Knowing When to Stop: Evaluation and Verification of Conformity to Output-size Specifications

Chenglong Wang♠
University of Washington
clwang@cs.washington.edu

Rudy Bunel♠
University of Oxford
rudy@robots.ox.ac.uk

Krishnamurthy Dvijotham
DeepMind
dvij@google.com

Po-Sen Huang
DeepMind
posenhuang@google.com

Edward Grefenstette♣
DeepMind
egrefen@fb.com

Pushmeet Kohli
DeepMind
pushmeet@google.com

Abstract

Models such as Sequence-to-Sequence *and* Image-to-Sequence *are widely used in real world applications. While the ability of these neural architectures to produce variable-length outputs makes them extremely effective for problems like Machine Translation and Image Captioning, it also leaves them vulnerable to failures of the form where the model produces outputs of undesirable length. This behaviour can have severe consequences such as usage of increased computation and induce faults in downstream modules that expect outputs of a certain length. Motivated by the need to have a better understanding of the failures of these models, this paper proposes and studies the novel output-size modulation problem and makes two key technical contributions. First, to evaluate model robustness, we develop an easy-to-compute differentiable proxy objective that can be used with gradient-based algorithms to find output-lengthening inputs. Second and more importantly, we develop a verification approach that can formally verify whether a network always produces outputs within a certain length. Experimental results on Machine Translation and Image Captioning show that our output-lengthening approach can produce outputs that are* 50 *times longer than the input, while our verification approach can, given a model and input domain, prove that the output length is below a certain size.*

1. Introduction

Neural networks with variable output lengths have become ubiquitous in several applications. In particular, recurrent neural networks (RNNs) such as LSTMs [17], used to form "sequence" models [30], have been successfully and extensively applied in in image captioning [34, 28, 21, 6, 12, 37, 26, 24, 1], video captioning [33, 38, 35, 40, 41], machine translation (MT) [30, 9], summarization [10], and in other sequence-based transduction tasks.

The ability of these sequence neural models to generate variable-length outputs is key to their performance on complex prediction tasks. However, this ability also opens a powerful attack for adversaries that try to force the model to produce outputs of specific lengths that, for instance, lead to increased computation or affect the correct operation of down-stream modules. To address this issue, we introduce the *output-length modulation* problem where given a *specification* of the form that the model should produce outputs with less than a certain maximum length, we want to find adversarial examples, *i.e.* search for inputs that lead the model to produce outputs with a larger length and thus show that the model under consideration violates the specification. Different from existing work on targeted or untargeted attacks where the goal is to perturb the input such that the output is another class or sequence in the development dataset (thus within the dataset distribution), the output-modulation problem requires solving a more challenging task of finding inputs such that the output sequences are outside of the training distribution, which was previously claimed difficult [5].

The naive approach to the solution of the output-length modulation problem involves a computationally intractable search over a large discrete search space. To overcome this, we develop an easy-to-compute differentiable proxy objective that can be used with gradient-based algorithms to find output-lengthening inputs. Experimental results on Machine Translation show that our adversarial output-lengthening approach can produce outputs that are 50 times longer than the input. However, when evaluated on the image-to-text image captioning model, the method is less

♠work done during an internship at DeepMind
♣now at Facebook AI Research

successful. There could have been two potential reasons for this result: the image-to-text architecture is truly robust, or the adversarial approach is not powerful enough to find adversarial examples for this model. To resolve this question, we develop a verification method for checking and formally proving whether a network is consistent with the output-size specification for the given range of inputs. To the best of our knowledge, our verification algorithm is the first formal verification approach to check properties of recurrent models with variable output lengths.

Our Contributions To summarize, the key contributions of this paper are as follows:

- We propose and formulate the novel output-size modulation problem to study the behaviour of neural architectures capable of producing variable length outputs, and we study its evaluation and verification problems.
- For evaluation, we design an efficiently computable differentiable proxy for the expected length of the output sequence. Experiments show that this proxy can be optimized using gradient descent to efficiently find inputs causing the model to produce long outputs.
- We demonstrate that popular machine translation models can be forced to produce long outputs that are 50 times longer than the input sequence. The long output sequences help expose modes that the model can get stuck in, such as undesirable loops where they continue to emit a specific token for several steps.
- We demonstrate the feasibility of formal verification of recurrent models by proposing the use of mixed-integer programming to formally verify that a certain neural image-captioning model will be consistent with the specification for the given range of inputs.

Motivations and Implications Our focus on studying the output-length modulation problem is motivated by the following key considerations:

- *Achieving Computational Robustness:* Many ML models are now offered as a service to customers via the cloud. In this context, ML services employing variable-output models could be vulnerable to denial-of-service attacks that cause the ML model to perform wasteful computations by feeding it inputs that induce long outputs. This is particularly relevant for *variable compute models*, like Seq2Seq [9, 30]. Given a trained instance of the model, no method is known to check for the consistency of the model with a specification on the number of computation steps. Understanding the vulnerabilities of ML models to such output-lengthening and computation-increasing attacks is important for the safe deployment of ML services.
- *Understanding and Debugging Models:* By designing inputs that cause models to produce long outputs, it is possible to reason about the internal representations learned by the model and isolate where the model exhibits undesirable behavior. For example, we find that an English to German sequence-to-sequence model can produce outputs that end with a long string of question marks ('?'). This indicates that when the output decoder state is conditioned on a sequence of '?'s, it can end up stuck in the same state.
- *Uncovering security vulnerabilities through adversarial stress-testing:* The adversarial approach to output-length modulation tries to find parts of the space of inputs where the model exhibits improper behavior. Such inputs does not only reveal abnormal output size, but could also uncover other abnormalities like the privacy violations of the kind that were recently revealed by [4] where an LSTM was forced to output memorized data.
- *Canonical specification for testing generalization of variable-output models:* Norm-bounded perturbations of images [31] have become the standard specification to test attacks and defenses on image classifiers. While the practical relevance of this particular specification can be questioned [14], it is still served as a useful canonical model encapsulating the essential difficulty in developing robust image classifiers. We believe stability of output-lengths can serve a similar purpose: as a canonical specification for variable output-length models. The main difficulties in studying variable output length models in an adversarial sense (the non-differentiability of the objective with respect to inputs) are exposed in output-lengthening attack, making it a fertile testing ground for both evaluating attack methods and defenses. We hope that advances made here will facilitate the study of robustness on *variable compute models* and other specifications for variable-output models such as monotonicity.

2. Related Work

There are several recent studies on generating adversarial perturbations on variable-output models. [27, 20] show that question answering and machine comprehension models are sensitive to attacks based on semantics preserving modification or the introduction of unrelated information. [11, 39] find that character-level classifiers are highly sensitive to small character manipulations. [29] shows that models predicting the correctness of image captions struggle against perturbations consisting of a single word change. [5] and [8] further study adversarial attacks for sequential-output models (machine-translation, image captioning) with specific target captions or keywords.

We focus on sequence output models and analyze the output-length modulation problem, where the models should produce outputs with at least a certain number of output tokens. We study whether a model can be adversarially perturbed to change the size of the output, which is a more challenging task compared to targeted attacks (see details in Section 3). On the one hand, existing targeted attack tasks aim to perturb the input such that the output is another sequence in the validation dataset (thus within the training distribution), but attacking output size requires the model to generate out-of-distribution long sequences. On the other hand, since the desired output sequence is only loosely constrained by the length rather than directly provided by the user, the attack algorithm is required to explore the output size to make the attack possible.

For models that cannot be adversarially perturbed, we develop a verification approach to show that it isn't simply a lack of power by the adversary but the sign of true robustness from the model. Similar approaches have been investigated for feedforward networks [3, 7, 32] but our work is the first to handle variable output length models and the corresponding decoding mechanisms.

3. Modulating Output-size

We study neural network models capable of producing outputs of variable length. We start with a canonical abstraction of such models, and later specialize to concrete models used in machine translation and image captioning.

We denote by x the input to the network and by $\mathcal{X}$ the space of all inputs to the network. We consider a set of inputs of interest $\mathcal{S}$, which can denote, for example, the set of "small"[1] perturbations of a nominal input. We study models that produce variable-length outputs sequentially. Let $y_t \in \mathcal{Y}$ denote the t-th output of the model, where $\mathcal{Y}$ is the output vocabulary of the model. At each timestep, the model defines a probability over the next element $P(y_{t+1}|x, y_{0:t})$. There exists a special end-of-sequence element $\mathbf{eos} \in \mathcal{Y}$ that signals termination of the output sequence.

In practice, different models adopt different decoding strategies for generating y_{t+1} from the probability $P(y_{t+1}|x, y_{0:t})$ [13, 19, 22]. In this paper, we focus on the commonly used deterministic *greedy decoding* strategy [13], where at each time step, the generated token is given by the argmax over the logits:

$$y_0 = \operatorname{argmax}\{P(\cdot|x)\} \tag{1a}$$

$$y_{t+1} = \operatorname{argmax}\{P(\cdot|x, y_{0:t})\} \text{ if } y_t \neq \mathbf{eos} \tag{1b}$$

Since greedy decoding is deterministic, for a given sample x with a finite length output, we can define the length of the greedily decoded sequence as:

$$\ell(x) = t \text{ s.t } \begin{array}{ll} y_t = \mathbf{eos} & \\ y_i \neq \mathbf{eos} & \forall i < t \\ y_{i+1} = \operatorname{argmax}\{P(.|x, y_{0:i})\} & \forall i < t \end{array} \tag{2}$$

Note that there is a unique t that satisfies the above conditions, which is precisely the first t at which $y_t = \mathbf{eos}$ when using greedy decoding.

Output length modulation specification A network is said to satisfy the *output length modulation specification* parameterized by $\mathcal{S}, \hat{K}$, if for all inputs x in $\mathcal{S}$, the model terminates within $\hat{K}$ steps under greedy decoding for all $x \in \mathcal{S}$, formally:

$$\forall x \in \mathcal{S} \qquad \ell(x) \leq \hat{K} \tag{3}$$

In Section 3.1, we study the problem of finding adversarial examples, i.e., searching for inputs that lead the model to produce outputs with a larger length and thus show that the model violates the specification. In Section 4, we use formal verification method to prove that a model is consistent with the specification for the given range of inputs, if such attacks are indeed impossible.

3.1. The Output-Size Modulation Problem

In order to check whether the specification, Eq. (3), is valid, one can consider a falsification approach that tries to find counterexamples proving that Eq. (3) is false. If an exhaustive search over $\mathcal{S}$ for such counterexamples fails, the specification is indeed true. However, exhaustive search is computationally intractable; hence, in this section we develop gradient based algorithms that can efficiently find counterexamples (although they may miss them even if they exist). To develop the falsification approach, we study the solution to the following optimization objective:

$$\max_{x \in \mathcal{S}} \ell(x) \tag{4}$$

where $\mathcal{S}$ is the valid perturbation space. If the optimal solution x in the space $\mathcal{S}$ has $\ell(x) > \hat{K}$, then (3) is false.

The attack spaces $\mathcal{S}$ we consider in this paper include both continuous inputs (for image-to-text models) and discrete inputs (for Seq2Seq models).

Continuous inputs: For continuous inputs, such as image captioning tasks, the input is an $n \times m$ image with pixel values normalized to be in the range $[-1, 1]$. x is an $n \times m$ matrix of real numbers and $\mathcal{X} = [-1, 1]^{n \times m}$. We define the perturbation space $\mathcal{S}(x, \delta)$ as follows:

$$\mathcal{S}(x, \delta) = \{x' \in \mathcal{X} \mid \|x' - x\|_\infty \leq \delta\}$$

i.e., the space of δ perturbations of the input x in the ℓ_∞ ball.

[1] The precise definition of small is specific to the application studied.

Discrete inputs: For discrete inputs, e.g., machine translation tasks, inputs are discrete tokens in a language vocabulary. Formally, given the vocabulary V of the input language, the input space $\mathcal{X}$ is defined as all sentences composed of tokens in V, i.e., $\mathcal{X} = \{(x_1, \ldots, x_n) \mid x_i \in V, n > 0\}$. Given an input sequence $x = (x_1, \ldots, x_n)$, we define the δ-perturbation space of a sequence as all sequences of length n with at most $\lceil \delta \cdot n \rceil$ tokens different from x (i.e., $\delta \in [0, 1]$ denotes the percentage of tokens that an attacker is allowed to modify). Formally, the perturbation space $\mathcal{S}(x, \delta)$ is defined as follows:

$$\mathcal{S}(x, \delta) = \{(x'_1, \ldots, x'_n) \in V^n \mid \sum_{i=1}^{n} \mathbb{1}[x_i \neq x'_i] \leq \lceil \delta \cdot n \rceil\}$$

3.2. Extending Projected Gradient Descent Attacks

In the projected gradient descent (PGD) attacks [25],[2] given an objective function $J(x)$, the attacker calculates the adversarial example by searching for inputs in the attack space to maximize $J(x)$. In the basic attack algorithm, we perform the following updates at each iteration:

$$x' = \Pi_{\mathcal{S}(x,\delta)} \left(x + \alpha \nabla_x J(x)\right) \tag{5}$$

where $\alpha > 0$ is the step size and $\Pi_{\mathcal{S}(x,\delta)}$ denotes the projection of the attack to the valid space $\mathcal{S}(x, \delta)$. Observe that the adversarial objective in Eq. (4) cannot be directly used as $J(x)$ to update x as the length of the sequence is not a differentiable objective function. This hinders the direct application of PGD to output-lengthening attacks. Furthermore, when the input space $\mathcal{S}$ is discrete, gradient descent cannot be directly be used because it is only applicable to continuous input spaces.

In the following, we show our extensions of the PGD attack algorithm to handle these challenges.

Greedy approach for sequence lengthening We introduce a differentiable proxy of $\ell(x)$. Given an input x whose decoder output logits are $(o_1, \ldots, o_k)$ (i.e., the decoded sequence is $y = (\text{argmax}(o_1), \ldots, \text{argmax}(o_k))$), instead of directly maximizing the output sequence length, we use a greedy algorithm to find an output sequence whose length is longer than k by minimizing the probability of the model to terminate within k steps. In other words, we minimize the log probability of the model to produce **eos** at any of the timesteps between 1 to k. Formally, the proxy objective $\tilde{J}$ is defined as follows:

$$\tilde{J}(x) = \sum_{t=1}^{k} \max \left\{ o_t[\mathbf{eos}] - \max_{z \neq \mathbf{eos}} o_t[z],\ -\epsilon \right\}$$

where $\epsilon > 0$ is a hyperparameter to clip the loss. This is piecewise differentiable w.r.t. the inputs x (in the same sense that the ReLU function is differentiable) and can be efficiently optimized using PGD.

[2]Here the adversarial objective is stated as maximization, so the algorithm is Projected Gradient *Ascent*, but we stick with the PGD terminology since it is standard in the literature

3.3. Continuous relaxation for discrete inputs

While we can apply the PGD attack with the proxy objective on the model with continuous inputs by setting the projection function $\Pi_{\mathcal{S}(x,\delta)}$ as the Euclidean projection, we cannot directly update discrete inputs. To enable a PGD-type attack in the discrete input space, we use the Gumbel trick [18] to reparameterize the input space to perform continuous relaxation of the inputs.

Given an input sequence $x = (x_1, \ldots, x_n)$, for each x_i, we construct a distribution $\pi_i \in \mathbb{R}^{|V|}$ initialized with $\pi_i[x_i] = 1$ and $\pi_i[z] = -1$ for all $z \in V \setminus \{x_i\}$. The softmax function applied to π_i is a probability distribution over input tokens at position i with a mode at x_i. With this reparameterization, instead of feeding $x = (x_1, \ldots, x_n)$ into the model, we feed the Gumbel softmax sampling from the distribution $(u_1, \ldots, u_n)$. The sample $\tilde{x} = (\tilde{x}_1, \ldots, \tilde{x}_n)$ is calculated as follows:

$$u_i \sim \text{Uniform}(0, 1); \quad g_i = -\log(-\log(u_i))$$
$$p = \text{softmax}(\pi); \quad \tilde{x}_i = \text{softmax}(\tfrac{g_i + \log p_i}{\tau})$$

where τ is the Gumbel-softmax sampling temperature that controls the discreteness of $\tilde{x}$. With this relaxation, we perform PGD attack on the distribution π at each iteration. Since π_i is unconstrained, the projection step in (5) is unnecessary.

When the final $\pi' = (\pi'_1, \ldots, \pi_n)$ is obtained from the PGD attack, we draw samples $x'_i \sim \text{Categorical}(\pi_i)$ to get the final adversarial example for the attack.

4. Verified Bound on Output Length

While heuristics approaches can be useful in finding attacks, they can fail due to the difficulty of optimizing non-differentiable nonconvex functions. These challenges show up particularly when the perturbation space is small or when the target model is trained with strong bias in the training data towards short output sequences (e.g., the Show-and-Tell model as we will show in Section 6). Thus, we design a formal verification approach for complete reasoning of the output-size modulation problem, i.e., finding provable guarantees that no input within a certain set of interest can result in an output sequence of length above a certain threshold.

Our approach relies on counterexample search using intelligent brute-force search methods, taking advantage of powerful modern integer programming solvers [15]. We encode all the constraints that an adversarial example should satisfy as linear constraints, possibly introducing additional binary variables. Once in the right formalism, these can be fed into an off-the-shelf Mixed Integer Programming (MIP) solver, which provably solves the problem, albeit with a potentially large computational cost. The constraints consist

of four parts: (1) the initial restrictions on the model inputs (encoding $\mathcal{S}(x, \delta)$), (2) the relations between the different activations of the network (implementing each layer), (3) the decoding strategy (connection between the output logits and the inputs at the next step), and (4) the condition for it being a counterexample (*ie*. a sequence of length larger than the threshold). In the following, we show how each part of the constraints is encoded into MIP formulas.

Our formulation is inspired by pior work on encoding feed-forward neural networks as MIPs [3, 7, 32]. The image captioning model we use consists of an image embedding model, a feedforward convolutional neural network that computes an embedding of the image, followed by a recurrent network that generates tokens sequentially starting with the initial hidden state set to the image embedding.

The image embedding model is simply a sequence of linear or convolutional layers and ReLU activation functions. Linear and convolutional layers are trivially encoded as linear equality constraints between their inputs and outputs, while ReLUs are represented by introducing a binary variable and employing the big-M method [16]:

$$x_i = \max(\hat{x}_i, 0) \Rightarrow \delta_i \in \{0,1\}, \quad x_i \geq 0 \tag{6a}$$
$$x_i \leq u_i \cdot \delta_i, \quad x_i \geq \hat{x}_i \tag{6b}$$
$$x_i \leq \hat{x}_i - l_i \cdot (1 - \delta_i) \tag{6c}$$

with l_i and u_i being lower and upper bounds of $\hat{x_i}$ which can be obtained using interval arithmetic (details in [3]).

Our novel contribution is to introduce a method to extend the techniques to handle greedy decoding used in recurrent networks. For a model with greedy decoding, the token with the most likely prediction is fed back as input to the next time step. To implement this mechanism as a mixed integer program, we employ a big-M method [36]:

$$o_{\text{max}} = \max_{y \in \mathcal{Y}}(o_y)$$
$$\Rightarrow \quad o_{\text{max}} \geq o_y, \quad \delta_y \in \{0,1\} \quad \forall y \in \mathcal{Y} \tag{7a}$$
$$o_{\text{max}} \leq o_y + (\mathbf{u} - l_y)(1 - \delta_y) \quad \forall y \in \mathcal{Y} \tag{7b}$$
$$\sum_{y \in \mathcal{Y}} \delta_y = 1 \tag{7c}$$

with l_y, u_y being a lower/upper bound on the value of o_y and $\mathbf{u} = \max_{y \in \mathcal{Y}} u_y$ (these can again be computed using interval arithmetic). Implementing the maximum in this way gives us both a variable representing the value of the maximum (o_{max}), as well as a one-hot encoding of the argmax (δ_y). If the embedding for each token is given by $\{\text{emb}_i \mid i \in \mathcal{Y}\}$, we can simply encode the input to the following RNN timestep as $\sum_{y \in \mathcal{Y}} \delta_y \cdot \text{emb}_y$, which is a linear function of the variables that we previously constructed.

With this mechanism to encode the greedy decoding, we can now unroll the recurrent model for the desired number of timesteps. To search for an input x with output length $\ell(x) \geq \hat{K}$, we unroll the recurrent network for $\hat{K}$ steps and attempt to prove that at each timestep, **eos** is not the maximum logit, as in (2). We setup the problem as:

$$\max \min_{t=1..\hat{K}} \left[\max_{z \neq \textbf{eos}} o_t[z] - o_t[\textbf{eos}] \right] \tag{8}$$

where $o(k)$ represents the logits in the k-th decoding step. We use an encoding similar to the one of Equation (7) to represent the objective function as a linear objective with added constraints. If the global optimal value of Eq. (8) is positive, this is a valid counterexample: at all timesteps $t \in [1..\hat{K}]$, there is at least one token greater than the **eos** token, which means that the decoding should continue. On the other hand, if the optimal value is negative, that means that those conditions cannot be satisfied and that it is not possible to generate a sequence of length greater than $\hat{K}$. The **eos** token would necessarily be predicted before. This would imply that our robustness property is True.

5. Target Model Mechanism

We use image captioning and machine translation models as specific target examples to study the output length modulation problem. We now introduce their mechanism.

Image captioning models The image captioning model we consider is an encoder-decoder model composed of two modules: a convolution neural network (CNN) as an encoder for image feature extraction and a recurrent neural network (RNN) as a decoder for caption generation [34].

Formally, the input to the model x is an $m \times n$ sized image from the space $\mathcal{X} = [-1, 1]^{m \times n}$, the CNN-RNN model computes the output sequence as follows:

$$i_0 = \text{CNN}(x); \quad h_0 = \mathbf{0}$$
$$o_t, h_{t+1} = \text{RNNCell}(i_t, h_t)$$
$$y_t = \arg\max(o_t); \quad i_{t+1} = \text{emb}(y_t)$$

where emb denotes the embedding function.

The captioning model first run the input image x through a CNN to obtain the image embedding and feed it to the RNN as the initial input i_0 along with the initial state h_0. At each decode step, the RNN uses the input i_t and state h_t to compute the new state h_{t+1} as well as the logits o_t representing the log-probability of the output token distribution in the vocabulary. The output y_t is the token in the vocabulary with highest probability based on o_t, and it is embedded into the continuous space using an embedding matrix W_{emb} as $W_{emb}[y_t]$. The embedding is fed to the next RNN cell as the input for the next decoding step.

Machine translation models The machine translation model is an encoder-decoder model [30, 9] with both the

encoder and the decoder being RNNs. Given the vocabulary V_{in} of the input language, the valid input space $\mathcal{X}$ is defined as all sentences composed of tokens in V_{in}, i.e., $\mathcal{X} = \{(x_1, \ldots, x_n) \mid x_i \in V, n > 0\}$. Given an input sequence $x = (x_1, \ldots, x_n)$, the model first calculates its embedding $f(x)$ RNN as follows (h_t^e and i_t^e denote the encoder hidden states and the inputs at the t-th time step, respectively. emb^e denotes the embedding function for each token in the vocabulary). The model then uses $f(x)$ as the initial state h_0 for the decoder RNN to generate the output sequence, following the same approach as in the image captioning model.

$$h_0^e = \mathbf{0}; \quad i_t^e = \text{emb}^e(x_t)$$
$$h_t^e = \text{RNNCell}^e(i_t^e, h_{t-1}^e); \quad f(x) = h_n^e$$

6. Experiments

We consider the following three models, namely, Multi-MNIST captioning, Show-and-Tell [34], and Neural Machine Translation (NMT) [30, 9, 2] models.

6.1. Details of models and datasets

Multi-MNIST. The first model we evaluate is a minimal image captioning model for Multi-MNIST dataset. The Multi-MNIST dataset is composed from the MNIST dataset (Figure 1 left). Each image in the dataset is composed from 1-3 MNIST images: each MNIST image (28 * 28) is placed on the canvas of size (28 * 112) with random bias on the x-axis. The composition process guarantees that every MNIST image is fully contained in the canvas without overlaps with other images. The label of each image is the list of MNIST digits appearing in the canvas, ordered by their x-axis values. The dataset contains 50,000 training images and 10,000 test images, where the training set is constructed from MNIST training set and the test set is constructed from MNIST test set. The images are normalized to $[-1, 1]$ before feeding to the captioning model. For this dataset, we train a CNN-RNN model for label prediction. The model encoder is a 4-layers CNN (2 convolution layers and 2 fully connected layers with ReLU activation functions applied in between). The decoder is a RNN with ReLU activation. Both the embedding size and the hidden size are set to 32. We train the model for 300 steps with Adam optimizer based on the cross-entropy loss. The model achieves 91.2% test accuracy, and all predictions made by the model on the training set have lengths no more than 3.

Show-and-Tell. Show and Tell model [34] is an image captioning model with CNN-RNN encoder-decoder architecture similar to the Multi-MNIST model trained on the MSCOCO 2014 dataset [23]. Show-and-Tell model uses Inception-v3 as the CNN encoder and an LSTM for caption generation. We use a public version of the pretrained model[3] for evaluation. All images are normalized to $[-1, 1]$ and all captions in the dataset are within length 20.

NMT. The machine translation model we study is a Seq2Seq model [30, 9] with the attention mechanism [2] trained on the WMT15 German-English dataset. The model uses byte pair segmentation (BPE) subword units [28] as vocabulary. The input vocabulary size is $36,548$. The model consists of 4-layer LSTMs of 1024 units with a bidirectional encoder, with the embedding dimension set to 1024. We use a publicly available checkpoint[4] with 27.6 BLEU score on the WMT15 test datasets in our evaluation. At training time, the model restricts the maximum decoding length to 50.

6.2. Adversarial Attacks

Our first experiment studies whether adversarial inputs exist for the above models and how they affect model decoding. For each model, we randomly select 100 inputs from the development dataset as attack targets, and compare the output length distributions from random perturbation and PGD attacks.

Multi-MNIST We evaluate the distribution of output lengths of images with an ℓ^∞ perturbation radius of $\delta \in \{0.001, 0.005, 0.01, 0.05, 0.1, 0.5\}$ using both random search and PGD attack. In random search, we generate 10,000 random images within the given perturbation radius for each image in the target dataset as new inputs to the model. In PGD attack, the adversarial inputs are obtained by running 10,000 gradient descent steps with an learning rate of 0.0005 using the Adam Optimizer.

Neither of the attack methods can find any adversarial inputs for $\delta \in \{0.001, 0.005, 0.01\}$ perturbation radius (i.e., no perturbation is found for any images in the target dataset within the above δ to generate an output sequence longer than the original one). Figure 2 shows the distribution of the output lengths for images with different perturbation radius. Results show that the PGD attack is successful at finding attacks that push the distribution of output lengths higher, particularly at larger values of δ. Examples of adversarial inputs found by the model are shown in Figure 1.

Show-and-Tell For the Show-and-Tell model, we generate attacks within an ℓ^∞ perturbation radius of $\delta = 0.5$ with both random search and PGD attack on 500 images randomly selected from the development dataset. However, except one adversarial input found by PGD attack that would cause the model to produce an output with size 25, no other adversarial inputs are found that can cause the model to produce outputs longer than 20 words, which is the training length cap. Our analysis shows that the difficulty of attacking the model is resulted from its strong bias on the output

[3] https://github.com/tensorflow/models/

[4] https://github.com/tensorflow/nmt

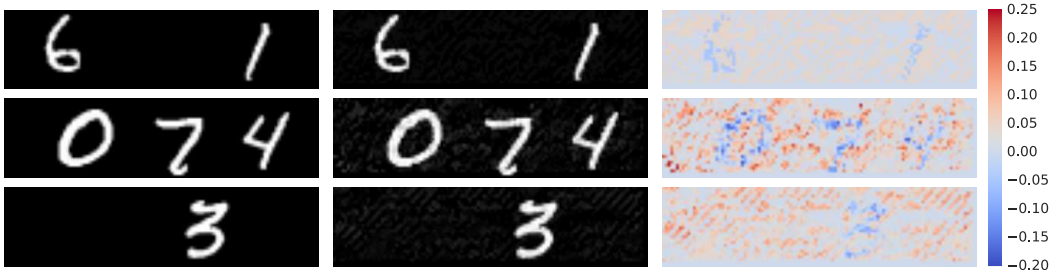

Figure 1. Multi-MNIST examples (left), adversarial examples found by PGD attack (mid), and their differences. For the first group, the model correctly predicts label $l_1 = [6, 1]$ on the original image but predicts $l'_1 = [6, 1, 1]$ for its corresponding adversarial input. Predictions on the original/adversarial inputs made by model for the second group are $l_2 = [0, 7, 4], l'_2 = [0, 1, 4, 3]$, and $l_3 = [3], l'_3 = [3, 3, 5, 3]$ for the third group. The adversarial inputs in the first/second/third groups are found within the perturbation radius $\delta_1 = 0.1, \delta_2 = 0.25, \delta_3 = 0.25$.

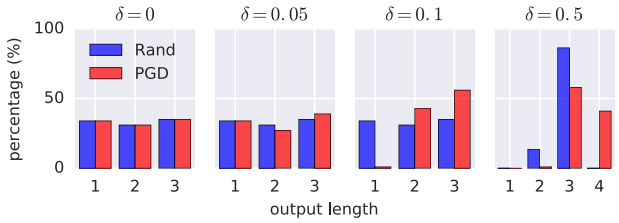

Figure 2. The distribution of output length for random search (denoted as **Rand**) and PGD attack with different perturbation radius δ. The x-axis denotes the output length and y-axis denotes the number of outputs with the corresponding length. $\delta = 0$ (no perturbation allowed) refers to the original output distribution of the target dataset.

sequence distribution and the saturation of sigmoid gates in the LSTM decoder. This result is also consistent with the result found by [5] that Show-and-Tell model is "only able to generate relevant captions learned from the training distribution".

NMT We evaluate the NMT model by comparing the output length distribution from adversarial examples generated from random search and PGD attack algorithms. We randomly draw 100 input sentences from the development dataset. The maximum input length is 78 and their corresponding translations made by the model are all within 75 tokens. We consider the perturbation $\delta \in \{0.3, 0.5, 1.0\}$.

1. *Random Search.* In each run of the random attack, given an input sequence with length n, we first randomly select $\lceil \delta \cdot n \rceil$ locations to modify, then randomly select substitutions of the tokens at these locations from the input vocabulary, and finally run the NMT model on the modified sequence. We run 10,000 random search steps for the 100 selected inputs, and show the distributions of all outputs obtained from the translation (in the total 1M output sequences).
2. *PGD Attack.* In PGD attack, we also start by randomly selecting $\lceil \delta \cdot n \rceil$ locations to modify for each input sequence with length n. We then run 800 iterations of PGD attack with Adam optimizer using an initial learning rate of 0.005 to find substitutions of the tokens at these selected locations. We plot the output length obtained from running these adversarial inputs through the translation model.

Figure 3 shows the distribution of output sequence lengths obtained from random search methods with different δ. We aggregate all sequences with length longer than 100 into the group '>100' in the plot. Results show that even random search approach could often craft inputs such that the corresponding output lengths are more than 75 and occasionally generates sentences with output length over 100. The random search algorithm finds 79, 11, 3 for δ =0.3, 0.5, 1, respectively, among the 1M translations that are longer than 100 tokens (at small δ, the search space is more restricted, and random search has a higher success rate of finding long outputs). Notably, the longest sequence found by the random search is a sequence with output length 312 tokens, where the original sequence is only 6.

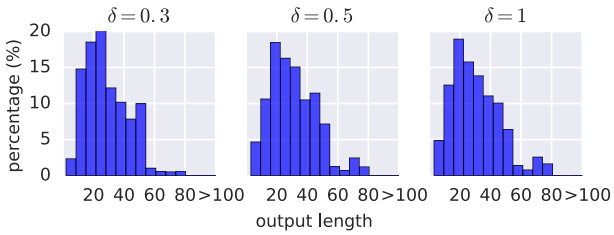

Figure 3. The histogram representing the output length distribution of the NMT model using random search with different perturbations ($\delta \in \{0.3, 0.5, 1\}$). The x-axis shows the output length. y-axis values are divided by 10,000, the number of random perturbation rounds per image.

Figure 4 shows the result from attacking the NMT model with PGD attack. Results show that PGD attack has relatively low success rate at lower perturbations compared

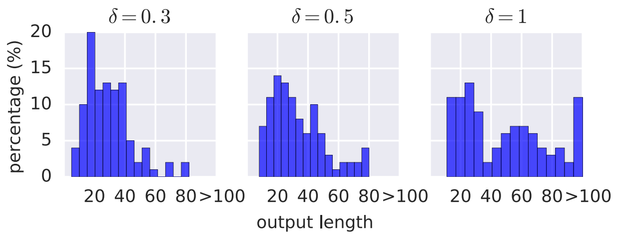

Figure 4. The histogram representing the output length distribution of the NMT model under PGD attack for different δ. x-axis shows the output length and y-axis shows the number of instances with the corresponding length.

(I) Die Waffe wird ausgestellt und durch den Zaun übergeben.
(O) The weapon is issued and handed over by the fence . ***eos***
(I′) Die namen name descri und ames utt origin i.e. meet grammatisch .
(O′) names name names name names grammatically name names ***eos***

Figure 5. An example of German to English translation where I, O refer to an original sequence in the dataset and the corresponding translation made by the model. I', O' refer to an adversarial example found by PGD attack and the corresponding model translation.

to larger perturbations. With an unconstrained perturbation $\delta = 100\%$, PGD attack algorithm discovers more adversarial inputs whose outputs are longer than 100 tokens (10% among all attacks), which is 1000× more often than random search. As an extreme case, PGD attack discovered an adversarial input with length 3 whose output length is 575. Examples of adversarial inputs and their corresponding model outputs are shown in Figure 5 and the Appendix; we find out that a common feature of the long outputs produced by the translation model is that the output sequences often end with long repetitions of one (or a few) words.

To analyze the bottleneck of PGD attack on the NMT model, we further run a variation of the PGD attack where the attack space is the (continuous) word embedding space as opposed to the (discrete) token space: we allow the attacker to directly modify token embeddings at selected attack locations to any other vector. PGD attack on this variation achieves a 100% success rate to find adversarial token embeddings such that the model outputs are longer than 500 tokens. This indicates that the discrete space is a bottleneck for consistently finding stronger attacks.

6.3. Verification

Our implementation of the verification algorithm using the mixed integer programming (8) is implemented using SCIP [15]. We run our verification algorithm on the Multi-

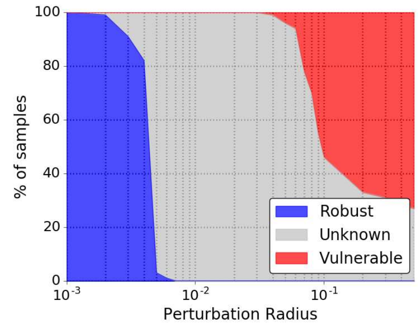

Figure 6. Proportion of samples being provably robust (in blue), vulnerable (in red) or of unknown robustness status (in gray) to an attack attempting to make the model generate an output sequence longer than the ground truth, as a function of the perturbation radius allowed to the attacker. For small radiuses, the MIP can prove that no attacks can be successful. For large radiuses, we are able to find successful attacks.

MNIST dataset, attempting to formally prove the robustness of the model to attacks attempting to generate an output longer than the ground truth. For each input image, we set a timeout of 30 minutes for the solver.

The results in Figure 6 show that our verification algorithm is able to verify formally that no attacks exists for small perturbation radiuses. However, as the perturbation radius increases, our verification algorithm times out and is not able to explore the full space of valid perturbations and thus cannot decide whether attacks exists in the given space. For this reason, the number of robust samples we report is only a lower bound on the actual number of robust samples. Conversely, the vulnerable samples that we exhibit give us an upper bound on the number of those robust samples. As shown by the large proportion of samples of unknown status, there is currently still a gap between the capabilities of formal verification method and attacks.

7. Conclusion

In this paper, we introduce the existence and the construction of the output-length modulation problem. We propose a differentiable proxy that can be used with PGD to efficiently find output-lengthening inputs. We also develop a verification approach to formally prove certain models cannot produce outputs greater than a certain length. We show that the proposed algorithm can produce adversarial examples that are 50 times longer than the input for machine translation models, and the image-captioning model can conform the output size is less than certain maximum length using the verification approach. In future work, we plan to study adversarial training of sequential output models using the generated attacks, to models that are robust against output lengthening attacks, and further, verify this formally.

References

[1] Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. Bottom-up and top-down attention for image captioning and visual question answering. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[3] Rudy Bunel, Ilker Turkaslan, Philip H. S. Torr, Pushmeet Kohli, and M. Pawan Kumar. A unified view of piecewise linear neural network verification. In *Advances in Neural Information Processing Systems*, 2018.

[4] Nicholas Carlini, Chang Liu, Jernej Kos, Úlfar Erlingsson, and Dawn Song. The secret sharer: Measuring unintended neural network memorization & extracting secrets. *CoRR*, abs/1802.08232, 2018.

[5] Hongge Chen, Huan Zhang, Pin-Yu Chen, Jinfeng Yi, and Cho-Jui Hsieh. Attacking visual language grounding with adversarial examples: A case study on neural image captioning. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2587–2597, 2018.

[6] Xinlei Chen and C. Lawrence Zitnick. Mind's eye: A recurrent visual representation for image caption generation. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, pages 2422–2431, 2015.

[7] Chih-Hong Cheng, Georg Nührenberg, and Harald Ruess. Maximum resilience of artificial neural networks. In *International Symposium on Automated Technology for Verification and Analysis*, pages 251–268. Springer, 2017.

[8] Minhao Cheng, Jinfeng Yi, Huan Zhang, Pin-Yu Chen, and Cho-Jui Hsieh. Seq2sick: Evaluating the robustness of sequence-to-sequence models with adversarial examples. *CoRR*, abs/1803.01128, 2018.

[9] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[10] Sumit Chopra, Michael Auli, and Alexander M Rush. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 93–98, 2016.

[11] Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. Hotflip: White-box adversarial examples for NLP. *arXiv preprint arXiv:1712.06751*, 2017.

[12] Hao Fang, Saurabh Gupta, Forrest N. Iandola, Rupesh Kumar Srivastava, Li Deng, Piotr Dollár, Jianfeng Gao, Xiaodong He, Margaret Mitchell, John C. Platt, C. Lawrence Zitnick, and Geoffrey Zweig. From captions to visual concepts and back. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, pages 1473–1482, 2015.

[13] Ulrich Germann, Michael Jahr, Kevin Knight, Daniel Marcu, and Kenji Yamada. Fast decoding and optimal decoding for machine translation. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, pages 228–235, 2001.

[14] J. Gilmer, R. P. Adams, I. Goodfellow, D. Andersen, and G. E. Dahl. Motivating the Rules of the Game for Adversarial Example Research. *ArXiv e-prints*, July 2018.

[15] Ambros Gleixner, Michael Bastubbe, Leon Eifler, Tristan Gally, Gerald Gamrath, Robert Lion Gottwald, Gregor Hendel, Christopher Hojny, Thorsten Koch, Marco E. Lübbecke, Stephen J. Maher, Matthias Miltenberger, Benjamin Müller, Marc E. Pfetsch, Christian Puchert, Daniel Rehfeldt, Franziska Schlösser, Christoph Schubert, Felipe Serrano, Yuji Shinano, Jan Merlin Viernickel, Matthias Walter, Fabian Wegscheider, Jonas T. Witt, and Jakob Witzig. The SCIP Optimization Suite 6.0. Technical report, Optimization Online, 2018.

[16] Ignacio E Grossmann. Review of nonlinear mixed-integer and disjunctive programming techniques. *Optimization and Engineering*, 3(3):227–252, 2002.

[17] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[18] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with Gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

[19] Frederick Jelinek. *Statistical Methods for Speech Recognition*. MIT press, 1997.

[20] Robin Jia and Percy Liang. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2021–2031, 2017.

[21] Andrej Karpathy and Li Fei-Fei. Deep visual-semantic alignments for generating image descriptions. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(4):664–676, 2017.

[22] Philipp Koehn, Franz Josef Och, and Daniel Marcu. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 48–54, 2003.

[23] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common objects in context. In *European Conference on Computer Vision*, pages 740–755, 2014.

[24] Jiasen Lu, Caiming Xiong, Devi Parikh, and Richard Socher. Knowing when to look: Adaptive attention via a visual sentinel for image captioning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

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

[26] Steven J Rennie, Etienne Marcheret, Youssef Mroueh, Jarret Ross, and Vaibhava Goel. Self-critical sequence training for image captioning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[27] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Semantically equivalent adversarial rules for debugging nlp models. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 856–865, 2018.

[28] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, 2016.

[29] Ravi Shekhar, Sandro Pezzelle, Yauhen Klimovich, Aurélie Herbelot, Moin Nabi, Enver Sangineto, and Raffaella Bernardi. FOIL it! find one mismatch between image and language caption. *CoRR*, abs/1705.01359, 2017.

[30] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112, 2014.

[31] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[32] Vincent Tjeng and Russ Tedrake. Verifying neural networks with mixed integer programming. *arXiv preprint arXiv:1711.07356*, 2017.

[33] Subhashini Venugopalan, Marcus Rohrbach, Jeffrey Donahue, Raymond Mooney, Trevor Darrell, and Kate Saenko. Sequence to sequence-video to text. In *Proceedings of the IEEE international Conference on Computer Vision*, pages 4534–4542, 2015.

[34] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3156–3164, 2015.

[35] Jingwen Wang, Wenhao Jiang, Lin Ma, Wei Liu, and Yong Xu. Bidirectional attentive fusion with context gating for dense video captioning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[36] Wayne L Winston and Munirpallam Venkataramanan. *Introduction to Mathematical Programming*, volume 1. Thomson Learning, 2002.

[37] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the 32nd International Conference on Machine Learning, ICML*, pages 2048–2057, 2015.

[38] Ran Xu, Caiming Xiong, Wei Chen, and Jason J Corso. Jointly modeling deep video and compositional text to bridge vision and language in a unified framework. In *AAAI*, 2015.

[39] Puyudi Yang, Jianbo Chen, Cho-Jui Hsieh, Jane-Ling Wang, and Michael I Jordan. Greedy attack and Gumbel attack: Generating adversarial examples for discrete data. *arXiv preprint arXiv:1805.12316*, 2018.

[40] Li Yao, Atousa Torabi, Kyunghyun Cho, Nicolas Ballas, Christopher Pal, Hugo Larochelle, and Aaron Courville. Describing videos by exploiting temporal structure. In *Proceedings of the IEEE international Conference on Computer Vision*, pages 4507–4515, 2015.

[41] Haonan Yu, Jiang Wang, Zhiheng Huang, Yi Yang, and Wei Xu. Video paragraph captioning using hierarchical recurrent neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4584–4593, 2016.