DACon: Data Augmentation for Entity Matching using Consistency Learning

Shen En Chen
Georgia Institute of Technology
Atlanta, Georgia
achen353@gatech.edu

Chieng Chang
Georgia Institute of Technology
Atlanta, Georgia
cchang397@gatech.edu

1 INTRODUCTION

With the development and adoption of deeper neural networks in almost all fields of computer science, including data management, the demand for high-quality labeled training data has surged. Data augmentation (DA), being one of the least expensive way to obtain more diverse data, becomes a common technique. Such technique is particularly crucial for the task of entity matching (EM), given that millions of records often contain only a few matched pairs. However, data augmentation could generate overly noisy examples that divert too much from the original data distribution and incur significant time cost to tune the large space of hyper-parameters. To tackle these two problems, we introduce consistency learning to entity matching systems' training process. More specifically, we will regulate the diversity of augmented examples and their impacts on the trained models through Jenson-Shannon divergence, formulating it as part of the loss function for the downstream pre-trained Tranformer-based language model used to perform EM. We attempt to achieve performance on EM comparable to state-of-the-art (SOTA) data augmentation models while reducing time complexity of the model training process. The model will be evaluated on common EM benchmark datasets in normal and low-resource settings. We will also analyze the contributions of different components of our DA frameworks and compare the training time against other DA techniques. We expect our framework to require less training time while preserving, if not, improving performance on EM tasks.

2 BACKGROUND

Before describing how we propose to leverage DA for more robust EM solutions, we present the main concepts behind EM and the state-of-the-art (SOTA) deep learning solutions. We also provide some background on DA.

2.1 Entity Matching and Deep EM Solutions

Entity Matching (EM) refers to the problem of determining whether two data entries refer to the same real-world entity. For example, given two datasets about the inventory of a company, the goal of EM is to determine the set of pairs of data entries, one entry from each table, so that each pair of entries refer to the same inventory item.

Just like many other domains in computer science, deep learning solutions for EM have also been proposed, such as DeepMatcher [12] and Ditto [8]. These models achieve SOTA performance on many EM benchmarks but at the same time require a significant amount of labeled training data. Our proposed DA approach builds on top of these deep EM solutions and seeks to fuel these models with fewer labeled data.

2.2 The Need for Data Augmentation

Modern machine learning models, such as deep neural networks, often adopt billions of parameters in order to model problems of high complexity. Accordingly, such models require massive labeled training datasets, which are often not available. DA tackles this data scarcity problem with relatively low costs. First gaining it popularity in computer vision (CV) tasks such as image classification, DA is also becoming increasingly common for natural language processing (NLP) [1] and reinforcement learning (RL) [7].

Heuristic data augmentation schemes often depend on the composition of a set of simple transformation functions (TFs) or DA operators, such as rotating and flipping an image or deleting and replacing tokens in a string of text. When chosen carefully, data augmentation schemes tuned by human experts can improve model performance. However, such heuristic strategies in practice can cause large variances in end model performance, and may not produce parameterizations and compositions needed for the models. Take binary text classification for example, overly deleting or replacing the tokens of a piece of text might drastically change the semantic of the text data such that it no longer fits the label (e.g. from "positive sentiment" to "negative sentiment"), but back-translation (the technique of translating the original text data to another language then back to the original language) could generate realistic and diverse expression while preserving the semantics and label. Heursitic data augmentation schemes also require significant human effort and it could grow exponentially with the increased number of candidate TFs.

3 RELATED WORK

3.1 Model Data Augmentation as Sequences of Transformation Functions

In response to the limitations of conventional DA approaches that manually select which transformation function(s) to apply, many research have been done to automatically explore the space of TFs efficiently and effectively to learn the best "DA policy". The common approach to this problem is to model DA as sequences of TFs provided by users. For example, a DA could be "randomly delete 3 tokens", followed by "randomly replacing 2 tokens with synonyms".

In TANDA [16], a TF sequence generator is trained to produce realistic images by having to fool a discriminator network, following the GANs framework [4]. That is, we can reasonably assume that we won't turn an image of a plane into one of a dog, but we might turn it

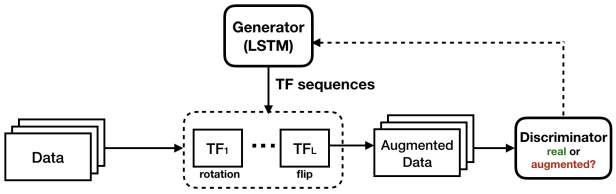

Figure 1: Modeling data augmentation as sequences of transformation functions with TANDA [16]. A TF sequence generator is trained adversarially to produce augmented images that are realistic compared to training data.

into an indistinguishable garbage image! Such an assumption allows us to leverage generative adversarial networks (GANs), where we simultaneously learn a generator and a discriminator. As shown in Figure 1, the objective for the generator is to produce sequences of TFs such that the augmented data point can fool the discriminator; whereas the objective for the discriminator is to produce values close to 1 for data points in the original training set and values close to 0 for augmented data points.

With a similar framework, AutoAugment [2] demonstrated SOTA performance using learned DA policies. In this work, a TF sequence generator learns to directly optimize for validation accuracy on the end model, instead of optimizing for realisticness of the augmented images as in TANDA. However, the search process can be computationally expensive due to the need to train a classification model in every gradient step for the generator. To address this issue, AutoAugment searches for augmentation policies on a surrogate dataset that is orders of magnitude smaller than the original dataset.

RandAugment [3] identified that simple random sampling over the transformation functions with grid search over the parameters of each transformation can outperform AutoAugment. Specifically, the authors replace the learned TF sequences and probabilities for applying each TF with a parameter-free procedure, which always selects a transformation with uniform probability. They reduce the search space by only learning the magnitude of each transformation function. RandAugment demonstrated 0.6% increase over the previous SOTA on ImageNet classification with less computational cost.

Adversarial AutoAugment [19] most recently proposed an adversarial framework to jointly optimize the end model training and augmentation policy search. The TF sequence generator attempts to increase the training loss of the end model through generating adversarial augmentation policies, while the downstream, end model is trained to be robust against those hard examples and therefore improves the generalization.

While modeling DA as sequences of TFs has been shown to achieve SOTA performance on many tasks, the overall search can still be expensive. Moreover, these model requires additional component, such as the LSTM-based generator in TANDA and the RNN-based controller in AutoAugment and Adversarial AutoAugment, that increase both the space and time complexity for training and inference. Our proposed approach, as we will explain in section 4, aims to learn an optimal DA policy without such overhead.

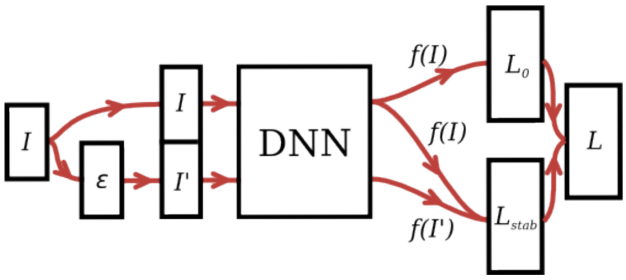

Figure 2: Consistency learning in [20]. For each labeled data I, a copy I' is corrupted with noise. Both the original and perturbed version are then processed by the neural network. The final loss L is the combination of the task objective (cross-entropy) L_0 and the stability loss $L_{stability}$ and propagates back through the network. The former is only evaluated on the output $f(I)$ of the original data, while the stability loss $L_{stability}$ uses the outputs of both versions.

3.2 Consistency Learning

Another perspective of automated DA, also mentioned in subsection 2.2, is the challenge of regulating the data operations to ensure the augmented examples are diverse enough but at the same time preserve the labels. That is, the augmented examples should be "hard to distinguish" from the original examples .

Kashefi and Hwa [6] used Kullback-Leibler (KL) divergence to quantify the hard-to-distinguish quality of the augmented text data. More specifically, they measured the distribution differences between the vector representations of original data and those of augmented data and showed that such metrics is indicative of how effective a given DA operation is on generating noisy but useful data to the model being trained.

In a similar vein, many research have applied KL-divergence on the label of the data instead of the latent data representation, given that the former should be more resilient to perturbations on data properties and is more intuitive to interpret for human practitioners. Furthermore, the KL divergence or other distribution divergence measures are commonly formulated as part of the optimization objective for the model, called consistency loss. Such learning of models is thus called consistency learning, and it has shown to be effective for different types of data including visual, audio, and text data.

Zheng et al. [20] applied consistency loss on image classification model and referred it stability loss. As shown in Figure 2, given an image from the training dataset, a noisy, or corrupted, version is generated and input to the model along with the original, uncorrupted copy. The training loss is measured as the sum of (1) the cross-entropy loss between the model prediction and ground truth and (2) the stability loss between the original and corrupted image.

Similarly, Iqbal et al. [5] investigated the use of consistency loss for audio classification tasks in the supervised learning setting. They proposed using the Jensen-Shannon (JS) divergence as a loss term along with the cross-entropy loss between the prediction and the ground truth. In their settings, JS divergence is chosen instead KL divergence in order to handle an arbitrary number of distributions

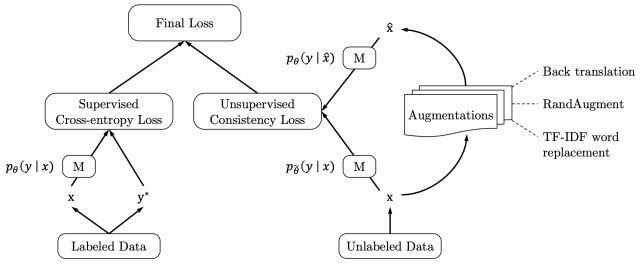

Figure 3: UDA [18] formulates its training loss as the sum of the cross-entropy of the labeled (supervised) data and the consistency loss on the augmented unlabeled (unsupervised) data.

created by different DA operations, allowing them to apply all types of DA operators available and let the training process to regulate the data diversity automatically through consistency loss.

Furthermore, UDA [18] calculated consistency loss for augmented unlabeled data instead of labeled data, finding that DA in supervised learning can also serve as a superior source of noise under the consistency enforcing semi-supervised framework. They conducted the experiments on both vision and language tasks and showed that such a framework can match and even outperform purely supervised learning that uses orders of magnitude more labeled data.

3.3 Data Augmentation for Entity Matching

Given the textual nature of most of the EM datasets, we could simply apply DA operations commonly used for text data, such as deleting, replacing, swapping, and inserting tokens or spans. To further utilize the tabular nature of EM datasets, we could also shuffle or delete the columns/attributes.

There are also more sophisticated DA operations include MixDA [11] and InvDA [10]. Both of these utilize the serialized sequence representation of table records. The former generates partially transformed sequences by interpolating the vector representation of the augmented sequence with that of the original sequence, whereas the latter utilize a seq2seq model that learns to fix corrupted/noisy sequence back to the original sequence. However, similar to those techniques that model DA as sequence of TFs, both of these techniques require training or fine-tuning a model additional to the downstream model performing the EM task.

4 APPLY CONSISTENCY LEARNING TO ENTITY MATCHING

In [10], the authors modeled the DA optimization problem by generating all possible augmented training data prior to the training, assigning a weight to each individual datum, and optimizing these weights through meta learning. However, a significant time overhead is induced through this approach because of the additional calculation of the validation loss. As a result, instead of weighing the training samples, we proposed to weigh the DA operators directly and regulate the training process with consistency loss on the RoBERTa-based model [9] in [10].

Our work can be broken down into two parts: the DA operators and the loss functions.

4.1 Data Augmentation Operators

An ideal set of DA operators should generate a set of augmented data with distribution that is neither too similar nor too different from the original set [5]. Operators that are too aggressive may lead to greater overfitting, while operators that are too conservative may result in poor performance through training on examples not representative of the given domain.

Common DA techniques for text data include token deletion, substitution with synonyms, substitution with antonyms, swap, random insertion, and many more. The aforementioned data augmentation operators are generally referred to as rule-based techniques, which are typically predetermined, easy to compute, and interpretable by humans.

In this work, we are using only rule-based DA operators as more advanced DA operators like MixDA [11] and InvDA [10] could results in additional fine-tuning cost (e.g. tuning additional seq2seq model for InvDA) and the risk of overfitting due to the high expressiveness of these techniques.

4.2 Loss Functions

Once we have the set of DA operators determined. We perform data augmentation as we train the model. Let the number of DA operators available be m (m is small). During each training iteration, for each training data x, we generate m augmented copies of x, $x' = \{x'_1, x'_2, \ldots, x'_m\}$, by applying the m DA operators independently on x. We pass x' together with x and update the model by aggregating the (1) cross-entropy loss and (2) consistency loss on the training data.

Cross-Entropy Loss

cross-entropy is a measure of the difference between two probability distributions for a given random variable or set of events. A skewed distribution has a low entropy, whereas a distribution where events have equal probability has a larger entropy. Entropy $H(x)$ can be calculated for a random variable with a set of x in X discrete states and their probability $P(x)$ as follows:

$$H(x) = -\sum_{x \in X} P(x) \cdot \log(P(x)) \tag{1}$$

The cross-entropy between two probability distributions Q from P, where P may be the target distribution and Q is the approximation of the target distribution, can be calculated using the probabilities of the events from P and Q as follows:

$$CE(P, Q) = -\sum_{x \in X} P(x) \cdot \log(Q(x)) \tag{2}$$

In the context of DA, we calculate the cross-entropy between the ground truth y and the model prediction given x.

$$l_{ce}(x, y) = CE(y^*, P_M(y|x)) \tag{3}$$

where $P_M(y|x)$ is the predicted distribution of model M based on x.

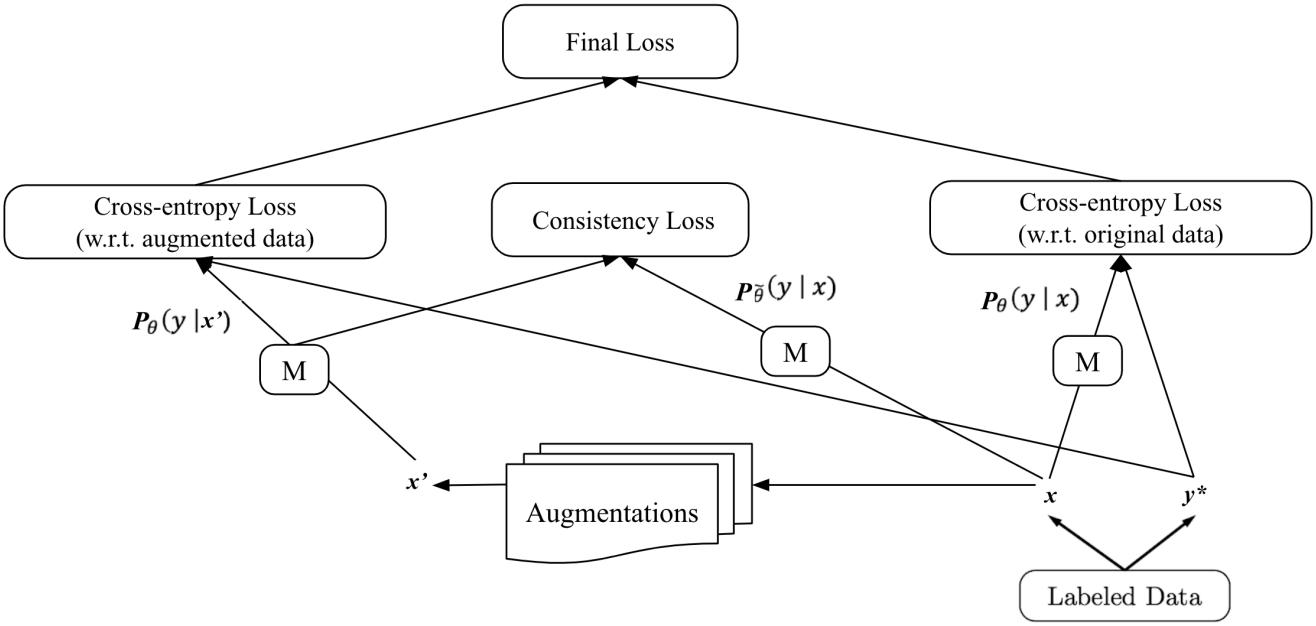

Figure 4: Our proposed consistency learning architecture

Recall that as a training data x passes through the model, m additional augmented data $x' = \{x'_1, x'_2, \ldots, x'_m\}$ are also fed. As a result, we calculate the cross-entropy for each of the augmented data x'_1 $(1 \leq i \leq m)$ in addition to the cross-entropy of the original x. We aggregation the cross-entropy losses for all augmented data and weigh this aggregated sum equally as the cross-entropy of x. Formally, for each training data x, the cross-entropy, including its augmented copies, is:

$$\mathcal{L}_{ce} = l_{ce}(x, y) + \frac{1}{m}\sum_{i=1}^{m} l_{ce}(x'_i, y) \tag{4}$$

Consistency Loss

Recall that an ideal augmented dataset should preserve the distribution of the original set. However, the cross-entropy loss does not explicitly enforce this property of consistency. To enforce consistency, we used Jensen-Shannon (JS) divergence similar to [5] to measure the similarities among the predicted distributions of x and x', namely $P_M(y|x)$, $P_M(y|x'_1)$, …, and $P_M(y|x'_m)$. We denote these distributions as P_M^x, $P_M^{x'_1}$, …, $P_M^{x'_m}$, respectively, and define the JS divergence, equivalently our consistency loss, as the following:

$$\begin{aligned}\mathcal{L}_{consistency} =& JSD(P_M^x, P_M^{x'_1}, \ldots, P_M^{x'_m}) \\ =& \frac{1}{m+1}[KL(P_M^x||C) + KL(P_M^{x'_1}||C) \\ &+ \cdots + KL(P_M^{x'_m}||C)]\end{aligned} \tag{5}$$

where $C = \frac{1}{m}(P_M^x + P_M^{x'_1} + \cdots + P_M^{x'_m})$ and $KL(P||Q)$ is the Kullback-Leibler (KL) divergence from Q to P. The primary reason for using the JS divergence is that it can handle an arbitrary number of distributions, while other divergences such as the KL divergence are defined for two distributions only.

Final Training Loss

Finally, we formulate our final training loss as:

$$\mathcal{L} = \mathcal{L}_{ce} + \mathcal{L}_{consistency} \tag{6}$$

We collectively minimize the cross-entropy loss and consistency loss. The former measures the how close our model approximates the true distribution and the latter ensures that learned representations of the downstream model do not deviate significantly from the original intent after certain transformations. A graphical representation of the final loss is illustrated in Figure 4. We refer to such a data augmentation framework as DACon.

5 EXPERIMENTS

We build our consistency learning framework by extending [10]'s implementation in PyTorch [14] and the Transformers library [17]. Similar to [10], we used the 12-layer RoBERTa [9] model on all EM datasets which we will introduce in subsection 5.4. Across all experiments, we set the batch size to be 32, learning rate to be 3e-5 and the max sequence length of 128 (recall that we're using the same downstream model as [10]). In each run, we fine-tune the LM (RoBERTa) and DACon jointly with Adam optimizer for 20 epochs via the loss on the training data. Then, we select the checkpoint with the highest F1 score on the test set, and report the test F1 score. For each dataset, we repeat the experiment five times. We ran all

experiments on a GeForce RTX 3090 GPU with Intel(R) Core(TM) i7-10700K CPU @ 3.80GHz.

5.1 Data Augmentation Operators

For DACon, we use 7 rule-based DA operators:

Table 1: DA Operators for DACon

Operators	Description
DEL	Randomly delete a span of arbitrary length
TOKEN_DEL_TFIDF	Sample a token based on TF-IDF and delete it
TOKEN_DEL	Sample a token randomly and delete it
SHUFFLE	Sample a token randomly and delete it
TOKEN_REPLACE_TFIDF	Sample a token based on TF-IDF and replace with a synonym
TOKEN_REPLACE	Sample a token randomly and replace it with a synonym
INSERT	Randomly insert a token

5.2 Baseline Methods

We evaluate and compare DACon against five previously mentioned SOTA methods:

RoBERTa: This is the baseline method in [10] that fine-tunes the pre-trained RoBERTa [9] on the original training examples without any data augmentation.

InvDA: This is an advanced DA operator developed in [10]. It performs seq2seq augmentations on data with a fine-tuned T5 model [15]. In this setting, only InvDA operator is used for augmentation.

Rotom: This is the the meta-learning framework in [10] that selects and weights the augmented examples to better fine-tune the target model. The DA operators used in this settings include rule-based operators, MixDA [11], and InvDA [10].

Rotom + SSL: This is a semi-supervised variant of Rotom that selects and weights the unlabeled data to train the target model. For each dataset, Rotom+SSL uses at most 10,000 uniformly sampled unlabeled examples.

DM + RoBERTa: DM, or DeepMatcher [13], is a classic deep learning EM method. Instead of fine-tuning LMs, DM trains a hybrid neural net consisting of RNN layers and the attention mechanism. It achieves good results in multiple EM tasks but requires a significant amount of (10k) training data. We consider a variant of DM with its RNN and word embedding layers replaced by a RoBERTa encoder to better compare the performance and time complexity. We denote this variant as DM+RoBERTa.

5.3 DACon

To better perform ablation analysis on different components of DACon, we have the following 4 configurations:

DACon Baseline: This is the most simplified version of DACon where only the cross-entropy with respect to the original data and that with respect to the augmented data is calculated. Moreover, there is no consistency loss and, for each training data x, only one single augmented x' is created based on an operator randomly selected from the 7 DA operators listed in subsection 5.1

DACon One-to-Many: This extends **DACon Baseline** by having 7 augmented $x' = x'_1, \ldots, x'_7$ for each training data x. Each of the x'_i $(1 \leq i \leq 7)$ is generated independently by one of the 7 DA operators in subsection 5.1. However, there is still no consistency loss calculated in this setting.

DACon Fixed Consistency: This builds upon **DACon One-to-Many** by bringing consistency loss formulated in subsection 4.2. This is also the same as the design in Figure 4.

DACon Consistency: This is an extension of the original formulation of DACon in subsection 4.2 and Figure 4. In this setting, for the consistency loss, instead of weighing all KL-divergence equally by $\frac{1}{m+1}$ as in Equation 5, we allow the weight of each divergence terms to be trainable during backpropagation.

5.4 Datasets

We evaluate the above 5 previous methods and 4 DACon variants on standard benchmark datasets of EM, listed in Table 2.

Table 2: EM datasets for the experiment

Datasets	#Train + #Valid	#Test
Abt-Buy	7,659	1,916
Amazon-Google	9,167	2,293
DBLP-ACM*	9,890	2,473
DBLP-Scholar*	22,965	5,742
Walmart-Amazon*	7,659	1,916

For each EM dataset, we create a sample of training/validation set of size from 300 to 750 and use the original test set for evaluation. Each EM dataset marked with a "$*$" also comes with a more challenging dirty version. Each test set consists of 20 uniformly sampled tuples.

6 EXPERIMENTAL RESULTS

6.1 Performance on Entity Matching Tasks

We evaluate the 4 variants of DACon and 5 previous approaches through F1 score. With 750 training and validation samples, as seen in Table 3, DACon's performance is comparable with RoBERTa, InvDA, Rotom, and Rotom with semi-supervised learning on average

Table 3: F1 scores on all EM datasets with at most 750 training+validation examples

Method	Abt-Buy	Amazon-Google	DBLP-ACM clean/dirty	DBLP-Scholar clean/dirty	Walmart-Amazon clean/dirty
DM + RoBERTa	85.71	82.35	97.92/97.92	92.63/91.49	72.73/67.92
RoBERTa	80.49	65.02	98.97/96.63	92.18/92.30	71.50/73.55
InvDA	84.26	59.70	96.97/96.51	91.99/91.69	71.85/73.48
Rotom	76.34	62.38	96.76/97.16	91.80/91.63	66.28/76.66
Rotom + SSL	81.89	62.34	98.09/97.20	92.88/92.97	72.19/71.55
DACon Baseline	81.06	62.95	96.73/96.98	92.42/92.00	73.71/71.91
DACon One-to-Many	80.21	61.06	96.96/97.19	91.94/91.45	76.88/75.14
DACon Fixed Consistency	83.60	62.87	92.29/96.40	91.33/91.69	78.93/74.18
DACon Consistency	80.81	60.30	97.29/95.95	92.02/91.86	74.60/72.11

over all datasets. When it comes to DeepMatcher + RoBERTa, however, DACon's performance differs across various datasets. DeepMatcher + RoBERTa performed much better than DACon on the Amazon-Google dataset, but performed worse than DACon on the Walmart-Amazon dataset.

Through DACon's performance on DBLP-ACM, DBLP-Scholar, and Walmart-Amazon datasets, we can see that DACon is capable of handling dirty datasets, as there isn't a significant performance drop when processing dirty datasets compared to clean datasets. Additionally. given that these datasets each contain labeled pairs of products or academic publication records from two different websites. We can see that DACon is suitable to be used on a wide range of domains.

Recall that DACon was designed into 4 different variants for further ablation analysis, where each contains one additional components compared to its preceding variant in the order of DACon Baseline, DACon One-to-Many, DACon Fixed Consistency, and DACon Consistency. However, as we see from Table 3, there is little trend consistent across all datasets. None performed better than the other three throughout. This is likely due to a few reasons. First, all the rule-based data augmentation operator used perform augmentation with random selections of token or spans for insertion, replacement, and deletion. Even if a consistency loss is imposed to weigh the effects of each type of augmentation, it's difficult to guarantee that the operators always produce similar augmented examples with the same aggressiveness throughout the training process. Secondly, we assume all the augmented examples preserve the labels of the original data from which they are augmented. However, this is not guaranteed and if the label is indeed corrupted for an augmented example, the regularization of the consistency loss probably would not be sufficient.

6.2 Labeling Budget

For each dataset, we create a training set of size 300, 450, 600, and 750, which will allow us to experiment how different labeling budgets affect performance. Additionally, each test set consists of 20 uniformly sampled tuples.

Looking at Figure 5, for most datasets, we can see that DACon Fixed Consistency and DACon Consistency starts with a lower performance when given a training set of size 300, but quickly improves as training set size increases. This is because consistency losses incorporate suitable similarity measures into the training objective. In other words, consistency losses are intended to preserve the semantics of data, and penalizes the model when augmented data deviates too much from training data. With a largely withheld set of training data, our consistency loss functions fitted itself to a training set that is different from a not withheld testing set. In contrast, DACon Baseline and DAcon One-to-Many showcases comparable performance to other previous methods across varying labeling budgets. This result indicates a significant potential saving of labeling effort in creating high-quality EM solutions.

Similar to the F1 scores on performance, there's little pattern among the label efficiency of the 4 DACon variants, likely due to the same reasons for the inconsistent performance in subsection 6.1.

6.3 Training Time

The results on average training time of different models and datasets with varying labeling budgets is presented in Figure 7. Compared to all baseline models, DACon's training time increases at a slower rate as the size of data increases. Specifically, between 300, 450, 600, and 780, DACon Baseline's training time increased by 8.38%, 7.27%, and 7.74% respectively and DACon Consistency's training time increased by 17.61%, 10.24%, and 10.99%, while DeepMatcher + RoBERTa's increased by 38.81%, 26.33%, and 21.73% and Rotom + SSL increased by 30.59%, 20.57%, and 20.28%.

Compared to Rotom and its variants, DACon repeats the process of data augmentation at each training iteration instead of pre-augmenting all training examples prior to training. On the other hand, at each training iteration, Rotom repeats the process of meta-learning, updating its filtering and weighting models by back-propagating the validation loss in additional to updating the parameters of the downstream language model with training loss. Because the data augmentation here is essentially string manipulation, the overhead of the two approaches depend on different processing units: CPU for DACon and GPU for Rotom. As a result, in Table 4 and Figure 6, even though DACon has a longer average training time with a training set size of 300, it runs faster when the training set size is larger. This is because the overhead added to

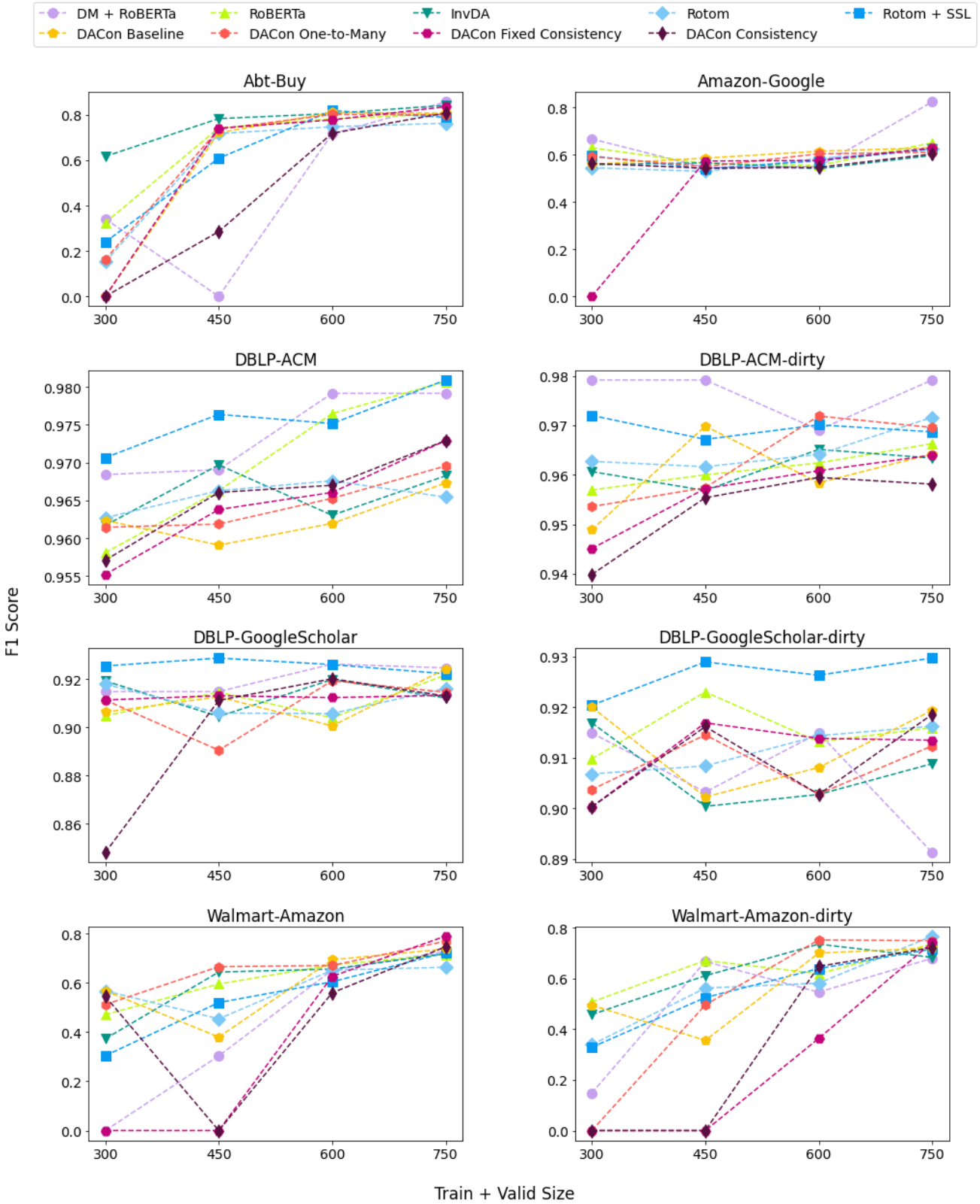

Figure 5: Performance of different models on different EM datasets with varying labeling budgets

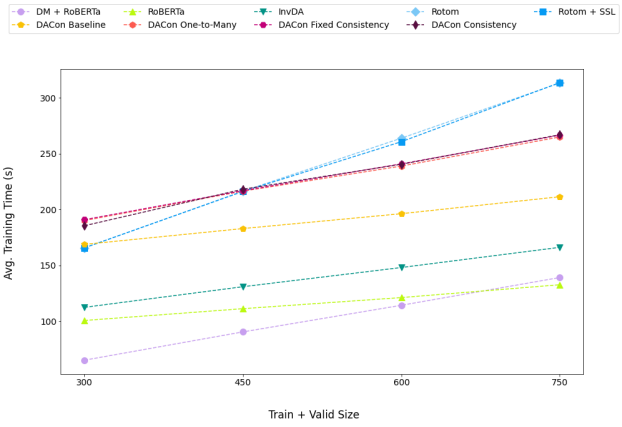

Figure 6: Training time of different models across all experiments with varying labeling budgets

CPU for data augmentation and GPU for backpropagation for every 100 additional training data is different, and as we see from the results, the time required for data augmentation with CPU scales better as the amount of data increases.

Still, for the training time, we find little consistency for the 4 DACon variants.

7 CONCLUSION

7.1 Key Contributions

In this report, we explore adding consistency loss functions as a way to regularize the latent space of machine learning models with respect to input transformations commonly used for data augmentation. We propose using the Jensen-Shannon divergence as a consistency loss term and use it to constrain the model output for texts used on entity matching tasks. Along with consistency learning, we have applied cross-entropy loss functions on each data augmentation operator. Instead of investigating individual augmented samples, we assign weights to each data augmentation operator. This has proven to reduce training time and is more prominent as the dataset size increases. The reduced training time did not worsen the model's performance on entity matching tasks.

7.2 Future Work

One of the major advantages of incorporating consistency loss to our training objective is that, unlike cross-entropy loss, consistency loss does not require ground truth labels. Thus, in future research, consistency loss can be adopted in semi-supervised and unsupervised settings. This will save significant data collection and data cleaning efforts, while ensuring high quality augmented data. We would also like to experiment with using DACon on other downstream tasks, such as data cleaning and text classification. Currently, DACon's cross-entropy losses are applied evenly across all data augmentation operators. This could become a trainable aspect of the model, which will allow cross-entropy loss functions to more accurately reflect on the suitability of each data augmentation operator.

REFERENCES

[1] Jacob Andreas. 2020. Good-Enough Compositional Data Augmentation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Online, 7556–7566. https://doi.org/10.18653/v1/2020.acl-main.676

[2] Ekin D. Cubuk, Barret Zoph, Dandelion Mané, Vijay Vasudevan, and Quoc V. Le. 2019. AutoAugment: Learning Augmentation Strategies From Data. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 113–123. https://doi.org/10.1109/CVPR.2019.00020

[3] Ekin D. Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V. Le. 2020. Randaugment: Practical automated data augmentation with a reduced search space. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 3008–3017. https://doi.org/10.1109/CVPRW50498.2020.00359

[4] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Networks. arXiv:1406.2661 [stat.ML]

[5] Turab Iqbal, Karim Helwani, Arvindh Krishnaswamy, and Wenwu Wang. 2021. Enhancing Audio Augmentation Methods with Consistency Learning. *CoRR* abs/2102.05151 (2021). arXiv:2102.05151 https://arxiv.org/abs/2102.05151

[6] Omid Kashefi and Rebecca Hwa. 2020. Quantifying the Evaluation of Heuristic Methods for Textual Data Augmentation. In *Proceedings of the Sixth Workshop on Noisy User-generated Text (W-NUT 2020)*. Association for Computational Linguistics, Online, 200–208. https://doi.org/10.18653/v1/2020.wnut-1.26

[7] Michael Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas. [n.d.]. Reinforcement Learning with Augmented Data. ([n. d.]). arXiv:2004.14990.

[8] Yuliang Li, Jinfeng Li, Yoshihiko Suhara, AnHai Doan, and Wang-Chiew Tan. 2020. Deep Entity Matching with Pre-Trained Language Models. *Proc. VLDB Endow.* 14, 1 (Sept. 2020), 50–60. https://doi.org/10.14778/3421424.3421431

[9] 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. *CoRR* abs/1907.11692 (2019). arXiv:1907.11692 http://arxiv.org/abs/1907.11692

[10] Zhengjie Miao, Yuliang Li, and Xiaolan Wang. 2021. Rotom: A Meta-Learned Data Augmentation Framework for Entity Matching, Data Cleaning, Text Classification, and Beyond. In *Proceedings of the 2021 International Conference on Management of Data* (Virtual Event, China) *(SIGMOD/PODS '21)*. Association for Computing Machinery, New York, NY, USA, 1303–1316. https://doi.org/10.1145/3448016.3457258

[11] Zhengjie Miao, Yuliang Li, Xiaolan Wang, and Wang-Chiew Tan. 2020. Snippext: Semi-Supervised Opinion Mining with Augmentednbsp;Data. In *Proceedings of The Web Conference 2020* (Taipei, Taiwan) *(WWW '20)*. Association for Computing Machinery, New York, NY, USA, 617–628. https://doi.org/10.1145/3366423.3380144

[12] Sidharth Mudgal, Han Li, Theodoros Rekatsinas, AnHai Doan, Youngchoon Park, Ganesh Krishnan, Rohit Deep, Esteban Arcaute, and Vijay Raghavendra. 2018. Deep Learning for Entity Matching: A Design Space Exploration. In *Proceedings of the 2018 International Conference on Management of Data* (Houston, TX, USA) *(SIGMOD '18)*. Association for Computing Machinery, New York, NY, USA, 19–34. https://doi.org/10.1145/3183713.3196926

[13] Sidharth Mudgal, Han Li, Theodoros Rekatsinas, AnHai Doan, Youngchoon Park, Ganesh Krishnan, Rohit Deep, Esteban Arcaute, and Vijay Raghavendra. 2018. Deep Learning for Entity Matching: A Design Space Exploration. In *Proceedings of the 2018 International Conference on Management of Data* (Houston, TX, USA) *(SIGMOD '18)*. Association for Computing Machinery, New York, NY, USA, 19–34. https://doi.org/10.1145/3183713.3196926

[14] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.), Vol. 32. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf

[15] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *CoRR* abs/1910.10683 (2019). arXiv:1910.10683 http://arxiv.org/abs/1910.10683

[16] Alexander J. Ratner, Henry R. Ehrenberg, Zeshan Hussain, Jared Dunnmon, and Christopher Ré. 2017. Learning to Compose Domain-Specific Transformations for Data Augmentation. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (Long Beach, California, USA) *(NIPS'17)*. Curran Associates Inc., Red Hook, NY, USA, 3239–3249.

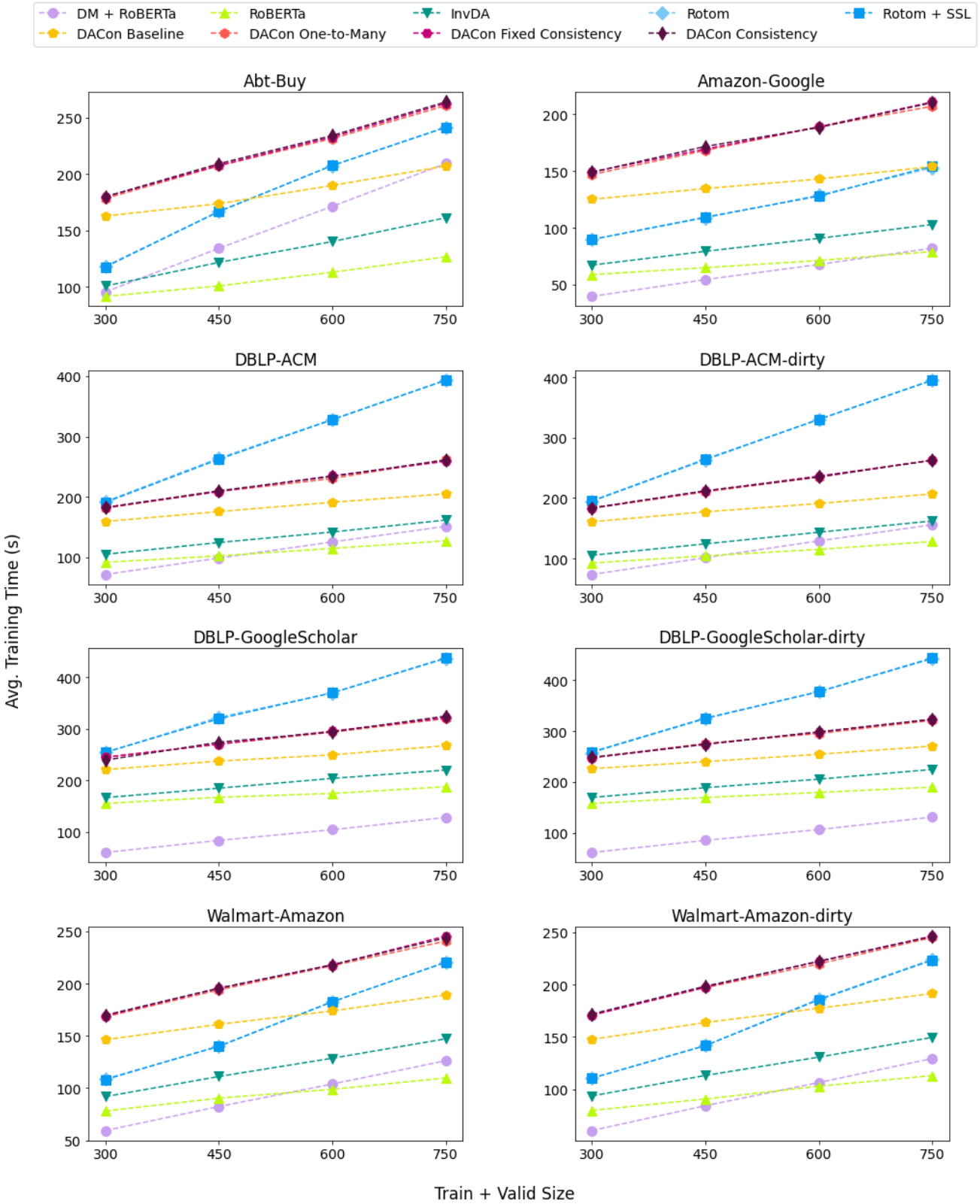

Figure 7: Average training time of different models by dataset with varying labeling budgets

Table 4: Average training time across all EM datasets in seconds by different training + validation size

Method	300	450	600	780
DM + RoBERTa	65.17	90.46	114.28	139.11
RoBERTa	100.67	111.24	121.17	132.65
InvDA	112.45	130.90	148.07	166.12
Rotom	165.73	216.40	263.81	313.18
Rotom + SSL	165.73	216.17	260.64	313.51
DACon Baseline	168.78	182.93	196.23	211.42
DACon One-to-Many	190.14	216.42	238.87	264.89
DACon Fixed Consistency	190.93	216.88	240.85	266.48
DACon Consistency	185.49	218.16	240.49	266.91

[17] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2019. HuggingFace's Transformers: State-of-the-art Natural Language Processing. *CoRR* abs/1910.03771 (2019). arXiv:1910.03771 http://arxiv.org/abs/1910.03771

[18] Qizhe Xie, Zihang Dai, Eduard H. Hovy, Minh-Thang Luong, and Quoc V. Le. 2019. Unsupervised Data Augmentation. *CoRR* abs/1904.12848 (2019). arXiv:1904.12848 http://arxiv.org/abs/1904.12848

[19] Xinyu Zhang, Qiang Wang, Jian Zhang, and Zhao Zhong. 2020. Adversarial AutoAugment. In *International Conference on Learning Representations*. https://openreview.net/forum?id=ByxdUySKvS

[20] Stephan Zheng, Yang Song, Thomas Leung, and Ian J. Goodfellow. 2016. Improving the Robustness of Deep Neural Networks via Stability Training. *CoRR* abs/1604.04326 (2016). arXiv:1604.04326 http://arxiv.org/abs/1604.04326