# Influence Tuning: Demoting Spurious Correlations via Instance Attribution and Instance-Driven Updates

**Xiaochuang Han** **Yulia Tsvetkov**
Paul G. Allen School of Computer Science & Engineering, University of Washington
{xhan77, yuliats}@cs.washington.edu

## Abstract

Among the most critical limitations of deep learning NLP models are their lack of interpretability, and their reliance on spurious correlations. Prior work proposed various approaches to interpreting the black-box models to unveil the spurious correlations, but the research was primarily used in human-computer interaction scenarios. It still remains underexplored whether or how such model interpretations can be used to automatically "unlearn" confounding features. In this work, we propose *influence tuning*—a procedure that leverages model interpretations to update the model parameters towards a plausible interpretation (rather than an interpretation that relies on spurious patterns in the data) in addition to learning to predict the task labels. We show that in a controlled setup, influence tuning can help deconfounding the model from spurious patterns in data, significantly outperforming baseline methods that use adversarial training.[1]

## 1 Introduction

Despite the huge success of contemporary deep learning models and various applications that they power, critical limitations persist. Among the most harmful issues are their lack of interpretability (Lipton, 2018; Guidotti et al., 2018), and the tendency to learn spurious correlations, in addition to the true signals of the task (Leino et al., 2019; Sagawa et al., 2020b). Both of these lead to corrosive outcomes, from reduced performance on datasets in which the confounds no longer hold (Jia and Liang, 2017; Gururangan et al., 2018; Glockner et al., 2018; McCoy et al., 2019; Kumar et al., 2019; Clark et al., 2019), to pernicious biases in model decisions (Sun et al., 2019; Blodgett et al., 2020; Field et al., 2021), and to overall reduced trust in technology (Ribeiro et al., 2016; Ehsan et al., 2019).

[1]This work was done at Carnegie Mellon University. Code is available at https://github.com/xhan77/influence-tuning.

Consequently, multiple approaches have been proposed to alleviate the issues of the growing inscrutability and brittleness of the models. Two prominent approaches to interpretability in NLP models are (1) *feature attribution*—identifying important tokens in the input span, e.g. via saliency maps (Li et al., 2016; Ribeiro et al., 2016); and (2) *instance attribution*—explaining the model decisions as a function of influential training data (Koh and Liang, 2017; Han et al., 2020; Pruthi et al., 2020b). Both lines of research aim to help users build trust in the model by showing the rationale behind the model decision.

Approaches to demoting the influence of spurious confounds in the data include (1) model-based approaches to learn confound-invariant representations, e.g., adversarial training (Pryzant et al., 2018; Elazar and Goldberg, 2018; Kumar et al., 2019); (2) data-based approaches to balance the training data, e.g., counterfactual data augmentation (Zmigrod et al., 2019; Kaushik et al., 2020); (3) optimization approaches to account for worst-case scenarios, e.g., distributionally robust optimization (Sagawa et al., 2020a); and (4) post-processing approaches, such as model ensembling (Clark et al., 2019).

The issues of interpretability and robust generalization are not unrelated. Interpretations can facilitate the discovery of the model's reliance on frequent spurious patterns. For example, in natural language inference models an over-reliance on lexical signals can be revealed via feature attribution (Gururangan et al., 2018), via instance attribution (Han et al., 2020), or through a combination of thereof (Pezeshkpour et al., 2021a). In this work, we investigate a closer interaction between interpretability and model robustness.

Our research hypothesis is that interpretations that discover confounds can be incorporated at training time, to proactively guide the model towards avoiding the confounds and improving generalization. Our method relies on instance attribution interpretation methods that determine the *influence* of training data on the model's decisions (§2). We show how this influence can help discover the model's reliance on some spurious patterns, first in an illustrative task (§3), and then more generally in our proposed framework *influence tuning*. Influence tuning aims to demote the spurious patterns by guiding the model to produce plausible interpretations via instance attribution (§4). We evaluate our approach on two datasets in a controlled setup (§5, §6). Our experiments show that the proposed influence tuning method outperforms the baselines that use adversarial training (Ganin et al., 2016; Pryzant et al., 2018). We conclude with a discussion of a potentially broader impact of influence tuning on various NLP tasks.

## 2 Interpretation via Instance Attribution

Our primary goal is to use model interpretations for deconfounding the model during training. We focus on instance attribution approaches since these interpretations may help capture higher-level attributes in addition to token- and phrase-level lexical features, e.g., span overlaps, the length of text, etc. In this section, we review the family of instance attribution methods.

Many NLP models share a same general formula for their decision process during testing: $\hat{y} = f(x_{\text{test}}; \theta)$, where $x_{\text{test}}$ is the test input tokens and $\theta$ is the parameters of the trained model. While feature attribution methods like saliency maps (Simonyan et al., 2014; Li et al., 2016) focus on interpreting an NLP model's decision by the importance of each individual tokens within $x_{\text{test}}$, instance attribution methods often look at the influence of $\theta$ on the decision, which is further influenced by the training examples the model uses during the training phase.

**Influence functions** Koh and Liang (2017) propose influence functions (IF) for ML models, following the vision from robust statistics. IF first approximates how upweighting a particular training example $z_{\text{train}} = (x_{\text{train}}, y_{\text{train}})$ in the training set $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ by an infinitesimal $\epsilon_{\text{train}}$ would change the learned model parameters $\theta$:

$$\frac{d\theta}{d\epsilon_{\text{train}}} = -H_\theta^{-1} \nabla_\theta \mathcal{L}(z_{\text{train}}, \theta),$$

where $H_\theta = \frac{1}{n}\sum_{i=1}^{n} \nabla_\theta^2 \mathcal{L}(z_i, \theta)$ is the Hessian of the model. We can then use the chain rule to measure how this change in the model parameters would in turn affect the loss of the probing input:[2]

$$\frac{d\mathcal{L}(z_{\text{probe}}, \theta)}{d\epsilon_{\text{train}}} = \nabla_\theta \mathcal{L}(z_{\text{probe}}, \theta) \cdot \frac{d\theta}{d\epsilon_{\text{train}}}$$

The final influence of a train example to a probing example is defined as: $\mathcal{I}(z_{\text{train}}, z_{\text{probe}}) = -\frac{d\mathcal{L}(z_{\text{probe}}, \theta)}{d\epsilon_{\text{train}}}$. That is, a training example is influential to a probing example if upweighting it in the training set would make the model more likely to make a correct decision over the probing example.[3]

**Gradient product** Computing the inverse Hessian $H_\theta^{-1}$ in the IF is expensive and requires further approximations if the model is non-convex.[4] Pruthi et al. (2020b) tackle the problem from a different perspective and arrive at a similar, but a first-order solution:[5]

$$\mathcal{I}(z_{\text{train}}, z_{\text{probe}}) = \sum_{i=1}^{k} \nabla_\theta \mathcal{L}(z_{\text{train}}, \theta_i) \cdot \nabla_\theta \mathcal{L}(z_{\text{probe}}, \theta_i),$$

where $\theta_i$ is the checkpoint of the model at each training epoch. The intuition behind this method is to approximate the total reduction in the probing loss $\mathcal{L}(z_{\text{probe}}, \theta)$ during the training process when the training example $z_{\text{train}}$ is used. Compared to IF, this gradient product method essentially drops the inverse Hessian term and reduces the problem to the dot product between the gradient of the training loss and the gradient of the probing loss.

**Gradient cosine** One potential problem of IF and gradient product is being dominated by some outlier training examples, where the norm of their training gradients is significantly larger than the rest of examples. This would lead the method to identify the same set of outlier training examples being influential to a large number of different probing examples. Barshan et al. (2020) points out this variance-lacking problem of IF and proposes a simple modification: substituting the dot product operation with cosine similarity, normalizing by the norm of the training gradients. Following the

[2]A probing input can be either obtained during test time or selected from the training set.

[3]More details about IF and its applications in NLP can be found in Koh and Liang (2017) and Han et al. (2020).

[4]Basu et al. (2021) also points out IF can be less accurate when used with deep neural networks.

[5]Here, we are presenting the equally weighted TracInCP variant from Pruthi et al. (2020b).

same intuition, we modify and further simplify the gradient product method:

$$\mathcal{I}(z_{\text{trn}}, z_{\text{prb}}) = \frac{\nabla_\theta \mathcal{L}(z_{\text{trn}}, \theta) \cdot \nabla_\theta \mathcal{L}(z_{\text{prb}}, \theta)}{\|\nabla_\theta \mathcal{L}(z_{\text{trn}}, \theta)\| \, \|\nabla_\theta \mathcal{L}(z_{\text{prb}}, \theta)\|}$$

We use this latter influence definition for the instance attribution interpretation method throughout this work.

## 3 A Toy Example: Predicting Text Length

Now let's imagine a simple synthetic task where an NLP model like BERT (Devlin et al., 2019) is trained for binary text classification. Class 0 contains random short sentences with a length sampled from $\mathcal{N}(\mu_{\text{short}}, \sigma^2)$; class 1 contains random long sentences with a length sampled from $\mathcal{N}(\mu_{\text{long}}, \sigma^2)$. Our classification task is to predict the text length.

However, there are confounds in this data. For every sentence, we insert a confounding token at the beginning of the sentence. Most of the time (e.g., 90%), token `A` would co-occur with a short sentence and `B` would co-occur with a long sentence; for the remaining sentences, this co-occurrence is flipped.

Our goal is to finetune the classifier so that it learns to predict the class labels (0 or 1) using, as intended, the text length information, instead of overfitting to the confounding tokens `A` or `B`. We refer to the text length as a *core attribute*, and to the confounding prefix tokens as a *spurious attribute*.[6]

Finetuning the classifier on our synthetic task yields an 100% accuracy on the training set (overfitting). We are more interested in interpreting what information the model relies on to make classification decisions. This drives us to apply the instance attribution interpretation methods.

To interpret each classification decision via instance attribution, we randomly sample a few examples within the training set[7] as our probing examples $z_{\text{prb}}$. We calculate the influence of each training example $z_{\text{trn}}$ on $z_{\text{prb}}$ using the gradient cosine method (§2). Our expectation is that the $z_{\text{trn}}$ instances that have the same core attribute as $z_{\text{prb}}$ should be influential to $z_{\text{prb}}$. In our example case, this means we expect the model to learn that the long training instances from class 1 are positively influential for labeling a long probing example with class 1. At the same time, the spurious attribute of $z_{\text{trn}}$ should not dominate the contribution to the training example's influence towards $z_{\text{prb}}$. Specifically, two long training examples, one with a confounding prefix `A` and the other with `B`, should be both influential to the long probing example with a confounding prefix, say, `B`.

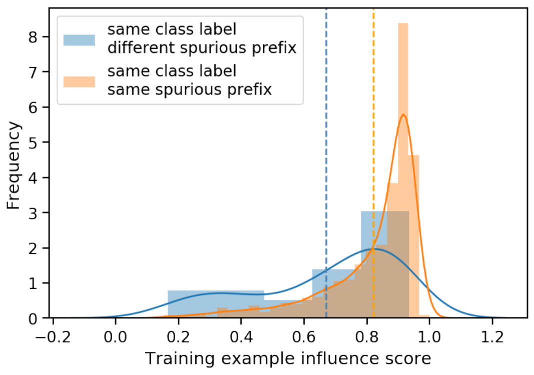


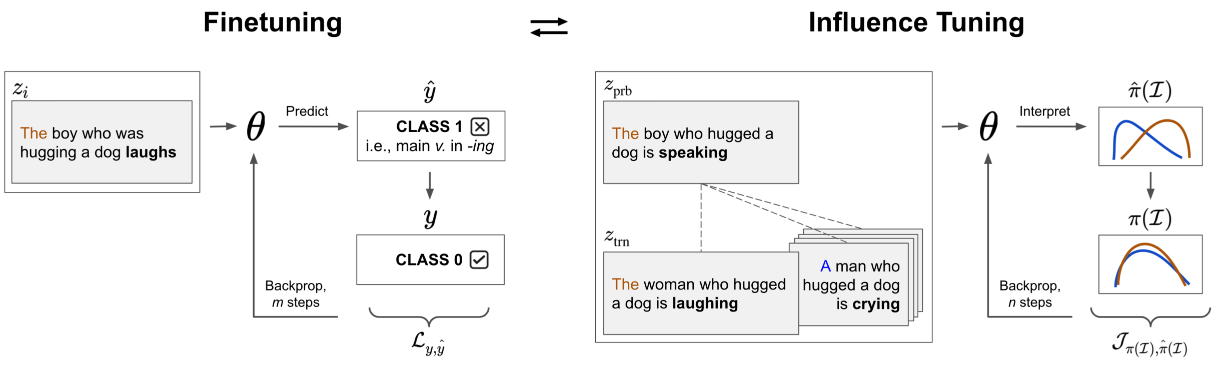
Figure 1: Distribution of each same-class training example's influence score $\mathcal{I}(z_{\text{trn}}, z_{\text{prb}})$ towards a typical probing example in TextLen (§5.2). The range of influence scores is $[-1, 1]$. The average score difference between the two groups is 0.15, and the difference is statistically significant via $t$-test.

Figure 1 illustrates the influence score distribution for a typical probing example. The probing example is from class 1 (long text) and has a confounding prefix `B`. The orange plot in the figure shows the influence distribution of all class 1 training examples with the same prefix `B`, whereas the blue plot shows the influence distribution of all class 1 training examples with the different spurious prefix `A`. We observe that there is a statistically significant influence difference between these two groups. However, as the spurious attribute should not influence the model's decision process, we conjecture that *the influence difference shows the model is confounded.*

As we show in this motivating example, research on interpretability via instance attribution can help us extract rationales behind the model decisions. When we know what are possible spurious patterns in the data, we can check whether the spurious confounds are influencing learning, yielding implausible interpretations. For *plausible* interpretations, such reliance on spurious attributes should not be as significant. In the next section, we propose a methodology to improve the model systematically, upon seeing an implausible rationale.

[6]This toy task is inspired by the numeric toy dataset in Sagawa et al. (2020b).

[7]In our experiments we choose probing examples from the training set, but it can be a held-out set as well.



Finetuning
Influence Tuning
$z_i$
The boy who was hugging a dog laughs
$\theta$
Predict
$\hat{y}$
CLASS 1 ☒
i.e., main v. in -ing
$y$
CLASS 0 ☑
Backprop, m steps
$\mathcal{L}_{y,\hat{y}}$
$z_{\text{prb}}$
The boy who hugged a dog is speaking
$z_{\text{trn}}$
The woman who hugged a dog is laughing
A man who hugged a dog is crying
Interpret
$\hat{\pi}(\mathcal{I})$
$\pi(\mathcal{I})$
Backprop, n steps
$\mathcal{J}_{\pi(\mathcal{I}),\hat{\pi}(\mathcal{I})}$

Figure 2: The influence tuning framework alternates between the **standard finetuning** phase (left) and the **influence tuning** phase (right). Illustrative examples are adapted from the MSGS dataset. **Standard finetuning** (left): As introduced in §5.2, the main verb in "the boy who was hugging a dog *laughs*" is not a present participle ending with "*-ing*", so the sentence should belong to `CLASS_0` ($y$). A model $\theta$ may initially predict `CLASS_1` with a high probability ($\hat{y}$). In the finetuning steps, we form a loss function $\mathcal{L}_{y,\hat{y}}$ over the labels and backpropagate into the model parameters. **Influence tuning** (right): For the influence tuning steps, we sample a few probing examples $z_{\text{prb}}$ and training examples $z_{\text{trn}}$ from the train set. In the figure $z_{\text{prb}}$ and $z_{\text{trn}}$ both belong to `CLASS_1` (main verb in *-ing* form) while the examples in $z_{\text{trn}}$ can have the same spurious attribute (sentence beginning with "*the*") or different spurious attribute (beginning with "*a*") as $z_{\text{prb}}$. A model $\theta$ may initially give the interpretation ($\hat{\pi}(\mathcal{I})$) that these examples in $z_{\text{trn}}$ have significantly different influence over $z_{\text{prb}}$. This could be a sign that the model is confounded; we form a loss function $\mathcal{J}_{\pi(\mathcal{I}),\hat{\pi}(\mathcal{I})}$ and backpropagate into the model parameters for a more plausible interpretation.

## 4 Influence Tuning

We propose a method to *tune the model towards providing plausible rationales behind its decisions*. Motivated by the example scenario in §3, we define this plausibility to be the difference between the influence of training examples with different spurious attributes. We therefore call the method *influence tuning*. Formally, we first randomly sample one probing example $z_{\text{prb}} = (x_{\text{prb}}, y_{\text{prb}})$ from the training set. We then sample a small group of training examples $\{z_{A_1}, \dots, z_{A_k}\} \subset \{z_{\text{trn}} \mid y_{\text{trn}} = y_{\text{prb}}, c_{\text{trn}} = c_{\text{prb}}\}$, that share the same label ($y$) and *the same* spurious attribute ($c$) as in $z_{\text{prb}}$ (e.g., samples from the orange distribution in Figure 1). Similarly, we sample a small group of training examples $\{z_{B_1}, \dots, z_{B_k}\} \subset \{z_{\text{trn}} \mid y_{\text{trn}} = y_{\text{prb}}, c_{\text{trn}} \neq c_{\text{prb}}\}$, that share the same label but with a spurious attribute that is *different* from the spurious attribute in $z_{\text{prb}}$ (e.g., samples from the blue distribution in Figure 1). Note that although in our example scenario $y$ and $c$ are both binary, they are not required to be so. Since the spurious attribute $c$ should not be a part of the rationale behind the model's decision, we expect the average influence of $\{z_{A_1}, \dots, z_{A_k}\}$ and $\{z_{B_1}, \dots, z_{B_k}\}$ on $z_{\text{prb}}$ to be close to each other. Therefore, it is natural to define a new loss function over the influence scores and incorporate it in the model training:

$$\mathcal{J} = \left(\frac{1}{k}\sum_{i=1}^{k}\mathcal{I}(z_{A_i}, z_{\text{prb}}) - \frac{1}{k}\sum_{i=1}^{k}\mathcal{I}(z_{B_i}, z_{\text{prb}})\right)^2.$$

To optimize for the influence loss $\mathcal{J}$, we need the gradient $\nabla_\theta \mathcal{J}$:

$$\nabla_\theta \mathcal{J} = \left(\frac{2}{k}\sum_{i=1}^{k}\mathcal{I}(z_{A_i}, z_{\text{prb}}) - \frac{2}{k}\sum_{i=1}^{k}\mathcal{I}(z_{B_i}, z_{\text{prb}})\right)$$
$$\left(\frac{1}{k}\sum_{i=1}^{k}\nabla_\theta\mathcal{I}(z_{A_i}, z_{\text{prb}}) - \frac{1}{k}\sum_{i=1}^{k}\nabla_\theta\mathcal{I}(z_{B_i}, z_{\text{prb}})\right),$$

where the key is in calculating $\nabla_\theta\mathcal{I}(z_{\text{trn}}, z_{\text{prb}})$ with arbitrary $z_{\text{trn}}$ being either $z_{A_i}$ or $z_{B_i}$.

Recall that with the gradient cosine influence definition, $\mathcal{I}(z_{\text{trn}}, z_{\text{prb}}) = \frac{\nabla_\theta\mathcal{L}(z_{\text{trn}},\theta)\cdot\nabla_\theta\mathcal{L}(z_{\text{prb}},\theta)}{\|\nabla_\theta\mathcal{L}(z_{\text{trn}},\theta)\|\,\|\nabla_\theta\mathcal{L}(z_{\text{prb}},\theta)\|}$. We can then derive $\nabla_\theta\mathcal{I}(z_{\text{trn}}, z_{\text{prb}})$ as:

$$\nabla_\theta\mathcal{I}(z_{\text{trn}}, z_{\text{prb}}) = \mathbf{p} + \mathbf{q} - \mathbf{r} - \mathbf{s}$$
$$\mathbf{p} = \frac{1}{\|\nabla_\theta\mathcal{L}(z_{\text{trn}})\|\,\|\nabla_\theta\mathcal{L}(z_{\text{prb}})\|}H_{\text{trn}}\nabla_\theta\mathcal{L}(z_{\text{prb}})$$
$$\mathbf{q} = \frac{1}{\|\nabla_\theta\mathcal{L}(z_{\text{trn}})\|\,\|\nabla_\theta\mathcal{L}(z_{\text{prb}})\|}H_{\text{prb}}\nabla_\theta\mathcal{L}(z_{\text{trn}})$$
$$\mathbf{r} = \frac{\nabla_\theta\mathcal{L}(z_{\text{trn}})\cdot\nabla_\theta\mathcal{L}(z_{\text{prb}})}{\|\nabla_\theta\mathcal{L}(z_{\text{trn}})\|^3\,\|\nabla_\theta\mathcal{L}(z_{\text{prb}})\|}H_{\text{trn}}\nabla_\theta\mathcal{L}(z_{\text{trn}})$$
$$\mathbf{s} = \frac{\nabla_\theta\mathcal{L}(z_{\text{trn}})\cdot\nabla_\theta\mathcal{L}(z_{\text{prb}})}{\|\nabla_\theta\mathcal{L}(z_{\text{trn}})\|\,\|\nabla_\theta\mathcal{L}(z_{\text{prb}})\|^3}H_{\text{prb}}\nabla_\theta\mathcal{L}(z_{\text{prb}})$$

where the Hessians $H_{\text{trn}} = \nabla^2_\theta\mathcal{L}(z_{\text{trn}})$ and $H_{\text{prb}} = \nabla^2_\theta\mathcal{L}(z_{\text{prb}})$. We omit $\theta$ in $\mathcal{L}(\cdot)$ for simplicity. Detailed derivations can be found in the appendix.[8]

[8]Intuitively, $\mathbf{p}$ and $\mathbf{q}$ find gradients that would help maximize the inner product of the training and probing model gradients in the next model update; $\mathbf{r}$ and $\mathbf{s}$ find gradients that would constrain the norm of the training and probing gradients for the next update.

Overall, obtaining the gradient $\nabla_\theta \mathcal{J}$ for the influence loss $\mathcal{J}$ defined over the tuple $(z_{\text{prb}}, \{z_{A_1}, \ldots, z_{A_k}\}, \{z_{B_1}, \ldots, z_{B_k}\})$ makes the optimization possible. For the actual training process, we alternate the optimization of $\theta$ over both the regular label prediction loss $\mathcal{L}$ and the influence loss $\mathcal{J}$, with the interval as a hyperparameter to select. That is, do $m$ steps of regular label loss propagation, $n$ steps of influence loss propagation, then back to label loss propagation, and so on. The goal is to find a set of model parameters, without changing the model architecture, that lead to both accurate label predictions and plausible rationales behind the decisions. We use a pretrained BERT model as our initial model $\theta$. Figure 2 summarizes our proposed method using the data examples that we will introduce in §5.2.[9]

### 4.1 A special case of influence tuning

The above section gives an influence tuning framework based on the influence score $\mathcal{I}(\cdot,\cdot)$ defined on the full set of model parameters $\theta$. Now we are going to investigate an interesting special case of the framework, which defines the influence score on a partial parameter set.

Recall that we are using a pretrained BERT model as our initial model $\theta$ in our setup, and finetuning the BERT model would require training a prediction head over the transformer layers. For text classification, the prediction head is just a linear projection layer $W$, projecting from the BERT `[CLS]` token embedding to the label space and connecting to the final cross entropy loss. Additionally in our setup, our sampled $z_{\text{prb}}$ and $z_{\text{trn}}$ have the same label $y$. Now let's define a small parameter subset $\theta_{\text{proj}} = W_{(y)}$, representing the row of the final projection layer $W$ corresponding to the label $y$.

Similar to the original gradient cosine influence definition, we define $\mathcal{I}_{\text{proj}}(z_{\text{trn}}, z_{\text{prb}}) = \frac{\nabla_{\theta_{\text{proj}}}\mathcal{L}(z_{\text{trn}},\theta)\cdot\nabla_{\theta_{\text{proj}}}\mathcal{L}(z_{\text{prb}},\theta)}{\|\nabla_{\theta_{\text{proj}}}\mathcal{L}(z_{\text{trn}},\theta)\|\,\|\nabla_{\theta_{\text{proj}}}\mathcal{L}(z_{\text{prb}},\theta)\|}$. We can further expand the label loss $\mathcal{L}$ with the parameter subset $W_{(y)}$ and the `[CLS]` embedding $h_{\texttt{[CLS]}}$:

$$\mathcal{L}(z,\theta) = -\log \frac{\exp(W_{(y)}h_{\texttt{[CLS]}})}{\sum_{y'}\exp(W_{(y')}h_{\texttt{[CLS]}})}$$

$$\begin{aligned}\nabla_{\theta_{\text{proj}}}\mathcal{L}(z,\theta) =& \frac{\exp(W_{(y)}h_{\texttt{[CLS]}})}{\sum_{y'}\exp(W_{(y')}h_{\texttt{[CLS]}})}h_{\texttt{[CLS]}} \\ &- h_{\texttt{[CLS]}} = (p(y)-1)h_{\texttt{[CLS]}}\end{aligned}$$

Therefore, $\frac{\nabla_{\theta_{\text{proj}}}\mathcal{L}(z_{\text{trn}},\theta)}{\|\nabla_{\theta_{\text{proj}}}\mathcal{L}(z_{\text{trn}},\theta)\|} = -\frac{h_{\text{trn}\,\texttt{[CLS]}}}{\|h_{\text{trn}\,\texttt{[CLS]}}\|}$, and similarly $\frac{\nabla_{\theta_{\text{proj}}}\mathcal{L}(z_{\text{prb}},\theta)}{\|\nabla_{\theta_{\text{proj}}}\mathcal{L}(z_{\text{prb}},\theta)\|} = -\frac{h_{\text{prb}\,\texttt{[CLS]}}}{\|h_{\text{prb}\,\texttt{[CLS]}}\|}$. We finally rewrite the partial influence definition $\mathcal{I}_{\text{proj}}(z_{\text{trn}}, z_{\text{prb}})$ as $\frac{h_{\text{trn}\,\texttt{[CLS]}}\cdot h_{\text{prb}\,\texttt{[CLS]}}}{\|h_{\text{trn}\,\texttt{[CLS]}}\|\,\|h_{\text{prb}\,\texttt{[CLS]}}\|}$, *which is essentially the cosine similarity between the training and probing example's `[CLS]` embeddings.*

The new definition $\mathcal{I}_{\text{proj}}(z_{\text{trn}}, z_{\text{prb}})$ leads to a new influence loss $\mathcal{J}_{\text{proj}}$. Different from the second-order influence tuning method that obtains $\nabla_\theta \mathcal{J}$, we can get $\nabla_\theta \mathcal{J}_{\text{proj}}$ by applying the regular gradient backward operation on the model and thus updating the model faster. All the other parts of the framework, like the data tuple selection and the alternating training objectives, remain the same. We call this special variant of influence tuning **embedding tuning**.

## 5 Experimental Setup

### 5.1 Adversarial training as a baseline

One notable feature of influence tuning is that it is designed to help deconfounding NLP models without adding additional modules to the network. A related line of research on deconfounding NLP models takes the intuition from domain adversarial training (Ganin et al., 2016; Pryzant et al., 2018; Kumar et al., 2019). These methods usually have two classifier modules built on top of a shared encoder. The objective for the model is adversarial: the model should be able to predict the target label $y$ of the input accurately, while failing at reconstructing the spurious attribute $c$ effectively, which potentially indicates that the confounding information regarding $c$ is not encoded by the model.

As a baseline for this work, we specifically modify a BERT model according to the method described in Pryzant et al. (2018). It uses a gradient reversal layer at the beginning of the confound classifier head that would multiply the gradient by $-1$ during the backward pass. All of the BERT transformer layers form the shared encoder for the label classifier and the confound classifier. The

[9] Instead of the alternating optimization we adopted, folding the influence loss into the standard finetuning loss as a regularizer may work as well. We did not explore it here since our initial hypothesis is whether we can use a plausible interpretation to help build a more generalizable model: the instance attribution interpretation methods assume some regular, untouched finetuning steps before interpreting.

implicit loss used by the model can then by written as $\mathcal{L} = \mathcal{L}_{\text{label}} - \lambda\mathcal{L}_{\text{confound}}$, where $\mathcal{L}_{\text{label}}$ and $\mathcal{L}_{\text{confound}}$ are both cross entropy loss, and $\lambda$ is a hyperparameter to select. Essentially, this method learns and unlearns—learns to predict the correct label while unlearning the information that could help reconstruct the confound attribute. Compared to other prior work tackling spurious correlations mentioned in §1, this method is also most suitable for a direct comparison with our proposed influence tuning framework, because both methods aim to explicitly demote certain known confounds for the model.

### 5.2 Datasets

To evaluate the proposed approaches for deconfounding NLP models, we conduct controlled experiments on two datasets.

**TextLen** TextLen is a synthetic dataset we create that follows the example scenario in §3. Specifically, for the training set, we randomly split 1500 sentences from the canonical CoNLL-2003 shared task dataset (Sang and De Meulder, 2003) to two classes 0 and 1 of equal sizes. Sentences from class 0 are trimmed to a length sampled from a normal distribution with $\mu_{\text{short}} = 15, \sigma = 4$; sentences from class 1 are trimmed with $\mu_{\text{long}} = 25, \sigma = 4$. We add prefix tokens A="Negative." and B="Positive." to the start of the sentences, such that 90% of the time, a class 0 sentence would receive the negative prefix and a class 1 sentence would receive the positive prefix. However, in the dev set and the test set of TextLen,[10] while trimmed with the same text length distribution, only 50% of the time the confounding prefix would correlate with the class label of the sentence. A deconfounded model should achieve a good classification performance on both the train and test splits.

**MSGS** The **M**ixed **S**ignals **G**eneralization **S**et is proposed by Warstadt et al. (2020) to investigate whether language models would acquire a preference for linguistic generalizations. The model is supposed to learn a classification task with an ambiguous training dataset. For example, a class 1 sentence could be "*the boy who hugged a cat is sneezing*", and a class 0 sentence could be "*a boy who is hugging the cat sneezed*". To distinguish the two classes, a model performing surface generalizations could potentially rely on whether the article "*the*" or "*a*" precedes the sentence. A model performing linguistic generalizations, however, could be deciding based on whether the main verb of the sentence is in the "*-ing*" form. The linguistic feature decides the class of the sentence in both the MSGS train and test sets, whereas the surface feature correlates highly with the classes only in the training set, and co-occurs randomly with the classes in the disambiguated test data. Specifically, we choose MSGS's SYNTACTIC CATEGORY as the core attribute and RELATIVE POSITION as the spurious attribute; for the training set, we also adopt an inoculation rate of 0.3% (Warstadt et al., 2020).

We show statistics of TextLen and MSGS in Table 1. We use BERT-base as our model template and the default BERT Adam optimizer for both tasks and all of the deconfounding methods. We perform hyperparameter search using the dev set for all of the methods. Detailed hyperparameters can be found in the appendix.

## 6 Results

### 6.1 Does influence tuning make the model interpretations more plausible?

We are interested in a preliminary research question first: having seen the confounded model interpretations discovered in §3, does our proposed method, influence tuning, make the model interpretations more plausible? To quantitatively measure how much the model relies on the spurious attribute to make decisions, for both tasks we randomly select 40 probing examples $z_{\text{prb}}$ from the training set. For each probing example $z_{\text{prb}}$, we put the training examples into two groups: $A = \{z_{\text{trn}} \mid y_{\text{trn}} = y_{\text{prb}}, c_{\text{trn}} = c_{\text{prb}}\}$ and $B = \{z_{\text{trn}} \mid y_{\text{trn}} = y_{\text{prb}}, c_{\text{trn}} \neq c_{\text{prb}}\}$, where $y$ is the true label and $c$ is the confounding spurious attribute. We define the *confound influence difference* (CID) to be the influence difference between the two groups to the probing example: $\frac{1}{|A|}\sum_{\text{trn}\in A}\mathcal{I}(z_{\text{trn}}, z_{\text{prb}}) - \frac{1}{|B|}\sum_{\text{trn}\in B}\mathcal{I}(z_{\text{trn}}, z_{\text{prb}})$.

We show in Figure 3 the average CID of three different models for TextLen during the training process: a model that is trained with the *regular fine-tuning* objective, a model that is trained using the *influence tuning* framework, and a control model that is trained over a *non-confounding* version of TextLen data (i.e., the spurious prefix token is removed). The final CIDs are 0.093, 0.035 and 0.019,

[10]Like the TextLen training set, the dev set content also comes from the CoNLL-2003 training set; the TextLen test set content instead comes from the CoNLL-2003 test set.

| | **TextLen** | **MSGS** |
|---|---|---|
| Size | 1500 / 480 / 500 | 5000 / 15000 / 15000 |
| Core attribute | Text length | SYNTACTIC CATEGORY |
| Spurious attribute | Prefix token | RELATIVE POSITION |
| Variance in core attribute | Yes | No |
| Spurious attribute train set success rate | 90.0% | 99.7% |
| Spurious attribute test set success rate | 50.0% | 66.7% |

Table 1: Dataset statistics

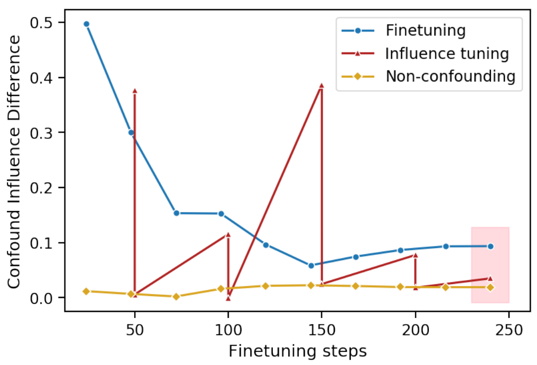


Figure 3: Average confound influence difference (CID) of different models in the TextLen task. The scale of the influence scores is $[-1, 1]$. The final CID is $0.093$ for the finetuning model, $0.035$ for the influence tuning model, and $0.019$ for the model trained on non-spurious data.

respectively for the three models. We observe that both the finetuning model and the influence tuning model start with a very high CID, indicating the confound attribute is exploited heavily at the beginning of the training process. However, for the influence tuning model, each influence tuning round happened after every 50 standard finetuning steps, helps the model achieve a near zero CID (as shown by the vertical drops within the influence tuning plot). The model does regain the CID during the following finetuning steps, but eventually arrives at a relatively low CID. The result on the MSGS data is similar, except that we do not have the non-confounding control model. The drops in CID caused by influence tuning answer our preliminary question: we do find that influence tuning makes the model interpretations more plausible in accordance with our expectation.

## 6.2 Does the guided plausibility transform well to the model generalizability?

Would a more plausible model, or more specifically a model that is *guided* to provide plausible interpretations, achieve a stronger performance in out-of-distribution test sets, where the confound information no longer helps the decision? To answer this question, we compare the different deconfounding approaches introduced in §4 and §5.1 over the TextLen and MSGS tasks.

We show our main results in Table 2. On TextLen, we observe that the adversarial training method gives a moderate improvement over the regular finetuning model. Both influence tuning and embedding tuning lead to significant accuracy gain, with the embedding tuning method achieves the highest 82.2% test accuracy. In §4.1 we derived why embedding tuning is a special case of influence tuning with a reduced set of parameters for the influence calculation. This could be enough for the TextLen dataset since the task is relatively easy. The upper bound model with the spurious attribute removed from the data still outperforms all of the deconfounding methods, leaving a gap to address in future work.

On MSGS, we observe a similar trend for the adversarial training method, which makes a moderate improvement over finetuning as expected. Again, both influence tuning and embedding tuning achieve significant improvements. However, the influence tuning method outperforms embedding tuning on this task. One contributing reason could be that the linguistic generalizations required by this task can be encoded across the full BERT transformer layers. Therefore, the influence defined only over the final projection layer in the embedding tuning case might be limiting. Overall, both our proposed influence tuning and the special case embedding tuning are effective at deconfounding the models in our experiments compared to the baselines.

| | Finetuning | Adversarial training | Influence tuning | Embedding tuning | No spurious |
|---|---|---|---|---|---|
| **TextLen** | 76.00 | 78.44 $_{(p=0.026)}$ | 80.48 $_{(p=0.003)}$ | **82.20** $_{(p<0.001)}$ | 84.96 $_{(p<0.001)}$ |
| **MSGS** | 75.33 | 83.41 $_{(p=0.179)}$ | **88.23** $_{(p=0.015)}$ | 87.45 $_{(p=0.042)}$ | - |

Table 2: Test set performance (accuracy) of different deconfounding approaches. All the experiments use five random seeds. The subscripts show the $p$-values of the $t$-tests comparing the deconfounding approaches with the regular finetuning model.

## 6.3 Can we use less data with influence tuning?

The advantage of influence tuning does not come without a price. For a general dataset, it would require at least some lightweight annotations in addition to the regular label information. For example, in §4 when we operate with the data tuple $(z_{\text{prb}}, \{z_{A_1}, \ldots, z_{A_k}\}, \{z_{B_1}, \ldots, z_{B_k}\})$, we would need information about the confound group an example belongs to. Though in our experiments with TextLen and MSGS we are sampling a relatively small set of $z_{\text{prb}}$, $z_{A_i}$ and $z_{B_i}$ (50–100 $z_{\text{prb}}$ each full influence tuning round, 3–5 $z_{A_i}$ and $z_{B_i}$ for each $z_{\text{prb}}$), the model still potentially has access to the confound information of the full dataset during the whole training process.[11] Therefore, in this section we are interested in whether we can strictly provide less accessible confound information to the model, and how this would affect the performance of our methods.

For both the TextLen and MSGS data, we randomly select subsets of the training set containing $m\%$ of the total examples. Then during the training process, we limit the model to sample $z_{\text{prb}}$, $z_{A_i}$ and $z_{B_i}$ only from the $m\%$ training subset where the confound group information is accessible. Note that this serves as a hard *upper limit* for the confound access, and the actual confound information queried by the influence tuning framework can be well under this limit.

In Table 3 we show the test performance of influence tuning and embedding tuning on TextLen and MSGS, using the same model hyperparameters as the results in Table 2. However, unlike the Table 2 results where every trial within the five random seeds succeeds in fitting the *training set*, the experiments with the data constraint sometimes fail to converge (i.e., not even fitting the train set). We exclude such failed trials from the average performance reported in Table 3, while we observe that at least three out of the five trials for each confound access rate can converge successfully.[12] We see that even with the hard constraint on the confound access rate, influence tuning and embedding tuning still outperform the baseline methods, using only 5%-20% examples. Generally, higher confound access rate would lead to stronger deconfounding performance, which creates a tradeoff to decide based on the need of the users.

| | Influence tuning | Embedding tuning |
|---|---|---|
| **TextLen** | | |
| 5% | 78.24 | 78.84 |
| 10% | 78.68 | 80.80 |
| 20% | 80.65 | 80.44 |
| 50% | 80.13 | 81.07 |
| 100% | 80.48 | 82.20 |
| **MSGS** | | |
| 5% | 81.74 | 84.90 |
| 10% | 80.48 | 81.56 |
| 20% | 83.90 | 81.56 |
| 50% | 93.57 | 82.80 |
| 100% | 88.23 | 87.45 |

Table 3: Performance of influence tuning and embedding tuning when there is an upper limit for the confound access rate. The accuracy shown is an average of at least three succeeded trials within the use of five random seeds.

## 6.4 Discussion

In this section we answered three questions regarding the influence tuning framework: whether it makes the model more plausible, whether it helps the model achieve a strong deconfounding performance, and whether it can be used with a reduced amount of data. We conducted experiments on a synthetic dataset and a linguistic probing dataset,

[11]Our baseline approach adversarial training also has access to and actually uses the confound information of the full data.

[12]Importantly, the trials are considered failed based on their training set performance (0.5 accuracy equivalent to random guess), not based on the dev set or test set performance. When deploying to an unknown test set, we would know when to re-train the model based on its known training performance.

but the potential application of our approach can be more impactful than the current tasks. For example, our method might be helpful for identifying and mitigating gender biases and racial biases in sentiment analysis or toxicity detection systems (Kiritchenko and Mohammad, 2018; Sap et al., 2019), by modeling the problem as a deconfounding task. One potential drawback is that these natural cases would inevitably require some extent of extra human annotations. However, we also believe the human feedback in NLP (Settles, 2011; Kaushik et al., 2020; Wang et al., 2021) is a crucial and controllable way to tackle model's exploitation of spurious correlations in the data, which happens as a result of the absence of proper supervision. Furthermore, if we define the influence objective differently in §4, e.g. modeling which groups of examples *should* be influential to the probing instance and to what extent, we may be able to implicitly *promote* the core attributes in the task in addition to demoting the confounds.

## 7 Related Work

Interpreting NLP models by token-level importance scores over the input span is a widely adopted approach (Belinkov and Glass, 2019). These scores can be gradient-based (Simonyan et al., 2014; Li et al., 2016), attention weights if supported by the model (Jain and Wallace, 2019; Wiegreffe and Pinter, 2019), or weights from a linear model fitting the local region of a deep model (Ribeiro et al., 2016). The models can achieve better performance or learn more efficiently if supervisions are provided for these feature importance scores (Ross et al., 2017; Zhong et al., 2019; Pruthi et al., 2020a).

Unlike the token-level interpretations, our focus in this work is on the instance attribution methods. Apart from influence functions (Koh and Liang, 2017) and TracIn (Pruthi et al., 2020b) that are already introduced, other instance attribution methods include representer point selection (Yeh et al., 2018) and $\theta$-relative influence functions (Barshan et al., 2020), with Pezeshkpour et al. (2021b) comparing the methods empirically in NLP tasks. However, these methods do not facilitate a systematic improvement for the model based on the plausibility of the interpretations, which is a gap addressed by this work. Models designed with explicit interpretability considerations like deep weighted averaging classifiers (Card et al., 2019) and SelfExplain (Rajagopal et al., 2021) may also support instance attribution, though the flexibility of the model architecture can be more limited.

One key use case of the proposed influence tuning framework is to deconfound the model from relying on spurious attributes during the decision process. Other works that aim at preventing neural models from using the spurious attributes include Elazar and Goldberg (2018) and Pryzant et al. (2018) which operate over a known set of confounds, and Kumar et al. (2019) which models unknown, latent confounds. They often involve the idea of learning invariant features across domains through adversarial training (Ganin et al., 2016; Xie et al., 2017). Spurious correlations can also be mitigated by data-based, optimization-based, and post-processing methods (Zmigrod et al., 2019; Kaushik et al., 2020; Sagawa et al., 2020a; Yaghoobzadeh et al., 2021; Clark et al., 2019). In this work, we mainly compare with the adversarial training method with gradient reversal in Pryzant et al. (2018) as a baseline, since both methods perform explicit demotions to some known confounds in the data used by the model. Future work can explore comparisons and potential combinations with other approaches addressing the spurious correlations.

## 8 Conclusion

NLP models that build upon deep neural networks are notoriously opaque about their decision process. Though instance attribution methods can be used to unveil problems of the model reflected by the implausible interpretations, a novel research question is whether or how the model training can benefit from interpretability methods in a systematic way. Our work addresses this question, by proposing the influence tuning framework that backpropagates a target instance attribution interpretation directly to the model. In two use cases of demoting spurious confounds in the data, we show that (1) influence tuning can eventually lead to more plausible model interpretations; (2) influence tuning can help build better-performing deconfounded models compared to those trained with the baseline methods; (3) influence tuning can still be reliable in lower-resource setups. Future work will explore more datasets and tasks, and other optimization methods. Additionally, we will explore guiding the model to learn to *promote* core attributes of the task in addition to *demoting* the spurious confounds.

## Acknowledgments

We thank Haoping Bai, Sachin Kumar, and the anonymous EMNLP reviewers and area chairs for helpful discussions of this work. This material is based upon work funded by the DARPA CMO under Contract No. HR001120C0124. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

## References

Elnaz Barshan, Marc-Etienne Brunet, and G. Dziugaite. 2020. RelatIF: Identifying explanatory training examples via relative influence. In *Proc. AISTATS*.

Samyadeep Basu, Philip Pope, and Soheil Feizi. 2021. Influence functions in deep learning are fragile. In *ICLR*.

Yonatan Belinkov and James Glass. 2019. Analysis methods in neural language processing: A survey. *TACL*.

Su Lin Blodgett, Solon Barocas, Hal Daumé III, and Hanna Wallach. 2020. Language (technology) is power: A critical survey of "bias" in NLP. In *Proc. ACL*.

Dallas Card, Michael Zhang, and Noah A. Smith. 2019. Deep weighted averaging classifiers. In *FAT**.

Christopher Clark, Mark Yatskar, and Luke Zettlemoyer. 2019. Don't take the easy way out: Ensemble based methods for avoiding known dataset biases. In *Proc. EMNLP*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proc. NAACL-HLT*.

Upol Ehsan, Pradyumna Tambwekar, Larry Chan, Brent Harrison, and Mark Riedl. 2019. Automated rationale generation: a technique for explainable ai and its effects on human perceptions. In *Proc. IUI*.

Yanai Elazar and Yoav Goldberg. 2018. Adversarial removal of demographic attributes from text data. In *Proc. EMNLP*.

Anjalie Field, Su Lin Blodgett, Zeerak Waseem, and Yulia Tsvetkov. 2021. A survey of race, racism, and anti-racism in NLP. In *Proc. ACL*.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *JMLR*.

Max Glockner, Vered Shwartz, and Yoav Goldberg. 2018. Breaking NLI systems with sentences that require simple lexical inferences. In *Proc. ACL*.

Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. 2018. A survey of methods for explaining black box models. *CSUR*.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel R. Bowman, and Noah A. Smith. 2018. Annotation artifacts in natural language inference data. In *Proc. NAACL-HLT*.

Xiaochuang Han, Byron C. Wallace, and Yulia Tsvetkov. 2020. Explaining black box predictions and unveiling data artifacts through influence functions. In *Proc. ACL*.

Sarthak Jain and Byron C. Wallace. 2019. Attention is not explanation. In *Proc. NAACL-HLT*.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Proc. EMNLP*.

Divyansh Kaushik, Eduard Hovy, and Zachary Chase Lipton. 2020. Learning the difference that makes a difference with counterfactually-augmented data. In *Proc. ICLR*.

Svetlana Kiritchenko and Saif Mohammad. 2018. Examining gender and race bias in two hundred sentiment analysis systems. In *Proc. *SEM*.

Pang Wei Koh and Percy Liang. 2017. Understanding black-box predictions via influence functions. In *Proc. ICML*.

Sachin Kumar, Shuly Wintner, Noah A Smith, and Yulia Tsvetkov. 2019. Topics to avoid: Demoting latent confounds in text classification. In *Proc. EMNLP*.

Klas Leino, Matt Fredrikson, Emily Black, Shayak Sen, and Anupam Datta. 2019. Feature-wise bias amplification. In *Proc. ICLR*.

Jiwei Li, Xinlei Chen, Eduard H. Hovy, and Dan Jurafsky. 2016. Visualizing and understanding neural models in NLP. In *Proc. NAACL-HLT*.

Zachary Chase Lipton. 2018. The mythos of model interpretability. *CACM*.

R. Thomas McCoy, Ellie Pavlick, and Tal Linzen. 2019. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proc. ACL*.

Pouya Pezeshkpour, Sarthak Jain, Sameer Singh, and Byron C. Wallace. 2021a. Combining feature and instance attribution to detect artifacts. *ArXiv*, abs/2107.00323.

Pouya Pezeshkpour, Sarthak Jain, Byron C. Wallace, and Sameer Singh. 2021b. An empirical comparison of instance attribution methods for NLP. In *Proc. NAACL-HLT*.

Danish Pruthi, Bhuwan Dhingra, Livio Baldini Soares, Michael Collins, Zachary C. Lipton, Graham Neubig, and William W. Cohen. 2020a. Evaluating explanations: How much do explanations from the teacher aid students? *ArXiv*, abs/2012.00893.

Garima Pruthi, Frederick Liu, Mukund Sundararajan, and Satyen Kale. 2020b. Estimating training data influence by tracking gradient descent. In *Proc. NeurIPS*.

Reid Pryzant, Kelly Shen, Dan Jurafsky, and Stefan Wagner. 2018. Deconfounded lexicon induction for interpretable social science. In *Proc. NAACL-HLT*.

Dheeraj Rajagopal, Vidhisha Balachandran, Eduard Hovy, and Yulia Tsvetkov. 2021. SelfExplain: A self-explaining architecture for neural text classifiers. In *EMNLP*.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "Why should I trust you?": Explaining the predictions of any classifier. In *Proc. KDD*.

Andrew Slavin Ross, Michael C. Hughes, and Finale Doshi-Velez. 2017. Right for the right reasons: Training differentiable models by constraining their explanations. In *IJCAI*.

Shiori Sagawa, Pang Wei Koh, Tatsunori B. Hashimoto, and Percy Liang. 2020a. Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization. In *Proc. ICLR*.

Shiori Sagawa, Aditi Raghunathan, Pang Wei Koh, and Percy Liang. 2020b. An investigation of why overparameterization exacerbates spurious correlations. In *Proc. ICML*.

Erik F. Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proc. CoNLL*.

Maarten Sap, Dallas Card, Saadia Gabriel, Yejin Choi, and Noah A. Smith. 2019. The risk of racial bias in hate speech detection. In *Proc. ACL*.

Burr Settles. 2011. Closing the loop: Fast, interactive semi-supervised annotation with queries on features and instances. In *EMNLP*.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2014. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *Proc. ICLR*.

Tony Sun, Andrew Gaut, Shirlyn Tang, Yuxin Huang, Mai ElSherief, Jieyu Zhao, Diba Mirza, Elizabeth Belding, Kai-Wei Chang, and William Yang Wang. 2019. Mitigating gender bias in natural language processing: Literature review. In *Proc. ACL*.

Zijie J. Wang, Dongjin Choi, Shenyu Xu, and Diyi Yang. 2021. Putting humans in the natural language processing loop: A survey. In *HCINLP*.

Alex Warstadt, Yian Zhang, Haau-Sing Li, Haokun Liu, and Samuel R. Bowman. 2020. Learning which features matter: Roberta acquires a preference for linguistic generalizations (eventually). In *Proc. EMNLP*.

Sarah Wiegreffe and Yuval Pinter. 2019. Attention is not not explanation. In *Proc. EMNLP*.

Qizhe Xie, Zihang Dai, Yulun Du, Eduard Hovy, and Graham Neubig. 2017. Controllable invariance through adversarial feature learning. In *Proc. NeurIPS*.

Yadollah Yaghoobzadeh, Soroush Mehri, R. Tachet, Timothy J. Hazen, and Alessandro Sordoni. 2021. Increasing robustness to spurious correlations using forgettable examples. In *EACL*.

Chih-Kuan Yeh, Joon Sik Kim, Ian En-Hsu Yen, and Pradeep Ravikumar. 2018. Representer point selection for explaining deep neural networks. In *Proc. NeurIPS*.

Ruiqi Zhong, Steven Shao, and Kathleen McKeown. 2019. Fine-grained sentiment analysis with faithful attention. *arXiv preprint arXiv:1908.06870*.

Ran Zmigrod, Sabrina J. Mielke, Hanna Wallach, and Ryan Cotterell. 2019. Counterfactual data augmentation for mitigating gender stereotypes in languages with rich morphology. In *Proc. ACL*.

## A Hyperparameters

For finetuning BERT, we use a learning rate of 2e-5 for both TextLen and MSGS. We tune 10 epochs for TextLen and 3 epochs for MSGS with a batch size of 64, resulting in around 250 steps for each dataset since the data size is also different.

For adversarial training, we use the same batch size of 64, but search the learning rate $\in$ {2e-5, 5e-5}, the number of training epochs $\in \{10, 20, 40\}$ for TextLen and $\in \{3, 6, 12\}$ for MSGS, and $\lambda \in \{0.1, 0.3, 1.0, 3.0\}$. For TextLen, the best hyperparameters are [5e-5, 20, 0.3]. For MSGS, the best hyperparameters are [5e-5, 12, 0.3].

For influence tuning and embedding tuning, we follow the vanilla finetuning and use the same learning rate of 2e-5, batch size of 64 and a total number of steps at around 250 for the regular label loss propagation steps. For the influence loss propagation steps, we search the tuning interval (i.e., how many label propagation steps needed before a round of influence propagation happens) $\in \{25, 50, 100\}$, number of epochs within one round of influence propagation $\in \{5, 10, 15\}$ for TextLen and $\in \{3\}$ for MSGS, batch size $\in \{4, 16, 64\}$, influence propagation optimizer's learning rate $\in \{$3e-5, 1e-5, 3e-6, 1e-6$\}$. The best hyperparameters for influence tuning are [50, 5, 64, 3e-5] for TextLen and [50, 3, 16, 3e-5] for MSGS. The best hyperparameters for embedding tuning are [50, 10, 64, 3e-5] for TextLen and [25, 3, 4, 1e-5] for MSGS. For TextLen, each influence tuning epoch we randomly sample 75 probing examples ($z_{\text{prb}}$) from the train set, and for each probing example we sample 5 positive and 5 negative train examples based on the confound information ($z_{A_i}$ and $z_{B_i}$). For MSGS, each influence tuning epoch we randomly sample 100 probing examples ($z_{\text{prb}}$) from the train set, and for each probing example we sample 3 positive and 3 negative train examples based on the confound information ($z_{A_i}$ and $z_{B_i}$). For all experiments with all the methods, we use 5 random seeds [2021, 2022, 2023, 2024, 2025].

## B Derivation of the influence gradients

To derive the gradient of the cosine influence, we first derive the gradient of the dot product influence:

$$\begin{aligned}
I =& \nabla_\theta L(x_1) \cdot \nabla_\theta L(x_2) \\
=& \frac{\partial L(x_1)}{\partial \theta_1}\frac{\partial L(x_2)}{\partial \theta_1} + \frac{\partial L(x_1)}{\partial \theta_2}\frac{\partial L(x_2)}{\partial \theta_2} + \ldots \\
\frac{\partial}{\partial \theta_1} I =& (\frac{\partial^2 L(x_1)}{\partial \theta_1^2}\frac{\partial L(x_2)}{\partial \theta_1} + \frac{\partial L(x_1)}{\partial \theta_1}\frac{\partial^2 L(x_2)}{\partial \theta_1^2}) \\
&+ (\frac{\partial^2 L(x_1)}{\partial \theta_1 \partial \theta_2}\frac{\partial L(x_2)}{\partial \theta_2} + \frac{\partial L(x_1)}{\partial \theta_2}\frac{\partial^2 L(x_2)}{\partial \theta_1 \partial \theta_2}) \\
&+ \ldots \\
=& H_{L1}[\text{row 1}] \cdot \nabla_\theta L(x_2) \\
&+ H_{L2}[\text{row 1}] \cdot \nabla_\theta L(x_1) \\
\ldots& \\
\nabla_\theta I =& H_{L1}\nabla_\theta L(x_2) + H_{L2}\nabla_\theta L(x_1)
\end{aligned}$$

Next we derive the gradient of the full cosine influence:

$$\begin{aligned}
I =& \frac{\nabla_\theta L(x_1) \cdot \nabla_\theta L(x_2)}{\|\nabla_\theta L(x_1)\| \, \|\nabla_\theta L(x_2)\|} = \frac{i(\theta)}{m(\theta)} \\
\nabla_\theta I =& \frac{m(\theta)\nabla_\theta i(\theta) - i(\theta)\nabla_\theta m(\theta)}{m(\theta)^2}
\end{aligned}$$

We already know the gradient of $i(\theta)$, so we are only interested in $\nabla_\theta m(\theta)$. We first calculate $\nabla_\theta \|\nabla_\theta L(x_1)\|$ and $\nabla_\theta \|\nabla_\theta L(x_2)\|$, and then apply product rule to combine:

$$\begin{aligned}
&\frac{\partial}{\partial \theta_1}\|\nabla_\theta L(x_1)\| \\
&= \frac{\partial}{\partial \theta_1}((\frac{\partial}{\partial \theta_1}L(x_1))^2 + (\frac{\partial}{\partial \theta_2}L(x_1))^2 + \ldots)^{1/2} \\
&= \frac{1}{2\,\|\nabla_\theta L(x_1)\|}(\frac{\partial}{\partial \theta_1}(\frac{\partial}{\partial \theta_1}L(x_1))^2 + \frac{\partial}{\partial \theta_1}(\frac{\partial}{\partial \theta_2}L(x_1))^2 \\
&\qquad + \ldots) \\
&= \frac{1}{2\,\|\nabla_\theta L(x_1)\|}(2\frac{\partial}{\partial \theta_1}L(x_1)\frac{\partial^2}{\partial \theta_1^2}L(x_1) + \\
&\qquad 2\frac{\partial}{\partial \theta_2}L(x_1)\frac{\partial}{\partial \theta_1}\frac{\partial}{\partial \theta_2}L(x_1) + \ldots) \\
&= \frac{1}{\|\nabla_\theta L(x_1)\|}(H_{L1}[\text{row 1}] \cdot \nabla_\theta L(x_1)) \\
&\ldots \\
&\nabla_\theta \|\nabla_\theta L(x_1)\| \\
&= \frac{1}{\|\nabla_\theta L(x_1)\|}H_{L1}\nabla_\theta L(x_1) \\
&\nabla_\theta \|\nabla_\theta L(x_2)\| \\
&= \frac{1}{\|\nabla_\theta L(x_2)\|}H_{L2}\nabla_\theta L(x_2) \\
&\nabla_\theta m(\theta) \\
&= \frac{\|\nabla_\theta L(x_2)\|}{\|\nabla_\theta L(x_1)\|}H_{L1}\nabla_\theta L(x_1) \\
&\quad + \frac{\|\nabla_\theta L(x_1)\|}{\|\nabla_\theta L(x_2)\|}H_{L2}\nabla_\theta L(x_2)
\end{aligned}$$

Combining the above terms, we have:

$$\begin{aligned}
\nabla_\theta I =& \frac{\|\nabla_\theta L(x_1)\| \, \|\nabla_\theta L(x_2)\|(H_{L1}\nabla_\theta L(x_2) + H_{L2}\nabla_\theta L(x_1))}{\|\nabla_\theta L(x_1)\|^2 \, \|\nabla_\theta L(x_2)\|^2} \\
&- \frac{(\nabla_\theta L(x_1) \cdot \nabla_\theta L(x_2))(\frac{\|\nabla_\theta L(x_2)\|}{\|\nabla_\theta L(x_1)\|}H_{L1}\nabla_\theta L(x_1))}{\|\nabla_\theta L(x_1)\|^2 \, \|\nabla_\theta L(x_2)\|^2} \\
&- \frac{(\nabla_\theta L(x_1) \cdot \nabla_\theta L(x_2))(\frac{\|\nabla_\theta L(x_1)\|}{\|\nabla_\theta L(x_2)\|}H_{L2}\nabla_\theta L(x_2))}{\|\nabla_\theta L(x_1)\|^2 \, \|\nabla_\theta L(x_2)\|^2}
\end{aligned}$$

For clarity, we rearrange the equation as below:

$$\begin{aligned}
\nabla_\theta I =& \mathbf{p} + \mathbf{q} - \mathbf{r} - \mathbf{s} \\
\mathbf{p} =& \frac{1}{\|\nabla_\theta L(x_1)\| \, \|\nabla_\theta L(x_2)\|}H_{L1}\nabla_\theta L(x_2) \\
\mathbf{q} =& \frac{1}{\|\nabla_\theta L(x_1)\| \, \|\nabla_\theta L(x_2)\|}H_{L2}\nabla_\theta L(x_1) \\
\mathbf{r} =& \frac{\nabla_\theta L(x_1) \cdot \nabla_\theta L(x_2)}{\|\nabla_\theta L(x_1)\|^3 \, \|\nabla_\theta L(x_2)\|}H_{L1}\nabla_\theta L(x_1) \\
\mathbf{s} =& \frac{\nabla_\theta L(x_1) \cdot \nabla_\theta L(x_2)}{\|\nabla_\theta L(x_1)\| \, \|\nabla_\theta L(x_2)\|^3}H_{L2}\nabla_\theta L(x_2)
\end{aligned}$$