OPTIMIZING DATA USAGE VIA DIFFERENTIABLE REWARDS

Anonymous authors
Paper under double-blind review

ABSTRACT

To acquire a new skill, humans learn better and faster if a tutor, based on their current knowledge level, informs them of how much attention they should pay to particular content or practice problems. Similarly, a machine learning model could potentially be trained better with a scorer that "adapts" to its current learning state and estimates the importance of each training data instance. Training such an adaptive scorer efficiently is a challenging problem; in order to precisely quantify the effect of a data instance at a given time during the training, it is typically necessary to first complete the entire training process. To efficiently optimize data usage, we propose a reinforcement learning approach called Differentiable Data Selection (DDS). In DDS, we formulate a scorer network as a learnable function of the training data, which can be efficiently updated along with the main model being trained. Specifically, DDS updates the scorer with an intuitive reward signal: it should up-weigh the data that has a similar gradient with a dev set upon which we would finally like to perform well. Without significant computing overhead, DDS delivers strong and consistent improvements over several strong baselines on two very different tasks of machine translation and image classification.[1]

1 INTRODUCTION

While deep learning models are remarkably good at fitting large data sets, their performance is also highly sensitive to the structure and domain of their training data. Training on out-of-domain data can lead to worse model performance, while using more relevant data can assist transfer learning. Previous work has attempted to create strategies to handle this sensitivity by selecting subsets of the data to train the model on (Jiang & Zhai, 2007; Wang et al.; Axelrod et al., 2011; Moore & Lewis, 2010), providing different weights for each example (Sivasankaran et al., 2017; Ren et al., 2018), or changing the presentation order of data (Bengio et al., 2009; Kumar et al., 2019).

However, there are several challenges with the existing work on better data usage strategies. Most work data filtering criterion or training curriculum rely on domain-specific knowledge and hand-designed heuristics, which can be sub-optimal. To avoid hand designed heuristics, several works propose to optimize a parameterized neural network to learn the data usage schedule, but most of them are tailored to specific use cases, such as handling noisy data for classification (Jiang et al., 2018), learning a curriculum learning strategy for NMT (Kumar et al., 2019), and actively selecting data for annotation (Fang et al., 2017; Wu et al., 2018). Fan et al. (2018) proposes a more general teacher-student framework that first trains a teacher network to select data that directly optimizes development set accuracy over multiple training runs. However, because running multiple runs of training simply to train this teacher network entails an n-fold increase in training time for n runs, this is infeasible in many practical settings. In addition, in preliminary experiments we also found the single reward signal provided by dev set accuracy at the end of training noisy to the extent that we were not able to achieve results competitive with simpler heuristic training methods.

In this paper, we propose an alternative: a general Reinforcement Learning (RL) framework for optimizing training data usage by training a *scorer network* tha minimizes the model loss on the development set. We formulate the scorer network as a function of the current training examples only, making it possible to re-use the model architecture which is designed and trained for the main task.

[1]We will make the code publicly available upon acceptance.

Thus, our method requires no heuristics and is generalizable to various tasks. To make the scorer adaptive, we perform frequent and efficient updates of the scorer network using a reward function inspired by recent work on learning using data from auxiliary tasks (Du et al., 2018; Liu et al., 2019b), which use the similarity between two gradients as a measure of task relevance. We propose to use the gradient alignment between the training examples and the dev set as a reward signal for *a parametric scorer network*, as illustrated in Figure 1. We then formulate our framework as an optimization problem found in many prior works such as meta-learning (Finn et al., 2017), noisy data filtering (Ren et al., 2018), and neural architecture search (Liu et al., 2019a), and demonstrate that our proposed update rules follow a direct differentiation of the scorer parameters to optimize the model loss on the dev set. Thus we refer to our framework as "Differentiable Data Selection" (DDS).

We demonstrate two concrete instantiations of the DDS framework, one for a more general case of image classification, and the other for a more specific case of neural machine translation (NMT). For image classification, we test on both CIFAR-10 and ImageNet. For NMT, we focus on a multilingual setting, where we optimize data usage from a multilingual corpus to improve the performance on a particular language. For these two very different and realistic tasks, we find the DDS framework brings significant improvements over the baselines for all settings.

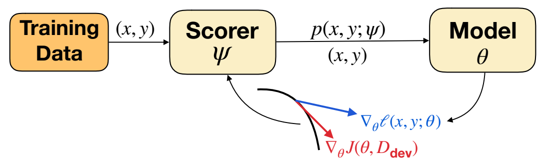

Figure 1: The general workflow of DDS.

2 Differentiable Data Selection

2.1 Risk, Training, and Development Sets

Commonly in machine learning, we seek to find the parameters θ^* that minimize the *risk* $J(\theta, P)$, the expected value of a loss function $\ell(x, y; \theta)$, where $\langle x, y\rangle$ are pairs of inputs and associated labels sampled from a particular distribution $P(X, Y)$:

$$\theta^* = \operatorname*{argmin}_{\theta} J(\theta, P) \quad \text{where} \quad J(\theta, P) = \mathbb{E}_{x,y\sim P(X,Y)}[\ell(x, y; \theta)] \tag{1}$$

Ideally, we would like the risk $J(\cdot)$ to be minimized over the data distribution that our system sees at test time, ie. $P_{\text{test}}(X, Y)$. Unfortunately, this distribution is unknown at training time, so instead we collect a training set $\mathcal{D}_{\text{train}} = \{(x_i, y_i) : i = 1, ..., N_{\text{train}}\}$ with distribution $P_{\text{train}}(X, Y) = \text{Uniform}(\mathcal{D}_{\text{train}})$, and minimize the *empirical risk* by taking $\langle x, y\rangle \sim P_{\text{train}}(X, Y)$. Since we need a sufficiently large training set $\mathcal{D}_{\text{train}}$ to train a good model, it is hard to ensure that $P_{\text{train}}(X, Y) \approx P_{\text{test}}(X, Y)$. In fact, we often accept that training data comes from a different distribution than test data. The discrepancy between $P_{\text{train}}(X, Y)$ and $P_{\text{test}}(X, Y)$ manifests itself in the form of problems such as overfitting (Zhang et al., 2017; Srivastava et al., 2014), covariate shift (Shimodaira, 2000), and label shift (Lipton et al., 2018).

However, unlike the large training set, we can collect a relatively small development set $\mathcal{D}_{\text{dev}} = \{(x_i, y_i) : i = 1, ..., N_{\text{dev}}\}$ with distribution $P_{\text{dev}}(X, Y)$ which is much closer to $P_{\text{test}}(X, Y)$. Since $\mathcal{D}_{\text{dev}}$ is a better approximation of our test-time scenario, we can use $\mathcal{D}_{\text{dev}}$ to get reliable feedback to learn to better utilize our training data from $\mathcal{D}_{\text{train}}$. In particular, we propose to train a *scorer* network, parameterized by ψ, to provide guidance on training data usage to minimize $J(\theta, \mathcal{D}_{\text{dev}})$.

2.2 Reinforcement Learning for Optimizing Data Usage

We propose to optimize the scorer's parameters ψ in an RL setting. Our *environment* is the model state θ and an example $\langle x, y\rangle$. Our RL *agent* is the scorer network ψ, which optimizes the data usage for the current model state. The agent's *reward* on picking an example approximates the dev set performance of the resulting model after the model is updated on this example.

Our scorer network is parameterized as a differentiable function that only takes as inputs the features of the example $\langle x, y\rangle$. Intuitively, it represents a distribution over the training data where more important data has a higher probability of being used, denoted $P(X, Y; \psi)$. Unlike prior methods which generally require complicated featurization of both the model state and the data as input to the

RL agent (Fan et al., 2018; Jiang et al., 2018; Fang et al., 2017), our formulation is much simpler and generalizable to different tasks. Since our scorer network does not consider the model parameters θ_t as input, we update it iteratively with the model so that at training step t, $P(X, Y; \psi_t)$ provides an up-to-date data scoring feedback for a given θ_t.

Although the above formulation is simpler and more general, it requires much more frequent updates to the scorer parameter ψ. Existing RL frameworks simply use the change in dev set risk as the regular reward signal, which makes the update expensive and unstable (Fan et al., 2018; Kumar et al., 2019). Therefore, we propose a novel reward function as an approximation to $\Delta J_{\text{dev}}(x, y)$ to quantify the effect of the training example $\langle x, y\rangle$. Inspired by Du et al. (2018) (which uses gradient similarity between two tasks to measure the adaptation effect between them, we use the agreement between the model gradient on data $\langle x, y\rangle$ and the gradient on the dev set to approximate the effect of $\langle x, y\rangle$ on dev set performance. This reward simply implies that we prefer data that moves θ in the direction that minimizes the dev set risk:

$$R(x, y) = \Delta J_{\text{dev}}(x, y) \approx \nabla_\theta \ell(x, y; \theta_{t-1})^\top \cdot \nabla_\theta J(\theta_t, \mathcal{D}_{\text{dev}}) \tag{2}$$

According to the REINFORCE algorithm (Williams, 1992), the update rule for ψ is thus

$$\psi_{t+1} \leftarrow \psi_t + \eta \underbrace{\nabla_\theta \ell(x, y; \theta_{t-1}) \cdot \nabla_\theta J(\theta_t, \mathcal{D}_{\text{dev}})}_{R(x,y)} \nabla_\psi \log(P(X, Y; \psi)) \tag{3}$$

where η is the learning rate for the scorer parameter ψ. The update rule for the model is simply

$$\theta_t \leftarrow \theta_{t-1} + \alpha \nabla_\theta J(\theta_{t-1}, P(X, Y; \psi)) \tag{4}$$

where α is the learning rate for the model. By alternating between Eqn. 4 and Eqn. 3, we can iteratively update θ using the guidance from the scorer network, and update ψ to optimize the scorer using feedback from the model.

Our formulation of scorer network as $P(X, Y; \psi)$ has several advantages. First, it provides the flexibility that we can either sample a training instance or equivalently scale the update from the training instance based on its score. Specifically, we provide an algorithm under the DDS framework for multilingual NMT (see Sec. 3.2), where the former is more efficient, and another more general algorithm for image classification (see Sec. 3.1), where the latter choice is natural. Second, it allows easy integration of prior knowledge of the data, which is shown to be effective in Sec. 4.

2.3 Deriving Rewards through Direct Differentiation

In this section, we show that the update for the scorer network in Eqn. 3 can be approximately derived as the solution of a bi-level optimization problem (Colson et al., 2007), which has been applied to many different lines of research (Baydin et al., 2018; Liu et al., 2019a; Ren et al., 2018).

Under our framework, the scorer samples the data by $\langle x, y\rangle \sim P(X, Y; \psi)$, and ψ will be chosen so that θ^* that optimizes $J(\theta, P(X, Y; \psi))$ will approximately minimize $J(\theta, P_{\text{dev}}(X, Y))$:

$$\psi^* = \underset{\psi}{\operatorname{argmin}}\, J(\theta^*(\psi), \mathcal{D}_{\text{dev}}) \text{ where } \theta^*(\psi) = \underset{\theta}{\operatorname{argmin}}\, \mathbb{E}_{x,y\sim P(X,Y;\psi)}\left[\ell(x, y; \theta)\right] \tag{5}$$

The connection between ψ and θ in Eqn. 5 shows that $J(\theta_t, \mathcal{D}_{\text{dev}})$ is differentiable with respect to ψ. Now we can approximately compute the gradient $\nabla_\psi J(\theta_t, \mathcal{D}_{\text{dev}})$ as follows:

$$\begin{aligned}
\nabla_\psi J(\theta_t, \mathcal{D}_{\text{dev}}) &= \nabla_{\theta_t} J(\theta_t, \mathcal{D}_{\text{dev}})^\top \cdot \nabla_\psi \theta_t(\psi) && \text{(chain rule)} \\
&= \nabla_{\theta_t} J(\theta_t, \mathcal{D}_{\text{dev}})^\top \cdot \nabla_\psi \left(\theta_{t-1} - \nabla_\theta J(\theta_{t-1}, \psi)\right) && \text{(substitute } \theta_t \text{ from Eqn 4)} \\
&\approx -\nabla_{\theta_t} J(\theta_t, \mathcal{D}_{\text{dev}})^\top \cdot \nabla_\psi \left(\nabla_\theta J(\theta_{t-1}, \psi)\right) && \text{(assume } \nabla_\psi \theta_{t-1} \approx 0\text{)} \\
&= \nabla_\psi \mathbb{E}_{x,y\sim P(X,Y;\psi)} \left[\nabla_\theta J(\theta_t, \mathcal{D}_{\text{dev}})^\top \cdot \nabla_\theta \ell(x, y; \theta_{t-1})\right] \\
&= \mathbb{E}_{x,y\sim P(X,Y;\psi)} \left[\left(\nabla_\theta J(\theta_t, \mathcal{D}_{\text{dev}})^\top \cdot \nabla_\theta \ell(x, y; \theta_{t-1})\right) \cdot \nabla_\psi \log P(x, y; \psi)\right]
\end{aligned} \tag{6}$$

Here, we make a Markov assumption that $\nabla_\psi \theta_{t-1} \approx 0$, assuming that at step t, given θ_{t-1} we do not care about how the values of ψ from previous steps led to θ_{t-1}. Eqn. 6 leads to a rule to update ψ using gradient descent, which is exactly the same as the RL update rule in Eqn. 3.

Note that our derivation above does not take into the account that we might use different optimizing algorithms, such as SGD or Adam (Kingma & Ba, 2015), to update θ. We provide detailed derivations for several popular optimization algorithms in Appendix A.1.

One potential concern with our approach is that because we optimize ψ_t directly on the dev set using $J(\theta_t, \mathcal{D}_{\text{dev}})$, we may risk indirectly overfitting model parameters θ_t by selecting a small subset of data that is overly specialized. However we do not observe this problem in practice, and posit that this because (1) the influence of ψ_t on the final model parameters θ_t is quite indirect, and acts as a "bottleneck" which has similarly proven useful for preventing overfitting in neural models Grézl et al. (2007), and (2) because the actual implementations of DDS (which we further discuss in Section 3) only samples a subset of data from $\mathcal{D}_{\text{train}}$ at each optimization step, further limiting expressivity.

3 Concrete Instantiations of DDS

We now turn to discuss two concrete instantiations of DDS that we use in our experiments: a more generic example of classification, which should be applicable to a wide variety of tasks, and a specialized application to the task of multilingual NMT, which should serve as an example of how DDS can be adapted to the needs of specific applications.

3.1 Formulation for Classification

Algorithm 1: Training a classification model with DDS.

Input : $\mathcal{D}_{\text{train}}$, $\mathcal{D}_{\text{dev}}$
Output : Optimal parameters θ^*
1 Initializer θ_0 and ψ_0
2 **for** $t = 1$ **to** *num_train_steps* **do**
3 Sample B training data points $x_i, y_i \sim \text{Uniform}(\mathcal{D}_{\text{train}})$
4 Sample B validation data points $x'_i, y'_i \sim \text{Uniform}(\mathcal{D}_{\text{dev}})$
▷ *Optimize θ*
5 Update $\theta_t \leftarrow \text{GradientUpdate}\Big(\theta_{t-1}, \sum_{i=1}^{B} p(x_i, y_i; \psi_{t-1}) \nabla_\theta \ell(x_i, y_i; \theta_{t-1})\Big)$
▷ *Evaluate θ_t on $\mathcal{D}_{dev}$*
6 Let $d_\theta \leftarrow \frac{1}{B} \sum_{j=1}^{B} \nabla_\theta \ell(x'_j, y'_j; \theta_t)$
▷ *Optimize ψ*
7 Let $d_\psi \leftarrow \frac{1}{B} \sum_{i=1}^{B} \Big[\Big(d_\theta^\top \cdot \nabla_\theta \ell(x_i, y_i; \theta_{t-1})\Big) \cdot \nabla_\psi \log p(x_i, y_i; \psi)\Big]$
8 Update $\psi_t \leftarrow \text{GradientUpdate}(\psi_{t-1}, d_\psi)$
9 **end**

Algorithm 1 presents the pseudo code for the training process on classification tasks, using the notations introduced in Section 2. The main classification model is parameterized by θ. The scorer $p(X, Y; \psi)$ is an identical network with the main model, but with independent weights, *i.e.* $p(X, Y; \psi)$ does not share weights with θ. For each example x_i in a minibatch uniformly sampled from $\mathcal{D}_{\text{train}}$, this DDS model outputs a scalar from the data x_i. All scalars are passed through a softmax function to compute the relative probabilities of the examples in the minibatch, and their gradients are scaled accordingly when applied to θ. Note that our actual formulation of $p(X, Y; \psi)$ does *not* depend on Y, but we keep Y in the notation for consistency with the formulation of the DDS framework. Note that we have two gradient update steps, one for the model parameter θ_t in Line 5 and the other for the DDS scorer parameter ψ in Line 8. For the model parameter update, we can simply use any of the standard optimization update rule. For the scorer ψ, we use the update rule derived in Section 2.3.

Per-Example Gradient. As seen from Line 7 of Algorithm 1, as well as from Eqn. 11, DDS requires us to compute $\nabla_\theta \ell(x_i, y_i; \theta_{t-1})$, *i.e.* the gradient for each example in a batch of training data. This operation is very slow and memory intensive, especially when the batch size is large, *e.g.* our experiments on ImageNet use a batch size of 4096 (see Section 4). Therefore, we propose an efficient approximation of this per-example gradient computation via the first-order Taylor expansion of $\ell(x_i, y_i; \theta_{t-1})$. In particular, for any vector $v \in \mathbb{R}^{|\theta|}$, with sufficiently small $\epsilon > 0$, we have:

$$v^\top \cdot \nabla_\theta \ell(x_i, y_i; \theta_{t-1}) \approx \frac{1}{\epsilon} \left(\ell\big(x_i, y_i; \theta_{t-1} + \epsilon v\big) - \ell\big(x_i, y_i; \theta_{t-1}\big)\right), \tag{7}$$

Eqn 7 can be implemented by keeping a shadow version of parameters θ_{t-1}, caching training loss $\ell(x_i, y_i; \theta_{t-1})$, and computing the new loss with $\theta_{t-1} + \epsilon v$. Here, v is d_θ as in Line 7 of Algorithm 1.

3.2 Formulation for Multilingual NMT

Next we demonstrate an application of DDS to multilingual models for NMT, specifically for improving accuracy on low-resource languages (LRL) (Zoph et al., 2016; Neubig & Hu, 2018). In this setting, we assume that we have a particular LRL S that we would like to translate into target language T, and we additionally have a multilingual corpus $\mathcal{D}_{\text{train}}$ that has parallel data between n source languages $(S_1, S_2, ..., S_n)$ and target language T. We would like to pick parallel data from any of the source languages to the target language to improve translation of a particular LRL S, so we assume that $\mathcal{D}_{\text{dev}}$ exclusively consists of parallel data between S and T. Thus, DDS will attempt to select data from $\mathcal{D}_{\text{train}}$ that improve accuracy on S-to-T translation as represented by $\mathcal{D}_{\text{dev}}$.

Algorithm 2: Training multilingual NMT with DDS.

Input : $\mathcal{D}_{\text{train}}$; K: number of data to train the NMT model before updating ψ; E: number of updates for ψ; α_1,α_2: discount factors for the gradient
Output : The converged NMT model θ^*

Initialize ψ_0, θ_0
▷ *Initialize the gradient of each source language*
$grad[S_i] \leftarrow 0$ **for** *i in n*
while θ *not converged* **do**
    $X, Y \leftarrow$ load_data$(\psi, \mathcal{D}_{\text{train}}, K)$
    ▷ *Train the NMT model*
    for x_i, y *in* X, Y **do**
        $\theta_t \leftarrow$ GradientUpdate $\left(\theta_{t-1}, \nabla_{\theta_{t-1}} \ell(x_i, y; \theta_{t-1})\right)$
        $grad[S_i] \leftarrow \alpha_1 \times$ grad$[S_i] + \alpha_2 \times \nabla_{\theta_{t-1}} \ell(x_i, y; \theta_{t-1})$
    end
    ▷ *Optimize* ψ
    for *iter in E* **do**
        sample B data pairs from $\mathcal{D}_{\text{train}}$
        $d_\psi \leftarrow \frac{1}{B} \sum_{j=1}^{B} \sum_{i=1}^{n} \left[\text{grad}[S_i]^\top \text{grad}[S] \cdot \nabla_{\psi_{t-1}} \log \left(p\left(S_i | y_j; \psi_{t-1}\right)\right)\right]$
        $\psi_t \leftarrow$ GradientUpdate$(\psi_{t-1}, d_{\psi_{t-1}})$
    end
end

To make training more efficient and stable in this setting, we make three simple modifications of the main framework in Section 2.3 that take advantage of the problem structure of multilingual NMT. First, instead of directly modeling $p(X, Y; \psi)$, we assume a uniform distribution over the target sentence Y, and only parameterize the conditional distribution of which source language sentence to pick given the target sentence: $p(X|y; \psi)$. This design follows the formulation of Target Conditioned Sampling (TCS; Wang & Neubig (2019)), an existing state-of-the-art data selection method that uses a similar setting but models the distribution $p(X|y)$ using heuristics. Second, we only update ψ after updating the NMT model for a fixed number of steps. Third, we sample the data according to $p(X|y; \psi)$ to get a Monte Carlo estimate of the objective in Eqn. 5. This significantly reduces the training time compared to using all data. The pseudo code of the training process is in Algorithm 2.

4 Experiments

We now discuss experimental results on both image classification, an instance of the general classification problem using Algorithm 1, and multilingual NMT using Algorithm 2.

4.1 Experimental Settings

Data. We apply our method on established benchmarks for image classification and multilingual NMT. For image classification, we use CIFAR-10 (Krizhevsky, 2009) and ImageNet (Russakovsky et al., 2015). For each dataset, we consider two settings: a reduced setting where only roughly 10% of the training labels are used, and a full setting, where all labels are used. Specifically, the reduced setting for CIFAR-10 uses the first 4000 examples in the training set, and with ImageNet, the reduced setting uses the first 102 TFRecord shards as pre-processed by Kornblith et al. (2019). We use the size of 224×224 for ImageNet.

For multilingual NMT, we use the 58-language-to-English TED dataset (Qi et al., 2018). Following prior work (Qi et al., 2018; Neubig & Hu, 2018; Wang et al., 2019b), we evaluate translation from four low-resource languages (LRL) Azerbaijani (`aze`), Belarusian (`bel`), Galician (`glg`), and Slovak (`slk`) to English, where each is paired with a similar high-resource language Turkish (`tur`),

Table 1: Results for image classification accuracy (left) and multilingual MT BLEU (right). For MT, the statistical significance is indicated with $*$ (p < 0.005) and † (p < 0.0001).

Methods	CIFAR-10 (WRN-28-k)		ImageNet (ResNet-50)	
	4K, $k=2$	Full, $k=10$	10%	Full
Uniform	82.60±0.17	95.55±0.15	56.36/79.45	76.51/93.20
SPCL	81.09±0.22	93.66±0.12	-	-
BatchWeight	79.61±0.50	94.11±0.18	-	-
MentorNet	83.11±0.62	94.92±0.34	-	-
DDS	83.63± 0.29	96.31± 0.13	**56.81/79.51**	**77.23/93.57**
retrained DDS	**85.56±0.20**	**97.91±0.12**	-	-

Methods	aze	bel	glg	slk
Uniform	10.31	17.21	26.05	27.44
SPCL	9.07	16.99	23.64	21.44
Related	10.34	15.31	27.41	25.92
TCS	11.18	16.97	27.28	27.72
DDS	10.74	17.24	27.32	**28.20***
TCS+DDS	**11.84***	**17.74**†	**27.78**	27.74

Russian (`rus`), Portugese (`por`), and Czech (`ces`) (details in Appendix A.3). We combine data from all 8 languages, and use DDS to optimize data selection for each LRL.

Models and Training Details. For image classification, on CIFAR-10, we use the pre-activation WideResNet-28 (Zagoruyko & Komodakis, 2016), with width factor $k = 2$ for the reduced setting and $k = 10$ for the normal setting. For ImageNet, we use the post-activation ResNet-50 (He et al., 2016). These implementations reproduce the numbers reported in the literature (Zagoruyko & Komodakis, 2016; He et al., 2016; Xie et al., 2017), and additional details can be found in Appendix A.4.

For NMT, we use a standard LSTM-based attentional baseline (Bahdanau et al., 2015), which is similar to previous models used in low-resource scenarios both on this dataset (Neubig & Hu, 2018; Wang et al., 2019b) and others (Sennrich & Zhang, 2019) due to its relative stability compared to other options such as the Transformer (Vaswani et al., 2017). Accuracy is measured using BLEU score (Papineni et al., 2002). More experiment details are noted in Appendix A.2.

Baselines and Our Methods. For both image classification and multi-lingual NMT, we compare the following data selection methods. **Uniform** where data is selected uniformly from all of the data that we have available, as is standard in training models. **SPCL** (Jiang et al., 2015), a curriculum learning method that dynamically updates the curriculum to focus more on the "easy" training examples based on model loss. **DDS**, our proposed method.

For image classification, we compare with several additional methods designed for filtering noisy data on CIFAR-10, where we simply consider the dev set as the clean data. **BatchWeight** (Ren et al., 2018), a method that scales example training loss in a batch with a locally optimized weight vector using a small set of clean data. **MentorNet** (Jiang et al., 2018), a curriculum learning method that trains a mentor network to select clean data based on features from both the data and the main model. For machine translation, we also compare with two state-of-the-art heuristic methods for multi-lingual data selection. **Related** where data is selected uniformly from the target LRL and a linguistically related HRL (Neubig & Hu, 2018). **TCS**, a recently proposed method of "target conditioned sampling", which uniformly chooses target sentences, then picks which source sentence to use based on heuristics such as word overlap (Wang & Neubig, 2019). Note that both of these methods take advantage of structural properties of the multi-lingual NMT problem, and do not generalize to other problems such as classification.

DDS is a flexible framework to incorporate prior knowledge about the data using the scorer network, which can be especially important when the data has certain structural properties such as language or domain. We test such a setting of DDS for both tasks. For image classification, we use **retrained DDS**, where we first train a model and scorer network using the standard DDS till convergence. The trained scorer network can be considered as a good prior over the data, so we use it to train the final model from scratch again using DDS. For multilingual NMT, we experiment with **TCS+DDS**, where we initialize the parameters of DDS with the TCS heuristic, then continue training.

4.2 Main Results

The results of the baselines and our method are listed in Table 1. First, comparing the standard baseline strategy of "Uniform" and the proposed method of "DDS" we can see that in all 8 settings DDS improves over the uniform baseline. This is a strong indication of both the consistency of the improvements that DDS can provide, and the generality – it works well in two very different settings. Next, we find that DDS outperforms SPCL by a large margin for both of the tasks, especially for multilingual NMT. This is probably because SPCL weighs the data only by their easiness, while

Figure 3: Example images from the ImageNet and their weights assigned by DDS.

ignoring their relevance to the dev set, which is especially important in settings where the data in the training set can have very different properties such as the different languages in multilingual NMT.

DDS also brings improvements over the state-of-the-art intelligent data utilization methods. For image classification, DDS outperforms MentorNet and BatchWeight on CIFAR-10 in all settings. For NMT, in comparison to Related and TCS, vanilla DDS performs favorably with respect to these state-of-the-art data selection baselines, outperforming each in 3 out of the 4 settings (with exceptions of slightly underperforming Related on `glg` and TCS on `aze`). In addition, we see that incorporating prior knowledge into the scorer network leads to further improvements. For image classification, retrained DDS can significantly improve over regular DDS, leading to the new state-of-the-art result on the CIFAR-10 dataset. For mulitlingual NMT, TCS+DDS achieves the best performance in three out of four cases (with the exception of `slk`, where vanilla DDS already outperformed TCS).[2]

DDS does not incur much computational overhead. For image classification and multilingual NMT respectively, the training time is about $1.5\times$ and $2\times$ the regular training time without DDS[3].

4.3 Analysis

Image Classification. Prior work on heuristic data selection has found that the model performs better if we feed higher quality or more domain-relevant data towards the end of training (van der Wees et al., 2017; Wang et al., 2019a). Here we verify this observation by analyzing the learned importance weight at the end of training for image classification. Figure 2 shows that at the end of training, DDS learns to balance the class distribution, which is originally unbalanced due to the dataset creation. Figure 3 shows that at the end of training, DDS assigns higher probabilities to images with clearer class content from ImageNet. These results show that DDS learns to focus on higher quality data towards the end of training.

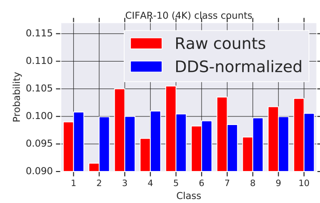

Figure 2: Class distributions of CIFAR-10 4K.

NMT. Next, we focus on multi-lingual NMT, where the choice of data directly corresponds to picking a language, which has an intuitive interpretation. Since DDS adapts the data weights dynamically to the model throughout training, here we analyze how the dynamics of learned weights.

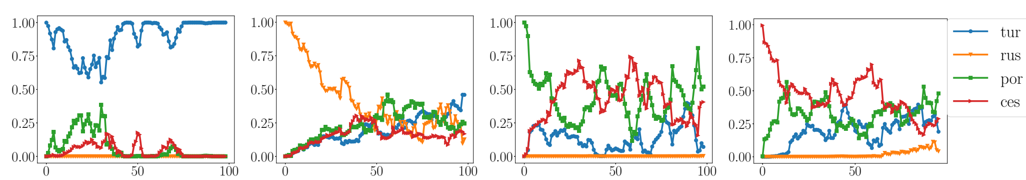

Figure 4: Language usage for TCS+DDS by training step. *From left to right*: `aze`, `bel`, `glg`, `slk`.

We plot the probability distribution of the four HRLs (because they have more data and thus larger impact on training) over the course of training. Figure 4 shows the change of language distribution for TCS+DDS. Since TCS selects the language with the largest vocabulary overlap with the LRL, the distribution is initialized to focus on the most related HRL. For all four LRLs, the percentage of their most related HRL starts to decrease as training continues. For `aze`, DDS quickly comes back to using its most related HRL. However, for `bel`, DDS continues the trend of using all four languages.

[2]For the NMT significance tests (Clark et al., 2011) find significant gains over the baseline for aze, slk, and bel. For glg the gain is not significant, but DDS-uniform without heuristics performs as well as the TCS baseline.

[3]The code for multilingual NMT is not optimized, so its training time could be reduced further

This shows that DDS is able to maximize the benefits of the multi-lingual data by having a more balanced usage of all languages.

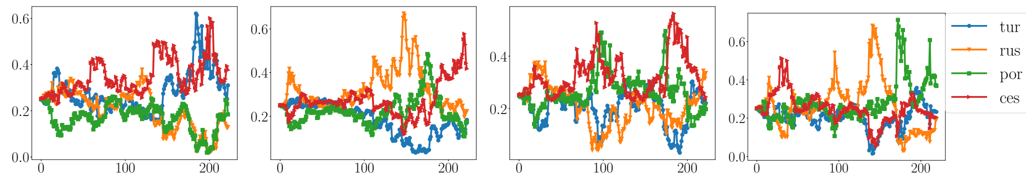

Figure 5: Language usage for DDS by training step. *From left to right*: `aze`, `bel`, `glg`, `slk`.

Figure 5 shows a more interesting trend of DDS without heuristic initialization. For both `aze` and `bel`, DDS focuses on the most related HRL after a certain number of training updates. Interestingly, for `bel`, DDS learns to focus on both `rus`, its most related HRL, and `ces`. Similarly for `slk`, DDS also learns to focus on `ces`, its most related HRL, and `rus`, although there is little vocabulary overlap between `slk` and `rus`. Also notably, the ratios change significantly over the course of training, indicating that different types of data may be more useful during different learning stages.

5 Related Work

Many machine learning approaches consider how to best present data to models. First, difficulty-based curriculum learning estimates the presentation order based on heuristic understanding of the hardness of examples (Bengio et al., 2009; Spitkovsky et al., 2010; Tsvetkov et al., 2016; Zhang et al., 2016; Graves et al., 2017; Zhang et al., 2018; Platanios et al., 2019). These methods, though effective, often generalize poorly because they require task-specific difficulty measures. On the other hand, self-paced learning (Kumar et al., 2010; Lee & Grauman, 2011) defines the hardness of the data based on the loss from the model, but is still based on the assumption that the model should learn from easy examples. Our method does not make these assumptions. Closest to the learning to teach framework (Fan et al., 2018) but their formulation involves manual feature design and requires expensive multi-pass optimization. Instead, we formulate our reward using bi-level optimization, which has been successfully applied for a variety of other tasks (Colson et al., 2007; Anandalingam & Friesz, 1992; Liu et al., 2019a; Baydin et al., 2018; Ren et al., 2018).

Data selection for domain adaptation for disparate tasks has also been extensively studied (Moore & Lewis, 2010; Axelrod et al., 2011; Ngiam et al., 2018; Jiang & Zhai, 2007; Foster et al., 2010; Wang et al.). These methods generally design heuristics to measure domain similarity. Submodular optimization (Kirchhoff & Bilmes, 2014; Tschiatschek et al., 2014) selects training data that are similar to dev set, but the criterion is often based on hand-designed features and the data usage is predefined before training. Besides domain adaptation, selecting also benefits training in the face of noisy or otherwise undesirable data (Vyas et al., 2018; Pham et al., 2018).

Our method is also related to works on training instance weighting (Sivasankaran et al., 2017; Ren et al., 2018; Jiang & Zhai, 2007; Ngiam et al., 2018). These methods reweigh data based on a manually computed weight vector, instead of using a parameterized neural network. Notably, Ren et al. (2018) tackles noisy data filtering for image classification, by using meta-learning to calculate a locally optimized weight vector for each batch of data. In contrast, our work focuses on the general problem of optimizing data usage. We train a parameterized scorer network that optimizes over the entire data space, which can be essential in preventing overfitting mentioned in Sec. 2; empirically our method outperform Ren et al. (2018) by a large margin in Sec. 4. (Wu et al., 2018; Kumar et al., 2019; Fang et al., 2017) propose RL frameworks for specific natural language processing tasks, but their methods are less generalizable and requires more complicated featurization.

6 Conclusion

We present Differentiable Data Selection, an efficient RL framework for optimizing training data usage. We parameterize the scorer network as a differentiable function of the data, and provide an intuitive reward function for efficiently training the scorer network. We formulate two algorithms under the DDS framework for two realistic and very different tasks, image classification and multilingual NMT, which lead to consistent improvements over strong baselines.

REFERENCES

G. Anandalingam and Terry L. Friesz. Hierarchical optimization: An introduction. *Annals OR*, 1992.

Amittai Axelrod, Xiaodong He, and Jianfeng Gao. Domain adaptation via pseudo in-domain data selection. In *EMNLP*, 2011.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *ICLR*, 2015.

Atilim Gunes Baydin, Robert Cornish, David Martínez-Rubio, Mark Schmidt, and Frank Wood. Online learning rate adaptation with hypergradient descent. In *ICLR*, 2018.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *ICML*, 2009.

Jonathan H. Clark, Chris Dyer, Alon Lavie, and Noah A. Smith. Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In *ACL*, 2011.

Benoît Colson, Patrice Marcotte, and Gilles Savard. An overview of bilevel optimization. *Annals OR*, 153(1), 2007.

Yunshu Du, Wojciech M. Czarnecki, Siddhant M. Jayakumar, Razvan Pascanu, and Balaji Lakshminarayanan. Adapting auxiliary losses using gradient similarity. *CoRR*, abs/1812.02224, 2018. URL http://arxiv.org/abs/1812.02224.

Yang Fan, Fei Tian, Tao Qin, Xiang-Yang Li, and Tie-Yan Liu. Learning to teach. In *ICLR*, 2018.

Meng Fang, Yuan Li, and Trevor Cohn. Learning how to active learn: A deep reinforcement learning approach. In *EMNLP*, pp. 595–605, 2017.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017.

George Foster, Cyril Goutte, and Roland Kuhn. Discriminative instance weighting for domain adaptation in statistical machine translation. In *EMNLP*, 2010.

Alex Graves, Marc G. Bellemare, Jacob Menick, Rémi Munos, and Koray Kavukcuoglu. Automated curriculum learning for neural networks. In *ICML*, 2017.

Frantisek Grézl, Martin Karafiát, Stanislav Kontár, and Jan Cernocky. Probabilistic and bottle-neck features for lvcsr of meetings. In *ICASSP*, volume 4, pp. IV–757. IEEE, 2007.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CPVR*, 2016.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

Jing Jiang and ChengXiang Zhai. Instance weighting for domain adaptation in nlp. In *ACL*, 2007.

Lu Jiang, Deyu Meng, Qian Zhao, Shiguang Shan, and Alexander G. Hauptmann. Self-paced curriculum learning. In *AAAI*, 2015.

Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li, and Li Fei-Fei. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *ICML*, 2018.

Diederik P. Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

Katrin Kirchhoff and Jeff A. Bilmes. Submodularity for data selection in machine translation. In *EMNLP*, 2014.

Simon Kornblith, Jonathon Shlens, and Quoc V. Le. Do better imagenet models transfer better? In *CVPR*, 2019.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

Gaurav Kumar, George Foster, Colin Cherry, and Maxim Krikun. Reinforcement learning based curriculum optimization for neural machine translation. In *NAACL*, pp. 2054–2061, 2019.

M. Pawan Kumar, Benjamin Packer, and Daphne Koller. Self-paced learning for latent variable models. In *NIPS*, 2010.

Yong Jae Lee and Kristen Grauman. Learning the easy things first: Self-paced visual category discovery. In *CVPR*, 2011.

Zachary C Lipton, Yu-Xiang Wang, and Alex Smola. Detecting and correcting for label shift with black box predictors. *arXiv preprint arXiv:1802.03916*, 2018.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: differentiable architecture search. 2019a.

Shikun Liu, Andrew J. Davison, and Edward Johns. Self-supervised generalisation with meta auxiliary learning. *CoRR*, abs/1901.08933, 2019b. URL `http://arxiv.org/abs/1901.08933`.

Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. In *ICLR*, 2017.

Robert C Moore and William Lewis. Intelligent selection of language model training data. In *ACL*, 2010.

Yurii E. Nesterov. A method for solving the convex programming problem with convergence rate $o(1/k^2)$. *Soviet Mathematics Doklady*, 1983.

Graham Neubig and Junjie Hu. Rapid adaptation of neural machine translation to new languages. *EMNLP*, 2018.

Jiquan Ngiam, Daiyi Peng, Vijay Vasudevan, Simon Kornblith, Quoc V. Le, and Ruoming Pang. Domain adaptive transfer learning with specialist models. *CVPR*, 2018.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *ACL*, 2002.

Minh Quang Pham, Josep Crego, Jean Senellart, and François Yvon. Fixing translation divergences in parallel corpora for neural MT. In *EMNLP*, 2018.

Emmanouil Antonios Platanios, Otilia Stretcu, Graham Neubig, Barnabas Poczos, and Tom Mitchell. Competence-based curriculum learning for neural machine translation. In *NAACL*, 2019.

Ye Qi, Devendra Singh Sachan, Matthieu Felix, Sarguna Padmanabhan, and Graham Neubig. When and why are pre-trained word embeddings useful for neural machine translation? *NAACL*, 2018.

Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. Learning to reweight examples for robust deep learning. In *ICML*, pp. 4331–4340, 2018.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 2015.

Rico Sennrich and Biao Zhang. Revisiting low-resource neural machine translation: A case study. In *ACL*, 2019.

Hidetoshi Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of statistical planning and inference*, 90(2):227–244, 2000.

Sunit Sivasankaran, Emmanuel Vincent, and Irina Illina. Discriminative importance weighting of augmented training data for acoustic model training. In *ICASSP*, 2017.

Valentin I. Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. From baby steps to leapfrog: How "less is more" in unsupervised dependency parsing. In *NAACL*, 2010.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. In *JMLR*, 2014.

Sebastian Tschiatschek, Rishabh K. Iyer, Haochen Wei, and Jeff A. Bilmes. Learning mixtures of submodular functions for image collection summarization. In *NIPS*, 2014.

Yulia Tsvetkov, Manaal Faruqui, Wang Ling, Brian MacWhinney, and Chris Dyer. Learning the curriculum with bayesian optimization for task-specific word representation learning. In *ACL*, 2016.

Marlies van der Wees, Arianna Bisazza, and Christof Monz. Dynamic data selection for neural machine translation. In *EMNLP*, 2017.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, pp. 5998–6008, 2017.

Yogarshi Vyas, Xing Niu, and Marine Carpuat. Identifying semantic divergences in parallel text without annotations. In *NAACL*, 2018.

Rui Wang, Masao Utiyama, Lemao Liu, Kehai Chen, and Eiichiro Sumita. Instance weighting for neural machine translation domain adaptation. In *EMNLP*.

Wei Wang, Isaac Caswell, and Ciprian Chelba. Dynamically composing domain-data selection with clean-data selection by "co-curricular learning" for neural machine translation. In *ACL*, 2019a.

Xinyi Wang and Graham Neubig. Target conditioned sampling: Optimizing data selection for multilingual neural machine translation. In *ACL*, 2019.

Xinyi Wang, Hieu Pham, Philip Arthur, and Graham Neubig. Multilingual neural machine translation with soft decoupled encoding. In *ICLR*, 2019b.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 1992.

Jiawei Wu, Lei Li, and William Yang Wang. Reinforced co-training. In *NAACL*, 2018.

Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *CVPR*, 2017.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *BMVC*, 2016.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *ICLR*, 2017.

Dakun Zhang, Jungi Kim, Josep Crego, and Jean Senellart. Boosting neural machine translation. *Arxiv 1612.06138*, 2016.

Xuan Zhang, Gaurav Kumar, Huda Khayrallah, Kenton Murray, Jeremy Gwinnup, Marianna J Martindale, Paul McNamee, Kevin Duh, and Marine Carpuat. An empirical exploration of curriculum learning for neural machine translation. *Arxiv, 1811.00739*, 2018.

Barret Zoph, Deniz Yuret, Jonathan May, and Kevin Knight. Transfer learning for low-resource neural machine translation. In *EMNLP*, 2016.

A APPENDIX

A.1 DERIVING GRADIENT OF ψ FOR DIFFERENT OPTIMIZERS

First, we rewrite the update rule of θ in Eqn. 4 to incorporate the effect of its specific optimization algorithm.

For a fixed value of ψ, $J(\theta, \psi)$ can be optimized using a stochastic gradient update. Specifically, at time step t, we update

$$\theta_t \leftarrow \theta_{t-1} - g\big(\nabla_\theta J(\theta_{t-1}, \psi)\big) \tag{8}$$

where $g(\cdot)$ is any function that may be applied to the gradient $\nabla_\theta J(\theta_{t-1}, \psi)$. For instance, in standard gradient descent $g(\cdot)$ is simply a linear scaling of $\nabla_\theta J(\theta_{t-1}, \psi)$ by a learning rate η_t, while with the Adam optimizer (Kingma & Ba, 2015) g also modifies the learning rate on a parameter-by-parameter basis.

Here we first derive $\nabla_\psi g$ for the general stochastic gradient descent (SGD) update, then provide examples for two other common optimization algorithms, namely Momentum (Nesterov, 1983) and Adam (Kingma & Ba, 2015).

SGD Updates. The SGD update rule for θ is as follows

$$\theta_t \leftarrow \theta_{t-1} - \eta_t \nabla_\theta J(\theta_{t-1}, \psi) \tag{9}$$

where η_t is the learning rate. Matching the updates in Eqn 9 with the generic framework in Eqn 8, we can see that g in Eqn 8 has the form:

$$g\big(\nabla_\theta J(\theta_{t-1}, \psi)\big) = \eta_t \nabla_\theta J(\theta_{t-1}, \psi) \tag{10}$$

This reveals a linear dependency of g on $\nabla_\theta J(\theta_{t-1,\psi})$, allowing the exact differentiation of g with respect to ψ. From Eqn **??**, we have

$$\begin{aligned}
&\nabla J(\theta_t, \mathcal{D}_{\text{dev}})^\top \cdot \nabla_\psi g\big(\nabla_\theta J(\theta_{t-1}, \psi)\big) \\
&= \eta_t \cdot \nabla_\psi \mathbb{E}_{x,y\sim p(X,Y;\psi)} \Big[J(\theta_t, \mathcal{D}_{\text{dev}})^\top \cdot \nabla_\theta \ell(x, y; \theta_{t-1}) \Big] \\
&= \eta_t \mathbb{E}_{x,y\sim p(X,Y;\psi)} \Big[\Big(J(\theta_t, \mathcal{D}_{\text{dev}})^\top \cdot \nabla_\theta \ell(x, y; \theta_{t-1}) \Big) \cdot \nabla_\psi \log p(x, y; \psi) \Big]
\end{aligned} \tag{11}$$

Here, the last equation follows from the log-derivative trick in the REINFORCE algorithm (Williams, 1992).

Momentum Updates. The momentum update rule for θ is as follows

$$\begin{aligned}
m_t &\leftarrow \mu_t m_{t-1} + \eta_t \nabla_\theta J(\theta_{t-1}, \psi) \\
\theta_t &\leftarrow \theta_{t-1} - m_t,
\end{aligned} \tag{12}$$

where μ_t is the momentum coefficient and η_t is the learning rate. This means that g has the form:

$$\begin{aligned}
g(x) &= \mu m_{t-1} + \eta_t x \\
g'(x) &= \eta_t
\end{aligned} \tag{13}$$

Therefore, the computation of the gradient ∇_ψ for the Momentum update is exactly the same with the standard SGD update rule in Eqn 11.

Adam Updates. We use a slightly modified update rule based on Adam (Kingma & Ba, 2015):

$$\begin{aligned}
g_t &\leftarrow \nabla_\theta J(\theta_{t-1}, \psi) \\
v_t &\leftarrow \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \\
\hat{v}_t &\leftarrow v_t / (1 - \beta_2^t) \\
\theta_t &\leftarrow \theta_{t-1} - \eta_t \cdot g_t / \sqrt{\hat{v}_t + \epsilon}
\end{aligned} \tag{14}$$

where β_2 and η_t are hyper-parameters. This means that g is a component-wise operation of the form:

$$\begin{aligned}
g(x) &= \frac{\eta_t \sqrt{1 - \beta_2^t} \cdot x}{\sqrt{\beta_2 v_{t-1} + (1 - \beta_2) x^2 + \epsilon}} \\
g'(x) &= \frac{\eta_t \sqrt{1 - \beta_2^t} (\beta_2 v_{t-1} + \epsilon)}{\big(\beta_2 v_{t-1} + (1 - \beta_2) x^2 + \epsilon\big)^{3/2}} \approx \eta_t \sqrt{\frac{1 - \beta_2^t}{\beta_2 v_{t-1}}},
\end{aligned} \tag{15}$$

the last equation holds because we assume v_{t-1} is independent of ψ. Here the approximation makes sense because we empirically observe that the individual values of the gradient vector $\nabla_\theta J(\theta_{t-1}, \psi)$, *i.e.* g_t, are close to 0. Furthermore, for Adam, we usually use $\beta_2 = 0.999$. Thus, the value $(1 - \beta_2)x^2$ in the denominator of Eqn 15 is negligible. With this approximation, the computation of the gradient ∇_ψ is almost the same with that for SGD in Eqn 11, with one extra component-wise scaling by the term in Eqn 15.

A.2 Hyperparameters for multilingual NMT

In this section, we give a detailed description of the hyperparameters used for the multilingual NMT experiments.

- We use a 1 layer LSTM with hidden size of 512 for both the encoder and decoder, and set the word embedding to size 128.
- The dropout rate is set to 0.3.
- For the NMT model, we use Adam optimizer with learning rate of 0.001. For the distribution parameter ψ, we use Adam optimizer with learning rate of 0.0001.
- We train all models for 20 epochs without any learning rate decay.
- We optimize both the NMT and DDS models with Adam, using learning rates of 0.001 and 0.0001 for θ and ψ respectively.

A.3 Dataset statistics for Multilingual NMT

LRL	Train	Dev	Test	HRL	Train
aze	5.94k	671	903	tur	182k
bel	4.51k	248	664	rus	208k
glg	10.0k	682	1007	por	185k
slk	61.5k	2271	2445	ces	103k

Table 2: Statistics of the multilingual NMT datasets.

A.4 Hyperparameters for image classification

In this section, we provide some additional details for the image classification task:

- We use the cosine learning rate decay schedule (Loshchilov & Hutter, 2017), starting at 0.1 for CIFAR-10 and 3.2 for ImageNet, both with 2000 warmup steps.
- We maintain a moving average of all model parameters with the rate of 0.999. Following Kornblith et al. (2019), we treat the moving statistics of batch normalization (Ioffe & Szegedy, 2015) as *untrained parameters* and also add them to the moving averages.
- For ImageNet, we use the post-activation ResNet-50 (He et al., 2016). The batch sizes for CIFAR-10 and for ImageNet are 128 and 4096, running for 200K steps and 40K steps, respectively.