	RTE Acc
Glitter ✨ (8x)	95.8	92.8	93.4	85.9
Glitter ✨ (6x)	94.7	92.7	93.7	86.3
Glitter ✨ (4x)	95.0	92.1	93.3	85.7
Glitter-Rnd (8x/2x)	94.3	91.4	93.2	85.2
Glitter-Rnd (8x/1x)	94.3	91.8	93.2	84.5

Table 12: Dev results of self-KD for studying the effect of augmentation size and the selection algorithm for 4 GLUE tasks.