Learn2Weight: Parameter Adaptation against Similar-domain Adversarial Attacks

Siddhartha Datta
University of Oxford
siddhartha.datta@cs.ox.ac.uk

Abstract

Recent work in black-box adversarial attacks for NLP systems has attracted much attention. Prior black-box attacks assume that attackers can observe output labels from target models based on selected inputs. In this work, inspired by adversarial transferability, we propose a new type of black-box NLP adversarial attack that an attacker can choose a similar domain and transfer the adversarial examples to the target domain and cause poor performance in target model. Based on domain adaptation theory, we then propose a defensive strategy, called Learn2Weight, which trains to predict the weight adjustments for a target model in order to defend against an attack of similar-domain adversarial examples. Using Amazon multi-domain sentiment classification datasets, we empirically show that Learn2Weight is effective against the attack compared to standard black-box defense methods such as adversarial training and defensive distillation. This work contributes to the growing literature on machine learning safety.

1 Introduction

As machine learning models are applied to more and more real-world tasks, addressing machine learning safety is becoming an increasingly pressing issue. Deep learning algorithms have been shown to be vulnerable to adversarial examples (Szegedy et al., 2013; Goodfellow et al., 2014; Pernot et al., 2016a). In particular, prior black-box adversarial attacks assume that the adversary is not aware of the target model architecture, parameters or training data, but is capable of querying the target model with supplied inputs and obtaining the output predictions. The phenomenon that adversarial examples generated from one model may also be adversarial to another model is known as adversarial transferability (Szegedy et al., 2013).

Motivated by adversarial transferability, we conjecture another black-box attack pipeline where the adversary does not even need to have access to the target model nor query labels from crafted inputs. Instead, as long as the adversary knows the task of the target, they can choose a similar domain to build a substitute model, and then attack the target model with adversarial examples that are generated from the attack domain.

The similar-domain adversarial attack may be more practical than prior blackbox attacks as label querying from the target model is not needed. This attack can be illustrated with the following example (Figure 1b) in medical insurance fraud (Finlayson et al., 2019). Insurance companies may use hypothetical opioid risk models to classify the likelihood (high/low) of a patient to abuse the opioids to be prescribed, based on the patient's medical history as text input. Physicians can run the original patient history through the attack pipeline to generate an adversarial patient history, where the original is more likely to be rejected ("High" risk) and the adversarial is more likely to be accepted ("Low" risk). Perturbations in patient history could be, for example, a slight perturbation from "alcohol abuse" to "alcohol dependence", and it may successfully fool the insurance company's model.

Based on domain adaptation theory (Ben-David et al., 2010), we conjecture that domain-variant features cause the success of the similar-domain attack. The adversarial examples with domain-variant features are likely to reside in the low-density regions (far away from decision boundary) of the empirical distribution of the target training data which could fool the target model (Zhang et al., 2019b). Literature indicates that worsened generalizability is a tradeoff faced by existing defenses such as adversarial training (Raghunathan et al., 2019) and domain generalization techniques (Wang et al., 2019). In trying to increase robustness against adversarial inputs, a model faces a tradeoff of weakened accuracy towards clean inputs. Given that an adversarial training loss function is composed of a loss against
To curb this issue, methods have been proposed to improve robustness without losing the target model’s standard performance. Our approach is able to improve robustness against the similar-domain adversarial attack over other baseline defense methods. Moreover, our approach is able to improve robustness against the similar-domain adversarial attack over existing defenses against the similar-domain adversarial attack.

Our contribution can be summarized as follows:

- We are among the first to demonstrate the similar-domain adversarial attack, leveraging domain adaptation to create adversarial perturbations that compromise NLP models. This attack pipeline relaxes the previous black-box attack assumption that the adversary has access to the target model and can query the model with crafted examples.
- We propose a defensive strategy for this attack based on domain adaptation theory and meta learning. Experiments show the effectiveness of our approach over existing defenses against the similar-domain adversarial attack.

We propose a defensive strategy for this attack, leveraging domain adaptation to create adversarial perturbations that compromise NLP models. This attack pipeline relaxes the previous black-box attack assumption that the adversary has access to the target model and can query the model with crafted examples.

To defend against this similar-domain adversarial attack, we propose a meta learning approach, Learn2Weight, so that the target model’s decision boundary can adapt to the examples from low-density regions. Experiments confirm the effectiveness of our approach against the similar-domain attack over other baseline defense methods. Moreover, our approach is able to improve robustness accuracy without losing the target model’s standard generalization accuracy.

Our contribution can be summarized as follows:

- We are among the first to demonstrate the similar-domain adversarial attack, leveraging domain adaptation to create adversarial perturbations that compromise NLP models. This attack pipeline relaxes the previous black-box attack assumption that the adversary has access to the target model and can query the model with crafted examples.
- We propose a defensive strategy for this attack based on domain adaptation theory and meta learning. Experiments show the effectiveness of our approach over existing defenses against the similar-domain adversarial attack.

Our contribution can be summarized as follows:

- We are among the first to demonstrate the similar-domain adversarial attack, leveraging domain adaptation to create adversarial perturbations that compromise NLP models. This attack pipeline relaxes the previous black-box attack assumption that the adversary has access to the target model and can query the model with crafted examples.
- We propose a defensive strategy for this attack based on domain adaptation theory and meta learning. Experiments show the effectiveness of our approach over existing defenses against the similar-domain adversarial attack.

2 Related Work

Zhang et al. (2020) provides a survey of adversarial attacks in NLP. Existing research proposes different attack methods for generating adversarial text examples (Moosavi-Dezfooli et al., 2016; Ebrahimi et al., 2018; Wallace et al., 2019). The crafted adversarial text examples have been shown to fool state-of-the-art NLP systems, e.g. BERT (Jin et al., 2019). A large body of adversarial attack research focuses on black-box attack where the adversary builds a substitute model by querying the target model with supplied inputs and obtaining the output predictions. The key idea behind such black-box attack is that adversarial examples generated from one model may also be misclassified by another model, which is known as adversarial transferability (Szegedy et al., 2013; Cheng et al., 2019). While prior work examines the transferability between different models trained over the same dataset, or the transferability between the same or different models trained over disjoint subsets of a dataset, our work examines the adversarial transferability between different domains, which we call a similar-domain adversarial attack.

3 Similar-domain Adversarial Attack

3.1 Adversarial attack background

Adversarial attacks modify inputs to cause errors in machine learning inference (Szegedy et al., 2013). We use the basic gradient-based attack method Fast Gradient Sign Method (FGSM) (Goodfellow et al., 2014), with perturbation rate $\varepsilon = 0.4$. Other NLP adversarial generation algorithms could also be used, such as Rand-FGSM (Tramèr et al., 2017), Basic Iterative Method (Kurakin et al., 2016c,a; Xie et al., 2018), DeepFool (Moosavi-Dezfooli et al., 2016), HotFlip (Ebrahimi et al., 2018), uni-
We present the architecture of similar-domain adversarial attacks. The defender, the attacker, and the target model parameters, collect attack data from a similar domain, \( X_j \sim X \), and train an attack model (parameters \( \theta_j \) ) \( \circledast \). They run the attack model on the test data \( X_j \) to obtain correctly-classified instances \( \circledast \). They choose an adversarial attack algorithm and generate a set of adversarial samples \( X_j^{a d v} \) \( \circledast \). They expose \( X_j^{a d v} \) to the target model, hoping \( X_j^{a d v} \) misleads the target model to produce an output of their choice \( \circledast \). The attacker’s objective is to maximize the misclassification per label and minimize the accuracy w.r.t. perturbed inputs \( (\max \text{ Eqt } 1) \), while the defender’s objective is to maximize the accuracy w.r.t. perturbed inputs \( (\min \text{ Eqt } 1) \). This type of attack works best as an adversarial attack that compromises systems that base decision-making on one-instance.

\[
\mathbb{E}_{x_j, y_j \sim X_j, Y_j} [f(\theta_j; X_j^{a d v}) - y_j]
\]

Definition 2. Similar-domain Adversarial Attack. Target model \( f \), trained on target domain data \( X_i \), is a deep neural network model with weights \( \theta_i \) mapping text instances to labels: \( Y_i = f(\theta_i; X_i) \). An adversary chooses source attack domain \( X_j \), builds substitute model \( f(\theta_j; X_j) \), and generates a set of adversarial examples \( X_j^{a d v} \) from \( X_j \) using \( \text{Adv}(\theta_j; X_j) \), such that during an attack \( f(\theta_j; X_j^{a d v}) = f(\theta_j; X_j^{a d v}) \).

4 Is the Attack Effective?

4.1 Setup

(Datasets) We sample domains from 25 domain datasets, each containing 1,000 positive and 1,000 negative reviews for an Amazon product category, sourced from the Amazon multi-domain sentiment classification benchmark (Blitzer et al., 2007).

(Models) We evaluated our setup on several architectures commonly-used for sentiment classification, including LSTM (Wang et al., 2018), GRU, BERT (Devlin et al., 2019), CNN (Kim, 2014), and Logistic Regression (Maas et al., 2011).
The greater the gap between the original and after-attack samples, denoted as the after-attack accuracy \( \hat{\text{a}} \), the more successful the attack. We first report the accuracy of the target model against the complete, unperturbed test set of the attack domain, to demonstrate that any drop in classification accuracy is not from domain shift alone but from adversarial transferability.

**Table 2: Domain shift & similarity**

<table>
<thead>
<tr>
<th>Attack Domain</th>
<th>book</th>
<th>magazine</th>
<th>baby</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unperturbed Accuracy</td>
<td>0.745</td>
<td>0.726</td>
<td>0.646</td>
</tr>
<tr>
<td>After-attack Accuracy</td>
<td>0.395</td>
<td>0.398</td>
<td>0.421</td>
</tr>
<tr>
<td>SharedVocab</td>
<td>0.455</td>
<td>0.381</td>
<td>0.255</td>
</tr>
<tr>
<td>Transfer Loss</td>
<td>0.000</td>
<td>0.017</td>
<td>0.071</td>
</tr>
</tbody>
</table>

Sorted in descending order of domain similarity, we observe a lower after-attack accuracy when domain similarity increases.

(Domain similarity) refers to the similarity between attacker’s chosen domain and defender’s domain. SharedVocab measures the overlap of unique words, in each of the datasets; a higher degree of overlapping vocabulary implies the two domains are more similar. We also use Transfer Loss, a standard metric for domain adaptation (Blitzer et al., 2007; Glorot et al., 2011), to measure domain similarity; lower loss indicates higher similarity. The test error from a target model trained on target domain \( X_i \) and evaluated on attack domain \( X_j \) returns transfer error \( e(X_i, X_j) \). The baseline error \( e(X_i, X_i) \) term is the test error obtained from target model trained on target domain (train) data \( X_i \) and tested on target domain (evaluation) data \( X_i \). This computes the transfer loss, \( tf(X_i, X_i) = e(X_i, X_i) - e(X_i, X_i) \).

(Accuracy) We first report the accuracy of the target models on the target domain test samples before the attack as the original accuracy. Then we measure the accuracy of the target models against adversarial samples crafted from the attack domain samples, denoted as the after-attack accuracy. Intra-attack accuracy denotes the after-attack accuracy where the attack domain is identical to the target domain. By comparing original and after-attack accuracy, we can evaluate the success of the attack. The greater the gap between the original and after-attack accuracy, the more successful the attack. Unperturbed accuracy measures the accuracy of the target model against the complete, unperturbed test set of the attack domain, to demonstrate that any drop in classification accuracy is not from domain shift alone but from adversarial transferability.

**4.2 Results**

The similar-domain adversarial attack results are presented in Table 2. We see a significant gap between original accuracy and after-attack accuracy, indicating that this attack can impose a valid threat to a target NLP system. After the similar-domain adversarial attack, the accuracy drops dramatically by a large margin. Take the book target domain as an example: when the attack domain is magazine, the after-attack accuracy drops to 0.398, and when the attack domain is baby, the accuracy is 0.421. Moreover, we observe a positive correlation between transfer loss and after-attack accuracy, and a negative correlation between shared vocab and after-attack accuracy.

5 Defending Against Similar-domain Adversarial Attack

In order to defend against a similarity based adversarial attack, it is critical to block adversarial transferability. Adversarial training is the most intuitive yet effective defense strategy for adversarial attack (Goodfellow et al., 2014; Madry et al., 2017). However, this may not be effective for two reasons. First, there is no formal guidance for generating similar-domain adversarial examples because the defender has no idea what the attack data domain is. Second, simply feeding the target model with adversarial examples may even hurt the generalization of the target model (Su et al., 2018; Raghunathan et al., 2019; Zhang et al., 2019a), which is also confirmed in our experiments.

**5.1 Parameter Adaptation**

Meta learning techniques that modify parameters (Ha et al., 2016; Hu et al., 2018; Kuen et al., 2019) are concerned with adapting weights from one model into another, and generating/predicting the complete set of weights for a model given the input samples. In our context, distinctly different
weights are produced for target models trained on inputs of different domains, and feature transferability (Yosinski et al., 2014) in the input space can be expected to translate to weights transferability in the parameter space. Rather than completely regenerating classification weights, our model robustification defense, Learn2Weight, predicts the perturbation to existing weights \( \theta^* = \theta_i + \Delta \theta \) for each new instance.

5.2 Learn2Weight (L2W)

We conjecture that an effective defense strategy is to perturb the target model weights depending on the feature distribution of the input instance. In inference (Algorithm 1), L2W recalculates the target model weights depending on the input. During training (Algorithm 2), L2W trains on sentences from different domains and a weight differential for that domain (the weight adjustment required to tune the target model’s weights to adapt to the input’s domain). We obtain the weight differential \( \Delta \theta \) by finding the difference between the weights \( \theta_j \) trained on sentence:label pairs from a specific domain \( X_j \sim \mathcal{X} \) and weights \( \theta_i \) trained on sentence:label pairs from the target domain \( X_i \). Other training models may be possible; here we trained a sequence-to-sequence network (Sutskever et al., 2014) on sentence: \( \Delta \theta \) pairs.

5.3 Perturbation Sets Generation

To generate synthetic domains of varying domain similarity \( \mathcal{S} = \{X_j : Y_j\}_{j=1}^T \) so that defenders defend their model using only target domain data \( X_i \), a defender iteratively generates perturbation sets that minimizes transfer loss while maximizing adversarial perturbations (Algorithm 3). A perturbation set is a set containing subsets of perturbed inputs (Alzantot et al., 2018; Wong et al., 2019). To construct one perturbation set (Eq 2), we utilize an iterative minimax algorithm, where we iteratively apply a maximizing adversarial perturbation factor \( \varepsilon \geq \varepsilon_{\text{min}} \), and accept the batch of perturbed inputs if it yields a minimizing input distance \( \text{dist} \leq d_{\text{max}} \). We repeat this \( T \) times. We use transfer loss as the distance metric to optimize for domain similarity. We retain FGSM as the adversarial attack algorithm.

\[
\begin{align*}
X \leftarrow \min \text{dist}(X^*, X_i) & \leq d_{\text{max}} \\
X^* & \leftarrow \min_{\varepsilon \leq \varepsilon_{\text{max}}} \max_{t} \text{dist}(\text{Adv}(\theta_t; X_i; \varepsilon), X_i) \\
X^* & \leftarrow \min_{\varepsilon \leq \varepsilon_{\text{max}}} \max_{t} [e(\text{Adv}(\theta_t; X_i; \varepsilon), X_i) - e(X_i, X_i)]
\end{align*}
\]

Algorithm 1: Learn2Weight (Inference)

\[
\begin{align*}
\text{inference} & (X_j^{ab}, h(\theta^m), f(\theta_t)) \\
\text{Input} & : \text{test-time inputs } X_j^{ab}; \text{L2W } h(\theta^m); \text{base learner } f(\theta_t) \\
\text{Output} & : \text{label } \hat{y} \\
\text{Compute parameter differential w.r.t. } & X_j^{ab} \\
\Delta \theta & \leftarrow h(\theta^m, X_j^{ab}) \\
\text{Update } & \theta^f \text{.} \\
\hat{y} & \leftarrow f(\theta_t + \Delta \theta; X_j^{ab}) \\
\text{return } & \hat{y}
\end{align*}
\]

Algorithm 2: Learn2Weight (Training)

\[
\begin{align*}
\text{train} & (S, D, \theta, E^f, E^m) \\
\text{Input} & : \text{domains (perturbation sets) } S, \text{target domain } D = \{X_i : Y_i\}, \text{base learner parameters } \theta, \text{epochs } E^f \text{ & } E^m \\
\text{Output} & : \text{L2W parameters } \theta^m \\
\text{Initialize empty set } & \Theta \text{ to store parameter differential.} \\
\Theta & \leftarrow \emptyset; \\
\text{Compute } & X_j \rightarrow \Delta \theta. \\
\text{foreach } & X_j : Y_j \in (D \cup S) \text{ do} \\
\text{for } c & \leftarrow 0 \text{ to } E^f \text{ do} \\
\theta^f_{j,c} & \leftarrow \theta^f_{j,c-1} - \sum_{X_{i,j}} X_{i,j} \frac{\partial L(X_{i,j})}{\partial \theta^f} \\
\Delta \theta & \leftarrow \theta^f_{j,c} - \theta^i; \\
\Theta & \leftarrow \Delta \theta; \\
\text{Compute } & \theta^m \text{.} \\
\text{for } c & \leftarrow 0 \text{ to } E^m \text{ do} \\
\theta^m & \leftarrow \theta^m_{c-1} - \sum_{X_{i,j}} X_{i,j} \Delta \theta \frac{\partial L(X_{i,j}, \Delta \theta)}{\partial \theta^m} \\
\text{return } & \theta^m
\end{align*}
\]

Algorithm 3: Perturbation Sets Generation

\[
\begin{align*}
\text{PerturbationSet} & (D, \theta; T; R; \text{dist}, d_{\text{max}}; \varepsilon, \gamma) \\
\text{Input} & : \text{target domain } D = \{X_i : Y_i\}, \text{parameters } \theta; \text{number of perturbation sets } T = 10, \text{max iterations } R = 10; \text{distance metric } \text{dist} = t f(X_i, X_j), \text{max distance } d_{\text{max}} = 0.1; \text{initial perturbation rate } \varepsilon = 0.9, \text{perturbation learning rate } \gamma = 0.05; \\
\text{Output} & : \text{set } S \text{ containing } T \text{ perturbation sets} \\
\text{Initialize empty } & S \text{ to store perturbation sets } S_t, \\
S & \leftarrow \emptyset; \\
\text{while } & t < T \text{ do} \\
\text{Run next iteration } & r \text{ until } S_t \text{ meets conditions.} \\
\text{for } r & \leftarrow 0 \text{ to } R \text{ do} \\
\text{Apply adversarial perturbations to } X. \\
S_{t,r} & \leftarrow \text{Adv}(\theta_t; X_i; \varepsilon); \\
\text{Evaluate distance conditions,} \\
\text{if } & \text{dist}(S_{t,r}, X_i) \leq d_{\text{max}} \text{ then} \\
\text{if } & \sigma^2(S_t \cup S_{t,r}) > \sigma^2(S) \text{ then} \\
S & \leftarrow \{S_{t,r}, Y_i\}; \\
\text{continue;} \\
\text{else} \\
\text{Adjust hyperparameters.} \\
\varepsilon & \leftarrow \varepsilon - \gamma; \\
t & \leftarrow t + 1; \\
\text{return } & S
\end{align*}
\]
5.4 Explanation: Blocking Transferability

To facilitate our explanation, we adapt from domain adaptation literature (Ben-David et al., 2010; Liu et al., 2019; Zhang et al., 2019c):

\[ e(X_j^{adv}, X_i) \leq e(X_i, X_i) + d_{H\Delta H}(X_j^{adv}, X_i) + \lambda \quad (3) \]

where \( H \) is the hypothesis space, \( h \) is a hypothesis function that returns labels \{0, 1\}, and \( e(X_i, X_i) \) and \( e(X_j^{adv}, X_i) \) are the generalization errors from passing target domain data \( X_i \) and adversarial data \( X_j^{adv} \) through a classifier trained on \( X_i \). \( d_{H\Delta H}(X_j^{adv}, X_i) \) is the \( H\Delta H \)-distance between \( X_i \) and \( X_j^{adv} \), and measures the divergence between the feature distributions of \( X_j^{adv} \) and \( X_i \).

\[ e_{X_j^{adv}}(h, h') \] and \( e_{X_i}(h, h') \) represent the probability that \( h \) disagrees with \( h' \) on the label of an input in the domain space \( X_j^{adv} \) and \( X_i \), respectively.

\[
d_{H\Delta H}(X_j^{adv}, X_i) = \sup_{h, h' \in H} |e_{X_j^{adv}}(h, h') - e_{X_i}(h, h')| \\
d_{H\Delta H}(X_j^{adv}, X_i) = \sup_{h, h' \in H} |E_{x_j \sim X_j}[(h(x_j) - h'(x_j))]| \\
- |E_{x_i \sim X_i}[(h(x_i) - h'(x_i))]| \\
\]

Divergence \( d_{H\Delta H} \) measures the divergence between feature distributions \( X_j^{adv} \) and \( X_i \). Higher \( d_{H\Delta H} \) indicates less shared features between 2 domains. The greater the intersection between feature distributions, the greater the proportion of domain-invariant features; one approach to domain adaptation is learning domain-invariant features representations (Zhao et al., 2019) to minimize \( d_{H\Delta H} \).

Explaining similarity-domain attacks. As demonstrated by empirical results, \( e(X_j^{adv}, X_i) \) increases in a similarity-based attack setting, and this would arise if \( d_{H\Delta H} \) increases correspondingly. \( d_{H\Delta H} \) computes inconsistent labels from inconsistent feature distributions, and attributes the success of the attack to domain-variant features.

FGSM and variants adjust the input data to maximize the loss based on the backpropagated gradients of a model trained on \( X_j \). As our pipeline used correctly-labelled sentences before adversarially perturbing them, we can infer that perturbations applied to \( X_j \) were not class-dependent (i.e. the success of the attack is not based on the removal of class-specific features), but class-independent features. It is already difficult for a model trained on \( X_j \) to classify when there is insufficient class-dependent features (hence a high \( tf(X_j^{adv}, X_i) \)); in a cross-domain setting, it must be even more difficult for a model trained on \( X_i \) to classify given a shortage of domain-invariant, class-dependent features.

\[
d_{H\Delta H} \geq e(X_j^{adv}, X_i) - e(X_i, X_i) - \lambda \\
d_{H\Delta H} \geq tf(X_j^{adv}, X_i) - \lambda \\
\]

Explaining Learn2Weight. L2W minimizes divergence by training on \( \{d_{H\Delta H}(X_j, X_i) : \Delta \theta \} \) pairs, such that \( \Delta \theta = L2W(d_{H\Delta H}(X_j, X_i)) \), where \( d_{H\Delta H}(X_j, X_i) \) is reconstructed from the difference between \( X_j \) and \( X_i \). The target model possesses a decision boundary (Liu et al., 2019) to classify inputs based on whether they cross the boundary or not; adversarial inputs have a tendency of being near the boundary and fooling it. Meta learning applies perturbations to the decision boundary such that the boundary covers certain adversarial inputs otherwise misclassified, and in this way blocks transferability. The advantage of training on multiple domains \( \{X_j\}_{j=1}^{T} \) is that the after-L2W divergence between \( X_j^{adv} \) and \( X_i \) is smaller because L2W’s weight perturbations render the decision boundary more precise in classifying inputs.

Explaining perturbation sets. We attributed why adversarial sentences \( X_j^{adv} \) are computed to be domain-dissimilar despite originating from \( X_j \) due to insufficient domain-invariant, class-dependent features resulting in low \( e(X_j^{adv}, X_i) \), i.e. low \( tf(X_j^{adv}, X_i) \). To replicate this phenomenon in natural domains, we iteratively perturb \( X_i \) to increase the proportion of class-independent features. This approximates the real-world similarity-based attack scenario where class-dependent features may be limited for inference. By generating the synthetic data, we are feeding L2W attack data with variations in \( d_{H\Delta H} \) and class-independent feature distributions. This prepares L2W to robustify weights \( \theta_i \) when such feature distributions are met.
6 Experiments

6.1 Baselines

Defensive distillation (Papernot et al., 2016c, 2017): The high-level implementation of defensive distillation is to first train an initial model against target domain inputs and labels, and retrieve the raw class probability scores. The predicted probability values would be used as the new labels for the same target sentences, and we would train a new model based on this new label-sentence pair.

Adversarial training (Goodfellow et al., 2014; Madry et al., 2017): It is shown that injecting adversarial examples throughout training increases the robustness of target neural network models. In this baseline, target model is trained with both original training data and adversarial examples generated from original training data. However, since the adversarial examples are still generated from the target domain, it is unlikely that the method can defend against a similar-domain adversarial attack, which is the result of domain-variant features.

Perturbation sets adversarial training: This ablation baseline tests for incremental performance to a baseline defense using domain-variant inputs. We adapt adversarial training to be trained on perturbation sets (synthetic domains) generated with Algorithm 3 with respect to target domain $X_i$.

6.2 Learn2Weight Performance

Defense performance. We present the results of different defense baselines in Table 3. First, we can see that L2W achieves the highest after-defense accuracy against the adversarial attack. Take the magazine as target domain for example: if the adversary chooses to use book data as the attack domain, it would reduce the target model accuracy to 0.343. However, L2W can improve the performance to 0.843, which is a significant and substantial improvement against the attack. This improvement also exist across different target/attack domain pairs. Second, we see that all defense methods can improve the accuracy to some extent which indicates the importance and effectiveness of having robust training for machine learning models.

Table 3: After-defense Accuracy: Learn2Weight outperforms the baseline and ablation methods.

<table>
<thead>
<tr>
<th>Target Domain</th>
<th>Attack Domain</th>
<th>After-attack Accuracy</th>
<th>After-defense Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>magazine</td>
<td>baby</td>
<td>0.381 0.366 0.343</td>
<td>0.365 0.386 0.401</td>
</tr>
<tr>
<td>baby</td>
<td>book</td>
<td>0.639 0.559 0.657</td>
<td>0.558 0.577 0.661</td>
</tr>
<tr>
<td>book</td>
<td>magazine</td>
<td>0.549 0.561 0.597</td>
<td>0.588 0.629 0.577</td>
</tr>
<tr>
<td>book</td>
<td>dvd</td>
<td>0.608 0.637 0.620</td>
<td>0.604 0.620 0.587</td>
</tr>
<tr>
<td>magazine</td>
<td>book</td>
<td><strong>0.796 0.842 0.843</strong></td>
<td><strong>0.774 0.751 0.737</strong></td>
</tr>
</tbody>
</table>

Table 4: Models: L2W retains high after-defense accuracy at varying attack model architectures.

<table>
<thead>
<tr>
<th>Target Domain</th>
<th>Attack Domain</th>
<th>After-Attack Accuracy</th>
<th>After-Defense Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>book</td>
<td>dvd</td>
<td>0.342 0.413 0.477</td>
<td>0.786 0.847 0.804</td>
</tr>
<tr>
<td>book</td>
<td>kitchenware</td>
<td>0.350 0.372 0.325</td>
<td>0.765 0.826 0.795</td>
</tr>
<tr>
<td>book</td>
<td>electronics</td>
<td>0.400 0.389 0.416</td>
<td>0.792 0.812 0.784</td>
</tr>
<tr>
<td>dvd</td>
<td>book</td>
<td>0.326 0.434 0.479</td>
<td>0.816 0.795 0.824</td>
</tr>
<tr>
<td>dvd</td>
<td>kitchenware</td>
<td>0.355 0.370 0.379</td>
<td>0.728 0.796 0.755</td>
</tr>
<tr>
<td>dvd</td>
<td>electronics</td>
<td>0.387 0.377 0.332</td>
<td>0.825 0.836 0.812</td>
</tr>
<tr>
<td>electronics</td>
<td>book</td>
<td>0.425 0.394 0.473</td>
<td>0.775 0.821 0.795</td>
</tr>
<tr>
<td>electronics</td>
<td>dvd</td>
<td>0.342 0.395 0.452</td>
<td>0.784 0.845 0.855</td>
</tr>
<tr>
<td>electronics</td>
<td>kitchenware</td>
<td>0.390 0.384 0.464</td>
<td>0.730 0.824 0.753</td>
</tr>
</tbody>
</table>
olutional layers with kernel widths of 3, 4, and 5, all with 100 output channels, and randomly-initialized, based on \textcite{Kim2014}. We configure Logistic Regression based on \textcite{Maas2011}. Based on \textcite{Devlin2019}, we initialize a pretrained BERT with its own embeddings. Models are trained until reaching state-of-the-art validation accuracy (early-stopping pauses training at loss 0.5).

We present the results of different attack model architectures in Table 4. First, the similar-domain adversarial attack is model-agnostic and it does not require the target and attack model to have identical architectures. We can see that all four attack model architectures are able to reduce the target model accuracy. Second, the results suggest that L2W is also model-agnostic as it can substantially improve the after-defense accuracy regardless which attack model is used.

7 Conclusion

In this newly-proposed, empirically-effective similar-domain adversarial attack, an adversary can choose a similar domain to the target task, build a substitute model and produce adversarial examples to fool the target model. We also propose a defense strategy, Learn2Weight, that learns to adapt the target model’s weight using crafted adversarial examples. Compared with other adversarial defense strategies, Learn2Weight can improve the target model robustness against the similar-domain attack. Our method demonstrates properties of a good adversarial defense, such as adopting a defense architecture that adapts to situations/inputs rather than compromising standard error versus robustness error, to leverage class-independent properties in domain-variant text, and factoring in domain similarity in adversarial robustness.
References


Siddhartha Datta and Nigel Shadbolt. 2022b. Interpolating compressed parameter subspaces.


Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In ACL, pages 142–150.


Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. 2019. Cutmix: Regularization strategy to train strong classifiers with localizable features.


