Parameter Adaptation for Joint Distribution Shifts

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

Abstract
While different methods exist to tackle distinct types of distribution shift, such as label shift (in the form of adversarial attacks) or domain shift, tackling the joint shift setting is still an open problem. Through the study of a joint distribution shift manifesting both adversarial and domain-specific perturbations, we not only show that a joint shift worsens model performance compared to their individual shifts, but that the use of a similar domain worsens performance than a dissimilar domain. To curb the performance drop, we study the use of perturbation sets motivated by input and parameter space bounds, and adopt a meta learning strategy (hypernetworks) to model parameters w.r.t. test-time inputs to recover performance.

1 Introduction
The problem of distribution shift, the divergence between the train-time and test-time distributions, manifests in different forms in NLP. Robustness against adversarial attacks, a form of label shift, have resulted in defenses such as such as adversarial training (Goodfellow et al., 2014; Madry et al., 2017), defensive distillation (Papernot et al., 2016b, 2017), denoising (Liao et al., 2018), many of which return parameters that are static at test-time. Robustness against domain shifts has resulted in various methods of adaptation from a source distribution to a target distribution (Liu et al., 2019; Ziser and Reichart, 2019; Cui et al., 2017).

In particular, we are motivated to study the problem of joint distribution shifts, where a target distribution can manifest perturbations from multiple sources distributions simultaneously. Given the bounded input space and non-conformity of perturbation sources, we evaluate a parameter adaptation strategy based in meta learning, where adapted parameters are mapped for distant tasks/distributions in the input space. We use hypernetworks (Ha et al., 2016) trained on perturbation sets as an interpolating input-parameter adaptation model.

Contributions We validate these two hypothesis for robustness to joint distributional shift in NLP:
- Model performance weakens when a test-time input manifests joint distribution shift, evaluated on adversarial and domain-specific perturbations. The performance worsens when a similar domain is used.
- A meta learner adapting the model parameters with respect to perturbed inputs can retain robust performance, evaluated with perturbation sets and hypernetworks.

2 Problem & Preliminaries
Terminology and notation is defined and used progressively through this section and onto the rest of the paper.

Proposition 1 Distribution shifts, manifesting as perturbations (in this case, adversarial and domain-specific), in NLP are bounded within a finite dictionary or embedding space. Any shifted distribution $X_j$ is located at a bounded distance from an (origin) source distribution $X_i$.

A distribution shift is denoted as the divergence between a source (train-time) and target (test-time) distribution. The cause for shift can vary by source of distribution (e.g. domain shift, task shift, label shift) and variations per source (e.g. multiple backdoor triggers, multiple domains). A joint distribution shift is denoted distribution shift attributed to multiple sources and/or variations per source; this is in contrast to disjoint distribution shift, attributed to a single source and variation of shift.

We denote a token $x$, which is an index mapped to a word/character (word, in our evaluation) in a
I purchased this toy for my son when he was 4 months old. At first, he seemed a little intimidated by the toys. 

I obtained this toy for my children when he was 4 weeks senior. At first, he hoped a modest harassed by the toy.

It felt like a big commitment for me to have to run the program 2 times a day, and near the end of my pregnancy I was annoyed with having anything strapped across my belly. 

It felt like a big committed for me to have to run the program 2 length a day, and near the end of my pregnancy I was annoyed with takes anything strapped across my belly.

This dvd gives a very good 60 minute workout. As others have pointed out the cardio is very dancy. The first time I did it, I felt a bit awkward with the steps. 

This dvd gives a awfully okay 60 minute exercise. As others have pointed out the cardio is very dancy. The first time I did it, I perceived a bit awkward with the steps.

| Source domain: baby, Shifted domain: books |
|-------------------------------------------|
| Original sentence (Actual label: Pos)     |
| I purchased this toy for my son when he was 4 months old. At first, he seemed a little intimidated by the toys. |
| Adversarial sentence                      |
| I obtained this toy for my children when he was 4 weeks senior. At first, he hoped a modest harassed by the toy. |
| Original sentence (Actual label: Pos)     |
| It felt like a big commitment for me to have to run the program 2 times a day, and near the end of my pregnancy I was annoyed with having anything strapped across my belly. |
| Adversarial sentence                      |
| It felt like a big committed for me to have to run the program 2 length a day, and near the end of my pregnancy I was annoyed with takes anything strapped across my belly. |

| Source domain: dvd, Shifted domain: baby |
|-----------------------------------------|
| Original sentence (Actual label: Pos)     |
| Fast times at ridgemont high is a clever, insightful, and wicked film! It is not just another teen movie. |
| Adversarial sentence                      |
| Sooner days at ridgement high is a sane, thoughtful, and wicked flick! It is not just another adolescent flick. |
| Original sentence (Actual label: Pos)     |
| This dvd gives a very good 60 minute workout. As others have pointed out the cardio is very dancy. The first time I did it, I felt a bit awkward with the steps. |
| Adversarial sentence                      |
| This dvd gives a awfully okay 60 minute exercise. As others have pointed out the cardio is very dancy. The first time I did it, I perceived a bit awkward with the steps. |

Table 1: Shifted inputs: Adversarial sentences shifted with respect to adversarial perturbations and a shifted domain are evaluated by a model trained on a source domain. Perturbations are in red, and prediction confidence in brackets.

| Target Domain | book | magazine | baby |
|---------------|------|----------|------|
| Acc(θ; X, Y)  | 0.880| 0.960    | 0.890|
| Acc(θ; Adv(θ; X), Y) | 0.525| 0.570    | 0.632|

| Attack Domain | magazine | book | dvd | baby | dvc | book | dvc | book | magazine |
|---------------|----------|------|-----|------|-----|------|-----|------|----------|
| Acc(θ; X, Y)  | 0.745    | 0.726| 0.646| 0.673| 0.663| 0.739| 0.652| 0.624| 0.665    |
| Acc(θ; Adv(θ; X), Y) | 0.395| 0.398| 0.421| 0.343| 0.366| 0.381| 0.386| 0.365| 0.401    |
| SharedVocab   | 0.455    | 0.381| 0.255| 0.381| 0.345| 0.260| 0.255| 0.270| 0.260    |
| Transfer Loss | 0.000    | 0.017| 0.071| 0.010| 0.022| 0.079| 0.050| 0.066| 0.069    |

Table 2: Domain shift & similarity: Sorted in descending order of domain similarity, we observe a lower after-attack accuracy when domain similarity increases. Original accuracy $\text{Acc}(\theta; X_i, Y)$ denotes the accuracy of the defender’s model $f(\theta_i)$ on the source distribution (test) data $X_i$. Intra-attack accuracy $\text{Acc}(\theta; Adv(\theta; X_i), Y)$ denotes the accuracy of the defender’s model $f(\theta_i)$ on adversarially-perturbed source distribution (test) data $Adv(\theta; x_i)$. Unperturbed accuracy $\text{Acc}(\theta; X_j, Y)$ denotes the accuracy of the defender’s model $f(\theta_i)$ on the domain-shifted (non-adversarial) distribution (test) data $X_j$. After-attack accuracy $\text{Acc}(\theta; Adv(\theta; X_j), Y)$ denotes the accuracy of the defender’s model $f(\theta_i)$ on the joint shifted (adversarial+domain) distribution (test) data $Adv(\theta; x_j)$.

finite and discrete dictionary (or embedding space) $\mathcal{K}$ where $|\mathcal{K}|=k$ (GloVe (Pennington et al., 2014), in our evaluation). A sentence is an $N$-token sequence $x = \{x\}^N$. A dataset containing $M$ labelled sequences is constructed as $X = \{x\}^M$ and mapped to their corresponding indexed labels $x \mapsto y, X \mapsto Y$.

We denote $\mathcal{X}$ as the input space. In the NLP setting, as $\mathcal{K}$ is discrete and finite, the maximum distance between any 2 arbitrary tokens (diameter of $\mathcal{X}$) approximates the bound in our evaluation $d_{\text{max}} := \max_{x_i, x_j \sim \mathcal{X}} \text{dist}(x_i, x_j)$. We denote a generic distance metric $\text{dist}$, and the properties of the metric should be inferred from the input arguments. To retain generality, we refer to the input (label) distribution as its corresponding input (label) space. A distribution shift $X_i \rightarrow X_j$ would be measured as the relative distance between the 2 subspaces $\text{dist}(X_i, X_j)$. Any point in these subspaces reside in $\mathcal{X}$ where $X_i, X_j \subseteq \mathcal{X}$, hence the distribution shift is also bounded $\text{dist}(X_i, X_j) \leq d_{\text{max}}$. A perturbation $\varepsilon_n$ is a change in value of the token $x$ w.r.t. $\mathcal{K}$ at the $n$th position of the sequence such that $x_j = \{x_i\}^n \cup \{x_i + \varepsilon_n\}^n$. We denote $\varepsilon$ as a vector composed of elements from set $\{0, \varepsilon_n\}$ such that $x_j = x_i + \varepsilon$, where the position $n$ is computed by $\text{Adv}$ (Sec 5). Perturbations can manifest as adversarial perturbations and/or domain-specific perturbations. As perturbations can cause an input point to shift from one sub-
space to another subspace, we compose distribution shifts in terms of perturbations, and transitive to shifts, perturbations are also bounded within $\mathcal{K}$ and $\mathcal{X}$. We extend on how distribution shifts can be manifested from domain shift and adversarial perturbations in Appendix: Sec8.1.

Suppose $f_i$ is the ground-truth function that defines the source distribution (subspace) of index $i$ mapping to the label distribution $\mathcal{Y}_i \mapsto \mathcal{X}_i$ such that $y = f_i(\{x\}_x \sim \mathcal{P}(\mathcal{X},N))$, where each token $x$ is sampled from $\mathcal{K}$ to form a sequence of length $N$ and returns ground-truth label $y$. $f$ is a function approximating $f_i$, accepting the input sequence $x$ and model parameters (approximating $f_i$) $\theta_i$ as arguments to return the predicted label $\hat{y} = f(\theta_i; x)$. $f$ is a generalized function, specifically a deep neural network of fixed architecture with respect to varying $\mathcal{X}$, and adaptation to $f$ with respect to varying $f_i$ is through varying $\theta_i$. $\theta_i$ is obtained through Stochastic Gradient Descent by minimizing the loss between the actual and approximated functions through the enumeration of samples from a dataset $\{x\}_x \sim \mathcal{X}_i$ such that $\theta_{i,t} := \theta_{i,t-1} - \sum_{x \sim \mathcal{X}_i} \partial \mathcal{L}(x, y) / \partial \theta_i$. As validated in Galanti and Wolf (2020), we retain the assumption that a mapped $f(\theta; X)$ exists for any sampled distribution $X \sim \mathcal{X}$.

Moreover, $f_i \mapsto \mathcal{P}_1(\mathcal{K}, N)$ the function is mapped to its corresponding probability density function, which is used for sampling tokens to construct a sequence from distribution $\mathcal{X}_i \mapsto \mathcal{P}_1(\mathcal{K}, N)$. $\mathcal{P}_1(\mathcal{K}, N)$ maps each $x$ in $\mathcal{K}$ to the probability of occurrence of token $x_n$ at the $n$th position $(n \in N)$ of a sequence in source distribution $\mathcal{X}_i$, i.e. $\{\mathcal{K}, N\} \mapsto \mathcal{P}_1(\mathcal{K}, N)$. This could also be approximated as obtaining the hidden state representations or activations returned at the $t$th layer of a model $f(\cdot; \theta)$ (e.g. BERT (Devlin et al., 2019)); $\mathcal{P}_1 \approx h_{t} = f_{t}(\theta; \cdot)$.

**Proposition 2** A distribution shift in the datasets $\mathcal{X}_i \mapsto \mathcal{X}_j$ in a bounded input space can be approximated by the distance between their optimized parameters $\theta_i \rightarrow \theta_j$ and difference in optimization trajectories per epoch.

Suppose we train models optimized for different shifted distributions starting from a constant, shared initialization (e.g. 1 random $\theta$, 1 pre-trained $\theta$). For 2 independent parameter optimization processes of distributions $\mathcal{X}_i$ and $\mathcal{X}_j$, in order for $\theta_i$ and $\theta_j$ to reside in a local subspace in the parameter space (i.e. minimize distance between parameters $\text{dist}(\theta_i, \theta_j) \approx 0$), the difference in gradient updates across their training epochs (assuming the same total epoch count $E$) should be minimal, where

$$\text{dist}(\theta_i, \theta_j) = \sum_{e} \left| \frac{\partial \mathcal{L}(x_i, y_i)}{\partial \theta_i,e} - \frac{\partial \mathcal{L}(x_j, y_j)}{\partial \theta_j,e} \right|$$

Measurements for $\text{dist}(\theta_i, \theta_j)$ include cosine distance (Wortsman et al., 2021), centered kernel alignment (Kornblith et al., 2019), set difference in subnetworks (Datta and Shadbolt, 2022b). We formulate the parameter distance as the distance in successive gradient updates based on a set of observations (not intended to be mutually-exclusive).

$E \in [0, E]$ is an arbitrary epoch.

**Observation 1** $\theta_i$ and $\theta_j$ converge (diverge) if their training distributions contain transferable (interfering) features.

**Observation 2** $\theta_i$ and $\theta_j$ may diverge, attributing to the presence of highly-contextual/semantic features (to optimize for feature density), or robust features (to optimize for feature sparsity).

From Observations 1 and 2 (details in Appendix: Sec8.1), we intend to make it clear that parameter adaptation does not mean a linear change in $\text{dist}(\mathcal{X}_i, \mathcal{X}_j)$ will map to a specific pattern of parameter change (e.g. feature sparsity/density, transfer/interference). Conversely, a linear change in adversarial perturbation factor $\varepsilon$ and/or domain similarity may not map to a linear change in distance in the input space $\text{dist}(\mathcal{X}_i, \mathcal{X}_j)$. As the mapping between the input-output spaces $\mathcal{X} \rightarrow \Theta$ may not be linearly-interpolable, the adaption function should be a non-linear function (e.g. hypernetwork). As the practical objective is to compute $\text{dist}(\theta_i, \theta_j)$ in order to compute $y$, we can inversely measure $\text{dist}(\mathcal{X}_i, \mathcal{X}_j)$ w.r.t. $\text{dist}(\theta_i, \theta_j)$ to simplify further analysis. We approximate the distance between 2 distributions in a bounded input space by the distance between their optimized parameters trained on a constant initialization. As no corresponding ground-truth input distribution may exist for the constant random initialization, the mapped origins for the input space $\mathcal{X}$ and parameter space $\Theta$ are the source distribution $\mathcal{X}_i$ and its mapped parameters $\theta_i$, respectively.

$$\text{dist}(\mathcal{X}_i, \mathcal{X}_j) \propto \text{dist}(\theta_i, \theta_j)$$

$$E \sum_{e} \left| \frac{\partial \mathcal{L}(x_i, y_i)}{\partial \theta_i,e} - \frac{\partial \mathcal{L}(x_j, y_j)}{\partial \theta_j,e} \right|$$
3 Adaptation to Joint Distribution Shifts

**Proposition 3** Suppose the $(\ell - 1)$th layer in an $\ell$-layer neural network $f$ is the layer returning class probabilities such that $f_{\ell-1}(\theta; x) = \{y : \rho\}$, and perturbations per sequence $\xi = \sum_{\lambda}^\Lambda \varepsilon_\lambda$, where $\Lambda$ are different sources/variations of shift. To mitigate the increase in error attributed to $\arg \max_{y \sim Y} f_{\ell-1}(\theta; \xi)$, for a shifted input $x_j$, we may adapt the parameter $\theta_i \rightarrow \theta_j$ to converge class probabilities w.r.t. $\xi$ to 0, where

$$\begin{align*}
    f_{\ell-1}(\theta_j; \xi) &= \sum_\Lambda \{y_\lambda : \rho_\lambda\} \\
    f_{\ell-1}(\theta_i; \xi) &= \sum_\Lambda \{y_\lambda : 0\}
\end{align*}$$

We provide details of Proposition 3 in Appendix: Sec8.1. Thus from Proposition 3, the objective of this work is to propose an adaptation technique against joint distribution shifts. We evaluate Hypotheses 1 and 2, the former proposing a suitable joint distribution setting, the latter proposing an adaptation method to tackle the scenario in Hypothesis 1. We describe the evaluation for Hypothesis 2 in Sec5. The hypotheses are affirmative: their intention is to illustrate the theoretical grounding for which the subsequent empirical observations will validate.

**Hypothesis 1** It is hypothesized that a perturbed sample manifesting joint distribution shift, in particular adversarial shift and domain shift (worse, similar domain shift), the model’s accuracy w.r.t. the perturbed input would be lower.

**(Hypothesis 1a)** It is hypothesized that an input manifesting joint distribution shift will yield lower model accuracy on shifted inputs. At train-time, we presume a model is trained on a source dataset $X_i \rightarrow Y_i$. At test-time, we presume a model encounters a sample that is joint-distributionally-shifted from $X_i$: we adopt 2 sources of perturbations $\xi$, (i) adversarial perturbations $\varepsilon_{\text{adv}}$, and (ii) domain shift $\varepsilon_{\text{domain}}$ from dataset $X_j \rightarrow Y_j$. Proposition 1 implies that despite the insertion of perturbations, the shifted distribution is a bounded distance from the source training distribution. However, Proposition 2 also indicates that trajectory-changing perturbations, such as the replacement of semantic/contextual structure through domain shift, will result in different optimal model parameters, implying an increased distance between the 2 distributions in the input space. If trajectory-changing perturbations manifest, then passing $X_j$ through a model with $\theta_i$ will likely result in the label-shifting case in Eqt 10, and thus reduce model accuracy.

**(Hypothesis 1b)** It is hypothesized that an adversarially-perturbed input originating from a similar domain to the training domain will return a lower accuracy on perturbed inputs than that of a dissimilar domain, for a small number of labels in $Y$ such that $1/|Y| \rightarrow 0$. Extending on Hypothesis 1a, we further hypothesize that adversarial inputs generated from a similar domain to the training distribution will result in a lower accuracy w.r.t. perturbed inputs than a dissimilar domain. Intuitively, one would expect a dissimilar domain to return a lower accuracy w.r.t. perturbed inputs, given the expectedly larger change in semantic structure, i.e. $\text{dist}(X_i, X_j) \uparrow \infty$. This hypothesis would be supported by literature in out-of-distribution robustness (Hendrycks and Gimpel, 2016).

Based on Proposition 2, $\theta_i$ and $\theta_j$ may be more distant w.r.t. dissimilar domains than similar domains. Though $\theta_i$ and $\theta_j$ are distant, the distance between parameters do not equate to reduction in accuracy. Theorem 1 (Appendix: Sec8.1) shows that when a test-time distribution is too distant from the source distribution (e.g. test-time distribution is random), then the predicted labels tend to uniformly sample the label distribution: $f_{\ell-1}(\theta_j; x_j) \approx \{y : 1/|Y|\}$. The low expected variance of the class probabilities $\sigma^2(\{y : 1/|Y|\}) \approx 0$ would suggest the expected accuracy w.r.t. perturbed inputs would also be $1/|Y|$. We assume the label set is finite and of small count, i.e. $\sum_{\{y : 1/|Y|\}} \approx 0$. For a small number of labels, we would expect an adversarially-perturbed input from a similar domain to return class probabilities skewed away from the ground-truth label $\max_{\text{dist}(f_{\ell-1}(\theta; x_{\text{similar}}), f_{\ell-1}(\theta; x_{\text{dissimilar}}))} = \text{dist}(\{y : \rho_{\text{similar}}\}, \{y : \rho_{\text{dissimilar}}\})$ while also retaining non-zero variance $\sigma^2_{\text{similar}}(\{y : 1/|Y|\}) > \sigma^2_{\text{dissimilar}}(\{y : 1/|Y|\})$. For a similar domain distribution that approximates or is near the source distribution in the input space with correspondingly low parameter distance (Proposition 2), as the parameters $\theta_i \approx \theta_j$, it follows that a (gradient-based) adversarial attack algorithm will be able to generate perturbations with respect to a close approximation of the parameters of the source distribution. For a large number of labels where $|Y| \rightarrow \infty$, this hypothesis may not hold, and a distant (dissimilar) distribution would attain an accuracy w.r.t. perturbed inputs of 0. We formalize this result in Theorem 2 (Appendix: Sec8.1).
Evaluation Setup We share the experimental details we use to evaluate Hypothesis 1; these configurations are retained in evaluating Hypothesis 2 in Sec5.

(Adversarial attack algorithm) We use the baseline gradient-based adversarial attack algorithm Fast Gradient Sign Method (FGSM) (Goodfellow et al., 2014). We denote $\text{Adv}(\theta; x; \epsilon)$ as the adversarial attack method, with a given perturbation rate $\epsilon = 0.4$. The goal of $\text{Adv}$ is to maximize the misclassification rate on perturbed inputs ($\max \text{ Eqt 3}$: $x^{\text{adv}} = \text{Adv}(\theta, x)$ s.t. $y \neq f(\theta; x^{\text{adv}})$. Details are in Appendix: Sec8.2.

\[ \mathbb{E}_{x_i, y \sim X_j, y_j}[f(\theta; \text{Adv}(\theta_j; x_j)) - y] \tag{3} \]

(Joint Distribution Shift) For a (different, attack) domain sampled from set of natural domains $X_j \sim X$, the attacker’s objective is to maximize the misclassification per label and minimize the accuracy w.r.t. perturbed inputs ($\max$ Eqt 3), while the defender’s objective is to maximize the accuracy w.r.t. perturbed inputs ($\min$ Eqt 3). 1 The defender trains their model of parameters $\theta_i$ on source dataset $X_i$. 2 The attacker collects their own dataset $X_j$ and trains a surrogate/substitute model to obtain parameters $\theta_j$. 3 To generate perturbed test-time inputs, the attacker first evaluates $f(\theta_j; x_j)$ on unseen, test data $x_j \sim X_j$ to filter for correctly-classified instances, then perturbs this data with a chosen adversarial attack algorithm to generate adversarial samples such that the predicted target label is $-y$: $x_j^{\text{adv}} = \text{Adv}(\theta_j; x_j) \equiv y$.

(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) (default configuration, unless otherwise specified), GRU (BERT (Devlin et al., 2019), CNN (Kim, 2014), and Logistic Regression (Maas et al., 2011).

(Metrics) We denote $\text{Acc}(\theta; X, Y)$ to measure the classification accuracy of base learner $f(\theta; X)$ such that $\text{Acc}(\theta; X, Y) = \frac{1}{|Y|} \sum_{x,y} [1 - \text{sign}(f(\theta; x) - y)]$. We utilize SharedVocab and Transfer Loss as distance metrics, as they manifest attributes recurring throughout the paper, such as inspecting similarities in input distributions w.r.t. tokens/dictionary, connecting distance between distributions w.r.t. cross-domain parameter efficacy, utilized as ablation defense, and utilized in shifted inputs generation. 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. Transfer Loss $t_f$ (Blitzer et al., 2007; Glorot et al., 2011) measures domain similarity by computing the difference between the test error (evaluating $X_j$ on a model $f(\theta_i)$) and the baseline error (evaluating $X_i$ on a model $f(\theta_i)$): $t_f(X_i, X_j) = e(X_i, \theta_i) - e(X_i, \theta_i)$ where $e(X, \theta) = \frac{1}{M} \sum_{m=1}^{M} [f(\theta; x_m) - y]^2$. A lower transfer loss indicates higher similarity.

(Evaluating Hypothesis 1 (Table 2)) We observe a significant gap between original accuracy and after-attack accuracy between different domain pairs. The after-attack accuracy is worse than both the intra-attack accuracy and unperturbed accuracy, indicating a joint shift worsens accuracy w.r.t. perturbed inputs than each individual shift separately (validating Hypothesis 1a). Moreover, we observe a positive correlation between transfer loss and after-attack accuracy, and a negative correlation between SharedVocab and after-attack accuracy (albeit a low variance of distance). Both indicate a joint shift manifesting similar domains lowers the accuracy further than that of dissimilar domains (validating Hypothesis 1b).

Table 2 also validates Proposition 1. The SharedVocab metric is high amongst all the domain pairs, and the variance between pairs is low. The transfer loss across all domain pairs are low and below 0.1, also with minimal variance. This empirically supports the notion that despite defined as being different domains of varying similarity, there is a common input space where all these distributions reside, and the distance between them is interpretatively low.

(Hypothesis 2) It is hypothesized the existence of a meta-learner that can compute high-accuracy parameters for inputs sampled from a joint-distributionally-shifted source distribution, where the perturbation sources are adversarial and domain-specific, and the input space is bounded; i.e. $\text{dist}(x_i, x_j) \mapsto \theta_i + \Delta \theta$. 
Algorithm 1: Perturbation Sets Generation

PerturbationSet(\(D; \theta_i; T; R; \text{dist}, d_{\text{max}}; \varepsilon, \gamma\))

Input: dataset \(D = \{X_i : Y_i\}\), parameters \(\theta_i\); number of perturbation sets \(T = 10\), max iterations \(R = 10\); distance metric \(\text{dist} = t_f(X_i, X_j)\), max distance \(d_{\text{max}} = 0.1\); initial perturbation rate \(\varepsilon = 0.9\), perturbation learning rate \(\gamma = 0.05\);

Output: set \(S\) containing \(T\) perturbation sets

1. Initialize empty \(S\) to store perturbation sets \(S_t\).
2. \(S_t \leftarrow \emptyset\);
3. while \(t < T\) do
   4. Run next iteration \(r\) until \(S_t\) meets conditions.
   5. for \(r \leftarrow 0\) to \(R\) do
      6. Apply adversarial perturbations to \(X\).
      7. \(S_{t,r} \leftarrow \text{Adversarial}(\theta_i; X_t; \varepsilon)\);
     8. Evaluate distance conditions.
      9. if \(\text{dist}(S_{t,r}, X_i) \leq d_{\text{max}}\) then
         10. if \(\sigma^2(S \cup S_{t,r}) > \sigma^2(S)\) then
            11. \(S \leftarrow \{S_{t,r} : Y_i\}\);
               12. continue;
      13. else
         14. Adjust hyperparameters.
         15. \(\varepsilon \leftarrow \varepsilon - \gamma\);
      16. \(t \leftarrow t + 1;\)
   17. return \(S\)

Algorithm 2: Hypernetwork: Training

\(\text{train}(S, D, \theta_i, E^f, E^{m_f})\)

Input: perturbation sets \(S\), dataset \(D = \{X_i : Y_i\}\), model parameters \(\theta_i\), epochs \(E^f\) & \(E^{m_f}\)

Output: meta learner parameters \(\theta^{m_f}\)

1. Initialize empty set \(\Theta\) to store parameter differential.
2. \(\Theta \leftarrow \emptyset\);
3. Compute \(X_j \to \Delta \theta\).
4. foreach \(X_j : Y_j \in (\{X_i : Y_i\} \cup S)\) do
   5. for \(e \leftarrow 0\) to \(E^f\) do
      6. \(\theta_{f_j}^e \leftarrow \theta_{f_j}^{e-1} - \sum X_j Y_j \partial\mathcal{L}(x, y) \partial \theta_{f_j}\);
      7. \(\Delta \theta \leftarrow \theta_{f_j}^e - \theta_{f_j};\)
      8. \(\Theta \leftarrow \Delta \theta;\)
6. Compute \(\theta^{m_f}\).
7. for \(e \leftarrow 0\) to \(E^{m_f}\) do
   8. \(\theta_{m_f}^{e-1} \leftarrow \sum X_j \partial \mathcal{L}(X_j, \Delta \theta) \partial \theta_{m_f}^{e-1}\);
9. return \(\theta^{m_f}\)

Algorithm 3: Hypernetwork: Inference

\(\text{inference}(X_j, \hat{h}(\theta^{m_f}), f(\theta_i))\)

Input: test-time inputs \(X_j\); meta learner \(\hat{h}(\theta^{m_f})\); base learner \(f(\theta_i)\)

Output: label \(\hat{y}\)

1. Compute parameter differential w.r.t. \(X_j\), \(\Delta \hat{h} \leftarrow \hat{h}(\theta^{m_f}, X_j)\)
2. Update \(\theta_i\).
3. \(\hat{y} \leftarrow f(\theta_i + \Delta \theta; X_j)\)
4. return \(\hat{y}\)

Proposition 2 indicates that a functional relationship may exist between \(\text{dist}(X_i, X_j)\) and \(\text{dist}(\theta_i, \theta_j)\). Proposition 1 implies that, since \(\mathcal{X}\) is a bounded space, \(\text{dist}(X_i, X_j)\) and \(\text{dist}(\theta_i, \theta_j)\) are bounded as well. Proposition 2 notes observations where the adaptation in parameters may not follow a linear pattern of increasing sparsity/density, increasing transferable/interfering subnetworks, etc; thus a non-linear adaptation function should be used. We elaborate on general meta learner assumptions in Appendix: Sec8.2 and specify our adaptation assumptions in Sec4.

4 Adaptation with Hypernetworks

Parameter Adaptation (Algorithms 2, 3) A hypernetwork \(\hat{h}(x, I) = f(x; m_f(\theta^{m_f}; I))\) is a pair of learners, the base learner \(f : \mathcal{X} \mapsto \mathcal{Y}\) and meta learner \(m_f : I \mapsto \Theta^{m_f}\), such that for the meta-data \(I\) of input \(x\) (where \(\mathcal{X} \mapsto I\)), \(m_f\) produces the base learner parameters \(\theta_I = m_f(\theta^{m_f}; I)\). The function \(m_f(\theta^{m_f}; I)\) takes a conditioning input \(I\) (e.g. meta-data, task header, few-shot samples, support set) to returns parameters \(\theta_I \in \Theta^{m_f}\) for \(f\). The meta learner parameters and base learner (of each respective distribution) parameters reside in their distinct parameter spaces \(\theta^{m_f} \in \Theta^{m_f}\) and \(\theta^f \in \Theta^{f}\). The learner \(f\) takes an input \(x\) and returns an output \(\hat{y} = f(\theta_i; x)\) that depends on both \(x\) and the task-specific input \(I\). In practice, \(m_f\) is a large neural network and \(f\) is a small neural network.

In our implementation, our hypernetwork is a sequence-to-sequence network (Sutskever et al., 2014), not using any conditioning input \(I\) (accepting \(x\) as sole input), computing the parameter change \(\Delta \theta\) to an origin parameter \(\theta_i\), and only has access to its source distribution (no other domains, including the attacker’s domain). Details are in Appendix: Sec8.2.

Our parameter adaptation architecture is implemented as follows. At train-time, \(\textbf{1.}\) \(T\) training sets perturbed with respect to the source distribution are generated (Algorithm 1). Their corresponding base learner parameters are computed. The parameter differential is computed \(\Delta \theta = \theta_j^f - \theta_i^f\) to obtain a meta-training set \(\{\{x_j : \theta_j^f - \theta_i\}\}^T\). \(\textbf{2.}\) \(m_f\) (Algorithm 2) optimizes its meta parameters \(\theta^{m_f}\) containing the source and perturbed distributions, and their corresponding predicted weight differentials. At test-time, \(\textbf{3.}\) \(m_f\) (Algorithm 3) computes the predicted weight dif-
ferential to update the base learner parameters, and return a prediction $\hat{y} = f(\theta_i + m f(x_j); x_j)$.

$$\text{dist}(x_i, x_j) = \sum_{n} P_i(x_n, K, N) \rightarrow \Delta \theta \quad (3)$$

$$\text{dist}(X_i, X_j) = \sum_{m} \sum_{n} P_i(x_{nm}, K, N) \rightarrow \Delta \theta \quad (4)$$

**Perturbation Sets (Algorithm 1)** Extending on Proposition 1, while Eqn 6 computed $\text{dist}(X_i, X_j)$ with respect to the distance between tokens in $K$, we can alternatively measure the distance in the total likelihood that each token exists in its $n$th index of a sequence $\sum_{n} P_i$ (Eqn 4).

This motivates our construction of a perturbation set for use in constructing robust models. A **perturbation set** is a set containing subsets of perturbed inputs. These perturbations may be generated with respect to word substitutions (Alzantot et al., 2018; Jia et al., 2019), Wasserstein balls (Wong et al., 2019), or distribution shifts (Sinha et al., 2018; Sagawa et al., 2020). The optimal $\theta^{mf}$ is required to adapt $\theta^f$ across varying $\sum_{n} P_i$. Hence, we are motivated to generate $T$ perturbation sets of diverging $\sum_{n} P_i$. An average $\theta^f$ (computed with static-adaptation methods) may not return high-accuracy across every point in the distribution of $\sum_{n} P_i$, further motivating the need for dynamic adaptation. We pursue the following implementation (Algorithm 1) to (i) generate $T$ training sets of diverging $\sum_{n} P_i$ (representing varying ambiguities) for a hypernetwork to adapt $\theta^f$ towards; (ii) avoid averaging randomly-generated $\sum_{n} P_i$ to a single static parameter; (iii) sample shifted distributions that retain sufficient similarities to the source distribution (high dissimilarity returns random labels (Thm 1)). We provide extended detail in Appendix: Sec8.2.

To construct one perturbation set (Eqn 5), 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. To retain the relational property in Eqn 1, an optimal distance metric would be transfer loss. In-line with the rest of the paper, we retain FGSM as the adversarial attack algorithm. The procedure for iterative perturbations are in-line with BIM (Kurakin et al., 2016). Iteratively evaluating perturbations to approximately invert a distance function $tf$ is in-line with Papernot et al. (2016a)’s inversion of $\frac{\partial f(\theta; x)}{\partial x}$.

$$X^* := \min_{\varepsilon \sim [\varepsilon_{\text{min}}, 1]} \text{dist}(X^*, X_i) \leq d_{\text{max}} \quad (5)$$

$$X^* := \min_{\varepsilon \sim [\varepsilon_{\text{min}}, 1]} \arg \max \text{dist}(\text{Adv}(\varepsilon, X_i), X_i) \quad (5)$$

$$X^* := \min_{\varepsilon \sim [\varepsilon_{\text{min}}, 1]} \arg \max [\varepsilon(\text{Adv}(\varepsilon, X_i), \theta_i) - v(X_i, \theta_i)]$$

## 5 Evaluation

### Evaluation Setup

We share additional experimental details to evaluate Hypothesis 2.

**Baselines & Ablations** We compare against 2 empirically-effective defenses. The implementation structure of **defensive distillation** (Papernot et al., 2016b, 2017) 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) shows that injecting adversarial examples throughout training increases the robustness of target neural network models. In this baseline, the model is trained with both original (source distribution) training data and adversarial examples generated from the source distribution.

A **SharedVocab defense** is an ablation to remove tokens/words in test-time inputs that are not in the source distribution’s vocabulary. Instead of manipulating $<\text{UNK}>$ tokens, we remove unknown words and re-concatenate the remaining tokens/words in order. **Perturbation sets adversarial training** is an ablation that adapts adversarial training to be trained on the joint-distributionally-shifted inputs w.r.t. the source distribution, generated with Algorithm 1.

**Evaluating Hypothesis 2** (Tables 3 & 4) Table 3 validates Hypothesis 2. NLP models factor in semantic structure with respect to $P_i$ through re-mapping unknown words to the $<\text{UNK}>$ token. Surprisingly, the SharedVocab defense occasionally outperforms other defenses in a joint distribution shift setting, which implies (with support from Proposition 1) that the source distribution retains sufficient similarity to natural domains and their shifted distributions in the input space to the extent that feature reuse (Raghu et al., 2020) can occur. The ablation adversarial training defense outperforms the baseline adversarial training, indicating that constructing diverse, joint-distributionally-shifted perturbations sets in contrast to random perturbation sets yield marginal
benefits. Though both our hypernetwork and the ablation adversarial training make use of the perturbation sets, our results indicate, not only that a mapping can indeed be constructed between the changing shifts/perturbations and required parameter adaptation, but that computing an average/static parameter across varying perturbation sets is not optimal compared to enabling the base learner to adapt parameters according to the summation of probabilities of occurrence per sequence. We additionally show in Table 4 that our proposed defense is scalable across different model architectures and capacities.

### 6 Related Work

There is sparsely-growing literature on joint distribution shifts, and we would be amongst the first in an NLP setting to study joint shifts and their methods for robustness. Datta and Shadbolt (2022a) demonstrated the low likelihood of backdooring a model in the presence of joint distribution shifts, including multiple perturbations of the same shift type (backdoor), and multiple perturbations of different shift types (backdoor, adversarial, domain). Qi et al. (2021) used text style transfer to perform adversarial attacks. Naseer et al. (2019) generated domain-invariant adversarial perturbations to fool models of different domains. Ganin et al. (2016) proposed domain-adapted adversarial training to improve domain adaptation. Geirhos et al. (2019) demonstrated the use of stylized perturbations with AdaIN can improve performance on corruptions dataset ImageNet-C. AdvTrojan (Liu et al., 2021) combined adversarial perturbations with backdoor trigger perturbations to craft stealthy triggers to perform backdoor attacks. Santurkar et al. (2020) synthesized distribution shifts by combining random noise, adversarial perturbations, and domain shifts to contribute subpopulation shift benchmarks. Rusak et al. (2020) proposed a robustness measure by augmenting a dataset with both adversarial noise and stylized perturbations, by evaluating a set of perturbation types including Gaussian noise, stylization and adversarial perturbations. Datta and Shadbolt (2022b) demonstrated a low-loss compressed subspace defense to tackle joint distribution shifts in a multi-agent backdoor attack setting.

### Table 3: After-defense Accuracy:

Perturbation Sets + Hypernetworks outperforms the baseline and ablation methods. **After-defense accuracy** $\text{Acc}(\theta_i, \Delta \theta; \text{Adv}(\theta_j; X_j, Y))$ denotes the accuracy of the base learner on the joint shifted (adversarial+domain) distribution (test) data $\text{Adv}(\theta_i; x_i)$, with changes to the model parameters $\theta_i + \Delta \theta$ and/or input processing $\text{Adv}(\theta_j; X_j) + \Delta X$ as part of the defense.

| Target Domain | Attack Domain | $\text{Acc}(\theta_i; \text{Adv}(\theta_j; X_j, Y))$ | $\text{Acc}(h(X_j); \text{Adv}(\theta_j; X_j, Y))$ |
|---------------|---------------|-------------------------------------|-------------------------------------|
| book | d – book | 0.342 0.413 | 0.350 0.372 |
| book | d – electronics | 0.400 0.389 | 0.387 0.377 |
| d – book | 0.326 0.434 | 0.355 0.370 |
| d – electronics | 0.387 0.377 | 0.400 0.389 |
| electronics | d – book | 0.425 0.394 | 0.342 0.395 |
| electronics | d – electronics | 0.390 0.384 | 0.464 0.329 |

### Table 4: Models:

Perturbation Sets + Hypernetworks ($\text{Acc}(h(X_j); \text{Adv}(\theta_j; X_j, Y))$) can retain high after-defense accuracy at varying base learner model capacities and/or architectures.

| Target Domain | Attack Domain | BERT | LSTM | GRU | CNN | LogReg | BERT | LSTM | GRU | CNN | LogReg |
|---------------|---------------|------|------|-----|-----|--------|------|------|-----|-----|--------|
| book | d – book | 0.414 | 0.471 | 0.335 | 0.447 | 0.786 | 0.847 | 0.804 | 0.816 | 0.782 |
| book | d – electronics | 0.350 | 0.372 | 0.325 | 0.353 | 0.765 | 0.826 | 0.795 | 0.742 | 0.767 |
| d – book | 0.400 | 0.389 | 0.416 | 0.315 | 0.460 | 0.792 | 0.812 | 0.784 | 0.770 | 0.725 |
| d – electronics | 0.387 | 0.377 | 0.332 | 0.348 | 0.455 | 0.825 | 0.836 | 0.812 | 0.834 | 0.796 |
| electronics | d – book | 0.425 | 0.394 | 0.473 | 0.338 | 0.474 | 0.775 | 0.821 | 0.795 | 0.782 | 0.712 |
| electronics | d – electronics | 0.390 | 0.384 | 0.464 | 0.329 | 0.432 | 0.730 | 0.824 | 0.753 | 0.724 | 0.678 |
7 Conclusion

We grounded our work in the problem of joint distribution shift, showing not only reduced performance to models in its exposure, but additional performance reduction when the domain shift is similar to the source distribution. With perturbation set construction motivated by our understanding of the input and parameter space bounds, and motivated by dynamic parameter adaptation, we propose an adaptation strategy that tackles joint distribution shifts in NLP settings.

References

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Sri- vastava, and Kai-Wei Chang. 2018. Generating natural language adversarial examples. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2890–2896, Brussels, Belgium. Association for Computational Linguistics.

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In ACL, pages 440–447.

Karsten M. Borgwardt, Arthur Gretton, Malte J. Rasch, Hans-Peter Kriegel, Bernhard Schölkopf, and Alex J. Smola. 2006. Integrating structured biological data by Kernel Maximum Mean Discrepancy. Bioinformatics, 22(14):e49–e57.

Xia Cui, Frans Coenen, and Danushka Bollegala. 2017. Tsp: Learning task-specific pivots for unsupervised domain adaptation. In Machine Learning and Knowledge Discovery in Databases, pages 754–771, Cham. Springer International Publishing.

Siddhartha Datta and Nigel Shadbolt. 2022a. Backdoors stuck at the frontdoor: Multi-agent backdoor attacks that backfire. In International Conference on Learning Representations Workshop: Gamification and Multiagent Solutions.

Siddhartha Datta and Nigel Shadbolt. 2022b. Low-loss subspace compression for clean gains against multi-agent backdoor attacks. arXiv preprint arXiv:2203.03692.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

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

Tomer Galanti and Lior Wolf. 2020. On the modularity of hypernetworks.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. J. Mach. Learn. Res., 17(1):2096–2030.

Robert Geirhos, Patricia Rubisch, Claudia Michaelis, Matthias Bethge, Felix A. Wichmann, and Wieland Brendel. 2019. Imagenet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. In International Conference on Learning Representations.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. In ICML, pages 513–520.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572.

David Ha, Andrew Dai, and Quoc V Le. 2016. Hypernetworks. arXiv preprint arXiv:1609.09106.

Marton Havasi, Rodolphe Jenatton, Stanislav Fort, Jeremiah Zhe Liu, Jasper Snoek, Balaji Lakshminarayanan, Andrew M. Dai, and Dustin Tran. 2021. Training independent subnetworks for robust prediction.

Dan Hendrycks and Kevin Gimpel. 2016. A baseline for detecting misclassified and out-of-distribution examples.

Robin Jia, Aditi Raghunathan, Kerem Göksel, and Percy Liang. 2019. Certified robustness to adversarial word substitutions. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 4129–4142, Hong Kong, China. Association for Computational Linguistics.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In EMNLP, pages 1746–1751.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. 2019. Similarity of neural network representations revisited. In Proceedings of the 36th International Conference on Machine Learning, volume 97 of Proceedings of Machine Learning Research, pages 3519–3529. PMLR.

Alexey Kurakin, Ian Goodfellow, and Samy Bengio. 2016. Adversarial examples in the physical world.
From word embeddings to document distances.
In Proceedings of the 32nd International Conference on Machine Learning, volume 37 of Proceedings of Machine Learning Research, pages 957–966, Lille, France. PMLR.

Fangzhou Liao, Ming Liang, Yinpeng Dong, Tianyu Pang, Xiaolin Hu, and Jun Zhu. 2018. Defense against adversarial attacks using high-level representation guided denoiser. In Proceedings of the 2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018, pages 1778–1787. Computer Vision Foundation / IEEE Computer Society.

Etai Littwin, Tomer Galanti, Lior Wolf, and Greg Yang. 2020. On infinite-width hypernetworks. Advances in Neural Information Processing Systems, 2020-December. Publisher Copyright: © 2020 Neural information processing systems foundation. All rights reserved.; null ; Conference date: 06-12-2020 Through 12-12-2020.

Guansiong Liu, Issa Khalil, Abdallah Khreishah, and NhatHai Phan. 2021. A synergetic attack against neural network classifiers combining backdoor and adversarial examples. In Proceedings of the 2021 ACM on Asia Conference on Computer and Communications Security, ASIA CCS ’21, page 506–519, New York, NY, USA. Association for Computing Machinery.

Hans-Peter Lüthi, Alina Meier, and Tobias Lang. 2018. Towards deep learning models resistant to adversarial attacks.

Hong Liu, Mingsheng Long, Jianmin Wang, and Michael Jordan. 2019. Transferrable adversarial training: A general approach to adapting deep classifiers.

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.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2017. Towards deep learning models resistant to adversarial attacks. arXiv preprint arXiv:1706.06083.

Muzammal Naseer, Salman H. Khan, Harris Khan, Fahad Shahbaz Khan, and Fatih Porikli. 2019. Cross-domain transferability of adversarial perturbations.

Shibani Santurkar, Dimitris Tsiapas, and Aleksander Madry. 2020. Breeds: Benchmarks for subpopulation shift.

Aman Sinha, Hongseok Namkoong, and John Duchi. 2018. Certifiable distributional robustness with principled adversarial training.
In International Conference on Learning Representations.
Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. 
*Dropout: A simple way to prevent neural networks from overfitting.* 
*Journal of Machine Learning Research*, 15(56):1929–1958.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. 
Sequence to sequence learning with neural networks. 
In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2, NIPS’14*, page 3104–3112, Cambridge, MA, USA. MIT Press.

Johannes von Oswald, Christian Henning, João Sacramento, and Benjamin F. Grewe. 2020. 
Continual learning with hypernetworks.

Jenq-Haur Wang, Ting-Wei Liu, Xiong Luo, and Long Wang. 2018. 
An LSTM approach to short text sentiment classification with word embeddings. In *ROCLING*, pages 214–223.

Zixin Wen and Yuanzhi Li. 2021. 
*Toward understanding the feature learning process of self-supervised contrastive learning.* 
In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 11112–11122. PMLR.

Eric Wong, Frank R. Schmidt, and J. Zico Kolter. 2019. 
*Wasserstein adversarial examples via projected sinkhorn iterations.* 
In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 6808–6817. PMLR.

Mitchell Wortsman, Maxwell Horton, Carlos Guestrin, Ali Farhadi, and Mohammad Rastegari. 2021. 
Learning neural network subspaces.

Mitchell Wortsman, Gabriel Ilharco, Samir Yitzhak Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S. Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. 2022. 
Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time.

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.*

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

Yftah Ziser and Roi Reichart. 2019. 
*Task refinement learning for improved accuracy and stability of unsupervised domain adaptation.* 
In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5895–5906, Florence, Italy. Association for Computational Linguistics.
8 Appendix

8.1 Theory

**Proposition 1 (extended)** Distribution shifts, manifesting as perturbations (in this case, adversarial and domain-specific), in NLP are bounded within a finite dictionary or embedding space. Any shifted distribution \( \mathcal{X}_j \) is located at a bounded distance from an (origin) source distribution \( \mathcal{X}_i \).

Distribution shift can manifest as domain shift \( \mathcal{X}_i \to \mathcal{X}_j \), which tend to manifest as a change in the underlying source distribution. Domain shift and/or text distance between 2 datasets \( \text{dist}(X_i, X_j) \) could be measured by the Kullback-Leibler divergence, Maximum Mean Discrepancy (Borgwardt et al., 2006), Word Mover’s Distance (Kusner et al., 2015), Transfer Loss (Blitzer et al., 2007; Glorot et al., 2011), etc.

As each input sequence is a vector, we can compute the distance between 2 datasets \( \text{dist}(X_i, X_j) \) could be measured by the Kullback-Leibler divergence, Maximum Mean Discrepancy (Borgwardt et al., 2006), Word Mover’s Distance (Kusner et al., 2015), Transfer Loss (Blitzer et al., 2007; Glorot et al., 2011), etc.

Domain shift and/or text distance between 2 datasets w.r.t. the average distance of their contained sequences, using any of the aforementioned distance metrics:

\[
\text{dist}(X_i, X_j) = \mathbb{E}_{x_i, x_j \sim X_i, X_j} \left[ \text{dist}(x_i, x_j) \right]
\]

\[
\approx \mathbb{E}_{x_i, x_j \sim X_i, X_j} \| x_i - x_j \|^2
\]

Unlike domain shift, adversarial perturbations tend to manifest as a change in the label distribution, specifically a change in the mapping between the source and label distribution. Perturbations from adversarial attack algorithms, due to the discrete nature of text input spaces, tend to manifest as token substitutions (i.e., a change to the token within a finite dictionary or embedding space. Any

\[
\text{dist}(X_i, X_j) = \mathbb{E}_{x_i, x_j \sim X_i, X_j} \left[ \text{dist}(x_i, x_j) \right]
\]

\[
\approx \mathbb{E}_{x_i, x_j \sim X_i, X_j} \| x_i - x_j \|^2
\]

**Observation 1** \( \theta_i \) and \( \theta_j \) converge (diverge) if their training distributions contain transferable (interfering) features.

The transfer-interference trade-off (Riemer et al., 2019) finds that if 2 arbitrary inputs for 2 independently trained networks contain transferable features, the gradient updates share the same direction; the updates share opposite directions if features interfere against each other. Specifically, we denote transfer and interference as:

\[
\begin{cases}
\text{Transfer: } \frac{\partial L(X, Y_i)}{\partial \theta_i} \cdot \frac{\partial L(X, Y_j)}{\partial \theta_j} > 0 \\
\text{Interference: } \frac{\partial L(X, Y_i)}{\partial \theta_i} \cdot \frac{\partial L(X, Y_j)}{\partial \theta_j} < 0
\end{cases}
\]

Neyshabur et al. (2020) finds that 2 parameters trained on 2 different distributions (e.g., domain-shifted) initialized on pre-trained weights will be optimized towards a shared flat basin in the loss landscape (though the likelihood for this occurrence weakens when a constant random initialization, or 2 independent random initializations are used). This body of work supports the notion that for 2 distributions \( \mathcal{X}_i \) and \( \mathcal{X}_j \), if the 2 distributions contain transferable features, then \( \text{dist}(\theta_i, \theta_j) \approx 0 \) for \( e \leq E \). If the 2 distributions contain interfering features, then \( \text{dist}(\theta_i, \theta_j) > 0 \) for \( e > E \). As the 2 distributions diverge in similarity, we would expect \( E \) to decrease so that the parameters can diverge accordingly \( \text{dist}(X_i, X_j) \propto \frac{1}{E} \); i.e., if the 2 distributions are the same or similar, \( E \approx E \).

**Observation 2** \( \theta_i \) and \( \theta_j \) may diverge, attributing to the presence of highly-contextual/semantic features (to optimize for feature density), or robust features (to optimize for feature sparsity).

Optimizing model parameters along the density-sparsity trade-off is a needed consideration. Parameters optimized for feature density pertains towards features that are more input instance specific or spuriously-correlated (e.g., features that are highly-specific to a given instance’s distribution, context, or semantics). Parameters optimized for feature sparsity pertains towards features that are more task-specific (e.g., features that are relatively agnostic to instance-specific features, but highly-specific to a given task’s distribution, context, or semantics). For example, a feature dense model would classify between a dolphin and a dog based on non-object-specific features, such as whether the background is blue. Robustness training procedures (e.g., data augmentation (Yun et al., 2019), adversarial training
(Goodfellow et al., 2014; Madry et al., 2017)) and sparse model training procedures (e.g. model pruning, dropout (Srivastava et al., 2014), contrastive learning (Wen and Li, 2021)) demonstrate improved model robustness and/or generalizability through sparse feature selection. These work show that robust (sparse; more distributionally-robust) features can be learnt to improve performance against natural or synthetic perturbations. In addition, many of these methods are static-adaptation techniques, where different methods and different hyperparameters per method return varying robustness accuracies (i.e. different model parameters). This indicates there is no one-fit-all parameter optimization strategy against different types of perturbations. This is shown in (Wortsman et al., 2022), where the authors construct a "soup" of models with varying augmentation techniques and hyperparameters to maximize robustness accuracy against varying types of perturbations. For example, the end-goal is not necessarily that we should optimize parameters for maximum feature sparsity or maximum feature density.

Parameter adaptation may also fall outside of the aforementioned observations. Ramé et al. (2021); Havasi et al. (2021) showed that a wide network can learn multiple subnetworks that may operate independently from each other.

From Observations 1 and 2, we aim to clarify that if we shift a source distribution \( \mathcal{X}_i \rightarrow \mathcal{X}_j \), it does not equate to the generation of interfering features, or the inducement of SGD to optimize towards feature density/sparsity. Different perturbations or distances between distributions in the input space may result in different parameter optimization strategies; for example, increasing adversarial perturbation factor \( \varepsilon \) or increasing domain dissimilarity \( \text{dist}(X_i, X_j) \) may not equate to a linear increase in feature sparsity of \( \theta \). One of the implications of this observation is that, though \( X_i \) and \( X_j \) may contain transferable features (due to similarity), if \( \theta_i \) and \( \theta_j \) are optimized independently, and either or both \( X_i \) and \( X_j \) contain features that induce density/sparsity, then it would result in divergence between \( \theta_i \) and \( \theta_j \) early in training: \( |\text{dist}(\theta_i, \theta_j)| > 0 \) for \( \varepsilon \leq E \), \( \varepsilon > E \).

In particular, we note that a linear change in adversarial perturbation factor \( \varepsilon \) or domain similarity may not translate into a linear change in distance in the input space \( \text{dist}(X_i, X_j) \). For example, if a low \( \varepsilon \) breaks the semantic structure of a sentence, then \( \text{dist}(X_i, X_j) \) may be perceived to be high. Based on this discussion, we approximate the distance between 2 distributions in a bounded input space by the distance between their optimized parameters trained on a constant initialization (Eqn 1); as no corresponding ground-truth input distribution may exist for the constant random initialization, the mapped origins for the input space \( \mathcal{X} \) and parameter space \( \Theta \) are the source distribution \( \mathcal{X}_i \) and its mapped parameters \( \theta_i \), respectively.

**Proposition 3** Suppose the \((\ell - 1)\)th layer in an \( \ell \)-layer neural network \( f \) is the layer returning class probabilities such that \( f_{\ell-1}(\theta; \mathbf{x}) = \{y : \rho \} \). and perturbations per sequence \( \xi = \sum_{\lambda} \varepsilon_{\lambda} \), where \( \Lambda \) are different sources/variations of shift. To mitigate the increase in error attributed to \( \arg\max_{\mathbf{x} \sim Y} f_{\ell-1}(\theta_i; \xi) \), for a shifted input \( \mathbf{x}_j \), we may adapt the parameter \( \theta_i \rightarrow \theta_j \) to converge class probabilities w.r.t. \( \xi \) to 0, where

\[
\begin{align*}
\{f_{\ell-1}(\theta_i; \xi) = \sum_{\Lambda} \{y_{\lambda_i} : \rho_{\lambda_i}\} \} \\
\{f_{\ell-1}(\theta_j; \xi) = \sum_{\Lambda} \{y_{\lambda_j} : 0\}\}
\end{align*}
\]

By distance metric (Eqn 6), we can decompose the distribution shift between 2 datasets into a set of perturbations per sequence \( \xi = \sum_{\lambda} \varepsilon_{\lambda} \), where \( \Lambda \) are different sources/variations of shift (e.g. domains, adversarial perturbation).

\[
\begin{align*}
X_j - X_i &= \{\{x_j\}^N \}^M - \{\{x_i\}^N \}^M \\
&= \{\{\sum_{\lambda} \varepsilon_{\lambda}\}^N \}^M
\end{align*}
\]

For an \( \ell \)-layer neural network \( f \), suppose the \((\ell - 1)\)th layer is the layer before the prediction layer that returns class probabilities \( \{y : \rho\} \) such that \( f_{\ell-1}(\theta; \mathbf{x}) = \{y : \rho\} \). This results in a decomposition of the class probabilities altered with respect to \( \xi \):

\[
\begin{align*}
f_{\ell-1}(\theta_i; \mathbf{x}_i) &= \arg\max_{\mathbf{y} \sim Y} \{f_{\ell-1}(\theta_i; \mathbf{x}_i) + \sum_{\lambda} f_{\ell-1}(\theta_i; \varepsilon_{\lambda})\} \\
&= \arg\max_{\mathbf{y} \sim Y} \{\{y_{\lambda_i} : \rho_{\lambda_i}\} + \sum_{\lambda} \{y_{\lambda_j} : \rho_{\lambda_j}\}\}
\end{align*}
\]

In particular, we find that distribution shift results in a misclassification (reduction in accuracy) when the perturbations \( \xi \) shifts the class probabilities towards a different class.
Under joint distribution shift, if the test-time distribution is too distant from the source distribution, then the predicted labels tend to uniformly sample the label distribution: $f_{\ell-1}(\theta; x_j) \approx \{ y : \frac{1}{|Y|} \}$.

**Proof sketch of Theorem 1.** Suppose we sample perturbations $(x_j = x_1 + \xi, (y_i \rightarrow y_j) \sim \mathcal{X}, \mathcal{Y}$.

$$x_j = x_i + \xi$$

$$\mathcal{L}(x_j, y) = \mathcal{L}(x_i, y) + \mathcal{L}(\xi, y)$$

$$\frac{\partial \mathcal{L}(x_j, y)}{\partial \theta} = \frac{\partial \mathcal{L}(x_i, y)}{\partial \theta} + \frac{\partial \mathcal{L}(\xi, y)}{\partial \theta}$$

This decomposition implies $\frac{\partial \mathcal{L}(x_i, y)}{\partial \theta}$ updates part of $\theta$ w.r.t. $x_1$, which we denote as $\theta \odot m_{x_1}$, and $\frac{\partial \mathcal{L}(\xi, y)}{\partial \theta}$ updates part of $\theta$ w.r.t. $\xi$, which we denote as $\theta \odot (m_{x_1} + m_{\xi})$. Given the distances (squared Euclidean norm) between the shifted inputs and outputs $x_1 \rightarrow x_j$ and $y_i \rightarrow y_j$, we can enumerate the following 4 cases. Case (1) is approximates minimal or negligible distribution shift, and is not evaluated. As the scope of (joint) distribution shift specifies a change in the input, a lack of shift in the input distribution invalidates consideration of Case (3).

$$\left\{ \begin{array}{ll}
|x_j - x_i|^2 \approx 0, & ||y_j - y_i||^2 \approx 0 \\
|x_j - x_i|^2 > 0, & ||y_j - y_i||^2 \approx 0 \\
|x_j - x_i|^2 \approx 0, & ||y_j - y_i||^2 > 0 \\
|x_j - x_i|^2 > 0, & ||y_j - y_i||^2 > 0 
\end{array} \right.$$

We denote a random distribution $\text{Rand} : s \sim \mathcal{U}(S)$ s.t. $\mathbb{P}(s) = \frac{1}{|S|}$, where an observation $s$ is uniformly sampled from (discrete) set $S$. For a set of perturbations per sequence $\xi = \sum_{\lambda} \epsilon_{\lambda}$, where $\Lambda$ are different sources/variations of shift, if $\xi \sim \text{Rand}$, then $\frac{\partial \mathcal{L}(\theta, x, y)}{\partial \theta} \sim \text{Rand}$ and $f(\theta; x_j) - f(\theta; x_i) \approx f(\theta; \xi) \sim \text{Rand}$ (by Lemma 1 and 2).

Hence, for each case of $\frac{\partial \mathcal{L}(\theta, x, y)}{\partial \theta}$:

1. if $\frac{\partial \mathcal{L}(\theta, x, y)}{\partial \theta} \neq 0$, given $\theta = \theta \odot (m_{x_1} + m_{\xi})$, then $f(\theta; \xi) \approx f(\theta; m_{x_1} + m_{\xi}) \sim \text{Rand}$;
2. if $\frac{\partial \mathcal{L}(\theta, x, y)}{\partial \theta} = 0$, given $m_{x_1} = 1^{(\theta)}$, then $f(\theta; \xi) \approx f(\xi; \theta + m_{x_1}) \sim \text{Rand}$.

In both cases, the predicted value of $f$ will be sampled randomly. Given it randomly samples from the label space $\mathcal{Y}$ for distributionally-shifted input/output Cases (2) and (4), it follows

$$\boxed{}$$
that, under the presence of distant (joint) distribution shift at test-time, a prediction $y \sim \mathcal{U}(\mathcal{Y})$ s.t. $f_{\ell-1}(y; \{\theta; x\}) = \hat{y}$. For the $(\ell - 1)$th layer that computes class probabilities, this results in $f_{\ell-1}(\theta_i; x_j) \approx \{y : \frac{1}{|\mathcal{Y}|}\}$. 

**Theorem 2** A joint shift with a similar domain and adversarial perturbations (0 ≤ $\xi$ ≤ $\infty$) will return a lower accuracy when the number of classes is finite and bounded w.r.t. the class probability $\xi$ at $|\mathcal{Y}| < \frac{1}{|\rho(\xi)|}$: if $|\mathcal{Y}| \rightarrow \infty$, then a joint shift with a dissimilar domain and adversarial perturbations ($\xi \rightarrow \infty$) will return a lower accuracy w.r.t. perturbed inputs.

**Proof sketch of Theorem 2.** We simplify $f_{\ell-1}(\theta; x) = \{y : \rho\}$ as $f_{\ell-1}(y; \{\theta; x\}) = \rho$ to compute the class probability of label $y$. Perturbations $\xi$ lower the class probabilities to $\rho(\xi)$ below the clean $\rho$: $f_{\ell-1}(y; \{\theta; x + \xi\}) = \rho(\xi) < \rho$. We simplify our analysis, such that when $\rho(\xi) < \rho$, then $f$ predicts $-y$ where $-y \neq y$, i.e. a lower $\rho(\xi)$ returns a lower accuracy w.r.t. perturbed inputs.

We consider 3 cases to evaluate how to maximize $\Delta \rho = \text{mathser} f_{\ell-1}(y; \{\theta; x + \xi\}) - f_{\ell-1}(y; \{\theta; x\})$:

$$
\begin{align*}
\Delta \rho_{\xi=0} & \approx 0 \\
\Delta \rho_{\xi \leq \xi \leq \infty} & = \rho(\xi) - \rho < 0 \\
\Delta \rho_{\xi \rightarrow \infty} & \approx \frac{1}{|\mathcal{Y}|} - \rho < 0 \quad (\text{Thm 1})
\end{align*}
$$

In order for perturbations w.r.t. a similar domain to reduce the accuracy lower than that w.r.t. a dissimilar domain $|\Delta \rho_{\xi \leq \xi \leq \infty}| > |\Delta \rho_{\xi \rightarrow \infty}|$, the number of classes $|\mathcal{Y}|$ needs to be greater than $\frac{1}{|\rho(\xi)|}$ for a given $\xi$:

$$
\Delta \rho_{\xi \leq \xi \leq \infty} < \Delta \rho_{\xi \rightarrow \infty} \\
\rho(\xi) < \frac{1}{|\mathcal{Y}|} \\
|\mathcal{Y}| < \frac{1}{|\rho(\xi)|}
$$

Given the bounds of the class probability w.r.t. $y$, we can conclude that a joint shift with a similar domain and adversarial perturbations (0 ≤ $\xi$ ≤ $\infty$) will return a lower accuracy when the number of classes is finite and bounded at $|\mathcal{Y}| < \frac{1}{|\rho(\xi)|}$; if $|\mathcal{Y}| \rightarrow \infty$, then a joint shift with a dissimilar domain and adversarial perturbations ($\xi \rightarrow \infty$) will return a lower accuracy w.r.t. perturbed inputs.

$$
0 \leq \rho(\xi) < \frac{1}{|\mathcal{Y}| \leq \frac{1}{|\rho(\xi)|}} \\
\frac{1}{|\mathcal{Y}|} \leq \rho(\xi) < \frac{1}{|\mathcal{Y}| \leq \frac{1}{|\rho(\xi)|}} \quad (11)
$$

**Hypothesis 2 (extended)** It is hypothesized the existence of a meta-learner that can compute high-accuracy parameters for inputs sampled from a joint-distributionally-shifted source distribution, where the perturbation sources are adversarial and domain-specific, and the input space is bounded; i.e. $\text{dist}(x_i, x_j) \rightarrow \theta_i + \Delta \theta$.

Suppose we construct a meta-learner $mf$ that maps a change in input $X_j - X_i$ to a change in the parameters $\theta_j = \theta_i + \Delta \theta$, and the adapted parameters are used to perform inference in the base learner $f$.

$$
\Delta \theta = mf(x_j) \\
\Delta \theta \approx mf(\text{dist}(x_i, f^{-1}(\theta_i))) \\
\Rightarrow \hat{y} = f(\theta_i + \Delta \theta; x_i)
$$

The meta learner may undergo different assumptions/constraints. The meta learner may be provided task-specific / distribution-specific meta-data which encodes information parameter adaptation, usually used in few-shot learning implementations such as MAML (Finn et al., 2017), Reptile (Nichol et al., 2018), Hypernetworks (von Oswald et al., 2020). In our setup, we do not presume any known information about the test-time distribution (i.e. no meta-data or headers). The model is assumed to only have its own source distribution / dataset, and has access to no other domain data (including the attacker’s domain). In many implementations, a meta-learner will accept $x_j$ (and/or meta-data) as input arguments. Meta-learners would need to compute a change in parameter distance $\Delta \theta \propto \text{dist}(\theta_i, \theta_j)$ based on $\text{dist}(X_i, X_j)$. The underlying source distribution may not be hard-coded within $f$ or $mf$, hence implicitly the source distribution would be approximated by inverting $\theta_i$ such that $X_i \leftarrow f^{-1}(\theta_i)$. Based on the given assumptions, a hypernetwork would be the optimal candidate to evaluate this parameter adaptation hypothesis: it contains a compressed, latent representation of the parameter space, and it can be implemented such
that meta-data is not required (and theoretically accepts continuously-shifted inputs, not constrained to a training set).

8.2 Methods

**Training** We use a 80 – 20% train-test split for both source data and domain-shifted data. For LSTM (GRU) of 64 cells, tokens embedded w.r.t. GloVe, sigmoid activation function, randomly-initialized and trained with Adam optimizer and 80% (60%) dropout until early-stopping pauses training at loss 0.5. For CNN accepting tokens embedded w.r.t. GloVe, of 3 convolutional layers with kernel widths of 3, 4, and 5, all with 100 output channels, randomly-initialized and trained until early-stopping pauses training at loss 0.5. We use the standard Logistic Regression configurations from scikit-learn. We initialize a pretrained BERT from huggingface (base, uncased) with its own embeddings and trained until loss 0.5. We use the standard Logistic Regressor and randomly-initialized and trained with Adam optimizer and gradient of the loss function w.r.t. inputs:

\[ \nabla \mathcal{L} = \frac{\partial \mathcal{L}}{\partial \theta} \]

In-line with the theoretical formulation of a hypernetwork.

**Defensive distillation** In-line with Papernot et al. (2016b, 2017), we first train an initial model against target domain inputs and labels, and retrieve the raw class probability distribution. 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 w.r.t. FGSM & Perturbations Set** In-line with Goodfellow et al. (2014); Madry et al. (2017), at each iteration we generate adversarial perturbations of \( \varepsilon = 0.4 \) with FGSM, where the size of the perturbed set is the size of the batch. For the Perturbation Sets variant, we do not generate perturbations per batch/epoch, and instead train the model on the pre-generated Perturbation Sets from Algorithm 1.

**Fast Gradient Sign Method (FGSM)** In-line with the FGSM implementation (Goodfellow et al., 2014), we generate adversarial samples by computing the sign of the gradient of the loss function w.r.t. inputs:

\[ \text{Adv}(\theta; x; \varepsilon) : x^{\text{adv}} = x + \varepsilon \cdot \text{sign}(\nabla \mathcal{L}(\theta; x, y)) \]

Image inputs are continuous, while text inputs are discrete, hence the two considerations are (i) which tokens in a sequence to perturb and (ii) measuring the perturbation per token. In-line with Papernot et al. (2016a), we iteratively insert perturbations until \( f(x^{\text{adv}}) \neq y \). Adapted from Papernot et al. (2016a), the proportion of the sequence to be perturbed is \( \varepsilon \) (and we randomly sample indices \( n \) until \( f(x^{\text{adv}}) \neq y \)), the perturbation measurement (sign of gradient direction) is based on FGSM, and the perturbed token \( w \) is the closest token in \( \mathcal{K} \) to the \( x + \varepsilon \cdot \text{sign} \) (where \( x \) would be interpreted as the position in the dictionary or embedding space). We set \( \varepsilon = 0.4 \) for \( \varepsilon \in [0, 1] \).

\[
\text{while } f(\theta; x^{\text{adv}}) \equiv y:
\qquad \text{for } n \in I \text{ where } n \in [1, N], |I| \leq \varepsilon N:
\quad w := x_n + \varepsilon \cdot \text{sign}(\nabla \mathcal{L}(\theta; x, y))
\quad x^{\text{adv}} \leftarrow w
\]\n
**Hypernetwork** We trained a sequence-to-sequence network (Sutskever et al., 2014) on each sequence pair \( x : \Delta \theta \). \( \Delta \theta \) is a flattened \( 1 \times |\theta| \)-dimensional vector, hence the input sequence length is \( N \) and output sequence length is \( |\theta| \). We enlarge the capacity of the sequence-to-sequence network \( \mathcal{C}^{mf} \) with respect to the capacity (parameter count) of the base learner such that the loss of the hypernetwork converges \( C^{mf} \geq C^f \) s.t. \( \mathcal{L}^{mf}(x, \Delta \theta) \), in-line with (von Oswald et al., 2020; Littwin et al., 2020) (we do not have a clear-cut ratio, as different architectural differences affect the compression ratio; for example, a wider but shallower base learner, with the same number of parameters as a narrower but deeper base learner, can be learnt by a smaller meta learner). This results in fine-tuned changes in sequence length, units/cells, etc. The hypernetwork is trained until early-stopping pauses at loss 0.5. Other training models may be possible, in-line with the theoretical formulation of a hypernetwork.

We adapt the implementation of a hypernetwork with respect to the constraints and formulation in Hypothesis 2. Namely, we do not use any conditioning input \( I \) (accepting \( x \) as sole input), and we compute the parameter change \( \Delta \theta \) to an origin parameter \( \Theta_i \) (i.e. the parameter space for different distributions are all identical \( \Theta_i \equiv \Theta_j \), given Proposition 2). For our implementation, \( \text{dist}(\Theta_i, \Theta_j) \equiv \Delta \theta \). Additional problem setup constraints are that the input space (with respect to the
word/character embedding space in language/text) is bounded, the model only has access to its own source distribution (no other data from other domains/distributions), and the label set is finite and small.

**Perturbation Sets** Our configurations are as follows: 10 perturbation sets, 10 maximum iterations for generating perturbations, transfer loss distance metric, maximum distance for transfer loss of 0.1, initial $\varepsilon = 0.9$ with perturbation learning rate of 0.05. We subsequently provide further details on our supporting motivations and theory for the use of perturbation sets.

Rather than optimizing $\theta^{mf}$ towards adapting $\theta^f$ with respect to adversarial tokens $x^\text{adv}_n$ (which represent perturbation-specific adaptation that may not arise at test-time), we aim to optimize $\theta^{mf}$ towards adapting $\theta^f$ with respect to varying $\sum_n^N P_i$ (which represent structure-specific adaptation, where for example an increase in ambiguous structural patterns or unknown tokens should correspondingly result in sparser feature selection in adapted parameters). In particular, we would like to generate $T$ training sets that include diverging $\sum_n^N P_i$ such that

$$\{\text{dist}((\sum_m^M \sum_n^N P_i(x_{i,n,m}), \sum_m^M \sum_n^N P_i(x_{i,n,m})))\}_{T \in T},$$

and for each unique $\sum_n^N P_i$, the required parameter adaptation $\Delta \theta$ to varying ambiguities represented by $\sum_n^N P_i$.

The use of perturbation sets, adversarial samples, and augmented data samples during training do indeed generate diverging $\sum_n^N P_i$. However, these static-adaptation methods map an average diverging $\sum_n^N P_i$ to a single (robust) parameter:

$$\frac{1}{M} \sum_m^M \sum_n^N P_i(x_{i,n,m}, K, N) \mapsto \Delta \theta.$$  

While multiple perturbation sets within the total $T$ sets may have similar $\sum_n^N P_i$, the required parameter adaptation $\Delta \theta$ for each unique $\sum_n^N P_i$ may differ, and thus benefit from a hypernetwork performing parameter-switching/adaptation. This is empirically validated in Table 3, where static-adaptation methods (even on our shifted perturbations set) perform sub-optimally across all domain pairs. Even the SharedVocab defense can occasionally outperform adversarial training, which validates that (i) there is sufficient shared vocabulary that the distance between the 2 domains are not too distant in the input space to still retain feature reuse (i) between distributions/domains, and (ii) the average ability of predicting under domain-specific ambiguity between an adversarially-trained model (no ability to deal with ambiguity) is similar (thus an improved architecture is required to handle domain-specific ambiguity).

The perturbation sets we construct are based solely on the source distribution, but should contain perturbations that resemble that originating from a joint distribution shift. We would like to sample distributions that have a high proportion of perturbations such that they are distant in the input space $\text{dist}(X_i, X_j)$. It is observed from (Zhang et al., 2017) that if the perturbed distributions are too dissimilar such that they tend to be random sentences or labels are randomly flipped, a base learner will be able to overfit and memorize these perturbed training instances, but will retain no generalizability at test-time. Hence, at the same time we would like to sample distributions to retain sufficient similarities to the source distribution, as indicated by Table 2 that natural domains have high SharedVocab and low transfer loss with respect to a source distribution. Distributions of high input distances are not expected at test-time (evaluation of hypothesis 1 indicates high-distance distributions would yield random predictions, and similar domains yield lower accuracy than dissimilar domains). Bounding the maximum distance in the input space also assists in bounding the parameter space, increasing the ease for the hypernetwork to generalize its output (base learner parameter) space.

An important objective is to ensure the transfer loss of the perturbation set is small/bounded and diverse, but we also regulate the $\varepsilon$, in addition to the aforementioned reasons, for the practical reason that not regulating $\varepsilon$ may result in the perturbation set containing no perturbations ($\varepsilon = 0$) to attain a low $tf$ (i.e. $X_i \equiv X_j$), thus to avoid this we ensure perturbations must manifest at a minimum extent $\varepsilon_{\text{min}}$. 

### Table 3

| Perturbation Sets | Score |
|-------------------|-------|
| No Perturbations   | Low   |
| One Perturbation   | High  |
| Two Perturbations  | Medium|

### Table 2

| Natural Domains   | Score |
|-------------------|-------|
| High SharedVocab  | Good  |
| Low SharedVocab   | Poor  |