Adversarial Reprogramming of Text Classification Neural Networks

Paarth Neekhara[1], **Shehzeen Hussain**[2], **Shlomo Dubnov**[1,3], **Farinaz Koushanfar**[2]
[1]Department of Computer Science
[2]Department of Electrical and Computer Engineering
[3]Department of Music
University of California San Diego
{pneekhar,ssh028}@ucsd.edu

Abstract

In this work, we develop methods to repurpose text classification neural networks for alternate tasks without modifying the network architecture or parameters. We propose a context based vocabulary remapping method that performs a computationally inexpensive input transformation to reprogram a victim classification model for a new set of sequences. We propose algorithms for training such an input transformation in both white box and black box settings where the adversary may or may not have access to the victim model's architecture and parameters. We demonstrate the application of our model and the vulnerability of neural networks by adversarially repurposing various text-classification models including LSTM, bi-directional LSTM and CNN for alternate classification tasks.

1 Introduction

While neural network based machine learning models serve as the backbone of many text and image processing systems, recent studies have shown that they are vulnerable to adversarial examples. Traditionally, an adversarial example is a sample from the classifier's input domain which has been perturbed in such a way that is intended to cause a machine learning model to misclassify it. While the perturbation is usually imperceptible, such an adversarial input results in the neural network model outputting an incorrect class label with higher confidence. Several studies have shown such adversarial attacks to be successful in both the continuous input domain (Szegedy et al., 2014; Papernot et al., 2016a, 2017, 2016b; Brown et al., 2017; Goodfellow et al., 2015) and discrete input spaces (Papernot et al., 2016c; Hu and Tan, 2018; Yang et al., 2018).

Adversarial Reprogramming (Elsayed et al., 2019) is a new class of adversarial attacks where a machine learning model is repurposed to perform a new task chosen by the attacker. The proposed attack is interesting because it allows an adversary to move a step beyond mere mis-classification of a victim network's output onto having the control to repurpose the network fully. The authors demonstrated how an adversary may repurpose a pre-trained ImageNet (Deng et al., 2009) model for an alternate classification task like classification of MNIST digits or CIFAR-10 images without modifying the network parameters. Since machine learning agents can be reprogrammed to perform unwanted actions as desired by the adversary, such an attack can lead to theft of computational resources such as cloud-hosted machine learning models. Besides theft of computational resources, the adversary may perform a task that violates the code of ethics of the system provider.

The adversarial reprogramming approach proposed by (Elsayed et al., 2019) trains an additive contribution θ to the inputs of the neural network to repurpose it for the desired alternate task. This approach assumes a white-box attack scenario where the adversary has full access to the network's parameters. Also, the adversarial program proposed in this work is only applicable to tasks where the input space of the the original and adversarial task is continuous.

In our work, we propose a method to adversarially *repurpose* neural networks which operate on sequences from a *discrete* input space. The task is to learn a simple transformation (adversarial program) from the input space of the adversarial task to the input space of the neural network such that the neural network can be repurposed for the adversarial task. We propose a context-based vocabulary remapping function as an adversarial program for sequence classification networks. We propose training procedures for this adversarial program in both white-box and black-

Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, pages 5216–5225,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

box scenarios. In the white-box attack scenario, where the adversary has access to the classifier's parameters, a Gumbel-Softmax trick (Jang et al., 2016) is used to train the adversarial program. Assuming a black-box attack scenario, where the adversary may not have access to the classifier's parameters, we present a REINFORCE (Williams, 1992) based optimization algorithm to train the adversarial program.

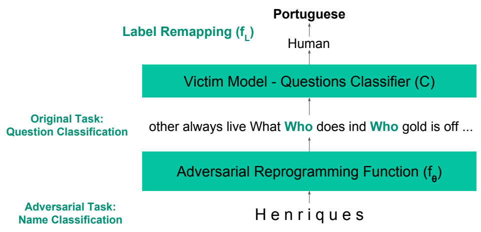

Portuguese
Label Remapping (f_L)
Human
Victim Model - Questions Classifier (C)
Original Task: Question Classification
other always live What Who does ind Who gold is off ...
Adversarial Reprogramming Function (f_θ)
Adversarial Task: Name Classification
H e n r i q u e s

Figure 1: Example of Adversarial Reprogramming for Sequence Classification. We aim to design and train the adversarial reprogramming function f_θ, such that it can be used to repurpose a pre-trained classifier C, for a desired adversarial task.

We apply our proposed methodology on various text classification models including Recurrent Neural Networks such as LSTMs and bidirectional LSTMs, and Convolutional Neural Networks (CNNs). We demonstrate experimentally, how these neural networks trained on a particular (original) text classification task can be repurposed for alternate (adversarial) classification tasks. We experiment with different text classification datasets given in table 1 as candidate original and adversarial tasks and adversarially reprogram the aforementioned text classification models to study the robustness of the attack.

2 Background and Related Work

2.1 Adversarial Examples

Traditionally, adversarial examples are intentionally designed inputs to a machine learning model that cause the model to make a mistake (Goodfellow et al., 2015). These attacks can be broadly classified into *untargeted* and *targeted* attacks. In the untargeted attack scenario, the adversary succeeds as long as the victim model classifies the adversarial input into *any* class other than the correct class, while in the *targeted* attack scenario, the adversary succeeds only if the model classifies the adversarial input into a specific incorrect class. In both these scenarios, the intent of the adversary is usually malicious and the outcome of the victim model is still limited to the original task being performed by the model.

Adversarial attacks of image-classification models often use gradient descent on an image to create a small perturbation that causes the machine learning model to mis-classify it (Szegedy et al., 2014; Biggio et al., 2013). There has been a similar line of adversarial attacks on neural networks with discrete input domains (Papernot et al., 2016c; Yang et al., 2018), where the adversary modifies a few tokens in the input sequence to cause mis-classification by a sequence model. In addition, efforts have been made in designing more general adversarial attacks in which the same modification can be applied to many different inputs to generate adversarial examples (Brown et al., 2017; Goodfellow et al., 2015; Moosavi-Dezfooli et al., 2017). For example, authors (Baluja and Fischer, 2018) trained an Adversarial Transformation Network that can be applied to all inputs to generate adversarial examples targeting a victim model or a set of victim models. In this work, we aim to learn such universal transformations of discrete sequences for a fundamentally different task: *Adversarial Reprogramming* described below.

2.2 Adversarial Reprogramming

Adversarial Reprogramming (Elsayed et al., 2019) introduced a new class of adversarial attacks where the adversary wishes to repurpose an existing neural network for a new task chosen by the attacker, without the attacker needing to compute the specific desired output. The adversary achieves this by first defining a hard-coded one-to-one label remapping function h_g that maps the output labels of the adversarial task to the label space of the classifier f; and learning a corresponding adversarial reprogramming function $h_f(.;\theta)$ that transforms an input $(\tilde{X})$ [1] from the input space of the new task to the input space of the classifier. The authors proposed an adversarial reprogramming function $h_f(.;\theta)$, for repurposing ImageNet models for adversarial classification tasks. An adversarial example X_{adv} for an input image $\tilde{X}$ can be generated using the following adversarial program: [2]

$$X_{adv} = h_f(\tilde{X};\theta) = \tilde{X} + \tanh(\theta)$$

[1] $\tilde{X}$ is an ImageNet size $(n \times n \times 3)$ padded input image

[2] Masking ignored because it is only a visualization convenience

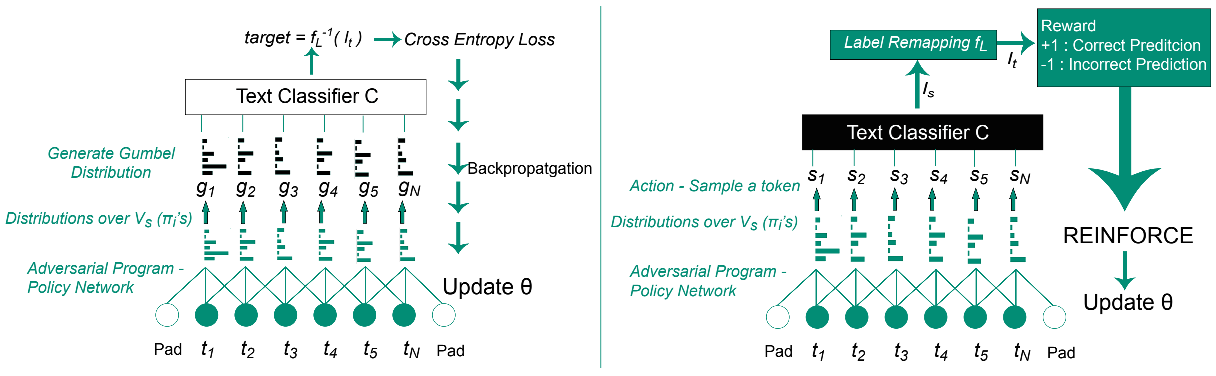

target = $f_L^{-1}(l_t)$
Cross Entropy Loss
Text Classifier C
Generate Gumbel Distribution
g_1 g_2 g_3 g_4 g_5 g_N
Backpropatgation
Distributions over V_S (π_i's)
Adversarial Program - Policy Network
Update θ
Pad t_1 t_2 t_3 t_4 t_5 t_N Pad
Label Remapping f_L
l_t
l_s
Reward
+1 : Correct Preditcion
-1 : Incorrect Prediction
Text Classifier C
Action - Sample a token
s_1 s_2 s_3 s_4 s_5 s_N
Distributions over V_S (π_i's)
Adversarial Program - Policy Network
REINFORCE
Update θ
Pad t_1 t_2 t_3 t_4 t_5 t_N Pad

Figure 2: **Adversarial Reprogramming Function and Training Procedures. Left:** *White-box* Adversarial Reprogramming. The adversary generates gumbel distributions g_i at each time-step which are passed as a soft version of one-hot vectors to the classifier C. The cross-entropy loss between the predictions and the mapped class is back-propagated to train the adversarial program θ. **Right:** *Black-box* Adversarial Reprogramming. The adversarial reprogramming function is used as a policy network and the sampled action (sequence s) is passed to the classifier C to get a reward based on prediction correctness. The adversarial program is then trained using REINFORCE.

where $\theta \in \mathbb{R}^{n \times n \times 3}$ is the learnable weight matrix of the adversarial program (where n is the ImageNet image width). Let $P(y|X)$ denote the probability of the victim model predicting label y for an input X. The goal of the adversary is to maximize the probability $P(h_g(y_{adv})|X_{adv})$ where y_{adv} is the label of the adversarial input X_{adv}. The following optimization problem that maximizes the log-likelihood of predictions for the adversarial classification task, can be solved using backpropagation to train the adversarial program parameterized by θ:

$$\hat{\theta} = argmin_{\theta} \left(-\log P(h_g(y_{adv})|X_{adv}) + \lambda||\theta||_2^2\right) \tag{1}$$

where λ is the regularization hyperparameter. Since the adversarial program proposed is a trainable additive contribution θ to the inputs, it's application is limited to neural networks with a continuous input space. Also, since the the above optimization problem is solved by back-propagating through the victim network, it assumes a white-box attack scenario where the adversary has gained access to the victim model's parameters. In this work, we describe how we can learn a simple transformation in the *discrete space* to extend the application of adversarial reprogramming on *sequence classification* problems. We also propose a training algorithm in the black-box setting where the adversary may not have access to the model parameters.

2.3 Transfer Learning

Transfer Learning (Raina et al., 2007) is a study closely related to Adversarial Reprogramming. During training, neural networks learn representations that are generic and can be useful for many related tasks. A pre-trained neural network can be effectively used as a feature extractor and the parameters of just the last few layers are retrained to realign the output layer of the neural network for the new task. Prior works have also applied transfer learning on text classification tasks (Do and Ng, 2006; Semwal et al., 2018). While transfer learning approaches exploit the learnt representations for the new task, they cannot be used to repurpose an exposed neural network for a new task without modifying some intermediate layers and parameters.

Adversarial Reprogramming (2019) studied whether it is possible to keep all the parameters of the neural network unchanged and simply learn an input transformation that can realign the outputs of the neural network for the new task. This makes it possible to repurpose *exposed* neural network models like cloud-based photo services to a new task where transfer learning is not applicable since we do not have access to intermediate layer outputs.

3 Methodology

3.1 Threat Model

Consider a sequence classifier C trained on the original task of mapping a sequence $s \in S$ to a class label $l_S \in L_S$ i.e $C : s \mapsto l_S$. An adversary wishes to repurpose the original classifier C for the adversarial task C' of mapping a sequence $t \in T$ to a class label $l_T \in L_T$ i.e $C' : t \mapsto l_T$. The adversary can achieve this by hard-coding a one-to-one label remapping function:

$$f_L : l_S \mapsto l_T$$

that maps an original task label to the new task label and learning a corresponding adversarial reprogramming function:

$$f_\theta : t \mapsto s$$

that transforms an input from the input space of the adversarial task to the input space of the original task. The adversary aims to update the parameters θ of the adversarial program f_θ such that the mapping $f_L(C(f_\theta(t)))$ can perform the adversarial classification task $C' : t \mapsto l_T$. Note that the victim model's parameters will be kept frozen while training the reprogramming function f_θ.

3.2 Adversarial Reprogramming Function

The goal of the adversarial reprogramming function $f_\theta : t \mapsto s$ is to map a sequence t to s such that it is labeled correctly by the classifier $f_L(C)$.

The tokens in the sequence s and t belong to some vocabulary lists V_S and V_T respectively. We can represent the sequence s as $s = s_1, s_2, .., s_N$ where s_i is the vocabulary index of the i_{th} token in sequence s in the vocabulary list V_S. Similarly sequence t can be represented as $t = t_1, t_2, .., t_N$ where t_i is the vocabulary index of the i_{th} token of sequence t in the vocabulary list V_T.

In the simplest scenario, the adversary may try to learn a vocabulary mapping from V_T to V_S using which each t_i can be independently mapped to some s_i to generate the adversarial sequence. Such an adversarial program has limited potential since the representational capacity of such a reprogramming function is very limited. We experimentally support this hypothesis by showing how such a transformation has limited potential for the purpose of adversarial reprogramming.

A more sophisticated adversarial program can be a sequence to sequence machine translation model (Sutskever et al., 2014) that learns a translation $t \mapsto s$ for adversarial reprogramming. While theoretically this is a good choice, it defeats the purpose of adversarial reprogramming. This is because the computational complexity of training and using such a machine translation model would be similar if not greater than that of a new sequence classifier for the adversarial task C'.

The adversarial reprogramming function should be computationally inexpensive but powerful enough for adversarial repurposing. To this end, we propose a context-based vocabulary remapping model that produces a distribution over the target vocabulary at each time-step based on the surrounding input tokens. More specifically, we define our adversarial program as a trainable 3-d matrix $\theta_{k\times|V_T|\times|V_S|}$ where k is the context size. Using this, we generate a probability distribution π_i over the vocabulary V_S at each time-step i as follows:

$$h_i = \sum_{j=0}^{k-1} \theta[j, t_{i+\lfloor k/2 \rfloor - j}] \qquad (2)$$

$$\pi_i = softmax(h_i) \qquad (3)$$

Both h_i and π_i are vectors of length $|V_S|$. To generate the adversarial sequence s we sample each s_i independently from the distribution π_i i.e $s_i \sim \pi_i$

In practice, during training, we implement this adversarial program as a single layer of 1-d convolution over the sequence of one-hot encoded vectors of adversarial tokens t_i with $|V_T|$ input channels and $|V_S|$ output channels with k-length kernels parameterized by $\theta_{k\times|V_T|\times|V_S|}$.

3.3 White-box Attack

In the white-box attack scenario, we assume that the adversary has gained access to the victim network's parameters and architecture. To train the adversarial reprogramming function f_θ, we use an optimization objective similar to equation 1. Let $P(l|s)$ denote the probability of predicting label l for a sequence s by classifier C. We wish to maximize the probability $P(f_L^{-1}(l_t)|f_\theta(t))$ which is the probability of the output label of the classifier being mapped to the correct class l_t for an input t in the domain of the adversarial task. Therefore we need to solve the following log-likelihood maximization problem:

$$\hat{\theta} = argmin_\theta(-\sum_t log(P(f_L^{-1}(l_t)|f_\theta(t)))) \qquad (4)$$

Note that that the output of the adversarial program $s = f_\theta(t)$ is a sequence of discrete tokens. This makes the above optimization problem non-differentiable. Prior works (Kusner and Hernández-Lobato, 2016; Gu et al., 2017; Yang et al., 2018) have demonstrated how we can smoothen such an optimization problem using the Gumbel-Softmax (Jang et al., 2016) distribution.

In order to backpropagate the gradient information from the classifier to the adversarial program, we smoothen the generated tokens s_i using Gumbel-Softmax trick as per the following:

For an input sequence t, we generate a sequence of Gumbel distributions $g = g_1, g_2, .., g_N$. The n_{th} component of distribution g_i is generated as follows:

$$g_i^n = \frac{\exp((\log(\pi_i^n) + r_n)/temp)}{\sum_j \exp((\log(\pi_i^j) + r_j)/temp)}$$

where π_i is the softmax distribution at the i_{th} time-step obtained using equation 3, r_n is a random number sampled from the Gumbel distribution (Gumbel, 1954) and $temp$ is the temperature of Gumbel-Softmax.

The sequence then passed to the classifier C is the sequence g which serves as a soft version of the one-hot encoded vectors of s_i's. Since the model is now differentiable, we can solve the following optimization problem using backpropagation:

$$\hat{\theta} = argmin_\theta(-\sum_t log(P(f_L^{-1}(l_t)|g)))$$

During training the temperature parameter is annealed from some high value t_{max} to a very low value t_{min}. The details of this annealing process for our experiments have been included in the appendix.

3.4 Black-box Attack

In the black-box attack scenario, the adversary can only query the victim classifier C for labels (and not the class probabilities). We assume that the adversary has the knowledge of the vocabulary V_S of the victim model. Since the adversarial program needs to produce a discrete output to feed as input to the classifier C, it is not possible to pass the gradient update from the classifier $f_L(C)$ to the adversarial program θ using standard backpropagation. Also, in the black-box attack setting it is not possible to back-propagate the cross entropy loss through the classifier C in the first place.

We formulate the sequence generation problem as a Reinforcement Learning problem (Bachman and Precup, 2015; Bahdanau et al., 2016; Yu et al., 2016) where the adversarial reprogramming function is the policy network. The state of the adversarial program is a sequence $t \in T$ and an action of our policy network is to produce a sequence of tokens $s \in S$. The adversarial program parameterized by θ, models the stochastic policy $\pi_{adv}(s|t;\theta)$ such that a sequence $s \in S$ may be sampled from this policy conditioned on $t \in T$. We use a simple reward function where we assign a reward +1 for a correct prediction and -1 for an incorrect prediction using the classifier $f_L(C)$ where f_L is the label remapping function and C is the classifier. Formally:

$$r(t,s) = \begin{cases} +1, & f_L(C(s)) = l_t \\ -1, & f_L(C(s)) \neq l_t \end{cases}$$

The optimization objective to train the policy network is the following:

$$\max_\theta J(\theta) \quad \text{where,} \quad J(\theta) = \mathbb{E}_{\pi_{adv}}[r(t,s)]$$

Following the REINFORCE algorithm (Williams, 1992) we can write the gradient of the expectation with respect to θ as per the following:

$$\begin{aligned}
\nabla_\theta J &= \nabla_\theta \left[\mathop{\mathbb{E}}_{\pi_{adv}} [r(t,s)] \right] \\
&= \nabla_\theta \left[\sum_s \pi_{adv}(s|t;\theta) r(t,s) \right] \\
&= \sum_s \pi_{adv}(s|t;\theta) \nabla_\theta \log(\pi_{adv}(s|t;\theta)) r(t,s) \\
&= \mathop{\mathbb{E}}_{\pi_{adv}} [r(t,s)\, \nabla_\theta \log(\pi_{adv}(s|t;\theta))] \\
&= \mathop{\mathbb{E}}_{\pi_{adv}} [r(t,s)\, \nabla_\theta \log(\pi_{adv}(s_1, .., s_N|t;\theta))] \\
&= \mathop{\mathbb{E}}_{\pi_{adv}} \left[r(t,s)\, \nabla_\theta \log(\prod_i \pi_{adv}(s_i|t;\theta)) \right] \\
&= \mathop{\mathbb{E}}_{\pi_{adv}} \left[r(t,s) \sum_i \nabla_\theta \log(\pi_{adv}(s_i|t;\theta)) \right]
\end{aligned}$$

Note that $\pi_{adv}(s_i|t;\theta)$ is the same as π_i defined in equation 3 which can be differentiated with respect to θ. The expectations are estimated as sample averages. Having obtained the gradient of expected reward, we can use mini-batch gradient ascent to update θ with a learning rate α as: $\theta \leftarrow \theta + \alpha \nabla_\theta J$.

4 Experiments

4.1 Datasets and Classifiers

We demonstrate the application of the proposed reprogramming techniques on various text-classification tasks. In our experiments, we design adversarial programs to attack both word-level and character-level text classifiers. Additionally, we aim to adversarially repurpose a character-level text classifier for a word-level classification task and vice-versa. To this end, we choose the following text-classification datasets as candidates for the original and adversarial classification tasks:

***1. Surname Classification Dataset* (Names-18, Names-5) (Robertson, 2017)**: The dataset categorizes surnames from 18 languages of origin. We use a subset of this dataset *Names-5* containing Names from 5 classes: *Dutch, Scottish, Polish, Korean* and *Portuguese*, as a candidate for adversarial task in the experiments.

***2. Experimental Data for Question Classification* (Questions) (Li and Roth, 2002)**: categorizes around 5500 questions into 6 classes: *Abbreviation, Entity, Description, Human, Location, Numeric*.

***3. Sentiment Classification Dataset (Abdulla et al., 2013)* (Arabic)** : contains 2000 binary labeled tweets on diverse topics such as politics and arts.

***4. Large Movie Review Dataset* (IMDB) for sentiment classification (Maas et al., 2011)**: contains 50,000 movie reviews categorized into binary class of positive and negative sentiment.

The statistics of the above mentioned datasets have been given in table 1. We train adversarial reprogramming functions to repurpose various text-classifiers based on Long Short-Term Memory (LSTM) network (Hochreiter and Schmidhuber, 1997), bidirectional LSTM network (Graves et al., 2005) and Convolutional neural network (Kim, 2014) models. All the aforementioned models can be trained for both word-level and character-level classification. We use character level classifiers for *Names-18* and *Names-5* datasets and word-level classifiers for *IMDB*, *Questions* and *Arabic* datasets. We use randomly initialized word/character embeddings which are trained along with the classification model parameters. For LSTM, we use the output at last timestep for prediction. For the Bi-LSTM, we combine the outputs of the first and last time step for prediction. For the Convolutional Neural Network we follow the same architecture as (Kim, 2014). Additionally, to understand the effectiveness of adversarial reprogramming, we train a sigle layer CNN with convolutional filter width 5 and discuss the results of this experiment in Section 4.4. The details of these classifiers have been included in table 2 of the supplementary material.

Data Set	# Classes	Train Samples	Test Samples	$\|V\|$	Avg Length
Names-18	18	115,028	28,758	90	7.1
Names-5	5	3632	909	66	6.5
Questions	6	4361	1091	1205	11.2
Arabic	2	1600	400	955	9.7
IMDB	2	25,000	25,000	10000	246.8

Table 1: Summary of datasets. $|V|$ denotes the vocabulary size of each dataset.

Data Set	LSTM	Bi-LSTM	CNN	1-CNN
Names-18	97.84	97.84	**97.88**	74.04
Names-5	**99.88**	**99.88**	99.77	71.51
Questions	96.70	**98.25**	98.07	83.77
Arabic	87.25	**88.75**	88.00	74.75
IMDB	86.83	89.43	**90.02**	83.32

Table 2: Test accuracy of various classification models. We use character-level models for *Names-5* and *Names-18* and word-level models for all other tasks. 1-CNN is a single layer CNN model with filter width 5.

4.2 Experimental Setup

As described in the methodology section, the label remapping function f_L we use, is a one-to-one mapping between the labels of the original task and the adversarial task. Therefore it is required to apply the constraint that the number of classes of the adversarial task are less than or equal to the number of classes of the original task. We choose *Names-5, Arabic* and *Question Classification* as candidates for the adversarial tasks and repurpose the models allowed under this constraint. We use context size $k = 5$ for all our experiments.

In white-box attacks, we use the Gumbel-Softmax based approach described in the methodology to train the adversarial program. The details of the temperature annealing process are included in table 1 of the supplementary material. For black-box attacks, we use the REINFORCE algorithm described in methodology, on mini-batches of sequences. Since the action space for certain reprogramming problems, (eg. reprogramming of IMDB classifier) is large ($|V_S| = 10000$), we restrict the output of the adversarial program to most frequent 1000 tokens in the vocabulary V_S. We use Adam optimizer (Kingma and Ba, 2015) for all our experiments. Hyperparameter details of all our experiments are included in table 1 of the supplementary material.

Original Task	Adversarial Task	Victim Model	Test Accuracy (%) Black box	White Box	White Box on Random Network
Questions	Names-5	LSTM	80.96	97.03	44.33
		Bi-LSTM	**93.51**	**99.66**	63.14
		CNN	88.90	99.22	93.06
Questions	Arabic	LSTM	73.50	87.50	50.00
		Bi-LSTM	81.75	83.50	70.00
		CNN	**82.25**	**87.25**	76.25
Names-18	Questions	LSTM	68.56	95.23	28.23
		Bi-LSTM	**94.96**	97.15	80.01
		CNN	71.03	**97.61**	33.45
Names-18	Arabic	LSTM	**83.00**	84.75	51.50
		Bi-LSTM	78.75	84.25	69.25
		CNN	80.75	**86.50**	60.00
IMDB	Arabic	LSTM	80.75	**88.25**	50.50
		Bi-LSTM	83.25	86.75	84.00
		CNN	**84.00**	87.00	84.25

Table 3: **Adversarial Reprogramming Experiments:** The accuracies of *white-box* and *black-box* reprogramming experiments on different combinations of original task, adversarial task and model. Figures in bold correspond to our best results on a particular adversarial task in the given attack scenario (black-box and white-box). *White-box on Random Network* column presents results of the white-box attack on an untrained neural network. Context size $k = 5$ is used for all our experiments.

4.3 Results and Discussion

The accuracies of all adversarial reprogramming experiments have been reported in table 3. To interpret the results in context, the accuracies achieved by the LSTM, Bi-LSTM and CNN text classification models on the adversarial tasks can be found in table 1.

We demonstrate how character-level models trained on Names-18 dataset can be repurposed for word-level sequence classification tasks like Question Classification and Arabic Tweet Sentiment Classification. Similarly, word-level classifiers trained on Question Classification Dataset can be repurposed for the character-level Surname classification task. Interestingly, classifiers trained on IMDB Movie Review Dataset can be repurposed for Arabic Tweet Sentiment Classification even though there is a high difference between the vocabulary size (10000 vs 955) and average sequence length(246.8 vs 9.7) of the two tasks. It can be seen that all of the three classification models are susceptible to adversarial reprogramming in both white-box and black-box setting.

White-box based reprogramming outperforms the black-box based approach in all of our experiments. In practice, we find that training the adversarial program in the black-box scenario requires careful hyper-parameter tuning for REINFORCE to work. We believe that improved reinforcement learning techniques for sequence generation tasks (Bahdanau et al., 2016; Bachman and Precup, 2015) can make the training procedure for black-box attack more stable. We propose such improvement as a direction of future research.

To assess the importance of the original task on which the network was trained, we also present results of white-box adversarial reprogramming on untrained random network. Our results are coherent with similar experiments on adversarial reprogramming of untrained ImageNet models (Elsayed et al., 2019) demonstrating that adversarial reprogramming is less effective when it targets untrained networks. The figures in table 3 suggest that the representations learned by training a text classifier on an original task, are important for repurposing it for an alternate task. However another plausible reason as discussed by Elsayed et al. is that the reduced performance on random networks might be because of simpler reasons like poor scaling of network weight initialization making the optimization problem harder.

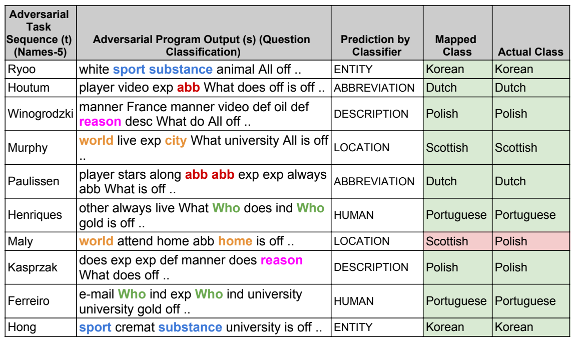
4.4 Complexity of Reprogramming Function

As motivated earlier in Section 3.2, computational efficiency of the adversarial program is critical for adversarial reprogramming to be of interest to an adversary. If the adversary can achieve the desired results using a computationally inexpensive classifier, it defeats the purpose of adversarial reprogramming. To understand if this is the case, we train a one-layer CNN with the same convolutional filter width as our adversarial program and average the activations across all time-steps to classify a given phrase. The results of such a classifier on various datasets have been reported in Table 2. We can observe that our white-box attack on pretrained networks, outperforms this classifier in all scenarios (refer to Table 3). Our best black-box attacks also outperform a one-layer CNN for all adversarial tasks. This experiment demonstrates that the reprogramming function exploits the learned feature representation of the victim model. Also, the observation that adversarial reprogramming is significantly less effective on randomly initialized untrained networks further reinforces the importance of utilizing a trained victim model.

Since the reprogramming function is a context based vocabulary remapping function, we can implement it as a look-up table that maps a combination of k tokens from the vocabulary V_t to a token in the source vocabulary V_s. The time complexity for transforming a sentence t to an adversarial sentence is just $O(length(t))$.

4.4.1 Adversarial Sequences

Figure 3 shows some adversarial sequences generated by the adversarial program for Names-5 Classification while attacking a CNN trained on the Question Classification dataset. A sequence t in the first column is transformed into the adversarial sequence s in the second column by the trained adversarial reprogramming function. While these adversarial sequences may not make semantic or grammatical sense, it exploits the learned representation of the classifier to map the inputs to the desired class. For example, sequences that should be mapped to HUMAN class have words like *Who* in the generated adversarial sequence. Similarly, sequences that should be mapped to LOCATION class have words like *world, city* in the adversarial sequence. Other such interpretable transformations are depicted via colored text in the adversarial sequences of Figure 3.



Adversarial Task Sequence (t) (Names-5)	Adversarial Program Output (s) (Question Classification)	Prediction by Classifier	Mapped Class	Actual Class
Ryoo	white **sport substance** animal All off ..	ENTITY	Korean	Korean
Houtum	player video exp **abb** What does off is off ..	ABBREVIATION	Dutch	Dutch
Winogrodzki	manner France manner video def oil def **reason** desc What do All off ..	DESCRIPTION	Polish	Polish
Murphy	**world** live exp **city** What university All is off ..	LOCATION	Scottish	Scottish
Paulissen	player stars along **abb abb** exp exp always abb What is off ..	ABBREVIATION	Dutch	Dutch
Henriques	other always live What **Who** does ind **Who** gold is off ..	HUMAN	Portuguese	Portuguese
Maly	**world** attend home abb **home** is off ..	LOCATION	Scottish	Polish
Kasprzak	does exp exp def manner does **reason** What does off ..	DESCRIPTION	Polish	Polish
Ferreiro	e-mail **Who** ind exp **Who** ind university university gold off ..	HUMAN	Portuguese	Portuguese
Hong	**sport** cremat **substance** university is off ..	ENTITY	Korean	Korean

Figure 3: Adversarial sequences generated by our adversarial program for Names-5 Classification (adversarial task), when targeting a CNN trained on the Question Classification dataset (original task). Interpretable transformations are shown as colored words in the second column.

4.4.2 Effect of Context Size

By varying the context size k of the convolutional kernel $\theta_{k \times |V_T| \times |V_S|}$ in our adversarial program we are able to control the representational capacity of the adversarial reprogramming function. Figure 4 shows the percentage accuracy obtained when training the adversarial program with different context sizes k on two different adversarial tasks: Arabic Tweets Classification and Name Classification. Using a context size $k = 1$ reduces the adversarial reprogramming function to simply a vocabulary remapping function from V_S to V_T. It can be observed that the performance of the adversarial reprogramming model at $k = 1$ is significantly worse than that at higher values of k. While higher values of k improve the performance of the adversarial program, they come at a cost of increased computational complexity and memory required for the adversarial reprogramming function. For the adversarial tasks studied in this paper, we observe that $k = 5$ is a reasonable choice for context size of the adversarial program.

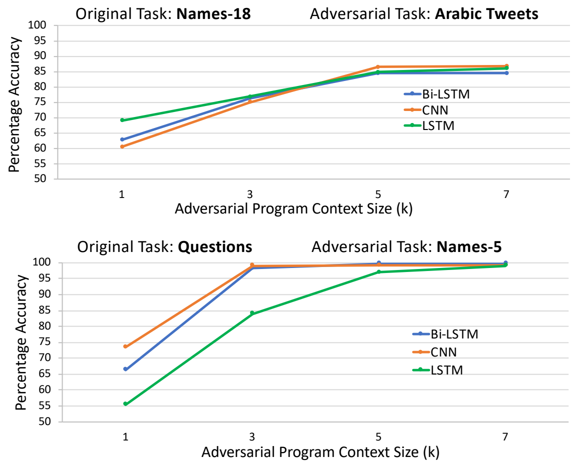


Figure 4: Accuracy vs Context size (k) plots for all 3 classification models on 2 different adversarial reprogramming experiments.

5 Conclusion

In this paper, we extend adversarial reprogramming, a new class of adversarial attacks, to target text classification neural networks. Our results demonstrate the effectiveness of such attacks in the more challenging black-box settings, posing them as a strong threat in real-world attack scenarios. We demonstrate that neural networks can be effectively reprogrammed for alternate tasks, which were not originally intended by a service provider. Our proposed end-to-end approach can be used to further understand the vulnerabilities and blind spots of deep neural network based text classification systems. We recommend future work to study the scope of adversarial reprogramming for other NLP applications such as machine translation, text to speech synthesis and text to image synthesis where the input space is discrete. Furthermore, due to the threat presented by adversarial reprogramming, we recommend future work to study defenses against such attacks.

References

N. A. Abdulla, N. A. Ahmed, M. A. Shehab, and M. Al-Ayyoub. 2013. Arabic sentiment analysis: Lexicon-based and corpus-based. In *2013 IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies (AEECT)*.

Philip Bachman and Doina Precup. 2015. Data generation as sequential decision making. In *Advances in Neural Information Processing Systems*, pages 3249–3257.

Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron C. Courville, and Yoshua Bengio. 2016. An actor-critic algorithm for sequence prediction. *CoRR*, abs/1607.07086.

Shumeet Baluja and Ian Fischer. 2018. Learning to attack: Adversarial transformation networks. In *AAAI*.

Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. 2013. Evasion attacks against machine learning at test time. In *Machine Learning and Knowledge Discovery in Databases*.

Tom B. Brown, Dandelion Mané, Aurko Roy, Martín Abadi, and Justin Gilmer. 2017. Adversarial patch. *CoRR*, abs/1712.09665.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. 2009. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*.

Chuong B. Do and Andrew Y. Ng. 2006. Transfer learning for text classification. In Y. Weiss, B. Schölkopf, and J. C. Platt, editors, *Advances in Neural Information Processing Systems 18*. MIT Press.

Gamaleldin F. Elsayed, Ian J. Goodfellow, and Jascha Sohl-Dickstein. 2019. Adversarial reprogramming of neural networks. In *ICLR*.

Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*.

Alex Graves, Santiago Fernández, and Jürgen Schmidhuber. 2005. Bidirectional lstm networks for improved phoneme classification and recognition. In *Proceedings of the 15th International Conference on Artificial Neural Networks: Formal Models and Their Applications - Volume Part II*, ICANN'05, pages 799–804, Berlin, Heidelberg. Springer-Verlag.

Jiatao Gu, Daniel Jiwoong Im, and Victor O. K. Li. 2017. Neural machine translation with gumbel-greedy decoding. *CoRR*, abs/1706.07518.

E.J. Gumbel. 1954. *Statistical theory of extreme values and some practical applications: a series of lectures*. Applied mathematics series. U. S. Govt. Print. Office.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8).

Weiwei Hu and Ying Tan. 2018. Black-box attacks against rnn based malware detection algorithms. In *Workshops at the Thirty-Second AAAI Conference on Artificial Intelligence*.

Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. *CoRR*, abs/1611.01144.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP*.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In Proc. *ICLR*.

Matt J. Kusner and José Miguel Hernández-Lobato. 2016. GANS for sequences of discrete elements with the gumbel-softmax distribution. *CoRR*, abs/1611.04051.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *Proceedings of the 19th International Conference on Computational Linguistics - Volume 1*, COLING '02, Stroudsburg, PA, USA. Association for Computational Linguistics.

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 *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, Portland, Oregon, USA.

S. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard. 2017. Universal adversarial perturbations. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 86–94.

Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. 2017. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*. ACM.

Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. 2016a. The limitations of deep learning in adversarial settings. In *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE.

Nicolas Papernot, Patrick D. McDaniel, and Ian J. Goodfellow. 2016b. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *CoRR*, abs/1605.07277.

Nicolas Papernot, Patrick D. McDaniel, Ananthram Swami, and Richard E. Harang. 2016c. Crafting adversarial input sequences for recurrent neural networks. *CoRR*, abs/1604.08275.

Rajat Raina, Alexis Battle, Honglak Lee, Benjamin Packer, and Andrew Y. Ng. 2007. Self-taught learning: Transfer learning from unlabeled data. In *Proceedings of the Twenty-fourth International Conference on Machine Learning*.

Sean Robertson. 2017. Classifying names with a character-level rnn - pytroch tutorial. `https://pytorch.org/tutorials/intermediate/char_rnn_classification_tutorial.html`.

Tushar Semwal, Gaurav Mathur, Promod Yenigalla, and Shivashankar B. Nair. 2018. A practitioners' guide to transfer learning for text classification using convolutional neural networks. *CoRR*, abs/1801.06480.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. In *ICLR*.

Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4).

Puyudi Yang, Jianbo Chen, Cho-Jui Hsieh, Jane-Ling Wang, and Michael I. Jordan. 2018. Greedy attack and gumbel attack: Generating adversarial examples for discrete data. *CoRR*, abs/1805.12316.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2016. Seqgan: Sequence generative adversarial nets with policy gradient. *CoRR*, abs/1609.05473.