Understanding Robust Generalization in Learning Regular Languages

Soham Dan [1] Osbert Bastani [1] Dan Roth [1]

Abstract

A key feature of human intelligence is the ability to generalize beyond the training distribution, for instance, parsing longer sentences than seen in the past. Currently, deep neural networks struggle to generalize robustly to such shifts in the data distribution. We study robust generalization in the context of using recurrent neural networks (RNNs) to learn regular languages. We hypothesize that standard end-to-end modeling strategies cannot generalize well to systematic distribution shifts and propose a compositional strategy to address this. We compare an end-to-end strategy that maps strings to labels with a compositional strategy that predicts the structure of the deterministic finite state automaton (DFA) that accepts the regular language. We theoretically prove that the compositional strategy generalizes significantly better than the end-to-end strategy. In our experiments, we implement the compositional strategy via an auxiliary task where the goal is to predict the intermediate states visited by the DFA when parsing a string. Our empirical results support our hypothesis, showing that auxiliary tasks can enable robust generalization. Interestingly, the end-to-end RNN generalizes significantly better than the theoretical lower bound, suggesting that it is able to achieve at least some degree of robust generalization.

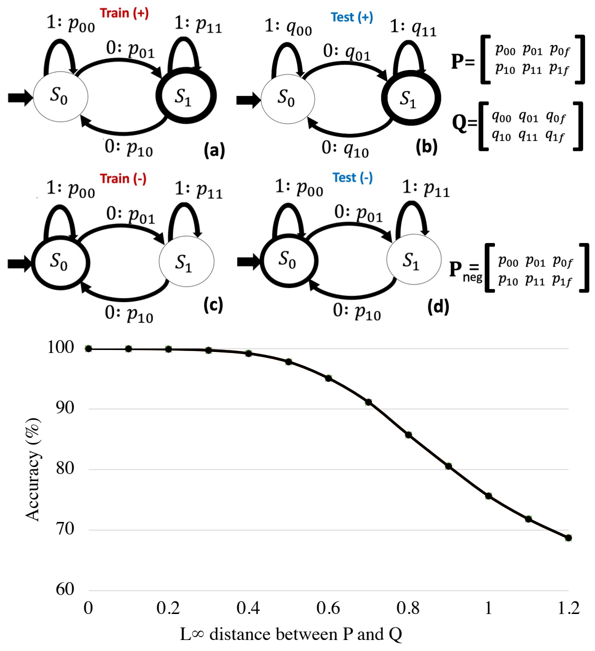

Figure 1. The *parity task* is to classify whether $x \in \{0,1\}^*$ has an odd ($y = 1$) or even ($y = 0$) number of zeros. We show the edge Markov Chains for generating (a) training, and (b) shifted test examples. We also plot test accuracies as a function of $\|P - Q\|_\infty$, where P and Q are the transition probabilities of the edge Markov chains (the train accuracy is always 100%). The test accuracy drops significantly as $\|P - Q\|_\infty$ increases. We plot accuracy w.r.t the ℓ_∞ norm in order to compare with our theoretical bounds as discussed in Sec. 5.3.

1. Introduction

A key challenge facing deep learning is its inability to generalize *robustly* to shifts in the underlying distribution. For example, Lake & Baroni (2018) demonstrate that recurrent neural networks (RNNs) have difficulty in generalizing to longer sentences than those seen during training and is also unable to understand novel combinations of familiar components, i.e., they lack *systematic compositionality*. Subsequent work have provided evidence of this failure across different architectures (Dessì & Baroni, 2019; Furrer et al., 2020) and tasks (Ruis et al., 2020; Kim & Linzen, 2020).

Humans, on the other hand, are remarkably robust to such shifts, suggesting that robust generalization is possible in practice. Thus, a key question is how to design deep learning algorithms that achieve this property. In the theoretical direction, most work has focused on learning in the presence of *covariate shift*—i.e., shift in the distribution over inputs (Blitzer et al., 2008; Pagnoni et al., 2018). However, these results largely rely on the shift being small (in particular,

[1]Department of Computer and Information Science, University of Pennsylvania. Correspondence to: Soham Dan <sohamdan@seas.upenn.edu>, Osbert Bastani <obastani@seas.upenn.edu>, Dan Roth <danroth@seas.upenn.edu>.

Proceedings of the 39th International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022. Copyright 2022 by the author(s).

bounded total variation (TV) distance), which does not hold for many of the shifts considered in robust generalization.

Alternatively, the empirical work has focused primarily on devising new generalization splits on which existing models fail (Lake & Baroni, 2018; Hupkes et al., 2020). However, without theoretical grounding, it is hard to interpret what degree of generalization can reasonably be expected.

In this paper, we consider the problem of classifying regular languages, which is sufficiently simple that we can theoretically analyze generalization in this setting. We consider two approaches to train neural networks. First, we consider an end-to-end approach that learns a model directly mapping strings to labels; this approach represents how neural networks are typically trained. Second, we consider a *compositional* approach that is given access to intermediate supervision on the sequence of states visited by a deterministic finite state automaton (DFA) representing the language. Based on this extra information, the compositional strategy learns a model to predict the DFA transitions and final states.

We consider a training distribution in the form of a Markov model over strings that is constructed based on the DFA (essentially a Markov chain, but where emissions are on the edges); then, our goal is to generalize in the presence of shifts to this Markov model (Fig. 1). Small shifts in the Markov model probabilities can lead to large shifts in the distribution over strings—for instance, leading to significantly longer strings on average.

We provide two theoretical characterizations for each approach. First, we prove closed-form lower bounds on generalization of each strategy; these bounds rely on bounding the TV distance between the training and test distributions. Our results suggest that compositional approaches can generalize significantly better than the end-to-end strategy. Intuitively, compositionality helps since the distribution over DFA transitions shifts significantly less than the distribution over strings.

In practice, our bounds on the total variation distance can be overly conservative. Thus, we additionally propose algorithms for producing unbiased estimates of the relevant TV distances, and use them to empirically study generalization. In our experiments, the end-to-end strategy is a recurrent neural network (RNN) trained to directly map strings to labels. For the compositional strategy, we train an RNN in the same way, but provide it with an auxiliary task where on each step, its goal is to predict the current state of the underlying DFA. Intuitively, this strategy should align the RNN representation with the DFA state, making it easy to predict whether the DFA accepts a given string.

Our empirical results demonstrate that the compositional strategy generalizes significantly better than the end-to-end strategy. Interestingly, however, the end-to-end strategy generalizes significantly better than expected according to the estimated bound, suggesting that even end-to-end learning can exhibit some degree of robust generalization. Finally, our compositional strategy relies on additional supervision; we demonstrate that it is possible to achieve some degree of robust generalization even if we relax this supervision.

Contributions. We formalize the problem of learning regular languages (Section 3). In the context of this problem, we provide a theoretical analysis of robust generalization for both end-to-end and compositional learning, including theoretical bounds demonstrating the benefits of compositional learning, along with algorithms for obtaining unbiased estimates of the relevant TV distances (Section 4). We then empirically demonstrate that an auxiliary task of predicting the DFA state achieves the compositional generalization bound and significantly outperforms the end-to-end strategy, though the end-to-end strategy exhibits some degree of robust generalization (Section 5). We also perform experiments to analyze the impact of "free" auxiliary signals, the number of examples, model cell choice, length of examples, and the confidence calibration of the end-to-end and the compositional models on the shifted distribution.

2. Related Work

Linguistics, automata, and RNNs. Linguistics has traditionally been tightly coupled to automata theory (Knight & May, 2009; Suresh et al., 2021). Markov used finite state processes to predict sequences of vowels and consonants in novels (Markov, 1956; Jurafsky & Martin), and Shannon extended this to predict letter sequences of English words using Markov processes (Shannon, 2001). Markov chains and DFAs have applications in transliteration, translation, lexical processing, speech recognition, optical character recognition, summarization, sequence tagging, and in sequence generation such as speech synthesis and text generation. (Knight & May, 2009; Gales & Young, 2008).

Recurrent neural networks (RNNs), which have been incredibly effective in natural language processing applications have renewed interest in automata theory in both linguistics and machine learning communities [1]. RNNs have expressiveness closely connected to that of DFAs (Rabusseau et al., 2019; Michalenko, 2019; Chen et al., 2018; Tiňo et al., 1998). For instance, it is possible to extract the DFA from an RNN trained on sequences generated by that DFA (Weiss et al., 2018; Giles et al., 1992b;a; Omlin & Giles, 1996; Gers & Schmidhuber, 2001; Firoiu et al., 1998); there has also been work trying to relate states of an RNN with those of a DFA (Tiňo et al., 1998; Michalenko, 2019). These prop-

[1]See (Ackerman & Cybenko, 2020) for a survey of the relationships between various state-of-the-art neural network architectures and formal languages.

erties make RNNs ideal models for us to study. While existing work has focused on showing that RNNs can represent DFAs, their ability to learn DFAs in a way that generalize robustly has not yet been studied.

Systematic generalization in deep learning. The ability for neural networks to generalize robustly has been a longstanding question in cognitive science (Fodor & Pylyshyn, 1988). Recently, several benchmarks have been proposed to investigate systematic generalization on carefully crafted train/test splits (e.g., the test set contains longer sequences than the training set) (Lake & Baroni, 2018; Lake, 2019; Hupkes et al., 2020; Loula et al., 2018; Tsarkov et al., 2020; Ruis et al., 2020; Kim & Linzen, 2020). Although they are all motivated to measure systematic compositionality, there is no theoretical framework to understand the generalization for the different choices of splits, making it hard to know if generalization is at all possible for a given split; our goal in this paper is to take a step towards bridging this gap.

Learning theory and covariate shift. There has been significant theoretical work studying generalization in the presence of *covariate shift* (Blitzer et al., 2008) (where the input distribution changes), with a large focus on *domain adaptation* (i.e., where we are given unlabeled examples from the target domain). This has been widely studied in machine learning (Pagnoni et al., 2018; Blitzer et al., 2008; Zhang et al., 2019; Koh et al., 2021) and natural language processing (Ben-David et al., 2021; Li, 2012). Redko et al. (2020) surveys theoretical work in this area. While covariate shift refers to a shift in the covariate (the independent, input variable) distribution between train and test, robust generalization refers to the property of a model to generalize under large covariate shifts (as seen in length generalization, for example (Lake & Baroni, 2018)). One key challenge is that in general, learning with covariate shift is only possible if the shift is small (e.g., small TV distance), yet the shifts in the robust generalization settings we consider, are typically large. Thus, we must leverage additional structure to learn in a provably generalizable way; we prove that compositional learning can do exactly this by leveraging the DFA structure.

3. Learning Regular Languages

We formalize the learning problem we study. To do so, we need to define the following objects: (i) a classifier $f^* : \mathcal{X} \to \mathcal{Y}$ that we want to learn, (ii) a training distribution P over $x \in \mathcal{X}$, and (iii) a test distribution Q over inputs $x \in \mathcal{X}$. Then, the problem is to train a model $\hat{f}$ on inputs $x_1, ..., x_n \sim P$ with labels $y_i = f^*(x_i)$, and then test $\hat{f}$ on inputs $x'_1, ..., x'_{n'} \sim Q$ with labels $y'_i = f^*(x'_i)$.

The classifier f^* is defined by a regular language $L(M)$ represented by a deterministic finite-state automaton (DFA) $M = (S, \Sigma, \delta, s_0, F)$, where S is a finite set of states, Σ is the alphabet, $\delta : S \times \Sigma \to S$ is the state transition function, $s_0 \in S$ is the initial state, and $F \subseteq S$ is a set of final states (Sipser, 1996). Given a string $x = \sigma_1...\sigma_T \in \Sigma^*$, we call T the *length* of x, and letting

$$s_t = \begin{cases} s_0 & \text{if } t = 1 \\ \delta(s_{t-1}, \sigma_{t-1}) & \text{otherwise,} \end{cases}$$

we call $z = s_1...s_{T+1} \in S^*$ the state sequence *induced* by x. We define $L(M) \subseteq \Sigma^*$ to be the strings accepted by M—i.e., $\sigma_1...\sigma_T \in L(M)$ if the induced state sequence $s_1...s_{T+1}$ satisfies $s_{T+1} \in F$. We let $\mathcal{X} = \Sigma^*$ be the strings over Σ, let $\mathcal{Y} = \{0, 1\}$, and let $f^*(x) = \mathbb{1}(x \in L(M))$ indicate whether x is accepted by M.

Next, P is defined by converting M to a variant of a Markov chain. In particular, we assume given (i) probabilities $P(\sigma \mid s)$ of emitting σ in state s, and (ii) probabilities $P(e \mid s)$ (where $e \in \{0, 1\}$) of terminating upon reaching state s. Then, we sample $x \sim P$ as follows: initialize $s_1 \leftarrow s_0$; on each step t, terminate with probability $P(e \mid s)$ and return $x = \sigma_1...\sigma_t$; otherwise, sample $\sigma_t \sim P(\cdot \mid s_t)$, transition $s_{t+1} = \delta(s_t, \sigma_t)$, and continue. We call P an *edge Markov chain* since it is essentially a Markov chain with edge emissions.

This strategy only samples positive examples $x \in L(M)$; to sample negative examples, we use the same strategy with the automaton M' for the complement $\mathcal{X} \setminus L(M)$. We sample positive and negative examples in equal proportion. Finally, we define the test distribution Q similarly.

4. Theoretical Analysis

Next, we characterize the ability of algorithms for learning DFAs to generalize to shifted distributions, considering both the case where $\hat{f}$ directly maps sequences to labels, as well as a compositional strategy where $\hat{f}$ learns the DFA structure.

4.1. Background on Covariate Shift

We provide background on generalization bounds in the presence of *covariate shift*—i.e., when the training and test distributions P and Q differ. Let

$$L_P(\hat{f}) = \mathbb{P}_{x \sim P}[\hat{f}(x) \neq f^*(x)]$$

be the loss of $\hat{f}$ on P, and similarly for $L_Q(\hat{f})$. Letting $\text{TV}(P(x), Q(x)) = \sum_{x \in \mathcal{X}} |P(x) - Q(x)|$ be the total variation distance, we have the following.

Lemma 4.1. $L_Q(\hat{f}) \leq L_P(\hat{f}) + \mathit{TV}(P(x), Q(x))$

We give a proof in Appendix A.1. In other words, we can characterize the loss of $\hat{f}$ on the test distribution Q

in terms of its loss on the training distribution P and $\mathrm{TV}(P(x), Q(x))$.

4.2. Learning DFAs with Covariate Shift

Next, we provide bounds on $\mathrm{TV}(P(x), Q(x))$. We focus on a single pair of edge Markov chains $P(x)$ and $Q(x)$; given bound $\mathrm{TV}(P^+(x), Q^+(x)) \leq \epsilon^+$ for the positive example distributions and bound $\mathrm{TV}(P^-(x), Q^-(x)) \leq \epsilon^-$ for the negative example distributions, it is easy to check that

$$\mathrm{TV}(P(x), Q(x)) \leq \frac{\epsilon^+ + \epsilon^-}{2},$$

where $P(x) = (P^+(x) + P^-(x))/2$ and similarly for $Q(x)$. First, we have the following worst-case bound on the shift, specialized to the case where all strings are of a fixed length T.

Lemma 4.2. *$TV(P(x), Q(x)) \leq 2T|S|^{T+1}\epsilon$, where $\epsilon = \max_{s \in S} TV(P(\sigma \mid s), Q(\sigma \mid s))$*

We give a proof in Appendix A.2.

Theorem 4.3. $L_Q(\hat{f}) \leq L_P(\hat{f}) + 2T|S|^{T+1}\epsilon$

This result follows immediately from Lemmas 4.1 & 4.2. This result says that the accuracy of $\hat{f}$ decays exponentially in the length of the inputs, and it decays linearly in ϵ, which quantifies the shift in the emission distributions of P and Q. However, this bound may be overly conservative. Given edge Markov chains $P(x)$ and $Q(x)$, we can estimate $\mathrm{TV}(P(x), Q(x))$ using the following.

Theorem 4.4. *We have*

$$TV(P(x), Q(x)) = \mathbb{E}_{x \sim P}\left[\left|1 - \frac{Q(x)}{P(x)}\right|\right].$$

We give a proof in Appendix A.3. Thus, given samples $x_1, ..., x_n \sim P$, we have

$$\mathrm{TV}(P(x), Q(x)) \approx \sum_{i=1}^{n} \left|1 - \frac{Q(x_i)}{P(x_i)}\right|.$$

Given $x = \sigma_1 ... \sigma_T$, we can compute

$$P(x) = \prod_{t=1}^{T} P(\sigma_t \mid s_t),$$

where $s_1 ... s_{T+1}$ is the state sequence induced by x, and similarly for $Q(x)$. Note that in this strategy, T is not fixed and can vary with x.

4.3. Compositional Learning of DFAs

So far, we have considered a classifier trained directly to predict labels from strings. Next, we consider a compositional strategy that is given access to the hidden states of the DFA, learns to predict transitions and final states, and then composes these predictions to form the overall prediction.

Consider a model $\hat{g} : S \times \Sigma \to S$ trained to predict transitions (i.e., $\hat{g}(s, \sigma) \approx \delta(s, \sigma)$), along with a model $\hat{h} : S \to \{0, 1\}$ trained to predict final states (i.e., $\hat{h}(s) \approx \mathbb{1}(s \in F)$). Then, we have

$$\hat{f}(x) = \mathbb{1}(s_{T+1} \in F),$$

where $s_1 ... s_{T+1}$ is the state sequence induced by x. In this case, because g and h take states as inputs, we directly consider shifts in the state distribution. In particular, the distribution P_t of states encountered by $\hat{g}$ and $\hat{h}$ on step t is given by $P_t(s') = \mathbb{1}(s' = s_0)$ if $t = 1$, and

$$P_t(s') = \mathbb{P}_{s \sim P_{t-1}, \sigma \sim P(\cdot \mid s)}[s' = \delta(s, \sigma)]$$

otherwise, and similarly for Q_t. In addition, we let

$$P_t(s, \sigma) = P_t(s) P(\sigma \mid s).$$

Then, $\hat{g}$ is trained on the distribution $P_t(s, \sigma)$ (for $t \in \{1, ..., T\}$), and $\hat{h}$ is trained on $P_{T+1}(s)$. Similar to before, we have the following worst-case bound on the shift, specialized to the case where all strings are of a fixed length T.

Lemma 4.5. *We have $TV(P_t(s), Q_t(s)) \leq (t-1)\epsilon$, and $TV(P_t(s, \sigma), Q_t(s, \sigma)) \leq t\epsilon$.*

We give a proof in Appendix A.4.

Theorem 4.6. *$L_Q(\hat{f}) \leq \tilde{L}_P(\hat{f}) + 2T^2\epsilon$, where $\epsilon = \max_{s \in S} TV(P(\sigma \mid s), Q(\sigma \mid s))$ and*

$$\tilde{L}_P(\hat{f}) = \sum_{t=1}^{T} L_{P_t}(\hat{g}) + L_{P_{T+1}}(\hat{h})$$

We give a proof in Appendix A.5. In this case, the accuracy of $\hat{f}$ scales quadratically in T, which is significantly better than Theorem 4.3, and linearly ϵ, which is the same as Theorem 4.3. As before, this bound can be overly conservative, so we estimate the error for given edge Markov chains $P(x)$ and $Q(x)$ based on the following.

Theorem 4.7. *We have*

$$L_Q(\hat{f}) \leq \tilde{L}_P(\hat{f}) + \sum_{t=1}^{T} TV(P_t(s, \sigma), Q_t(s, \sigma)) + TV(P_{T+1}(s), Q_{T+1}(s))$$

This result follows by the same argument as the proof of Theorem 4.6. We can estimate $P_t(s)$ by drawing samples $x_1, ..., x_n \sim P$, and computing

$$\hat{P}_t(s) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}(s = s_{i,t}),$$

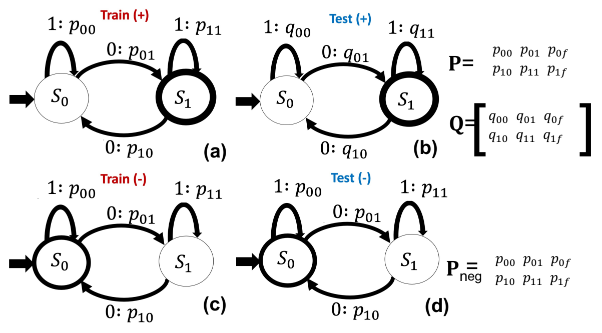
Train (+)
1: p_{00}
0: p_{01}
1: p_{11}
0: p_{10}
S_0
S_1
(a)
Test (+)
1: q_{00}
0: q_{01}
1: q_{11}
0: q_{10}
(b)
P= p_{00} p_{01} p_{0f} p_{10} p_{11} p_{1f}
Q= q_{00} q_{01} q_{0f} q_{10} q_{11} q_{1f}
Train (-)
(c)
Test (-)
(d)
P_{neg}= p_{00} p_{01} p_{0f} p_{10} p_{11} p_{1f}

Figure 2. The eMC$_{\text{id}}$ and eMC$_{\text{ood}}$ used to generate the train and o.o.d. test positive examples, and the eMC$_{\text{neg}}$ used to generate negative examples. Here, $\rightarrow$ denotes the start state; bold circles denote end states; P, Q, and P_{neg} denote the transition matrices for eMC$_{\text{id}}$, eMC$_{\text{ood}}$, and eMC$_{\text{neg}}$, respectively; and $0f, 1f$ denotes the end probability in states S_0 and S_1, respectively, with $p_{0f} = 0$ in eMC$_{\text{id}}$ and $p_{1f} = 0$ in eMC$_{\text{ood}}$.

where $s_{i,1}...s_{i,T_i+1}$ is the state sequence induced by x_i, and similarly for $\hat{Q}_t(s)$. Then, we have

$$\text{TV}(P_t(s), Q_t(s)) \approx \sum_{s\in S} |\hat{P}_t(s) - \hat{Q}_t(s)|.$$

Finally, we have $\hat{P}_t(s,\sigma) = \hat{P}_t(s)P(\sigma \mid s)$, and similarly for $\hat{Q}_t(s,\sigma)$, in which case

$$\text{TV}(P_t(s,\sigma), Q_t(s,\sigma)) \approx \sum_{s\in S} |\hat{P}_t(s,\sigma) - \hat{Q}_t(s,\sigma)|.$$

These estimates can be used in conjunction with Theorem 4.7. In our experiments, we make a modification—to reduce variance, rather than compute estimates for each t separately, we aggregate states across all steps and use the average for all $\hat{P}_t(s)$, and similarly for $\hat{Q}_t(s)$. Finally, we heuristically take T to be the average length of $x \sim P$.

5. Experiments

Next, we describe our experiments investigating whether end-to-end models can learn regular languages in a way that generalizes to distribution shifts, and whether different types of auxiliary supervision can help do so, and therefore enable robust generalization.

5.1. Experimental Setup

Classification problem. We consider the regular language classification problem. We construct an edge Markov chain eMC$_{\text{id}}$ to generate training examples and in-domain (i.d.) test examples for each language by assigning transition probabilities to each edge of the DFA for that language. To generate out-of-domain (o.o.d.) test examples, we perturb some of the edge probabilities of eMC$_{\text{id}}$ to obtain eMC$_{\text{ood}}$. To generate negative examples, we use the same strategy for the complement language $\mathcal{L}^C = \mathcal{X}\backslash\mathcal{L}$ to obtain eMC$_{\text{neg}}$; for simplicity, we do not shift the negative example distribution.

We focus on the parity language $\mathcal{L}_{par}$ as a case-study, and include additional results for other languages in Appendix B. This language has $\Sigma = \{0, 1\}$, and consists of all strings containing an odd number of zeros. Our DFA for $\mathcal{L}_{par}$ has two states $S = \{s_0, s_1\}$, where s_0 is the start state, and final states $F = \{s_1\}$. Fig. 2 shows the edge-Markov chains eMC$_{\text{id}}$, eMC$_{\text{ood}}$, and eMC$_{\text{neg}}$ we use. Note that by increasing the loop probabilities, we generate longer sequences; thus, the o.o.d. test distribution contains longer strings on average than the training distribution.

We also consider a broader class of *modulo languages*: regular languages over $\Sigma = \{0, 1\}$ of the form $\mathcal{L}_{mod-k}$, where the number of zeros are a multiple of k, for $k \in \{3, 4, 5\}$.

Classifier. We train an RNN binary classifier on examples generated by eMC$_{\text{id}}$ (positive) and eMC$_{\text{neg}}$ (negative). We evaluate it on i.d. test examples generated by eMC$_{\text{id}}$, eMC$_{\text{neg}}$, and on o.o.d. test examples from eMC$_{\text{ood}}$,eMC$_{\text{neg}}$.

To improve performance, we propose *state sequence auxiliary supervision (SSAS)*, where we additionally train the RNN on an auxiliary supervised learning task. In particular, given an input example $x \sim P$, where P is an edge Markov chain, in addition to training the RNN to predict the ground truth label $f^*(x)$, we train its representation z_t at each step t to predict the state s_t visited by P at step t. This supervision task is a multi-class classification problem, so we use the cross-entropy loss. This auxiliary loss is jointly optimized with the binary classification loss for the main task (the losses are weighted equally). Importantly, the auxiliary supervision signal is only provided during training; thus, at test time, the RNN acts identically to an RNN trained only on the main task (i.e., without SSAS).

We use an RNN with LSTM cells, with an embedding dimension of 50 and a hidden layer with dimension 50, optimized using stochastic gradient descent (SGD) with a learning rate of 0.01 [2]. We use $N^+_{train} = 1600$ positive and $N^-_{train} = 1600$ negative train examples, $N^+_{dev} = N^-_{dev} = 200$ dev examples, and use $N^+_{test} = 2000$ positive and $N^-_{test} = 2000$ negative examples for each of the i.d. and o.o.d. test sets.

Metrics. We perform several different experiments to compare our theoretical bounds with the empirical accuracy, studying the impact of the number of training examples, model cell variations (vanilla RNN, LSTM, or GRU), asymmetry in length generalization, the use of "free" auxiliary tasks that do not require knowledge of the DFA, and the

[2]Hyper-parameters chosen based on accuracy on i.d. dev-set.

Task	E2E	SSAS	RI (%)
$\mathcal{L}_{mod-3}$	66.10	**98.68**	49.49
$\mathcal{L}_{mod-4}$	64.61	**97.58**	51.03
$\mathcal{L}_{mod-5}$	65.15	**93.00**	42.75

Table 1. Accuracy (%) of the end-to-end (E2E) vs. SSAS training on the o.o.d. test set, with the relative improvements (RI) of SSAS over E2E for the *modulo languages*: $\mathcal{L}_{mod-3}$, $\mathcal{L}_{mod-4}$, $\mathcal{L}_{mod-5}$.

confidence calibration of the models.

5.2. End-to-End vs. SSAS Training

First, we compare the o.o.d. accuracy of end-to-end and SSAS training (in both cases, the i.d. test accuracy is 100%). Table 1 shows the o.o.d. test accuracy of each approach for the case $\mathrm{TV}(P, Q) = 1.3$ for each of the modulo-languages considered [3]. As can be seen, SSAS training significantly improves accuracy compared to end-to-end training. Strictly speaking, SSAS is not a compositional learning algorithm, but it guides the RNN to learn representations that correctly encode the compositional structure of the DFA. Thus, these results validate our theoretical finding that compositional training significantly improves generalization.

5.3. Theoretical vs. Empirical Generalization

Focusing on the parity language, we compare the empirical accuracy on the o.o.d. test set to our estimates of the accuracy based on Theorems 4.4 & 4.7 in Section 4. We consider $\mathrm{eMC_{id}}$ with transitions

$$P = \begin{bmatrix} 0.2 & 0.8 & 0 \\ 0.7 & 0.2 & 0.1 \end{bmatrix},$$

$\mathrm{eMC_{ood}}$ with transitions

$$Q = \begin{bmatrix} \delta & 1-\delta & 0 \\ 0.9-\delta & \delta & 0.1 \end{bmatrix},$$

where $\delta \in \{0.2, 0.25, ..., 0.85\}$, and $\mathrm{eMC_{neg}}$ with

$$P_{neg} = \begin{bmatrix} 0.7 & 0.2 & 1 \\ 0.8 & 0.2 & 0 \end{bmatrix}.$$

In Fig. 3, we plot the following four quantities as a function of $\|P - Q\|_\infty$ (for our choices of P and Q, we have $\epsilon = 2\|P - Q\|_\infty$, where ϵ is defined in Theorem 4.3):

- Empirical accuracy on the o.o.d. test set (solid lines), for end-to-end (red) and SSAS (black)
- Theoretical estimates of the accuracy (dashed lines) for end-to-end (red) and SSAS (black)

[3]Additional plots for the *modulo languages* are in Appendix B.

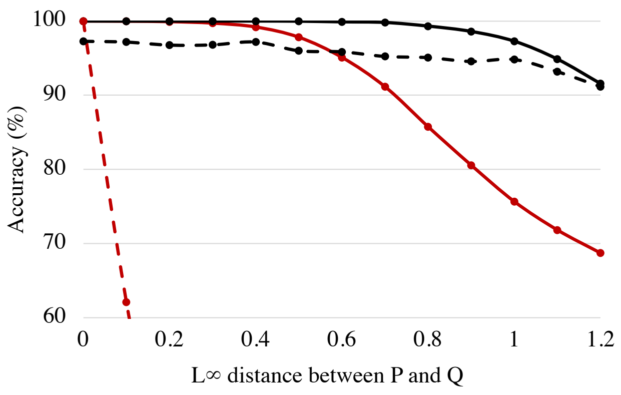

Figure 3. We plot the empirical o.o.d. test set accuracies for the end-to-end model (red solid) and the SSAS model (black solid), and the theoretical estimate of the o.o.d. accuracies for the end-to-end model (red dashed) and the SSAS model (black dashed).

For the theoretical estimate of the accuracy: (i) for end-to-end, we draw $n = 10000$ samples from eMC_{id} and compute the lower bound based on Theorem 4.4, and (ii) for SSAS, we use 10000 samples from each eMC_{id} and eMC_{ood}, and estimate the lower bound based on Theorem 4.7; for both, we average across 10 random repetitions. Further, the reported empirical accuracies are the average over 10 random runs.

As can be seen, the theoretical estimates are tight for SASS, indicating the SASS matches the expected generalization rate for compositional models. These results support our intuition that SASS enables the RNN to learn the compositional structure of the DFA. Interestingly, the end-to-end model significantly outperforms the theoretical estimate. Since the estimate of the TV distance converges to its true value, so the gap between the theoretical and empirical values must either be due to the inequality in Lemma 4.1 or the fact that the RNN is learning some compositional structure. We expect that the latter must be happening to some degree to explain such a large gap—importantly, the gap is substantially larger than the gap for SASS.

5.4. Free Auxiliary Signals

While SSAS improves generalization, a critical shortcoming is that it requires knowledge of the underlying DFA; in particular, it uses the DFA to construct ground truth state sequences used to train the RNN. In this section, we study whether auxiliary tasks computed without such prior knowledge can improve generalization. As an example, we consider the *count auxiliary task*; intuitively, a model that can count zeros is more likely to correctly learn the parity concept. More precisely, we consider a 10-class classifica-

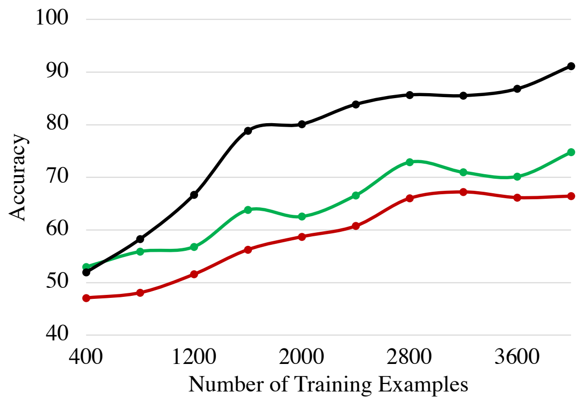

Figure 4. For the setting described in Sec. 5.4, we plot the accuracies of the baseline (red), SSAS model (black), and the model with free auxiliary supervision (green), for varying number of training examples. We see that *auxiliary count supervision* helps the baseline model consistently across varying amounts of training data.

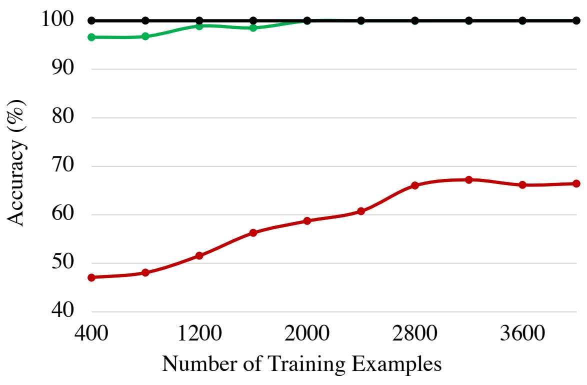

Figure 5. We plot the train (black), i.d. test (green), and o.o.d. test (red) accuracies as a function of the number of training examples drawn from the eMC_{id} for the baseline (end-to-end) model. The plot shows that having more in-domain training examples, by itself, is insufficient to achieve robust generalization.

tion task, where zero counts greater than 8 are represented by the 10^{th} class. As with SSAS, we equally weigh the cross-entropy loss for this auxiliary task with the binary cross-entropy loss for the main task.

We let P, Q, and P_{neg} be as in Section 5.3, with $\delta = 0.85$. In Fig. 4, we plot the o.o.d. test accuracy (estimated on 4000 examples) as a function of the number of training examples, ranging from 400 to 4000. As can be seen, the count auxiliary task improves performance by $5 - 8\%$. The improvement is less than in SSAS, which is to be expected since the task in SSAS gives information directly relevant to the main task. These results demonstrate that even without knowledge of the DFA, we can improve generalization via auxiliary supervision.

5.5. Effect of Number of Training Examples

Next, we show that simply increasing the number of training examples is insufficient for achieving robust generalization, demonstrating that novel techniques such as SSAS are required to generalize robustly.

We consider P, Q, and P_{neg} as in Section 5.3, with $\delta = 0.85$. In Fig. 5, we plot the empirical train, i.d. test, and o.o.d. test accuracies as a function of the the number of training examples varying from 400 to 4000; in all cases, we estimate accuracy on 4000 test examples. As can be seen, the i.d. test accuracy quickly reaches 100%, but the o.o.d. test accuracy remains significantly lower, stabilizing around 66% at 2800 training examples (the train accuracy is 100% throughout). These results also justify our choice of 3200 training examples in our other experiments.

5.6. Effect of the Model Cell

Next, we study whether changes in the model cell architecture affect generalization. We consider long short term memory (LSTM) units, recurrent neural network (RNN) units, and gated recurrence units (GRU).

We set P, Q, and P_{neg} as in Section 5.3. In Fig. 6, we plot the o.o.d. test accuracy (estimated using 4000 examples) as a function of $\|P - Q\|_\infty$ for each of the three choices. As before, the train and i.d. test accuracy are 100% for all choices. For models trained end-to-end, LSTM and GRU cells perform comparably in terms of o.o.d. accuracy whereas RNN cells perform significantly worse. These results are in line with the fact that RNNs are worse at capturing long-range dependencies, which are necessary for solving the parity task. For models trained using SSAS, LSTMs perform the best; interestingly, GRUs perform well at first but become worse than RNNs as $\|P-Q\|_\infty$ becomes large. The superior performance of LSTMs over other cell choices, justifies our use of them in the other experiments.

5.7. Asymmetric Length Generalization

Next, we study how different training distributions affects generalization. In particular, in the context of the parity language, we consider two training distributions given by edge Markov chains with transition probabilities P_1 and P_2 and a test distribution given by an edge Markov chain with

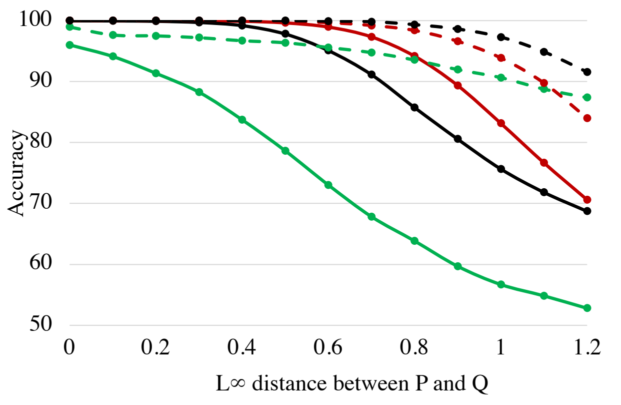

Figure 6. We plot the accuracy of models trained end-to-end (solid) and using SSAS (dashed) as a function $\|P - Q\|_\infty$, for models with LSTM (black), GRU (red), and RNN (green) cells.

transition probabilities Q, where

$$P_1 = \begin{bmatrix} 0.2 & 0.8 & 0 \\ 0.7 & 0.2 & 0.1 \end{bmatrix}$$

$$P_2 = \begin{bmatrix} 0.8 & 0.2 & 0 \\ 0.1 & 0.8 & 0.1 \end{bmatrix}$$

$$Q = \begin{bmatrix} 0.5 & 0.5 & 0 \\ 0.4 & 0.5 & 0.1 \end{bmatrix}.$$

The negative examples are generated from eMC$_{\text{neg}}$ described in Section 5.3. Importantly, we have $\|P_1 - Q\|_\infty = \|P_2 - Q\|_\infty = 0.6$. In Fig. 7, we show the results of evaluating models trained on examples from P_1 (red) and P_2 (black) on test examples generated using P_1, P_2, and Q. As can be seen, the model trained on examples from P_1 generalizes well to test examples from Q and P_2, whereas the model trained from P_2 generalizes poorly to test examples from Q and P_1. In this case, P_1 tends to generate longer sequences than P_2 (with Q being in between); thus, our results show that training on longer sequences can generalize to shorter sequences, whereas training on shorter sequences cannot generalize to training on longer sequences.

5.8. Confidence Calibration

We study confidence calibration of the baseline (E2E) and SSAS models as a function of increasing separation, $\|P - Q\|_\infty$. Calibration measures how well the posterior probabilities of a model are aligned with the empirical likelihoods (Guo et al., 2017). We use the following metrics:

• *Brier Score (BS)* (Brier et al., 1950) is a proper scoring rule for measuring the accuracy of predicted probabilities. It is defined as the mean squared error between the predicted probabilities and the actual targets. If n denotes the total number of examples and Ψ_i denotes the prediction

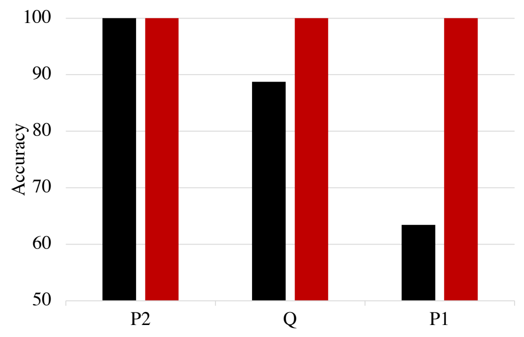

Figure 7. As described in Section 5.7, we train models on 3200 examples generated from two different eMCs with transition matrices P_1 (red) and P_2 (black), and tested on 4000 test examples separately generated from three different eMCs P_1, P_2, and Q.

Metric	Model	0.0	0.4	0.8	1.2
BS	E2E	5e-06	9e-04	0.073	0.262
	SSAS	**3e-06**	**1e-04**	**0.008**	**0.051**
ECE	E2E	4e-04	0.003	0.097	0.303
	SSAS	**3e-04**	**0.002**	**0.035**	**0.125**

Table 2. Brier Score (BS) and Expected Calibration Error (ECE) as a function of increasing $\|P - Q\|_\infty = \{0, 0.4, 0.8, 1.2\}$ for the E2E and SSAS models. Note that lower values are better.

probability that $f(x_i) = 1$, then the Brier Score is:

$$BS = \frac{1}{n}\sum_{i=1}^{n}(\Psi_i - y_i)^2.$$

• *Expected Calibration Error (ECE)* (Guo et al., 2017) measures the difference in expectation between confidence and accuracy. Empirically this is approximated by dividing the data into M confidence based bins, $B_1, ..., B_M$, where B_m contains all datapoints x_i for which Ψ_i lies in $(\frac{m-1}{M}, \frac{m}{M}]$. If $acc(B_m)$ and $conf(B_m)$ denotes the average accuracy and prediction confidence for the points in B_m, then ECE is:

$$ECE = \sum_{m=1}^{M}\frac{|B_m|}{n}|acc(B_m) - conf(B_m)|.$$

In Table 2 we see that while calibration degrades with increasing $\|P - Q\|_\infty$ for both models, SSAS is significantly better calibrated than E2E under the distribution shift.

6. Conclusion

We have studied robust generalization of RNNs learning regular languages, providing both theoretical and empirical evidence that compositional strategies generalize more robustly to shifted distributions compared to end-to-end strategies. In particular, leveraging an auxiliary task in the form of state supervision (which we call SSAS) achieves the theoretical rate anticipated by our compositional generalization guarantee. We have also demonstrated that generic auxiliary tasks such as counting zeros can also improve generalization. Interestingly, we find that end-to-end learning outperforms the theoretical rate for end-to-end learning, suggesting that end-to-end approaches can still achieve some degree of robust generalization.

A key direction for future work is to explore the effectiveness of other techniques for enabling robust generalization. It would also be interesting to explore how these results generalize to other kinds of tasks beyond regular languages [4], as well as to study the performance of state-of-the-art architectures such as transformers in this setting.

Acknowledgements

Research was sponsored by the Army Research Office and was accomplished under Grant Number W911NF-20-1-0080. This work was supported by Contract FA8750-19-2-0201 with the US Defense Advanced Research Projects Agency (DARPA). The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense, the Army Research Office or the U.S. Government.

References

Ackerman, J. and Cybenko, G. A survey of neural networks and formal languages. *arXiv preprint arXiv:2006.01338*, 2020.

Ben-David, E., Cohen, S. B., McDonald, R., Plank, B., Reichart, R., Rotman, G., and Ziser, Y. Proceedings of the second workshop on domain adaptation for nlp. In *Proceedings of the Second Workshop on Domain Adaptation for NLP*, 2021.

Blitzer, J., Crammer, K., Kulesza, A., Pereira, F., and Wortman, J. Learning bounds for domain adaptation. 2008.

Brier, G. W. et al. Verification of forecasts expressed in terms of probability. *Monthly weather review*, 78(1):1–3, 1950.

Chen, Y., Gilroy, S., Maletti, A., May, J., and Knight, K. Recurrent neural networks as weighted language recognizers. In *NAACL-HLT*, 2018.

Dessì, R. and Baroni, M. Cnns found to jump around more skillfully than rnns: Compositional generalization in seq2seq convolutional networks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 3919–3923, 2019.

[4](Suzgun et al., 2019) empirically evaluates the learning capabilities of LSTMs for some simple context-sensitive and context-free languages.

Firoiu, L., Oates, T., and Cohen, P. R. Learning a deterministic finite automaton with a recurrent neural network. In *International Colloquium on Grammatical Inference*, pp. 90–101. Springer, 1998.

Fodor, J. A. and Pylyshyn, Z. W. Connectionism and cognitive architecture: A critical analysis. *Cognition*, 28(1-2): 3–71, 1988.

Furrer, D., van Zee, M., Scales, N., and Schärli, N. Compositional generalization in semantic parsing: Pretraining vs. specialized architectures. *arXiv preprint arXiv:2007.08970*, 2020.

Gales, M. and Young, S. The application of hidden markov models in speech recognition. 2008.

Gers, F. A. and Schmidhuber, E. Lstm recurrent networks learn simple context-free and context-sensitive languages. *IEEE Transactions on Neural Networks*, 12(6):1333–1340, 2001.

Giles, C. L., Miller, C. B., Chen, D., Chen, H.-H., Sun, G.-Z., and Lee, Y.-C. Learning and extracting finite state automata with second-order recurrent neural networks. *Neural Computation*, 4(3):393–405, 1992a.

Giles, C. L., Miller, C. B., Chen, D., Sun, G.-Z., Chen, H.-H., and Lee, Y.-C. Extracting and learning an unknown grammar with recurrent neural networks. In *Advances in neural information processing systems*, pp. 317–324, 1992b.

Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. On calibration of modern neural networks. In *International Conference on Machine Learning*, pp. 1321–1330. PMLR, 2017.

Hupkes, D., Dankers, V., Mul, M., and Bruni, E. Compositionality decomposed: how do neural networks generalise? *Journal of Artificial Intelligence Research*, 67: 757–795, 2020.

Jurafsky, D. and Martin, J. H. Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition.

Kim, N. and Linzen, T. Cogs: A compositional generalization challenge based on semantic interpretation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 9087–9105, 2020.

Knight, K. and May, J. Applications of weighted automata in natural language processing. In *Handbook of Weighted Automata*, pp. 571–596. Springer, 2009.

Koh, P. W., Sagawa, S., Xie, S. M., Zhang, M., Balsubramani, A., Hu, W., Yasunaga, M., Phillips, R. L., Gao, I., Lee, T., et al. Wilds: A benchmark of in-the-wild distribution shifts. In *International Conference on Machine Learning*, pp. 5637–5664. PMLR, 2021.

Lake, B. and Baroni, M. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *International Conference on Machine Learning*, pp. 2873–2882. PMLR, 2018.

Lake, B. M. Compositional generalization through meta sequence-to-sequence learning. *Advances in Neural Information Processing Systems*, 32, 2019.

Li, Q. Literature survey: domain adaptation algorithms for natural language processing. *Department of Computer Science The Graduate Center, The City University of New York*, pp. 8–10, 2012.

Loula, J., Baroni, M., and Lake, B. Rearranging the familiar: Testing compositional generalization in recurrent networks. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pp. 108–114, 2018.

Markov, A. Essai d'une recherche statistique sur le texte du roman" eugene onegin" illustrant la liaison des epreuve en chain ('example of a statistical investigation of the text of" eugene onegin" illustrating the dependence between samples in chain'). izvistia imperatorskoi akademii nauk (bulletin de l'académie impériale des sciences de st.-pétersbourg), 7: 153–162, 1913. *English Translation by Morris Halle*, 1956.

Michalenko, J. J. *Representing formal languages: A comparison between finite automata and recurrent neural networks*. PhD thesis, Rice University, 2019.

Omlin, C. W. and Giles, C. L. Extraction of rules from discrete-time recurrent neural networks. *Neural networks*, 9(1):41–52, 1996.

Pagnoni, A., Gramatovici, S., and Liu, S. Pac learning guarantees under covariate shift. *arXiv preprint arXiv:1812.06393*, 2018.

Rabusseau, G., Li, T., and Precup, D. Connecting weighted automata and recurrent neural networks through spectral learning. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 1630–1639. PMLR, 2019.

Redko, I., Morvant, E., Habrard, A., Sebban, M., and Bennani, Y. A survey on domain adaptation theory: learning bounds and theoretical guarantees. *arXiv preprint arXiv:2004.11829*, 2020.

Ruis, L., Andreas, J., Baroni, M., Bouchacourt, D., and Lake, B. M. A benchmark for systematic generalization in grounded language understanding. *Advances in Neural Information Processing Systems*, 33, 2020.

Shannon, C. E. A mathematical theory of communication. *ACM SIGMOBILE mobile computing and communications review*, 5(1):3–55, 2001.

Sipser, M. Introduction to the theory of computation. *ACM Sigact News*, 27(1):27–29, 1996.

Suresh, A. T., Roark, B., Riley, M., and Schogol, V. Approximating probabilistic models as weighted finite automata. *Computational Linguistics*, 47(2):221–254, 2021.

Suzgun, M., Belinkov, Y., and Shieber, S. M. On evaluating the generalization of lstm models in formal languages. *Proceedings of the Society for Computation in Linguistics (SCiL)*, pp. 277–286, 2019.

Tiňo, P., Horne, B. G., Giles, C. L., and Collingwood, P. C. Finite state machines and recurrent neural networks—automata and dynamical systems approaches. In *Neural networks and pattern recognition*, pp. 171–219. Elsevier, 1998.

Tsarkov, D., Tihon, T., Scales, N., Momchev, N., Sinopalnikov, D., and Schärli, N. *-cfq: Analyzing the scalability of machine learning on a compositional task. *arXiv preprint arXiv:2012.08266*, 2020.

Weiss, G., Goldberg, Y., and Yahav, E. Extracting automata from recurrent neural networks using queries and counterexamples. In *International Conference on Machine Learning*, pp. 5247–5256. PMLR, 2018.

Zhang, Y., Liu, T., Long, M., and Jordan, M. Bridging theory and algorithm for domain adaptation. In *International Conference on Machine Learning*, pp. 7404–7413. PMLR, 2019.

A. Proofs

A.1. Proof of Lemma 4.1

We have

$$L_Q(\hat{f}) = \sum_{x\in\mathcal{X}} \mathbb{1}(\hat{f}(x) \neq f^*(x))Q(x) = \sum_{x\in\mathcal{X}} \mathbb{1}(\hat{f}(x) \neq f^*(x))(P(x)+Q(x)-P(x)) \\ \leq L_P(\hat{f}) + \sum_{x\in\mathcal{X}} |Q(x)-P(x)|,$$

as claimed. $\square$

A.2. Proof of Lemma 4.2

The given emission probabilities $P(\sigma \mid s)$ induce transition probabilities over the DFA states—in particular, the probability of transitioning from s to s' is

$$P(s' \mid s) = \sum_{\sigma\in\Sigma} P(\sigma \mid s)\cdot \mathbb{1}(s' = \delta(s,\sigma)).$$

Then, the probability of a sequence of states is $P(s_1...s_{T'}) = \prod_{t=1}^{T} P(s_{t+1} \mid s_t)$, where $T' = T+1$, and similarly for $Q(s_1...s_{T'})$. Now, we prove the claim. First, note that

$$\begin{aligned} |P(x)-Q(x)| &= \left|\sum_{z\in S^{T'}} P(x\mid z)P(z) - \sum_{z\in S^{T'}} Q(x\mid z)Q(z)\right| \\ &\leq \left|\sum_{z\in S^{T'}} (P(x\mid z)P(z) - P(x\mid z)Q(z))\right| + \left|\sum_{z\in S^{T'}} (P(x\mid z)Q(z) - Q(x\mid z)Q(z))\right| \\ &\leq \sum_{z\in S^{T'}} P(x\mid z)|P(z)-Q(z)| + \sum_{z\in S^{T'}} |P(x\mid z)-Q(x\mid z)|Q(z). \end{aligned}$$

Thus, we have

$$\begin{aligned} \text{TV}(P(x),Q(x)) &= \sum_{x\in\Sigma^T} |P(x)-Q(x)| \\ &\leq \sum_{z\in S^{T'}}\sum_{x\in\Sigma^T} P(x\mid z)|P(z)-Q(z)| + \sum_{z\in S^{T'}}\sum_{x\in\Sigma^T} |P(x\mid z)-Q(x\mid z)|Q(z) \\ &= \text{TV}(P(z),Q(z)) + \mathbb{E}_{z\sim Q}[\text{TV}(P(x\mid z),Q(x\mid z))]. \end{aligned} \tag{1}$$

Consider the first term of (1). Letting $z = s_1...s_{T'}$, note that

$$\begin{aligned} |P(z)-Q(z)| &= \left|\prod_{t=1}^{T} P(s_{t+1}\mid s_t) - \prod_{t=1}^{T} Q(s_{t+1}\mid s_t)\right| \\ &= \left|\sum_{\tau=1}^{T}\left(\prod_{t=1}^{\tau} P(s_{t+1}\mid s_t)\prod_{t=\tau+1}^{T} Q(s_{t+1}\mid s_t) - \prod_{t=1}^{\tau-1} P(s_{t+1}\mid s_t)\prod_{t=\tau}^{T} Q(s_{t+1}\mid s_t)\right)\right| \\ &= \left|\sum_{\tau=1}^{T}\left[\prod_{t=1}^{\tau-1} P(s_{t+1}\mid s_t)\right]\left[\prod_{t=\tau+1}^{T} Q(s_{t+1}\mid s_t)\right](P(s_{\tau+1}\mid s_t) - Q(s_{\tau+1}\mid s_t))\right| \\ &\leq \sum_{\tau=1}^{T} |P(s_{\tau+1}\mid s_t) - Q(s_{\tau+1}\mid s_t))| \\ &\leq T\max_{s\in S}\text{TV}(P(s'\mid s),Q(s'\mid s)). \end{aligned}$$

Furthermore, note that for any $s \in S$, we have

$$\begin{aligned}
\text{TV}(P(s' \mid s), Q(s' \mid s)) &= \sum_{s' \in S} |P(s' \mid s) - Q(s' \mid s)| \\
&= \sum_{s' \in S} \left| \sum_{\sigma \in \Sigma} (P(\sigma \mid s) - Q(\sigma \mid s)) \cdot \mathbb{1}(s' = \delta(s, \sigma)) \right| \\
&\leq \sum_{\sigma \in \Sigma} |P(\sigma \mid s) - Q(\sigma \mid s)| \sum_{s' \in S} \mathbb{1}(s' = \delta(s, \sigma)) \\
&= \text{TV}(P(\sigma \mid s), Q(\sigma \mid s)), \qquad (2)
\end{aligned}$$

As a consequence, we have

$$\begin{aligned}
\sum_{z \in S^{T'}} |P(z) - Q(z)| &\leq T|S|^{T'} \max_{s \in S} \text{TV}(P(s' \mid s), Q(s' \mid s)) \\
&\leq T|S|^{T'} \max_{s \in S} \text{TV}(P(\sigma \mid s), Q(\sigma \mid s)).
\end{aligned}$$

Now, consider the second term of (1). Letting $x = \sigma_1 ... \sigma_T$ and $z = s_1 ... s_{T'}$, we have

$$\begin{aligned}
|P(x \mid z) - Q(x \mid z)| &= \left| \prod_{t=1}^{T} P(\sigma_t \mid s_t) - \prod_{t=1}^{T} Q(\sigma_t \mid s_t) \right| \\
&= \left| \sum_{\tau=1}^{T} \prod_{t=1}^{\tau} P(\sigma_t \mid s_t) \prod_{t=\tau+1}^{T} Q(\sigma_t \mid s_t) - \prod_{t=1}^{\tau-1} P(\sigma_t \mid s_t) \prod_{t=\tau}^{T} Q(\sigma_t \mid s_t) \right| \\
&= \left| \sum_{\tau=1}^{T} \left[\prod_{t=1}^{\tau-1} P(\sigma_t \mid s_t) \right] \left[\prod_{t=\tau+1}^{T} Q(\sigma_t \mid s_t) \right] (P(\sigma_\tau \mid s_\tau) - Q(\sigma_\tau \mid s_\tau)) \right| \\
&\leq \sum_{\tau=1}^{T} \left[\prod_{t=1}^{\tau-1} P(\sigma_t \mid s_t) \right] \left[\prod_{t=\tau+1}^{T} Q(\sigma_t \mid s_t) \right] |P(\sigma_\tau \mid s_\tau) - Q(\sigma_\tau \mid s_\tau))|.
\end{aligned}$$

As a consequence, we have

$$\begin{aligned}
&\sum_{x \in \Sigma^T} |P(x \mid z) - Q(x \mid z)| \\
&\leq \sum_{\sigma_1 ... \sigma_T \in \Sigma^T} \sum_{\tau=1}^{T} \left[\prod_{t=1}^{\tau-1} P(\sigma_t \mid s_t) \right] \left[\prod_{t=\tau+1}^{T} Q(\sigma_t \mid s_t) \right] |P(\sigma_\tau \mid s_\tau) - Q(\sigma_\tau \mid s_\tau))| \\
&= \sum_{\tau=1}^{T} \left[\prod_{t=1}^{\tau-1} \sum_{\sigma_t \in \Sigma} P(\sigma_t \mid s_t) \right] \left[\prod_{t=\tau+1}^{T} \sum_{\sigma_t \in \Sigma} Q(\sigma_t \mid s_t) \right] \sum_{\sigma_\tau \in \Sigma} |P(\sigma_\tau \mid s_\tau) - Q(\sigma_\tau \mid s_\tau))| \\
&= \sum_{\tau=1}^{T} \sum_{\sigma_\tau \in \Sigma} |P(\sigma_\tau \mid s_\tau) - Q(\sigma_\tau \mid s_\tau))| \\
&= \sum_{\tau=1}^{T} \text{TV}(P(\sigma \mid s_\tau), Q(\sigma \mid s_\tau)).
\end{aligned}$$

Therefore, we have

$$\begin{aligned}
\mathbb{E}_{z \sim Q}[\text{TV}(P(x \mid z), Q(x \mid z)] &= \sum_{\tau=1}^{T} \mathbb{E}_{z \sim Q}[\text{TV}(P(\sigma \mid s_\tau), Q(\sigma \mid s_\tau))] \\
&\leq T \max_{s \in S} \text{TV}(P(\sigma \mid s), Q(\sigma \mid s)).
\end{aligned}$$

The claim follows since $T\epsilon + T|S|^{T+1}\epsilon \leq 2T|S|^{T+1}\epsilon$. $\square$

A.3. Proof of Theorem 4.4

Note that

$$\text{TV}(P(x),Q(x)) = \sum_{x\in\mathcal{X}} |P(x)-Q(x)| = \sum_{x\in\mathcal{X}} \left|1-\frac{Q(x)}{P(x)}\right| P(x) = \mathbb{E}_{x\sim P}\left[\left|1-\frac{Q(x)}{P(x)}\right|\right],$$

as claimed. □

A.4. Proof of Lemma 4.5

We prove the first claim by induction on t. The base case $t=1$ is trivial, since $P_1(s)=Q_1(s)=\mathbb{1}(s=s_0)$. For the inductive case, note that

$$\begin{aligned}
&\text{TV}(P_t(s'),Q_t(s')) \\
&= \sum_{s'\in S} |P_t(s')-Q_t(s')| \\
&= \left|\sum_{s'\in S}\sum_{s\in S} P_{t-1}(s)P(s'\mid s) - Q_{t-1}(s)Q(s'\mid s)\right| \\
&\le \left|\sum_{s'\in S}\sum_{s\in S} P_{t-1}(s)P(s'\mid s) - P_{t-1}(s)Q(s'\mid s)\right| + \left|\sum_{s'\in S}\sum_{s\in S} P_{t-1}(s)Q(s'\mid s) - Q_{t-1}(s)Q(s'\mid s)\right| \\
&\le \sum_{s'\in S}\sum_{s\in S} P_{t-1}(s)|P(s'\mid s)-Q(s'\mid s)| + \sum_{s'\in S}\sum_{s\in S} |P_{t-1}(s)-Q_{t-1}(s)|Q(s'\mid s) \\
&\le \max_{s\in S} \text{TV}(P(s'\mid s),Q(s'\mid s)) + \text{TV}(P_{t-1}(s),Q_{t-1}(s)) \\
&\le \max_{s\in S} \text{TV}(P(\sigma\mid s),Q(\sigma\mid s)) + \text{TV}(P_{t-1}(s),Q_{t-1}(s)) \\
&\le (t-1)\epsilon,
\end{aligned}$$

where the second-to-last step follows from (2) in the proof of Theorem 4.3, so the first claim follows. For the second claim, note that

$$\begin{aligned}
\text{TV}(P_t(s,\sigma),Q_t(s,\sigma)) &= \sum_{s\in S}\sum_{\sigma\in\Sigma} |P_t(s,\sigma)-Q_t(s,\sigma)| \\
&= \sum_{s\in S}\sum_{\sigma\in\Sigma} |P_t(s)P(\sigma\mid s)-Q_t(s)Q(\sigma\mid s)| \\
&\le \sum_{s\in S}\sum_{\sigma\in\Sigma} |P_t(s)P(\sigma\mid s)-P_t(s)Q(\sigma\mid s)| + |P_t(s)Q(\sigma\mid s)-Q_t(s)Q(\sigma\mid s)| \\
&= \sum_{s\in S}\sum_{\sigma\in\Sigma} P_t(s)|P(\sigma\mid s)-Q(\sigma\mid s)| + |P_t(s)-Q_t(s)|Q(\sigma\mid s) \\
&= \max_{s\in S} \text{TV}(P(\sigma\mid s),Q(\sigma\mid s)) + \text{TV}(P_t(s),Q_t(s)) \\
&\le t\epsilon,
\end{aligned}$$

so the second claim follows as well. □

A.5. Proof of Theorem 4.6

First, by a union bound, we have

$$\begin{aligned}
L_Q(\hat{f}) &= \mathbb{P}_{x\sim Q}[\hat{f}(x) \neq f^*(x)] \\
&= \mathbb{P}_{x\sim Q}\left[\bigvee_{t=1}^{T} \hat{g}(s_t,\sigma_t) \neq \delta(s_t,\sigma_t) \vee \hat{h}(s_{T+1}) \neq \mathbb{1}(s_{T+1} \in F)\right] \\
&\leq \sum_{t=1}^{T} \mathbb{P}_{x\sim Q}[\hat{g}(s_t,\sigma_t) \neq \delta(s_t,\sigma_t)] + \mathbb{P}_{x\sim Q}[\hat{h}(s_{T+1}) \neq \mathbb{1}(s_{T+1} \in F)] \\
&= \sum_{t=1}^{T} \mathbb{P}_{(s_t,\sigma_t)\sim Q_t}[\hat{g}(s_t,\sigma_t) \neq \delta(s_t,\sigma_t)] + \mathbb{P}_{s_{T+1}\sim Q_{T+1}}[\hat{h}(s_{T+1}) \neq \mathbb{1}(s_{T+1} \in F)] \\
&= \sum_{t=1}^{T} L_{Q_t}(\hat{g}) + L_{Q_{T+1}}(\hat{h}).
\end{aligned}$$

By Lemmas 4.1 & 4.5, the loss of $\hat{g}$ on step t satisfies

$$L_{Q_t}(\hat{g}) \leq L_{P_t}(\hat{g}) + T\epsilon,$$

and similarly the loss of $\hat{h}$ satisfies

$$L_{Q_{T+1}}(\hat{h}) \leq L_{P_{T+1}}(\hat{h}) + T\epsilon.$$

Thus, we have

$$L_Q(\hat{f}) \leq \left[\sum_{t=1}^{T} L_{P_t}(\hat{g}) + L_{P_{T+1}}(\hat{h})\right] + T(T+1)\epsilon,$$

so the claim follows since $T(T+1)\epsilon \leq 2T^2\epsilon$ (since $T \geq 1$). $\square$

B. Additional Experiments

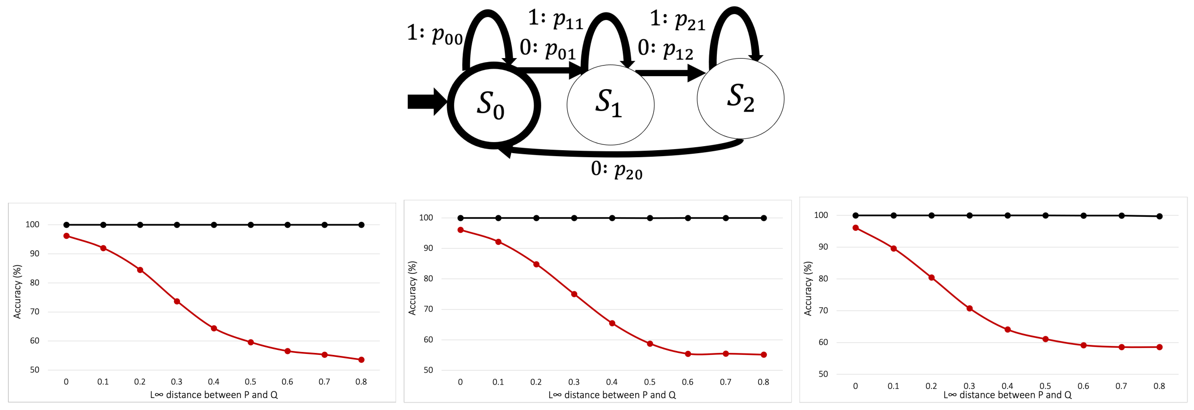

Figure 8. The *modulo-k task* is to classify whether the number of zeros in $x \in \{0,1\}^*$ is divisible by k $(y = 1)$ or not $(y = 0)$. We show the edge Markov Chains for generating examples for $\mathcal{L}_{mod-3}$, and is similar for $\mathcal{L}_{mod-4}$ and $\mathcal{L}_{mod-5}$ with additional states in between. The shifts are introduced similar to the parity language by perturbing the loop probabilities. We plot test accuracies for $\mathcal{L}_{mod-3}, \mathcal{L}_{mod-4}, \mathcal{L}_{mod-5}$ (in order from L-R) as a function of $\|P - Q\|_\infty$. For the baseline model (solid red line), the test accuracy drops significantly as $\|P - Q\|_\infty$ increases while our proposed SSAS approach (solid black line) ensures robustness to distribution shift.