Data Pruning for Efficient Model Pruning in Neural Machine Translation

Abdul Hameed Azeemi and Ihsan Ayyub Qazi and Agha Ali Raza
Lahore University of Management Sciences
{abdul.azeemi, ihsan.qazi, agha.ali.raza}@lums.edu.pk

Abstract

Model pruning methods reduce memory requirements and inference time of large-scale pre-trained language models after deployment. However, the actual pruning procedure is computationally intensive, involving repeated training and pruning until the required sparsity is achieved. This paper combines data pruning with movement pruning for Neural Machine Translation (NMT) to enable efficient fine-pruning. We design a dataset pruning strategy by leveraging cross-entropy scores of individual training instances. We conduct pruning experiments on the task of machine translation from Romanian-to-English and Turkish-to-English, and demonstrate that selecting hard-to-learn examples (top-k) based on training cross-entropy scores outperforms other dataset pruning methods. We empirically demonstrate that data pruning reduces the overall steps required for convergence and the training time of movement pruning. Finally, we perform a series of experiments to tease apart the role of training data during movement pruning and uncover new insights to understand the interplay between data and model pruning in the context of NMT.

1 Introduction

Large-scale pre-trained language models have demonstrated encouraging performance in various NLP tasks at the cost of over-parametrized networks, and high memory requirements (Devlin et al., 2019; Raffel et al., 2020). This has led to the development of several pruning approaches for reducing model size, such as magnitude pruning (Han et al., 2015; Gale et al., 2019), movement pruning (fine-pruning) (Sanh et al., 2020), block movement pruning (Lagunas et al., 2021) and lottery ticket hypothesis for BERT (Chen et al., 2020). Although model pruning is effective at reducing the inference time after deployment, the actual pruning procedure is computationally intensive and unsuitable to be performed in resource-constrained settings. For example, BERT requires six iterations to reach 40% sparsity with Iterative Magnitude Pruning (IMP) —requiring training to convergence, pruning, and retraining to recover the lost accuracy (Chen et al., 2020, 2021). Recent work (Chen et al., 2021) attempts to decrease the training time by identifying structured winning tickets early in training but the implementation does not allow general application to other model pruning algorithms.

In contrast, we examine the problem of increasing the efficiency of model pruning techniques through the lens of reducing data requirements and thus ask the following questions — How much data is superfluous in fine-pruning language models for machine translation? Can we develop a metric for identifying the informative training examples and significantly prune training data to decrease the training time and memory requirements during language model pruning?

In this work, we develop a dataset pruning algorithm for efficient movement pruning of T5 language model (Raffel et al., 2020) on the task of neural machine translation (NMT) across two datasets, WMT16 En-Ro and En-Tr. We begin by leveraging the training dynamics and use cross-entropy score for ranking each example according to its difficulty. We utilize this ranking to prune the datasets by selecting hard-to-learn training examples and pruning the remaining examples. We compare this approach with multiple data pruning baselines used in standard vision and speech tasks, including stratified (representative) selection (Azeemi et al., 2022b), random selection, and easiest example selection (Sorscher et al., 2022; Paul et al., 2022). The pruned subsets are then used for movement pruning at varying levels of model sparsity. Finally, we perform a series of experiments in the context of NMT to tease apart the role of training data during movement pruning and make the following contributions.
1.1 Contributions

1. We find that fine-pruning T5 on hard-to-learn examples identified through cross-entropy score yields better BLEU score on two NMT datasets than training on easy-to-learn, random or stratified subsets of the training data.

2. We demonstrate that selecting hard-to-learn examples leads to the least reduction of vocabulary in the pruned dataset which helps explain higher performance achieved through these examples.

3. We observe that an unpruned model is better for ranking the examples and pruning the data as reduced model capacity asymmetrically reduces the capability of identifying hard-to-learn examples.

4. We find that the score rankings are transferable to other models —the subsets generated through one model (e.g., T5) can be used for fine-pruning another model (e.g., BART).

2 Related Work

Scoring individual instances The problem of scoring individual instances has been studied extensively for classification tasks in NLP. Swayamdipta et al. (2020) use data maps to visualize and score instances using training dynamics and identify three broad instance classes—easy-to-learn, hard-to-learn, and ambiguous by measuring the confidence of the true prediction and its variability across epochs. They find that high performance can be achieved by training on ambiguous instances while easy-to-learn instances aid optimization. However, this approach is not directly applicable to tasks other than classification, e.g., NMT, where the confidence and variability of individual examples need to be defined differently. Earlier work on standard vision tasks (Paul et al., 2021) has demonstrated that high-scoring, hard-to-learn examples primarily drive learning in neural networks, based on the observation that data-dependent Neural Tangent Kernel (NTK) submatrix for harder examples evolves faster during training than other examples. Our work considers intrinsic and extrinsic example scoring metrics in the context of NMT for identifying informative examples.

Data Pruning The primary aim of data pruning methods for deep learning models is the identification of informative training examples using different heuristics and removing redundant samples from the dataset (Kaushal et al., 2019; Saadatfar et al., 2020; Durga et al., 2021; Kothawade et al., 2021; Killamsetty et al., 2021; Paul et al., 2021; Ahia et al., 2021). Toneva et al. (2018) consider the ‘forgetfulness’ of a training example by measuring the number of times it is misclassified after being classified correctly during training, i.e., the forgetting score. The repeatedly forgotten examples are selected to construct a smaller training subset without significantly affecting the generalization performance. Paul et al. (2021) show that gradient norm (GraNd) can be used for removing a large number of less informative examples while retaining the test accuracy on multiple standard vision datasets (CIFAR-10 and CIFAR-100) and convolutional neural networks (ResNet). Paul et al. (2022) conduct an empirical study on the impact of data subsets for iterative magnitude pruning during neural network pre-training in image classification. They find that pre-training on the easier training examples reduces the number of steps required for finding a suitable initialization in iterative magnitude pruning.

Active learning The primary goal in active learning is to select the most informative examples from a pool of unlabelled examples that should be labeled first. In NLP, active learning has been studied for text classification (Ru et al., 2020; Yu et al., 2021), visual question answering (Karamcheti et al., 2021) and sentiment analysis (Venugopalan and Gupta, 2022) amongst other domains. We do not consider active learning methods in this work since our core objective is to prune data by selecting the examples from a fully labeled dataset (with text and reference translations).

Data subset selection for NMT Apart from active learning, several other subset selection methods have been proposed for NMT to achieve better performance. This includes noisy data filtering (Pham et al., 2018; Ramírez-Sánchez et al., 2020), contrastive data selection (Moore and Lewis, 2010), scoring sentence pairs with dual cross-entropy (Junczys-Dowmunt, 2018; Koehn et al., 2020), multilingual data, (Chaudhary et al., 2019; Wang and Neubig, 2019) and bilingual mappings (Lo and Joannis, 2020). Ahia et al. (2021) evaluate the impact of model pruning with low-resourced data for NMT.
They find that in a low-data regime, less model capacity rather than more aids out-of-distribution generalization. Additionally, they observe that pruning affects performance on long-tail of data distribution more than prototypical instances. This work considers the resource-constrained environment at deployment time. In contrast, the primary focus of our work is to consider pruning within the constraints at training time. Additionally, to the best of our knowledge, our work is the first to combine movement pruning with data pruning for NMT.

3 Preliminaries

In this section, we first introduce Neural Machine Translation (NMT) and then present model pruning and data pruning methods in the context of NMT.

3.1 Neural Machine Translation

The fundamental goal of an NMT model is to translate a source sentence $X = \{x_1, \ldots, x_S\}$ into a target sentence $Y = \{y_1, \ldots, y_T\}$, where $S$ and $T$ are the number of tokens in $X$ and $Y$ respectively. The following chain rule describes the probability of each token in the target sentence conditioned on the source sentence:

$$P(Y | X; \theta) = \prod_{i=1}^{T} p(y_i | y_{0:i-1}, X; \theta)$$  

(1)

where $\theta$ represents the model parameters. The NMT models optimize cross-entropy (CE) loss by minimizing negative log-likelihood of the samples during training:

$$\mathcal{L}_{CE}(\theta) = -\sum_{i=1}^{N} \log p(y_i | y_{0:i}, X; \theta)$$  

(2)

In the inference phase, the probabilities for the target tokens are generated through an autoregressive process. These probabilities are utilized for the selection of high probability tokens using search heuristics like beam search.

3.2 Model Pruning

The goal of model pruning methods is to reduce the memory footprint and increase the efficiency of neural networks through sparsity induction. The two primary approaches for pruning language models are (i) structured and (ii) unstructured. Structured pruning aims to remove network blocks, whereas unstructured pruning removes the least important weights wherever they occur in the network.

3.2.1 Magnitude Pruning

Magnitude pruning is an unstructured pruning approach where the weights to be pruned are determined by the importance scores $S$ assigned to each weight $i, j$ in the weight matrix $W$. The parameter mask $M$ is used to retain the top $k\%$ weights and zero out the others.

$$M(S) = \begin{cases} 1, & S_{i,j} \text{ in top } k\% \\ 0, & \text{otherwise} \end{cases}$$  

(3)

The model is pruned by replacing the original weight matrix with the masked version.

$$W' = W \odot M(S)$$  

(4)

3.2.2 Movement Pruning

Movement pruning is an unstructured pruning method that considers changes in weights (i.e., their movement) during fine-tuning (Sanh et al., 2020). It involves joint fine-tuning and compression in the fine-pruning phase during which the sparsity of the model is gradually increased from an initial value $s_i$ to a final value $s_f$ over $n$ pruning steps through automated gradual pruning (Zhu and Gupta, 2017). The key difference between this approach and magnitude pruning is that the weights can be pruned if they shrink during training, regardless of their magnitude. Hence, it considers the 1st-order information instead of the 0th-order information used in magnitude pruning. For high sparsity levels, movement pruning can perform better than magnitude pruning. It is better suited for the transfer learning regime as it combines fine-tuning and compression into a fine-pruning step.

4 Method

We consider a language model $l(x; \theta) (\theta \in \mathcal{R}^d)$ pre-trained on a generic dataset through objective $\mathcal{L}_p$. This model is fine-pruned for the downstream task of machine translation through movement pruning on the dataset $x \in \mathcal{D}_t$. $\mathcal{D}_t$ consists of sequence pairs $(x_i, y_i)$ where $x_i$ is the source sentence in one language and $y_i$ is the translation in another language. Our goal is to prune $\mathcal{D}_t$ through different heuristics to obtain a smaller dataset $\mathcal{D}_s$ and analyze the impact of fine-pruning the NMT model $l(x; \theta)$ using this limited data. Specifically, we consider the changes in test BLEU performance of pruned model and the impact on the training time during movement pruning using limited data.
4.1 Pruning Metric

The existing data pruning methods for neural networks in vision and speech tasks leverage pruning metrics—for example, normed error (EL2N) (Paul et al., 2021), forgetting scores (Toneva et al., 2018), forgetting norm (Azeemi et al., 2022a)—for ranking the training examples according to their difficulty. The data pruning method then operates on this ranking to construct an informative data subset by selecting the easy/hard examples according to the task properties. Drawing inspiration from this, we leverage training dynamics of language models and propose two pruning metrics for ranking examples on the task of NMT—(i) Cross-entropy loss (Eq. 2) and (ii) the BLEU score of individual examples during training (§6.1). CE loss can be considered as an intrinsic ranking metric that compares the model output with labels, while BLEU score is an extrinsic metric that compares the candidate translation with the reference translation. These metrics are used in the dataset pruning algorithm, which we present next.

![Image of Data pruning with movement pruning for NMT](image)

**Figure 1:** Data pruning with movement pruning for NMT. In step (1), we fine-tune a pre-trained language model on the complete NMT dataset and record training scores for each training example (e.g., cross-entropy score). In step (2), a data subset is created by ranking the examples according to the score and selecting the easy/hard examples according to the pruning strategy. This pruned dataset is used in step (3) to fine-prune the model and evaluate the test set.

4.2 Dataset Pruning Algorithm

We now present the dataset pruning algorithm for NMT (Algorithm 1). We first fine-tune the pre-trained language model for NMT on the complete (unpruned) dataset $D_s$. At the end of fine-tuning, we compute the training scores for each example through the pruning metric $e$, e.g., cross-entropy score. We then prune the dataset through the computed scores and the pruning strategy $s$. We consider three pruning strategies, Top-K, Bottom-K and Stratified, which select the hardest, easiest and representative examples respectively using the computed scores.

**How does data pruning enable efficient fine-pruning?** The initial fine-tuning to compute the ranking of the training examples (step 1 in Fig. 1) is done only once before the actual fine-pruning. This ranking is then used to create an optimal subset through the pruning strategy. This pruned dataset can be utilized for the actual fine-pruning in resource-constrained settings as it requires less time and memory (§6.2). Thus, the cost of initial fine-tuning run to compute the scores is amortized across the efficiency improvements achieved via multiple fine-prunings done using the pruned data, potentially on other models (see §6.3).

**Algorithm 1 Dataset Pruning for NMT**

<table>
<thead>
<tr>
<th>Input: Pre-trained language model $l$, Dataset $D_s$, Data Pruning Fraction $p$, Pruning strategy $s$, Pruning metric $e$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S \leftarrow$ Fine-tune $f$ on $D_s$ and compute scores for each example through $e$</td>
</tr>
<tr>
<td>$size \leftarrow (1 - p) \ast \text{len}(D_s)$</td>
</tr>
<tr>
<td>$S \leftarrow \text{sortDescending}(S)$</td>
</tr>
<tr>
<td>if $s = \text{topK}$ then</td>
</tr>
<tr>
<td>$D_l \leftarrow S[0 : size]$</td>
</tr>
<tr>
<td>else if $s = \text{bottomK}$ then</td>
</tr>
<tr>
<td>$D_l \leftarrow S[\text{len}(D_s) - size : \text{len}(D_s)]$</td>
</tr>
<tr>
<td>else if $s = \text{stratified}$ then</td>
</tr>
<tr>
<td>$D_l \leftarrow \text{stratifiedSampling}(D_s, size)$</td>
</tr>
<tr>
<td>else if $s = \text{random}$ then</td>
</tr>
<tr>
<td>$D_l \leftarrow \text{randomSampling}(D_s, size)$</td>
</tr>
<tr>
<td>end if</td>
</tr>
</tbody>
</table>

5 Experiments

5.1 Setup

**Datasets.** We evaluate our approach on WMT16 En–Ro and WMT16 En–Tr parallel datasets (Bojar et al., 2016). En–Ro is selected as a medium difficulty dataset while En–Tr is selected as a challenging dataset for NMT due to the rich agglutinative
Figure 2: Test BLEU for fine-pruning T5 on subsets selected through training cross-entropy scores across different strategies — Top-K: Selecting **hard-to-learn** examples — Bottom-K: Selecting **easy-to-learn** examples — Random: Selecting random examples — Stratified: Selecting examples through stratified sampling of cross-entropy scores. For each result, we do two runs and report the mean BLEU score.

---

**morphology** of Turkish and differences in word order (SVO in English, SOV in Turkish). We consider the translation tasks of Ro → En and Tr → En for evaluation. The statistics are shown in Table 1.

<table>
<thead>
<tr>
<th></th>
<th>Train</th>
<th>Dev</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>En → Ro</td>
<td>610,320</td>
<td>1,999</td>
<td>1,999</td>
</tr>
<tr>
<td>En → Tr</td>
<td>205,756</td>
<td>1,001</td>
<td>3,000</td>
</tr>
</tbody>
</table>

Table 1: Size of the train, development and test sets for En-Tr and En-Ro datasets.

**Model** We use the T5 multi-lingual pre-trained language model for evaluation. T5 (Raffel et al., 2020) is an encoder-decoder transformer model (Vaswani et al., 2017) which frames every task as a text-to-text problem. This allows using the same model and the loss function on multiple NLP tasks. We use the T5-small variant pre-trained on the 750 GB C4 dataset containing text from the public web scrape of the common crawl. This variant has 60 million parameters, 6 layers in the encoder and decoder each, and 8-headed attention.

**Model Pruning Setup.** We fine-prune the language model through movement pruning to different levels of target sparsity {10%, 50%} using data subsets at pruning fractions of {20%, 40%, 60%, 80%}. We compute sparsity as the number of pruned parameters divided by the model size. The attention heads and dense layers are pruned during training by gradually increasing the sparsity level through a cubic sparsity scheduler. The model is fine-pruned until convergence.

**Baselines.** We choose Random selection and Stratified sampling as our baselines. For random selection, we prune the training set randomly according to the specified pruning percentage and then fine-prune the model on the pruned subset. For the second baseline, we compute the cross entropy scores of individual examples similar to Algorithm 1 and perform stratified sampling. This constructs a representative subset by selecting examples from every sub-population, which results in a subset containing examples of varying difficulty and has been shown to outperform random sampling for speech tasks (Azeemi et al., 2022b).

6 Results and Discussion

Figure 2 shows the complete results for BLEU on the development datasets after fine-pruning T5 on the subsets selected through Top-K, Bottom-K, Random and Stratified strategies. The sweep across pruning percentage demonstrates consistently higher BLEU for fine-pruning on subsets consisting of hard-to-learn examples (Top-K strategy). Bottom-K performs the worst, indicating that the selection of the easy-to-learn examples is not
a good choice in a limited data regime, especially for challenging datasets like En-Tr.

To identify the subpopulations being selected by each pruning strategy, we analyze the distribution of training cross-entropy scores (Figure 3). For En-Ro dataset, we observe a long-tail of hard-to-learn examples and thus Top-K strategy is selecting these examples to an extent determined by pruning percentage. In contrast, En-Tr being a challenging dataset, has a significantly smaller number of easy-to-learn examples. Despite having different training distribution, the better performance of Top-K compared to other strategies (Table 2) signifies the appropriateness of selecting hard-to-learn examples for fine-pruning T5 for NMT. We hypothesize that this is due to the greater inclusion of informative examples in Top-K subsets which we verify in §6.4.

**Data pruning without model pruning.** The first column in Fig. 2 shows the result of pruning data without pruning the language model. We notice that up to 60% pruning, regular and sparse models demonstrate comparable test BLEU score. Beyond this—one extreme pruning percentages (≥ 80%)—the decrease in BLEU is greater for model pruning. This suggests that for the majority of data pruning percentages, sparse models are indeed suitable for practical usage.

### 6.1 Can we use an extrinsic pruning metric?

The original pruning algorithm considers the cross-entropy loss of individual examples as the pruning metric. We now consider using the BLEU score of individual training examples for data pruning. This is an extrinsic metric that compares the candidate translation with the reference translation. The distribution of the training BLEU score (shown in Figure 4) is different from the cross-entropy distribution (Figure 3). Particularly, the long-tail we observe in the distribution of cross-entropy scores of En-Ro dataset corresponding to the rare, hard-to-learn examples is not present for training BLEU scores distribution of En-Ro.

We next evaluate our pruning algorithm with BLEU training scores instead of the cross-entropy pruning scores (Table 2) on 10% model sparsity. No strategy consistently outperforms random subset selection for Ro → En implying that the BLEU score is not a suitable pruning metric as compared to cross-entropy training scores.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Strategy</th>
<th>Pruning Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>En-Ro</td>
<td>Random</td>
<td>20% 40% 60% 80%</td>
</tr>
<tr>
<td></td>
<td>Top-K</td>
<td>25.74 25.89 25.57 23.04</td>
</tr>
<tr>
<td></td>
<td>Bottom-K</td>
<td>25.59 25.56 25.62 22.39</td>
</tr>
<tr>
<td></td>
<td>Stratified</td>
<td>25.93 25.83 25.82 21.70</td>
</tr>
<tr>
<td></td>
<td>Stratified</td>
<td>25.82 25.56 25.93 22.73</td>
</tr>
</tbody>
</table>

Table 2: Test BLEU for fine-pruning on subsets selected through training BLEU scores across different strategies at 10% model sparsity. No single pruning method consistently performs better than the random pruning baseline.
6.2 Does data pruning reduce the fine-pruning time?

We conduct an experiment to quantify the reduction in training time and the impact on convergence steps during movement pruning. In Figure 5, we observe a significant reduction in the overall steps required for convergence for pruned subsets—for example, fine-pruning with 40% data is 48.9% faster than training with 80% data for En-Tr dataset. For En-Ro, increasing the pruning percentage from 20% to 60% reduces the convergence steps by 29.6% (54000 → 38000) while only decreasing the BLEU by 2.22% (26.08 → 25.50) for Top-K strategy (Figure 6). This demonstrates that data pruning reduces the memory and time requirements during fine-pruning, thus enabling training in compute-restricted environments. Moreover, we observe that the convergence steps are linearly proportional to the number of examples, implying that pruned datasets consisting of high-scoring examples do not negatively affect the convergence rate. This finding is consistent with recent work on data pruning in vision tasks (Sorscher et al., 2022), which demonstrated that the convergence time for pruned datasets is primarily determined by the number of training examples.

Figure 5: The convergence steps for fine-pruning T5 on the pruned subsets for En-Ro and En-Tr at 10% sparsity for movement pruning. Dataset pruning significantly reduces the steps required for convergence and hence the real time required for fine-pruning.

Practical efficiency improvements. The initial computation of the pruning metric before fine-pruning needs to be done once for a particular dataset. Hence, the initial setup cost amortizes over the efficiency improvements achieved with every subsequent fine-pruning done using the pruned dataset. Finally, the choice of the dataset pruning percentage can be made according to the compute constraints present at training time. Alternatively, the desired final BLEU range can be used to determine the corresponding pruning fraction and subsequently the most suitable compute environment for fine-pruning the model.

6.3 Are the pruning scores transferable across models?

The pruned subsets are generated by ranking the training examples through a pruning metric. Intuitively, these subsets should reflect the properties of the training data instead of a specific model. We now perform an empirical analysis to determine if the pruned subsets generated through one model can be used for fine-pruning another model i.e., if the score rankings are transferable. We consider the subsets of En-Tr dataset pruned through the T5 cross-entropy scores and use them for fine-pruning BART-base, which is another transformer encoder-decoder model that works well for translation tasks (Lewis et al., 2019). The results (Fig. 7) are intriguing; we observe that the same pruned subsets are effective for fine-pruning BART-base. From these observations, we hypothesize that the relative ranking of cross-entropy scores and thus the pruned subsets are dataset-specific and model-agnostic which allows them to be used for different models. Experiments on other datasets would serve to validate these findings.
6.4 How does data pruning change the training distribution of NMT datasets?

To understand the changes in the distribution of pruned subsets that are contributing to better performance of Top-K strategy, it is essential to analyze the vocabulary of pruned subsets. We perform empirical analysis to determine the reduction in vocabulary for En→Tr (Figure 8) and En→Ro (Figure 9) datasets pruned through Top-K, Bottom-K and Stratified strategy.

We observe the least reduction in vocabulary size for Top-K pruning with a decrease of 83,911 unique tokens for Tr (at 60% pruning) as compared to a reduction of 107,370, 107,906, and 145,834 unique tokens for Stratified, Random and Bottom-K respectively. As noted earlier, Top-K shows the highest test BLEU for multiple pruning percentages (Fig. 2). This signifies that hard-to-learn examples are essential for learning during fine-pruning, regardless of their distribution in the unpruned dataset.

6.5 Why is an unpruned model better for ranking the examples?

The original pruning strategy (§4.2) computes the cross-entropy scores and ranks the examples by fine-tuning the unpruned model. We compare this with an alternate strategy of computing the scores for the complete dataset through the sparse model, i.e., after fine-pruning. Fig. 10 shows the difference between the distribution of scores computed with an unpruned T5 model and a pruned T5 model (at 10% sparsity) for En→Tr dataset. We find that the absolute cross-entropy scores computed through the pruned model are shifted to the left with a visibly longer tail of harder examples, suggesting that reduced model capacity asymmetrically reduces the capability of identifying hard-to-learn examples. We also observe a sharp peak of the examples with a training cross-entropy score close to zero for the sparse model, which is not present for the unpruned model, indicating that pruned model exhibits a lower training error on easy-to-learn examples. However, this is not necessarily beneficial due to the clumped-together scores, which prevent the deterministic ranking of easier examples for subsequent pruning.
7 Limitations

We list the potential limitations of our work below:

Other datasets. We evaluated our data pruning algorithm on two NMT datasets —En–Ro and En–Tr. Further empirical evaluation will help verify the generalization of our approach to other types of NMT datasets —for example, high-resource languages like WMT14 En–De (Bojar et al., 2014), noiser datasets like MTNT (Michel and Neubig, 2018), datasets with high OOV rate like Gnome and the ones from a different domain like the Ubuntu technical dataset.

Cheaper pruning metrics. The proposed pruning method requires a fine-tuning run to compute the cross-entropy scores and construct the ranking for all the training examples. Although this procedure is done before fine-pruning, it still contributes to the end-to-end cost. Cheaper example scoring metrics, e.g., self-supervised metrics that do not require a complete training run (Sorscher et al., 2022) might reduce the initial cost of data pruning and yield more efficient results.

8 Conclusion

In this work, we leverage training dynamics to devise a dataset pruning algorithm for efficient movement pruning in NMT. Experiments on two NMT datasets of varying difficulty show the advantage of selecting hard-to-learn examples for fine-pruning T5 language model. Finally, we demonstrate the desirable properties of the proposed pruning method, including minimal vocabulary changes and transferability to other models. Future work includes experimentation with the proposed pruning strategy on other downstream tasks and an in-depth analysis of the pruned subsets.

9 Ethical Impact

The data pruning strategies do not explicitly prevent unbalanced pruning of different subpopulations within the translation datasets. This can lead to the under-representation of certain groups in the source and target language subsets and introduce potential bias against certain entities. To mitigate these concerns, a comprehensive explainability and fairness evaluation of the models trained on pruned data should be conducted.


Song Han, Jeff Pool, John Tran, and William Dally. 2015. Learning both weights and connections for efficient neural network. Advances in neural information processing systems, 28.


A Implementation Details

We extend the HuggingFace nn-pruning\(^1\) package and implement data pruning methods for movement pruning specifically for machine translation tasks. Our implementation can be easily extended to allow any translation dataset\(^2\) and model\(^3\) available on HuggingFace (Wolf et al., 2019) to leverage the data pruning algorithm for computing the pruning metrics and generating a data subset for fine-pruning.

A.1 Models

We use the T5 model publicly available on HuggingFace transformers (Wolf et al., 2019). The HuggingFace repository is available under the Apache License 2.0 license.

A.2 Datasets

We use the WMT16 En–Ro and En–Tr datasets available under the CC-BY-SA license.

B Reproducibility and Hyperparameters

We report the hyperparameters used in our experiments in Table 3 tuned through the validation dataset.

<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>fine-pruning learning rate</td>
<td>3e (-5)</td>
</tr>
<tr>
<td>train batch size</td>
<td>32</td>
</tr>
<tr>
<td>eval batch size</td>
<td>32</td>
</tr>
<tr>
<td>num beams</td>
<td>4</td>
</tr>
<tr>
<td>pad-to-max-length</td>
<td>true</td>
</tr>
<tr>
<td>initial-threshold (density)</td>
<td>100%</td>
</tr>
<tr>
<td>label-smoothing</td>
<td>0.1</td>
</tr>
</tbody>
</table>

Table 3: Hyperparameters for the experiments.

C Resources

We use a single 48GB NVIDIA A6000 GPU for running our experiments on a privately hosted server. We consumed a total of 2710 GPU hours for the entirety of the project including early experiments and the final results.

\(^1\)https://github.com/huggingface/nn_pruning
\(^2\)https://huggingface.co/datasets?task_categories=task_categories:translation
\(^3\)https://huggingface.co/models?pipeline_tag=translation