Learning by Ignoring

Pengtao Xie [1] and Xingchen Zhao [1]

[1]Affiliation not available

October 30, 2023

Abstract

Learning by ignoring, which identifies less important things and excludes them from the learning process, is an effective learning technique in human learning. There has been psychological studies showing that learning to ignore certain things is a powerful tool for helping people focus. We are interested in investigating whether this powerful learning technique can be borrowed from humans to improve the learning abilities of machines. We propose a novel learning approach called learning by ignoring (LBI). Our approach automatically identifies pretraining data examples that have large domain shift from the target distribution by learning an ignoring variable for each example and excludes them from the pretraining process. We propose a three-level optimization framework to formulate LBI which involves three stages of learning: pretraining by minimizing the losses weighed by ignoring variables; finetuning; updating the ignoring variables by minimizing the validation loss. We develop an efficient algorithm to solve the LBI problem. Experiments on various datasets demonstrate the effectiveness of our method.

Learning by Ignoring

Xingchen Zhao XIZ168@PITT.EDU

Pengtao Xie* P1XIE@ENG.UCSD.EDU

University of California San Diego

Abstract

Learning by ignoring, which identifies less important things and excludes them from the learning process, is an effective learning technique in human learning. There has been psychological studies showing that learning to ignore certain things is a powerful tool for helping people focus. We are interested in investigating whether this powerful learning technique can be borrowed from humans to improve the learning abilities of machines. We propose a novel learning approach called learning by ignoring (LBI). Our approach automatically identifies pretraining data examples that have large domain shift from the target distribution by learning an ignoring variable for each example and excludes them from the pretraining process. We propose a three-level optimization framework to formulate LBI which involves three stages of learning: pretraining by minimizing the losses weighed by ignoring variables; finetuning; updating the ignoring variables by minimizing the validation loss. We develop an efficient algorithm to solve the LBI problem. Experiments on various datasets demonstrate the effectiveness of our method.

1. Introduction

Learning by ignoring is a broadly-used method in human learning. For example, in course learning, given a large collection of practice problems provided in the textbook, the teacher selects a subset of problems as homework for the students to practice instead of using all problems in the textbook. Some practice problems are ignored because 1) they are too difficult which might confuse the students; 2) they are too simple which are not effective in helping the students to practice their knowledge learned during lectures; 3) they are repetitive. The study in (Cunningham and Egeth, 2016) shows that learning to ignore certain things is a powerful tool for helping people focus.

Inspired by this ignoring-driven learning technique of humans, we are interested in investigating whether this methodology is helpful for improving machine learning as well. We propose a novel learning framework called learning by ignoring (LBI). In this framework, a model is trained to perform a target task. The model consists of a data encoder and a task-specific head. The encoder is trained in two phrases: pretraining and finetuning, conducted on a pretraining dataset and a finetuning dataset. Some pretraining data examples have a domain difference with the finetuning dataset, rendering them not suitable to pretrain the data encoder that will be used for performing the target task. We would like to identify such out-of-domain data examples and exclude them from the pretraining process. To achieve this goal, we associate each pretraining example with an ignoring variable $a \in [0, 1]$. If a is

. *Corresponding author.

close to 0, it means that this example should be ignored. We develop a three-level framework (involving three learning stages) to automatically learn these ignoring variables. In the first learning stage, we pretrain a data encoder V by minimizing a weighted pretraining loss: the loss defined on each pretraining example is multiplied with the ignoring variable of this example. If an example x should be ignored, its a is close to 0; multiplying a to the loss of x makes this loss close to 0, which effectively excludes x from the pretraining process. In this stage, these ignoring variables $A = \{a\}$ are fixed. They will be updated at a later stage. Note that the optimal solution $V^*(A)$ is a function of A since $V^*(A)$ is a function of the weighted loss and the weighted loss is a function of A. In the second stage, we train another data encoder W on the finetuning dataset. During the training of W, we encourage it to be close to the optimal encoder $V^*(A)$ trained in the first stage by minimizing the squared L2 distance between W and $V^*(A)$. $V^*(A)$ contains information distilled from the pretraining examples $D^{(\text{pre})}$ after ignoring. Encouraging W to be close to $V^*(A)$ effectively achieves the goal of pretraining W on $D^{(\text{pre})}$. Note that the optimal solution $W^*(V^*(A))$ is a function of $V^*(A)$. In the third stage, we apply $W^*(V^*(A))$ to make predictions on the validation set of the target task and update A by minimizing the validation loss. To summarize, we aim to learn an ignoring variable for each pretraining example so that the data encoder pretrained on non-ignored examples achieves the optimal performance on the validation set after finetuning. The three stages are performed jointly end-to-end, where different stages influence each other. Experiments on various datasets demonstrate the effectiveness of our method.

The major contributions of this paper is as follows:

- Inspired by the ignoring-driven learning technique of humans, we propose a novel machine learning approach called learning by ignoring (LBI). Our approach automatically identifies pretraining data examples that have large domain shift from the target distribution by learning an ignoring variable for each example and excludes them from the pretraining process.
- We propose a multi-level optimization framework to formulate LBI which involves three stages of learning: pretraining by minimizing the losses weighed by ignoring variables; finetuning; updating the ignoring variables by minimizing the validation loss.
- We develop an efficient algorithm to solve the LBI problem.
- Experiments on various datasets demonstrate the effectiveness of our method.

The rest of the paper is organized as follows. Section 2 and 3 present the method and experiments respectively. Section 4 reviews related works. Section 5 concludes the paper.

2. Methods

In our framework, the goal is to train a model to perform a target task T. The model consists of a data encoder W and a task-specific head H. For example, if the target task is image classification, W could be a convolutional neural network which extracts visual features of images and H could be a linear classifier which takes the visual features extracted by W

Notation	Meaning
M	Number of pretraining examples
N	Number of finetuning examples
O	Number of validation examples
$d_i^{(\text{pre})}$	The i-th pretraining data example
a_i	Ignoring variable of $d_i^{(\text{pre})}$ in pretraining
b_i	Ignoring variable of $d_i^{(\text{pre})}$ in finetuning
$d_i^{(\text{tr})}$	The i-th finetuning data example
$d_i^{(\text{val})}$	The i-th validation example
W	Encoder in finetuning
V	Encoder in pretraining
H	Head of the target task
J	Head of the pretraining task

Table 1: Notations in learning by ignoring

as input and predicts class labels. The learning is performed in two phrases: pretraining and finetuning. In the pretraining phrase, we pretrain the data encoder W on a pretraining dataset $D^{(\text{pre})} = \{d_i^{(\text{pre})}\}_{i=1}^M$ by solving a pretraining task P. P could be the same as T. In this case, W and the task-specific head H are trained jointly on $D^{(\text{pre})}$. P could be different from T as well. Under such circumstances, P has its own task-specific head J while sharing the same encoder W with T. J and W are trained jointly on $D^{(\text{pre})}$. Oftentimes, some pretraining data examples have a domain shift with the finetuning dataset. This domain shift renders these examples not suitable for pretraining the encoder. We aim to automatically identify such examples using ignoring variables and exclude them from the pretraining process. To achieve this goal, we multiply the loss of a pretraining example x with an ignoring variable $a \in [0, 1]$. If a is close to 0, it means that this example should be ignored; accordingly, the loss (after multiplied with a) is made close to 0, which effectively excludes x from the pretraining process. We aim to automatically learn the values of these ignoring variables, which will be detailed later. After pretraining, the encoder is finetuned on the training dataset $D^{(\text{tr})} = \{d_i^{(\text{tr})}\}_{i=1}^N$. For mathematical convenience, we formulate "pretraining" in the following way: in the pretraining phrase, we train another encoder V; in the finetuning phrase, we encourage the encoder W to be close to the optimally pretrained encoder V^* where the closeness is measured using squared L2 distance $\|W - V^*\|_2^2$.

Overall, the learning is organized into three stages. In the first stage, the model trains the encoder V and the head J specific to the pretraining task P on the pretraining dataset $D^{(\text{pre})} = \{d_i^{(\text{pre})}\}_{i=1}^M$, with the ignoring variables $A = \{a_i\}_{i=1}^M$ fixed:

$$V^*(A) = \min_{V,J} \sum_{i=1}^{M} a_i L(V, J, d_i^{(\text{pre})}) \tag{1}$$

After training, the optimal head is discarded. The optimal encoder $V^*(A)$ is retained for future use. The ignoring variables A are used to define the weighted training loss. But they are not updated in this stage. If A are learned by minimizing this training loss, a trivial

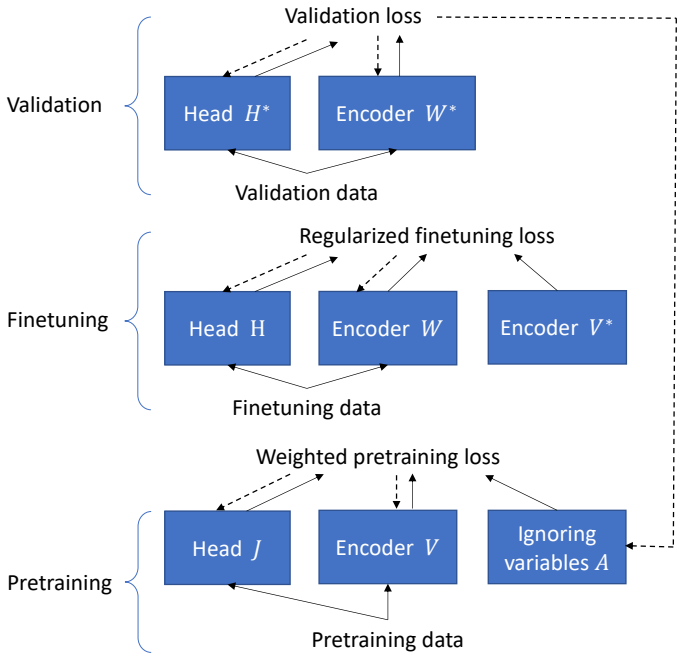

Figure 1: Illustration of learning by ignoring. The solid arrows denote the process of making predictions and calculating losses. The dotted arrows denote the process of updating learnable parameters by minimizing corresponding losses.

solution will be yielded where all ignoring variables are set to zero. Note that $V^*(A)$ is a function of A since $V^*(A)$ is a function of the weighted loss which is a function of A.

In the second stage, the model trains its data encoder W and task-specific head H by minimizing the training loss of the target task T. During training, W is encouraged to be close to $V^*(A)$ trained in the pretraining phrase by minimizing the squared L2 distance $\|W - V^*(A)\|_2^2$. This implicitly achieves the effect of pretraining W on the pretraining dataset $D^{(\text{pre})}$. The optimization problem in the second stage is:

$$W^*(V^*(A)), H^* = \min_{W,H} \ \sum_{i=1}^{N} L(W, H, d_i^{(\text{tr})}) + \lambda \|W - V^*(A)\|_2^2 \tag{2}$$

where λ is a tradeoff parameter. Note that $W^*(V^*(A))$ is a function of $V^*(A)$ since it is a function of $\|W - V^*(A)\|_2^2$, which is a function of $V^*(A)$.

In the third stage, we use the trained model consisting of $W^*(V^*(A))$ and H^* to make predictions on the validation dataset $D^{(\text{val})}$ of the target task T. We update A by minimizing the validation loss.

$$\min_A \ \sum_{i=1}^{O} L(W^*(V^*(A)), H^*, d_i^{(\text{val})}) \tag{3}$$

The three stages are mutually dependent: $V^*(A)$ learned in the first stage is used to define the objective function in the second stage; $W^*(V^*(A))$ learned in the second stage is used to define the objective function in the third stage; the updated A in the third stage

in turn changes the objective function in the first stage, which subsequently renders $V^*(A)$ and $W^*(V^*(A))$ to be changed.

Putting these pieces together, we have the following LBI framework, which is a three-level optimization problem:

$$\begin{array}{ll} \min_A & \sum_{i=1}^{O} L(W^*(V^*(A)), H^*, d_i^{(\text{val})}) \\ s.t. & W^*(V^*(A)), H^* = \min_{W,H} \ \sum_{i=1}^{N} L(W, H, d_i^{(\text{tr})}) + \lambda \|W - V^*(A)\|_2^2 \\ & V^*(A) = \min_{V,J} \ \sum_{i=1}^{M} a_i L(V, J, d_i^{(\text{pre})}) \end{array} \tag{4}$$

This formulation nests three optimization problems. On the constraints of the outer optimization problem are two inner optimization problems corresponding to the first and second learning stage respectively. The objective function of the outer optimization problem corresponds to the third learning stage.

If the pretaining task and target task are the same, in the second stage we can use the pretraining data to train W as well. Due to domain difference, not all pretraining examples are suitable for training W. To exclude such examples, we associate each pretraining example $d_i^{(\text{pre})}$ with an ignoring variable $b_i \in [0, 1]$. Note that b_i is different from a_i. b_i is used to determine whether $d_i^{(\text{pre})}$ should be ignored during training W and a_i is used to determine whether $d_i^{(\text{pre})}$ should be ignored during training V. The corresponding formulation is:

$$\begin{array}{ll} \min\limits_{A,B} & \sum_{i=1}^{O} L(W^*(V^*(A), B), H^*(B), d_i^{(\text{val})}) \\ s.t. & W^*(V^*(A), B), H^*(B) = \min\limits_{W,H} \ \sum\limits_{i=1}^{N} L(W, H, d_i^{(\text{tr})}) + \lambda \|W - V^*(A)\|_2^2 + \gamma \sum\limits_{i=1}^{M} b_i L(W, H, d_i^{(\text{pre})}) \\ & V^*(A) = \min\limits_{V,J} \ \sum_{i=1}^{M} a_i L(V, J, d_i^{(\text{pre})}) \end{array} \tag{5}$$

where γ is a tradeoff parameter and $B = \{b_i\}_{i=1}^{M}$. Note that W^* and H^* are both functions of B.

2.1. Optimization Algorithm

In this section, we derive an optimization algorithm to solve the LBI problem defined in Eq.(4). Inspired by (Liu et al., 2018), we approximate $V^*(A)$ using one-step gradient descent update of V with respect to $\sum_{i=1}^{M} a_i L(V, J, d_i^{(\text{pre})})$. We plug the approximation V' of $V^*(A)$ into $\sum_{i=1}^{N} L(W, H, d_i^{(\text{tr})}) + \lambda \|W - V^*(A)\|_2^2$ and obtain an approximated objective O_W. Then we approximate $W^*(V^*(A))$ using one-step gradient descent update of W with respect to O_W. Finally, we plug the approximation W' of $W^*(V^*(A))$ into $\sum_{i=1}^{O} L(W^*(V^*(A)), H^*, d_i^{(\text{val})})$ and get an approximated objective O_A. We perform a gradient-descent update of A with respect to O_A. In the sequel, we use $\nabla^2_{Y,X} f(X, Y)$ to denote $\frac{\partial f(X,Y)}{\partial X \partial Y}$.

First of all, we approximate $V^*(A)$ using:

$$V' = V - \xi_V \nabla_V \sum_{i=1}^{M} a_i L(V, J, d_i^{(\text{pre})}) \tag{6}$$

where ξ_V is a learning rate. We update J as:

$$J \leftarrow J - \xi_J \sum_{i=1}^{M} a_i \nabla_J L(V, J, d_i^{(\text{pre})}). \tag{7}$$

Plugging V' into $\sum_{i=1}^{N} L(W, H, d_i^{(\text{tr})}) + \lambda \|W - V^*(A)\|_2^2$, we obtain an approximated objective $\sum_{i=1}^{N} L(W, H, d_i^{(\text{tr})}) + \lambda \|W - V'\|_2^2$. Then we approximate $W^*(V^*(A))$ using one-step gradient descent update of W with respect to O_W:

$$\begin{aligned} W' &= W - \xi_W \nabla_W (\textstyle\sum_{i=1}^{N} L(W, H, d_i^{(\text{tr})}) + \lambda \|W - V'\|_2^2) \\ &= W - \xi_W (\textstyle\sum_{i=1}^{N} \nabla_W L(W, H, d_i^{(\text{tr})}) + 2\lambda (W - V')) \end{aligned} \tag{8}$$

Similarly, we approximate H^* with:

$$\begin{aligned} H' &= H - \xi_H \nabla_H (\textstyle\sum_{i=1}^{N} L(W, H, d_i^{(\text{tr})}) + \lambda \|W - V'\|_2^2) \\ &= H - \xi_H \textstyle\sum_{i=1}^{N} \nabla_H L(W, H, d_i^{(\text{tr})}) \end{aligned} \tag{9}$$

Finally, we plug W' and H' into $\sum_{i=1}^{O} L(W^*(V^*(A)), H^*, d_i^{(\text{val})})$ and get $O_A = \sum_{i=1}^{O} L(W', H', d_i^{(\text{val})})$. We update A by descending the gradient of O_A w.r.t A:

$$\begin{aligned} A &\leftarrow A - \xi_A \nabla_A \textstyle\sum_{i=1}^{O} L(W', H', d_i^{(\text{val})}) \\ &= A - \xi_A \textstyle\sum_{i=1}^{O} \frac{\partial V'}{\partial A} \frac{\partial W'}{\partial V'} \nabla_{W'} L(W', H', d_i^{(\text{val})}) \end{aligned} \tag{10}$$

where

$$\frac{\partial V'}{\partial A} = \frac{\partial (V - \xi_V \nabla_V \sum_{i=1}^{M} a_i L(V, J, d_i^{(\text{pre})}))}{\partial A} = -\xi_V \nabla_{A,V}^2 \sum_{i=1}^{M} a_i L(V, J, d_i^{(\text{pre})}) \tag{11}$$

and

$$\frac{\partial W'}{\partial V'} = \frac{\partial (W - \xi_W (\sum_{i=1}^{N} \nabla_W L(W, H, d_i^{(\text{tr})}) + 2\lambda (W - V')))}{\partial V'} = 2\xi_W \lambda I \tag{12}$$

where I is an identity matrix.

For the formulation in Eq.(5), the optimization algorithm is similar. $V^*(A)$ is approximated using Eq.(6). We approximate $W^*(V^*(A), B)$ using

$$W' = W - \xi_W \nabla_W (\sum_{i=1}^{N} L(W, H, d_i^{(\text{tr})}) + \lambda \|W - V^*(A)\|_2^2 + \gamma \sum_{i=1}^{M} b_i L(W, H, d_i^{(\text{pre})})) \tag{13}$$

and approximate $H^*(B)$ using

$$H' = H - \xi_H \nabla_H (\sum_{i=1}^{N} L(W, H, d_i^{(\text{tr})}) + \lambda \|W - V^*(A)\|_2^2 + \gamma \sum_{i=1}^{M} b_i L(W, H, d_i^{(\text{pre})})) \tag{14}$$

Algorithm 1 Optimization algorithm for learning by ignoring

while *not converged* **do**

1. Update encoder V in the pretraining phrase using Eq.(6)
2. Update task-specific head J in the pretraining phrase using Eq.(7)
3. Update encoder W in the finetuning phrase using Eq.(8)
4. Update task-specific head H in the finetuning phrase using Eq.(9)
5. Update ignoring variables A using Eq.(10)

end

The update of A is the same as that in Eq.(10). The update of B is as follows:

$$\begin{aligned} B &\leftarrow B - \xi_B \nabla_B \textstyle\sum_{i=1}^{O} L(W', H', d_i^{(\text{val})}) \\ &= B - \xi_B \textstyle\sum_{i=1}^{O} (\frac{\partial W'}{\partial B} \nabla_{W'} L(W', H', d_i^{(\text{val})}) + \frac{\partial H'}{\partial B} \nabla_{H'} L(W', H', d_i^{(\text{val})})) \end{aligned} \tag{15}$$

where

$$\frac{\partial W'}{\partial B} = -\xi_W \gamma \nabla_{B,W}^2 \sum_{i=1}^{M} b_i L(W, H, d_i^{(\text{pre})}) \tag{16}$$

and

$$\frac{\partial H'}{\partial B} = -\xi_H \gamma \nabla_{B,H}^2 \sum_{i=1}^{M} b_i L(W, H, d_i^{(\text{pre})}) \tag{17}$$

3. Experiments

3.1. Datasets

We perform experiments on Office-Home (Venkateswara et al., 2017) and Office-31 (Saenko et al., 2010). Office-Home consists of 15,500 images of daily objects from 65 categories and 4 domains, including Art (Ar), Clipart (Cl), Product (Pr), and Real-World (Rw). Each category has an average of about 70 images and a maximum of 99 images. Office-31 contains 4,110 images belonging to 31 classes and 3 domains, including Amazon website (A), web camera (W), and digital SLR camera (D). The number of images in domain A, W, and D is 2817, 795, and 498 respectively.

3.2. Baselines

We compare with the following baselines.

- **Baseline 1**. In this baseline, we ignore the pretraining dataset and directly train a model on the finetuning dataset.
- **Baseline 2**. In this baseline, we combine the pretraining dataset and the finetuning dataset as a single dataset, then train a model on the combined dataset.

	Baseline 1	Baseline 2	Baseline 3	Ours ($\lambda = 0, \gamma = 1$)	Ours ($\lambda = 1, \gamma = 0$)
Ar→Cl	65.28	66.09	**66.90**	68.15	66.67
Ar→Pr	85.70	84.98	84.62	85.94	**86.06**
Ar→Rw	71.76	70.60	73.15	72.00	**74.19**
Cl→Ar	57.08	58.75	61.46	60.49	**63.75**
Cl→Pr	85.22	84.86	85.22	85.70	**86.78**
Cl→Rw	70.60	71.30	73.50	70.43	**73.61**
Pr→Ar	59.38	58.54	59.79	61.38	**62.08**
Pr→Cl	**69.91**	66.55	67.36	69.11	68.87
Pr→Rw	71.18	70.49	72.69	72.60	**73.73**
Rw→Ar	59.79	60.83	**66.46**	64.06	65.63
Rw→Cl	**69.44**	65.05	69.21	70.31	68.75
Rw→Pr	86.78	84.62	**87.74**	87.38	86.66
Average	71.01	70.22	72.34	72.30	**73.07**

Table 2: Accuracy (%) on the Office-Home dataset. In the $A \rightarrow B$ notion, A denotes the pretraining data and B denotes the finetuning data.

- **Baseline 3**. In this baseline, we first train a model M_1 on the pretraining dataset. Then we train a model M_2 on the finetuning dataset. When training M_2, we encourage its weights to be close to the optimally trained weights of M_1 by minimizing their squared L2 distance.

3.3. Results

Table 2 shows the results on the Office-Home dataset. In the $A \rightarrow B$ notion, A denotes the pretraining dataset and B denotes the finetuning dataset. From this table, we make the following observations. **First**, our method (where λ is set to 1 and γ is set to 0) performs better than the three baselines in 7 out of the 12 $A \rightarrow B$ cases and achieves a better averaged accuracy than the baselines. This demonstrates the effectiveness of our method. Our method learns to ignore out-of-domain pretraining data examples, which avoids the data encoder to be distorted by such examples. Baseline 1 simply ignores all pretraining examples, including the useful ones, which is a waste of data. Baseline 3 uses all pretraining examples, including the out-of-domain ones, where the encoder may be biased by these out-of-domain examples. Baseline 2 directly uses the pretraining data for finetuning, which yields a low-quality model due to the domain discrepancy between pretraining data and finetuning data. **Second**, when $\lambda = 0$ and $\gamma = 1$, our method performs less well. This is because under this setting, the selected pretraining data is used for finetuning instead of pretraining. Due to the large domain shift, pretraining data is more suitable for pretraining instead of finetuning. However, Ours ($\lambda = 0$,$\gamma = 1$) outperforms Baseline 2 which does not perform ignoring. This further demonstrates the effectiveness of data ignoring.

Table 3 shows the results on the Office-31 dataset. From this table, we make the following observations. **First**, Ours ($\lambda = 0, \gamma = 1$) which uses pretraining data for finetuning

	Baseline 1	Baseline 2	Baseline 3	Ours ($\lambda = 1, \gamma = 0$)	Ours ($\lambda = 0, \gamma = 1$)
A→W	99.38	96.88	97.5	97.5	**100**
A→D	95.83	96.88	97.92	95.83	**98.44**
D→W	98.12	96.88	98.12	98.12	**99.22**
D→A	**86.52**	84.38	83.79	85.55	85.94
W→D	98.96	96.88	98.96	97.92	**100**
W→A	**85.55**	85.16	85.16	85.35	85.35
Average	94.06	92.84	93.58	93.38	**94.83**

Table 3: Accuracy (%) on the Office-31 dataset. In the $A \rightarrow B$ notion, A denotes the pretraining data and B denotes the finetuning data.

outperforms Ours ($\lambda = 0, \gamma = 1$) which uses pretraining data for pretraining. This is because in the Office-31 dataset, the domain difference between pretraining and finetuning data is smaller than that in the Office-Home dataset. When the domain difference is small, using the pretraining data directly for finetuning is an more efficient way for data utilization. **Second**, Ours ($\lambda = 0, \gamma = 1$) outperforms Baseline 2 which does not perform data ignoring. This further demonstrates the effectiveness of data ignoring.

4. Related Works

Several approaches have been proposed for data selection. Matrix column subset selection (Deshpande and Rademacher, 2010; Boutsidis et al., 2009) aims to select a subset of data examples that can best reconstruct the entire dataset. Similarly, coreset selection (Bachem et al.) chooses representative training examples in a way that models trained on the selected examples have comparable performance with those trained on all training examples. These methods perform data selection and model training separately. As a result, the validation performance of the model cannot be used to guide data selection. Ren et al. (2018) propose a meta learning method to learn the weights of training examples by performing a meta gradient descent step on the weights of the current mini-batch of examples. Shu et al. (2019) propose a method which can adaptively learn an explicit weighting function directly from data. These works focus on selecting training examples using a meta-learning framework (or bi-level optimization framework) while our work focuses on selecting pretraining examples using a three-level optimization framework.

5. Conclusions

In this paper, we propose a new machine learning approach – learning by ignoring (LBI), inspired by the ignoring-driven learning technique of human. In LBI, an ignoring variable is learned for each pretraining data example to identify examples that have significant domain difference with the target distribution. We propose a multi-level optimization framework to formalize LBI which involves three learning stages: an encoder is trained by minimizing a weighted pretraining loss where the loss of each data example is weighted by its ignoring

variable; another encoder is finetuned where during the finetuning this encoder is encouraged to be close to the previously pretrained encoder; the ignoring variables are updated by minimizing the validation loss calculated using the finetuned encoder. We conduct experiments on various datasets where the results demonstrate the effectiveness of our method.

References

Olivier Bachem, Mario Lucic, and Andreas Krause. Practical coreset constructions for machine learning.

Christos Boutsidis, Michael W Mahoney, and Petros Drineas. An improved approximation algorithm for the column subset selection problem. In *Proceedings of the twentieth annual ACM-SIAM symposium on Discrete algorithms*, pages 968–977. SIAM, 2009.

Corbin A Cunningham and Howard E Egeth. Taming the white bear: Initial costs and eventual benefits of distractor inhibition. *Psychological science*, 27(4):476–485, 2016.

Amit Deshpande and Luis Rademacher. Efficient volume sampling for row/column subset selection. In *2010 ieee 51st annual symposium on foundations of computer science*, pages 329–338. IEEE, 2010.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018.

Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. Learning to reweight examples for robust deep learning. *arXiv preprint arXiv:1803.09050*, 2018.

Kate Saenko, Brian Kulis, Mario Fritz, and Trevor Darrell. Adapting visual category models to new domains. In *European conference on computer vision*, pages 213–226. Springer, 2010.

Jun Shu, Qi Xie, Lixuan Yi, Qian Zhao, Sanping Zhou, Zongben Xu, and Deyu Meng. Meta-weight-net: Learning an explicit mapping for sample weighting. In *Advances in Neural Information Processing Systems*, pages 1919–1930, 2019.

Hemanth Venkateswara, Jose Eusebio, Shayok Chakraborty, and Sethuraman Panchanathan. Deep hashing network for unsupervised domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5018–5027, 2017.