TASK LOSS ESTIMATION FOR SEQUENCE PREDICTION

Dzmitry Bahdanau, Dmitriy Serdyuk, Philémon Brakel, Nan Rosemary Ke
Université de Montréal

Jan Chorowski
University of Wrocław

Aaron Courville, Yoshua Bengio*
Université de Montréal

ABSTRACT

Often, the performance on a supervised machine learning task is evaluated with a *task loss* function that cannot be optimized directly. Examples of such loss functions include the classification error, the edit distance and the BLEU score. A common workaround for this problem is to instead optimize a *surrogate loss* function, such as for instance cross-entropy or hinge loss. In order for this remedy to be effective, it is important to ensure that minimization of the surrogate loss results in minimization of the task loss, a condition that we call *consistency with the task loss*. In this work, we propose another method for deriving differentiable surrogate losses that provably meet this requirement. We focus on the broad class of models that define a score for every input-output pair. Our idea is that this score can be interpreted as an estimate of the task loss, and that the estimation error may be used as a consistent surrogate loss. A distinct feature of such an approach is that it defines the desirable value of the score for every input-output pair. We use this property to design specialized surrogate losses for Encoder-Decoder models often used for sequence prediction tasks. In our experiment, we benchmark on the task of speech recognition. Using a new surrogate loss instead of cross-entropy to train an Encoder-Decoder speech recognizer brings a significant 9% relative improvement in terms of Character Error Rate (CER) in the case when no extra corpora are used for language modeling.

1 INTRODUCTION

There has been an increase of interest in learning systems that can solve tasks in an "end-to-end" fashion. An early example of such a system is a highly successful convolutional network handwriting recognition pipeline (LeCun et al., 1998). More recent examples are deep convolutional networks designed for image recognition (Krizhevsky et al., 2012), neural translation systems (Sutskever et al., 2014; Bahdanau et al., 2015a), and speech recognizers (Graves & Jaitly, 2014; Hannun et al., 2014a; Chorowski et al., 2015; Bahdanau et al., 2015b). Parts of end-to-end systems, such as image features extracted by convolutional networks, often successfully replace hand-designed ones (Yosinski et al., 2014). This demonstrates how useful it can be that all parts of a system are learned to solve the relevant task.

In practice however, it often happens that the relevant *task loss* function, such as error rate in classification, word error rate in speech recognition, or BLEU score in machine translation, is only used for model evaluation, while a different *surrogate loss* is used to train the model. There are several reasons for the evaluation loss – training loss discrepancy: the evaluation criterion may be non-differentiable, it can be non-convex or otherwise inconvenient to optimize, or one may want to emphasize certain problem-agnostic model properties, such as a class separation margin (Vapnik, 1998). For instance, classification models are often evaluated based on their error rates, which corresponds to a 0-1 task loss. However, people often minimize surrogate losses like the cross-entropy (Bishop, 2006) or the hinge loss (Vapnik, 1998) instead. For classification, these surrogate losses are well-motivated and their minimization tends to lead to a low error rate. It is not clear, however,

*Yoshua Bengio is a CIFAR Senior Fellow

that the same methods should be preferred for structured output problems, in which typically there is a gradation in the quality of answers.

In this work, we revisit the problem of choosing an appropriate surrogate loss for training. We focus on the broad class of models that define a score for every input-output pair and make predictions by looking for the output with the lowest score. Our main idea is that if the scores defined by the model are approximately equal to the task loss, then the task loss of the model's prediction should be low. We hence propose to define the surrogate loss as the estimation error of a score function that is trained to mimic the task loss, a method we will refer to as *task loss estimation*. We prove that minimization of such a surrogate loss leads to the minimization of the targeted task loss as well, a property that we call *consistency* with the task loss. The main distinct feature of our new approach is that it prescribes a target value for the score of every input-output pair. This target value does not depend on the score of other outputs, which is the key property of the proposed method and the key difference from other approaches to define consistent surrogate losses, such as the generalized hinge loss used in Structured Support Vector Machines (Tsochantaridis et al., 2005).

Furthermore, we apply the task loss estimation principle to derive new surrogate losses for sequence prediction models of the Encoder-Decoder family. The Decoder, typically a recurrent network, produces the score for an input-output pair by summing terms associated with every element of the sequence. The fact that the target for the score is fixed in our approach allows us to define targets for each of the terms separately. By doing so we strive to achieve two goals: to facilitate faster training and to ensure that the greedy search and the beam search used to obtain predictions from an Encoder-Decoder work reasonably well. To validate our ideas we carry out an experiment on a speech recognition task. We show that when no external language model is used using a new surrogate loss indeed results in a relative 9% improvement of the CER compared to cross-entropy training for an Encoder-Decoder speech recognizer.

2 Task Loss Estimation for Supervised Learning

Basic Definitions Consider the broad class of supervised learning problems in which the trained learner is only allowed to deterministically produce a single answer $\hat{y} \in \mathcal{Y}$ at run-time, when given an input $x \in \mathcal{X}$. After training, the learner's performance is evaluated in terms of the task loss $L(x, \hat{y})$ that it suffered from outputting $\hat{y}$ for x. We assume that the task loss is non-negative and that there exists a unique ground truth answer $y = g(x)$ such that $L(x, g(x)) = 0$.[1] During the training, the learner is provided with training pairs (x_i, y_i), where $y_i = g(x_i)$. We assume that given the ground truth y_i, the loss $L(x, \hat{y})$ can be efficiently for any answer $\hat{y}$.

The training problem is then defined as follows. Given a family of parametrized mappings $\{h_\alpha\}, \alpha \in \mathcal{A}$ from $\mathcal{X}$ to $\mathcal{Y}$, try to choose one that minimizes (as much as possible) the *risk functional*:

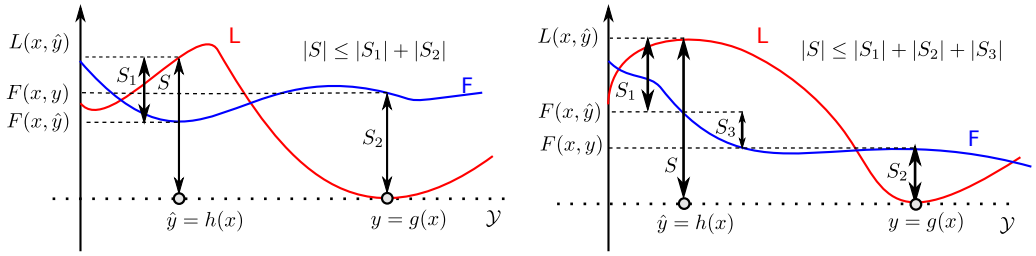
$$R(\alpha) = \int_x L(x, h_\alpha(x)) P(x) dx, \tag{1}$$

where P is an unknown data distribution. The choice must be made using only a sample $S = \{x_i\}_{i=1}^N$ from the distribution P with ground truth answers $\{y_i\}_{i=1}^N$ available for $x_i \in S$.

Here are two examples of task losses that are pretty much standard in some key supervised learning problems:

- the 0-1 loss used in classification problems is $L(x, y) = \begin{cases} 1, & g(x) = y \\ 0, & g(x) \neq y \end{cases}$;
- the Levenshtein distance used in speech recognition is $L(x, y) = \rho_{\text{levenstein}}(g(x), y)$ is the minimum number of changes required to transform a transcript y into the correct transcript $g(x)$.

[1] Both assumptions are made to keep the exposition simple, they are not cricial for applicability of the task loss estimation approach.



(a) (b)

Figure 1: A graphical illustration of how the loss estimation error provides an upper bound for the task loss L, which is the underlying idea of Theorem 1. The segments S, S_1, S_2, S_3 on the picture stand for the four main terms of the theorem statement, $L(x,\hat{y})$, $|L(x,\hat{y}) - F(x,\hat{y})|$, $|F(x,y)|$, $F(x,\hat{y}) - F(x,y)$ respectively. The location of the segments related to each other explains why the loss estimation error gives a bound on the task loss $L(x,\hat{y})$ of the prediction $\hat{y}$. Figure 1a displays the case when the mininum of $F(x)$ is successfully found by $h_\alpha(x)$. Figure 1b explains the term $F(x,\hat{y}) - F(x,y)$ which appears when $h_\alpha(x)$ is an approximate minimizer incapable to find an output with a score lower than $F(x,y)$.

Empirical Risk and Surrogate Losses Under the assumptions that S is big enough and the family $\mathcal{A}$ is limited or some form of regularization is introduced, the *empirical risk* $\hat{R}(\alpha)$ can be minimized

$$\hat{R}(\alpha) = \frac{1}{N}\sum_{i=1}^{N} L(x_i, h_\alpha(x_i)), \tag{2}$$

instead of R (Vapnik, 1998).

A common practical issue with minimizing the empirical risk functional $\hat{R}(\alpha)$ is that $L(x,y)$ is often not differentiable with respect to y, which in turn renders $\hat{R}(\alpha)$ non-differentiable with respect to α and therefore difficult to optimize. The prevalent workaround is to define $h_\alpha(x)$ as the minimum of a *scoring function* $F_\alpha(x,y)$ (often also called energy):

$$h_\alpha^{\min}(x) = \operatorname{argmin}_y F_\alpha(x,y).$$

Parameters α of the scoring function are chosen to minimize a (technically empirical) *surrogate risk* $\mathcal{R}(\alpha)$ defined as the average *surrogate loss* $\mathcal{L}(x_i,\cdot)$:

$$\mathcal{R}(\alpha) = \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}(x_i, F_\alpha(x_i)), \tag{3}$$

where $F_\alpha(x_i) \in \mathbb{R}^{|\mathcal{Y}|}$ is the vector of scores computed on all elements of $\mathcal{Y}$[2].

We argue that, for the transition from the empirical risk $\hat{R}$ to the surrogate risk $\mathcal{R}$ to be helpful, a number of conditions should hold:

1. It must be easy to compute predictions $h_\alpha^{\min}(x)$. Thus $F_\alpha(x,y)$ must be easy to minimize over y, at least in an approximate sense. For instance, in most classification problems this is not an issue at all because the output space $\mathcal{Y}$ is small. On the other hand, for structured output prediction this might be a significant issue.
2. $\mathcal{R}$ should be simpler to optimize than $\hat{R}$.
3. Optimization of $\mathcal{R}$ should result in optimization of $\hat{R}$.

Let us consider two examples of surrogate losses

[2]Without loss of generality, we assume here that the output space is discrete.

- The *cross-entropy surrogate loss* $\mathcal{L}_{CE}$ is applicable when the scores $F_\alpha(x, y)$ are interpreted as unnormalized negative log-probabilities:

$$\mathcal{L}_{\text{CE}}(x, F_\alpha(x)) = F_\alpha(x, g(x)) - \log(\sum_{y' \in \mathcal{Y}} \exp(F_\alpha(x, y'))), \tag{4}$$

$$\mathcal{R}_{\text{CE}}(\alpha) = \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}_{\text{CE}}(x_i, F_\alpha(x_i)). \tag{5}$$

  With $\mathcal{L}_{\text{CE}}$ choosing α that minimizes $\mathcal{R}_{\text{CE}}(\alpha)$ corresponds to Maximum Likelihood Estimation (MLE).

- A generalized hinge loss used in Structured Support Vector Machines (Tsochantaridis et al., 2005):

$$\mathcal{L}_{\text{hinge}}(x, F_\alpha(x)) = \max_y \left(F_\alpha(x, g(x)) - F_\alpha(x, y) + L(g(x), y), 0\right).$$

  The respective surrogate risk $\mathcal{R}_{\text{hinge}}$ is defined similarly to $\mathcal{R}_{\text{CE}}$.

Both of these surrogate loss functions have properties that make them relatively simple to optimize. The cross-entropy is both differentiable and convex. The hinge loss is piecewise differentiable and convex as well. We refer the reader to LeCun et al. (2006) for a survey of surrogate loss functions (note that their definition of a loss function differs slightly from the one we use in this text).

Popular surrogate losses are often agnostic to the task loss L, the cross-entropy surrogate loss $\mathcal{L}_{\text{CE}}$ being a good example. Even if we find parameters α_{CE} which make the cross-entropy $\mathcal{R}_{\text{CE}}(\alpha_{\text{CE}})$ arbitrary small, there is no guarantee that the empirical risk $\hat{R}(\alpha_{\text{CE}})$ will also be small. However, some surrogate losses, such as the generalized hinge loss $\mathcal{L}_{\text{hinge}}$, provide certain guarantees for the empirical risk. Specifically, one can see that $L(x, h_\alpha^{\min}(x)) \leq \mathcal{L}_{\text{hinge}}(x, F(x))$, which implies $\hat{R}(\alpha) \leq \mathcal{R}_{\text{hinge}}(\alpha)$, or simply put, minimizing $\mathcal{R}_{\text{hinge}}$ *necessarily* pushes $\hat{R}$ down.

Task Loss Estimation In this paper we introduce a novel paradigm for building surrogate losses with guarantees similar to those of $\mathcal{L}_{\text{hinge}}$. Namely, we propose to interpret the scoring function F as an estimate of the task loss L itself. In other words we want $F_\alpha(x, y) \approx L(x, y)$.

We can motivate this approach by showing that for the empirical risk to be low, it is sufficient for the task loss and the score to be similar at only two points: the ground truth $g(x)$ and the prediction $h_\alpha(x)$. We combine the estimation errors for these two outputs to obtain a new surrogate loss $\mathcal{L}_{\text{min,min}}$ which we call the *min-min loss*.

Theorem 1. *Let $\mathcal{L}_{min,min}$ be defined as follows:*

$$\mathcal{L}_{min,\ min}(L(x), F_\alpha(x)) = |F_\alpha(x, y)| + |L(x, \hat{y}) - F_\alpha(x, \hat{y})|, \tag{6}$$

here $y = g(x)$, $\hat{y} = h_\alpha(x)$. Then the respective surrogate risk $\mathcal{R}_{min,min}$ provides the following bound on $\hat{R}$

$$\hat{R}(\alpha) \leq \mathcal{R}_{min,\ min}(\alpha) + M(\alpha), \tag{7}$$

where

$$M(\alpha) = \frac{1}{N} \sum_{i=1}^{N} \max\left(F(x_i, \hat{y}_i) - F(x_i, y_i), 0\right).$$

Figure 1 illustrates the statement of Theorem 1. Simply put, the theorem says that if $h_\alpha = h_\alpha^{min}$, or if h_α is a good enough approximation of h_α^{min} such that the term $M(\alpha)$ is small, the surrogate loss $\mathcal{R}_{\text{min,min}}$ is a sensible substitute for $\hat{R}$. Please see Appendix for a formal proof of the theorem.

The key difference of our new approach from the generalized hinge loss is that it assigns a fixed target $L(x, y)$ for the score $F(x, y)$ of every pair $(x, y) \in \mathcal{X} \times \mathcal{Y}$. This target is independent of the values of $F(x, y')$ for all other $y' \in \mathcal{Y}$. The knowledge that L is the target can be used at the stage of

designing the model $F_\alpha(x, y)$. For example, when y has a structure, a $L(x, y)$ might be decomposed into separate targets for every element of y, thereby making optimization of $\mathcal{R}$ more tractable.

In consideration of optimization difficulties, our new surrogate loss $\mathcal{L}_{\text{min, min}}$ is piece-wise smooth like $\mathcal{L}_{hinge}$, but it is not convex and even not continuous. In practice, we tackle the optimization by fixing the outputs $h_\alpha(x)$ for a subset of the sample S, improving $\mathcal{L}_{\text{min, min}}$ with the fixed outputs by e.g. a gradient descent step, and doing the same iteratively.

3 Task Loss Estimation for Sequence Prediction

In sequence prediction problems the outputs are sequences over an alphabet C. We assume that the alphabet is not too big, more specifically, that a loop over its elements is feasible. In addition we extend the alphabet C with a special end-of-sequence token \$, creating the extended alphabet $\overline{C} = C \cup \{\$\}$. For convenience, we assume that all valid output sequences must end with this token. Now we can formally define the output space as the set of all sequences which end with the end-of-sequence token $\mathcal{Y} = \{y\$: y \in C^*\}$, where C^* denotes a set of all finite sequences over the alphabet C.

We will now describe how task loss estimation can be applied to sequence prediction for the following specific scenario:

- The score function is an Encoder-Decoder model.
- The prediction $h_\alpha^{\min}$ is approximated with a beam search or a greedy search.

3.1 Encoder-Decoder Model

A popular model for sequence prediction is the Encoder-Decoder model. In this approach, the Decoder is trained to model the probability $P(y^j|z(x), y^{1\cdots j-1})$ of the next token y^j given a representation of the input $z(x)$ produced by the Encoder, and the previous tokens $y^{1\cdots j-1}$, where $y = g(x)$ is the ground truth output. Decoders are typically implemented using recurrent neural networks. Using the terminology of this paper, one can say that a standard Encoder-Decoder implements a parametrized function $\delta_\alpha(c, x, y^{1\cdots j-1})$ that defines the scoring function as follows:

$$F_\alpha^{ED1}(x, y) = \sum_{j=1}^{|y|} -\log q_\alpha(y^j, x, y^{1\cdots j}), \tag{8}$$

$$q_\alpha(y^j, x, y^{1\cdots j}) = \frac{\exp\left(\delta_\alpha(y^j, x, y^{1\cdots j})\right)}{\sum_{c\in\overline{C}} \exp\left(\delta_\alpha(c, x, y^{1\cdots j})\right)}. \tag{9}$$

The cross-entropy surrogate loss can be used for training Encoder-Decoders. Since the score function (8) defined by an Encoder-Decoder is a proper distribution, the exact formula for the surrogate loss is simpler than in Equation 4

$$\mathcal{L}_{CE}(x, F_\alpha^{ED1}(x)) = F_\alpha^{ED1}(x, y) = \sum_{j=1}^{|y|} -\log q_\alpha(y^j, x, y^{1\cdots j-1}),$$
$$\text{where } y = g(x).$$

Exactly computing $h_\alpha^{\min}$ is not possible for Encoder-Decoder models. A beam search procedure is used to compute an approximation h_α^B, where B is the beam size. In beam search at every step k the beam, that is a set of B "good prefixes" of length k, is replaced by a set of good prefixes of length $k + 1$. The transition is done by considering all continuations of the beam's sequences and leaving only those B candidates for which the partial sum of $\log q_\alpha$ is minimal.

3.2 Applying Task Loss Estimation to Encoder-Decoders

Adapting the Min-Min Loss. We want to keep the structure of the scoring function defined in Equation (8). However, the normalization carried out in (9) is not necessary any more, so our new

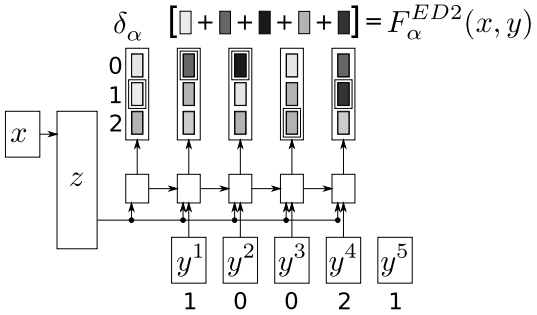


Figure 2: A schematic representation of an Encoder-Decoder architecture implementing the score function $F_\alpha^{ED2}(\cdot)$. For this example, the score of a sequence of labels $\{y^1, \cdots, y^5\}$ and an input sequence x is computed, where each label y^j is from the alphabet $\{0, 1, 2\}$. For each label in the sequence, the decoder produces a vector δ_α that represents the predicted change δ_o in the optimistic loss for each possible symbol at the next time step. The score for the whole sequence is computed by summing $\delta_\alpha(y^j, y^{1\cdots j-1}, x)$ for all j. Note that at each timestep, the decoder also uses the representation $z(x)$ computed by the encoder.

scoring function is simply the sum of δ_α:

$$F_\alpha^{ED2}(x, y) = \sum_{j=1}^{|y|} \delta_\alpha(y^j, x, y^{1\cdots j-1}).$$

Now, in theory, the min-min loss $\mathcal{L}_{\text{min, min}}$ could be used for training F_α^{ED2}. However, there are two concerns which render this straight-forward approach less attractive:

- Intuitively, constraining only the sum of δ_α might provide not enough supervision for training. Namely, the gradient of $\mathcal{L}_{\text{min, min}}$ would be the same with respect to all $\delta_\alpha(y^j, x, y^{1\cdots j-1})$, which might hamper gradient-based optimization methods.
- There is no guarantee that the beam search will be able to work with δ_α values learnt this way.

To circumvent both of these potential issues, we propose to break the target loss $L(x, y)$ into subtargets $\delta_L^j(x, y)$ assigned token-wise. We define the *optimistic task loss* $L_o(x, y)$ for an output prefix y as the loss of the best possible continuation of the prefix y. For completed output sequences, that is those ending with the end-of-sequence token, we say that the optimistic task loss is equal to the task loss. This results in the following formal definition:

$$L_o(x, y) = \begin{cases} \min\limits_{z \in B^*} L(x, yz\$), & y \in C^*; \\ L(x, y), & y \in \mathcal{Y}, \end{cases} \tag{10}$$

We argue that the change of the optimistic task loss $\delta_o(y^j, x, y^{1\cdots j-1}) = L_o(x, yc) - L_o(x, y)$ is a good target for $\delta_\alpha(y^j, x, y^{1\cdots j-1})$. Indeed, the pruning during beam search is done by looking at the sum $s(x, y^{1\cdots k}) = \sum\limits_{j=1}^{k} \delta_\alpha(y^j, x, y^{j-1})$ for the prefixes y from the beam. Informally, the pruning procedure should remove prefixes whose continuations are unlikely to be beneficial. The optimistic loss $L_o(x, y)$ tells us what is the lowest loss one can obtain by continuing y in an arbitrary way, and hence, it can be used for selecting the prefixes to be continued. Assuming that the network learns to output $\delta_\alpha(c, x, y^{1\cdots j}) \approx \delta_o(c, x, y^{1\cdots j})$, we can hope that pruning by $s_k(x, y^{1\cdots j}) \approx L_{\text{opt}}(x, y^{1\cdots k})$ will keep the good prefixes in.

Our new surrogate loss consisting of the sum of token-wise errors looks as follows:

$$\mathcal{L}^{ED}_{\text{min, min}}(x, \delta_\alpha(x)) = \sum_{j=1}^{|y|} |\delta_\alpha(y^j, x, y^{1\dots j-1}) - \delta_o(y^j, x, y^{1\dots j-1})| \tag{11}$$

$$+ \sum_{j=1}^{|\hat{y}|} |\delta_\alpha(\hat{y}^j, x, \hat{y}^{1\dots j-1}) - \delta_o(\hat{y}^j, x, \hat{y}^{1\dots j-1})|, \tag{12}$$

where $y = g(x)$, $\hat{y} = h^{\text{min}}_\alpha(x)$. Note, that $\mathcal{L}^{ED}_{\text{min,min}}$ extends our previous surrogate loss definition from (3) by working not on $F_\alpha(x)$ but on its additive components $\delta_\alpha(y^j, x, y^{1\dots j-1})$. One can also see that $\mathcal{L}^{ED}_{\text{min, min}}(x, \delta_\alpha(x)) \geq \mathcal{L}_{\text{min, min}}(x, \delta_\alpha(x))$ because of the triangle inequality, which implies that the respective surrogate risk is a bound for the empirical risk $\mathcal{R}^{ED}_{\text{min, min}} \geq \hat{R}(\alpha)$.

A careful reader might have noticed, that in practice we do not have access to $\mathcal{L}^{ED}_{\text{min, min}}$, because we can not compute $h^{\text{min}}_\alpha(x)$. The best we can have is $\mathcal{L}^{ED}_{\text{min},B}(x, y)$ defined in a similar way but using the beam search to compute $\hat{y} = h^B_\alpha(x)$ instead of the intractable exact minimization. However, according to Theorem 1 minimizing $\mathcal{L}^{ED}_{\text{min},B}$ guarantees low empirical risk for beam search predictions $h^B_\alpha(x)$, as long as the beam search finds an output with a score that is lower than the score of the ground truth. In our experience, this is usually the case for Encoder-Decoder models.

A Loss for the Greedy Search One disadvantage of the approach with $\mathcal{L}^{ED}_{\text{min},B}$ is that computing the surrogate loss, and therefore also its gradients, becomes quite expensive. We address this issue by proposing another surrogate loss which only involves beam search with the beam size $B = 1$, also often called greedy search. The new surrogate loss $\mathcal{L}^{ED}_{\text{greedy}}$ is defined as follows:

$$\mathcal{L}^{ED}_{\text{greedy}}(x, \delta_\alpha(x)) = \sum_{j=1}^{|\hat{y}|} |\delta_\alpha(\hat{y}^j, x, \hat{y}^{1\dots j-1}) - \delta_o(\hat{y}^j, x, \hat{y}^{1\dots j-1})| + |\delta_\alpha(c^j_{\text{min}}, x, \hat{y}^{1\dots j-1})|, \tag{13}$$

where $\hat{y} = h^1_\alpha(x)$, $c^j_{\text{min}} = \text{argmin}_{c \in \overline{C}} \delta_o(c, x, y^{1\dots j-1})$. We can show, that optimizing the respective surrogate risk $\mathcal{R}^{ED}_{\text{greedy}}$ necessarily improves the performance of greedy search:

Theorem 2. *The empirical risk $\hat{R}_{greedy}$ associated with using h^1_α for giving predictions is bounded by $\mathcal{R}^{ED}_{greedy}$, that is $\hat{R}_{greedy}(\alpha) \leq \mathcal{R}^{ED}_{greedy}(\alpha)$.*

The proof can be found in the Appendix. Now, with the greedy search, the gradient of $\hat{R}_{\text{greedy}}(\alpha)$ can be computed just as fast as the gradient of the average cross-entropy, since the computation of the gradient can be combined with the search.

Tricks of the Trade Driven by our intuition about the training process we make two further changes to the loss $\mathcal{L}_{\text{greedy}}$. First, we change Equation 13 by adding all characters into consideration:

$$\mathcal{L}^{ED}_{\text{greedy1}}(x, \delta_\alpha(x)) = \sum_{j=1}^{|\hat{y}|} \sum_{c \in \overline{C}} |\delta_\alpha(c, x, \hat{y}^{1\dots j-1}) - \delta_o(c, x, \hat{y}^{1\dots j-1})|. \tag{14}$$

Our reasoning is that by providing a more informative training signal at each step we help optimization. We note, that the bound on the empirical risk provided by the respective surrogate risk $\mathcal{R}^{ED}_{\text{greedy1}}(\alpha)$ is looser then the one by $\mathcal{R}^{ED}_{\text{greedy}}(\alpha)$ since $\mathcal{R}^{ED}_{\text{greedy}} \leq \mathcal{R}^{ED}_{greedy1}$. On the other hand, $\mathcal{R}^{ED}_{\text{greedy1}}$ enforces a margin between the best next token and all the worse ones, which can possibly help generalization.

Finally, we found $\mathcal{L}^{ED}_{\text{greedy1}}$ hard to optimize because the gradient of $|a - b|$ is always either +1 or -1, that is it does not get smaller when a and b approach each other. To tackle this we replaced the absolute value by the square:

$$\mathcal{L}^{ED}_{\text{greedy2}}(x, \delta_\alpha(x)) = \sum_{j=1}^{|\hat{y}|} \sum_{c \in \overline{C}} (\delta_\alpha(c, x, \hat{y}^{1\dots j-1}) - \delta_o(c, x, \hat{y}^{1\dots j-1}))^2.$$

Example: Edit Distance We explain how the decomposition of the task loss $L(x, y)$ into a sum $\sum_{j=1}^{|y|} \delta_o(y^j, x, y^{1\dots j-1})$ works on the example of the edit distance. The edit distance $\rho_{\text{levenstein}}(s_1, s_2)$ between two strings $s_1, s_2 \in C^*$ is the minimal number of actions required to transform s_1 into s_2, where the actions allowed are token deletion, insertion and substitution. If the loss $L(x, y)$ is defined as the edit distance $\rho_{\text{levenstein}}(g(x), y)$, there is a compact expression for the optimistic loss $L_o(x, y)$:

$$L_o(x, y) = \begin{cases} \min_{k=0}^{k=|g(x)|} \rho_{\text{levenstein}}(y, g(x)^{1\dots k}), & y \in C^*, \\ \rho_{\text{levenstein}}(y, g(x)), & y \in \mathcal{Y}. \end{cases} \tag{15}$$

Equation (15) formalizes the consideration that the optimal way to continue a prefix y is to append a suffix of the ground truth $g(x)$. From the obtained expression for $L_o(x, y)$ one can see that $\delta_o(c, x, y)$ can only be 0 or -1 when $c \neq \$$. Indeed, by definition $\delta_o \geq 0$, and also adding a character c to a prefix y can only change the edit distance $\rho(y, g(x)^{1\dots k})$ by 1 at most. For the case of $c = \$$ the value $\delta_o(\$, x, y)$ can be an arbitrarily large negative number, in particular for prefixes y which are shorter then $g(x)$. It would be a waste of the model capacity to try to exactly approximate such larger numbers, and in practice we clip the values $\delta_o(\$, x, y)$ to be at most -5.

An attentive reader might have noticed, that for complex loss functions such as e.g. BLEU and METEOR computing the loss decomposition like we did it above might be significantly harder. However, we believe that by considering all ground truth suffixes one can often find a close to optimal continuation.

4 Related Work

In an early attempt to minimize the empirical risk for speech recognition models, word error rate scores were used to rescale a loss similar to the objective that is often referred to as Maximum Mutual Information (Povey & Woodland, 2002). For each sequence in the data, this objective requires a summation over all possible sequences to compute the expected word error rate from the groundtruth, something that is possible for only a restricted class of models. A recent survey (He et al., 2008) explains and documents improvements in speech recognition brought by other methods of discriminative training of speech recognition systems.

In the context of Encoder-Decoders for sequence generation, a curriculum learning (Bengio et al., 2009) strategy has been proposed to address the discrepancy between the training and testing conditions of models trained with maximum likelihood (Bengio et al., 2015). It was shown that the performance on several sequence prediction tasks can be improved by gradually transitioning from a fully guided training scheme to one where the model is increasingly conditioned on symbols it generated itself to make training more similar to the decoding stage in which the model will be conditioned on its own predictions as well. While this approach has an intuitive appeal and clearly works well in some situations, it doesn't take the task loss into account and to our knowledge no clear theoretical motivation for this method has been provided yet. Another issue is that one needs to decide how fast to transition between the two different types of training schemes.

Recently, methods for direct empirical risk minimization for structured prediction have been proposed that treat the model and the approximate inference procedure as a single black-box method for generating predictions (Stoyanov et al., 2011; Domke, 2012). The gradient of the loss is back-propagated through the approximate inference procedure itself. While this approach is certainly more direct than the optimization of some auxiliary loss, it requires the loss to be differentiable.

Hazan et al. (2010) propose a method for direct loss minimization that approximates the gradient of the task loss using a loss adjusted inference procedure. This method has been extended to Hidden Markov Models and applied to phoneme recognition (Keshet et al., 2011).

For a model that provides a distribution over structured output configurations, the gradient with respect to any expectation over that distribution can be estimated using a sampling approach. This technique has been used for speech recognition (Graves & Jaitly, 2014) to estimate the gradient of the transcription loss (i.e., the word error rate) and is equivalent to the REINFORCE method (Williams, 1992) from reinforcement learning. A downside of this method is that in many cases the gradient estimates have high variance. The method also assumes that it is possible and computationally

feasible to sample from the model. A related approach is to use an inference method to generate a list of the n best candidate output predictions according to the model (note that for this the model doesn't need to be probabilistic) and approximate the expected loss using an average over these candidate predictions Gao & He (2013). Similarly, one can anneal from a smooth expectation approximated with a large number of candidates towards the loss of a single prediction Smith & Eisner (2006).

5 Experimental Setup and Results

For experimental confirmation[3] of the theory discussed in Sections 2 and 3, we use a character-level speech recognition task similar to Bahdanau et al. (2015b). Like in our previous work, we used the Wall Street Journal (WSJ) speech corpus for our experiments. The model is trained on the full 81 hour 'train-si284' training set, we use the 'dev93' development set for validation and model selection, and we report the performance on the 'eval92' test set. The inputs to our models were sequences of feature vectors. Each feature vector contained the energy and 40 mel-filter bank features with their deltas and delta-deltas, which means that the dimensionality of the feature vector is 123. We use the standard trigram language model shipped with the WSJ dataset; in addition we experiment with its extended version created by Kaldi WSJ s5 recipe (Povey et al., 2011).

Our main baseline is an Encoder-Decoder from our previous work on end-to-end speech recognition (Bahdanau et al., 2015b), trained with the cross-entropy surrogate loss. We trained a model with the same architecture but using the task loss estimation $\mathcal{L}^{ED}_{\text{greedy2}}$ criterion, which involves greedy prediction of the candidate sequence $\hat{y}$ during training. Algorithm 1 formally describes our training procedure.

Our main result is the 9% relative improvement of Character Error Rate that task loss estimation training brings compared to the baseline model when no external language model is used (see Table 1). This setup, being not typical for speech recognition research, is still an interesting benchmark for sequence prediction algorithms. We note, that the Word Error Rate of 18% we report here is the best in the literature. Another class of models for which results without the language model are sometimes reported are Connectionist Temporal Classification (CTC) models (Graves & Jaitly, 2014; Miao et al., 2015; Hannun et al., 2014b), and the best result we are aware of is 26.9% reported by Miao et al. (2015).

In our experiments with the language models we linearly interpolated the scores produced by the neural networks with the weights of the Finite State Transducer (FST), similarly to (Miao et al., 2015) and (Bahdanau et al., 2015b). Addition of language models resulted in a typical large performance improvement, but the advantage over the cross-entropy trained model was largely lost. Both the baseline and the experimental model perform worse than a combination of a CTC-trained network and a language model. As discussed in our previous work (Bahdanau et al., 2015b), we attribute it to the overfitting from which Encoder-Decoder models suffers due to their implicit language modelling capabilities.

while $L^{ED}_{greedy2}$ *improves on the validation set* **do**
  fetch a batch of input sequences B;
  generate $\hat{y}_i$ for each x_i from B using the greedy search;
  compute the score components $\delta_\alpha(c, x_i, \hat{y}_i^{1\ldots j-1})$;
  compute the component-wise targets $\delta_o(c, x_i, \hat{y}_i^{1\ldots j-1})$ as changes of the optimistic task loss;
  $L^{ED}_{greedy2} = \frac{1}{|B|} \sum_{i=1}^{|B|} \sum_{j=1}^{|\hat{y}|} \sum_{c \in \overline{C}} \Big(\delta_\alpha(c, x_i, \hat{y}_i^{1\ldots j-1}) - \max\Big(\delta_o(c, x_i, \hat{y}_i^{1\ldots j-1}), -5 \Big) \Big)^2$;
  compute the gradient of $L^{ED}_{greedy2}$ and update the parameters α;
end

Algorithm 1: The training procedure used in our experiments. Note, that generation of $\hat{y}_i$ and gradient computation can be combined in an efficient implementation, making it exactly as fast as cross-entropy training.

[3] Our code is available at https://github.com/rizar/attention-lvcsr

Table 1: Character, word, and sentence error rates (CER, WER, and SER) for the cross-entropy (CE) and the task loss estimation (TLE) models. The first three sections of the table present performance of the considered models with no language model integration, with a standard trigram language model (std LM), and with an extended language model (ext LM). The last section contains results from Graves & Jaitly (2014) and Miao et al. (2015). We found that increasing the beam size over 100 for the CE model does not give any improvement. In addition to the results on the test set (eval92) we reported the performance on the validation set (dev93).

Model	Beam size	Eval92 set			Dev93 set		
		CER%	WER%	SER%	CER%	WER%	SER%
CE, no LM	1	7.2	20.4	88.3	8.8	23.9	90.3
TLE, no LM	1	6.2	18.6	85.6	7.8	23.0	91.9
CE, no LM	10	6.4	18.6	85.0	8.8	23.9	90.3
TLE, no LM	10	5.8	17.6	84.7	7.6	22.1	89.9
CE + std LM	100	4.8	10.8	63.4	6.5	14.6	75.0
TLE + std LM	100	4.4	10.5	64.6	6.0	14.2	73.8
CE + ext LM	100	3.9	9.3	61.0	5.8	13.8	73.6
TLE + ext LM	100	4.1	9.6	62.2	5.7	13.7	74.2
TLE + ext LM	1000	4.0	9.1	61.9	5.4	12.9	73.2
Graves et al., CTC, no LM	–	–	27.3	–	–	–	–
Miao et al., CTC, no LM	–	–	26.9	–	–	–	–
Miao et al., CTC + LM	–	–	9.0	–	–	–	–
Miao et al., CTC + ext LM	–	–	7.3	–	–	–	–

It is notable, that the performance of the experimental model changes very little when we change the beam size from 10 to 1. An unexpected result of our experiments is that the sentence error rate for the loss estimation model is consistently lower. Cross-entropy is de-facto the standard surrogate loss for classifiers, and the sentence error rate is essentially the classification error, for which reasons we did not expect an improvement of this performance measure. This result suggests that for classification problems with very big number of classes the cross-entropy might be a non-optimal surrogate loss.

6 Conclusion and Discussion

The main contributions of this work are twofold. First, we have developed a method for constructing surrogate loss functions that provide guarantees about the task loss. Second, we have demonstrated that such a surrogate loss for sequence prediction performs better than the cross-entropy surrogate loss at minimizing the character error rate for a speech recognition task.

Our loss function is somewhat similar to the one used in the Structured SVM (Tsochantaridis et al., 2005). The main difference is that while the structured SVM uses the task loss to define the *difference* between the energies assigned to the correct and incorrect predictions, we use the task loss to directly define the desired score for all outputs. Therefore, the target value for the score of an output does not change during training.

We can also analyze our proposed loss from the perspective of score-landscape shaping (LeCun et al., 2006). Maximum likelihood loss applied to sequence prediction pulls down the score of correct sequences, while directly pulling up on the score of sequences differing in only one element. This is also known as teacher-forcing – the model is only trained to predict the next element of a correct prefixes of training sequences. In contrast, our proposed loss function defines the desired score level for all possible output sequences. Thus it is not only possible to train the model by lowering the score of the correct outputs and raising the score of neighboring incorrect ones, but by precisely raising the score of any incorrect one. Therefore, the model can be trained on its own mistakes.

Future work should investigate the applicability of our framework to other task loss functions like the BLEU score. Our results with the language models stress the importance of developing methods of joint training of the whole system, including the language model. Finally, theoretical work needs to

be done to extend our framework to different approximate inference algorithms as well and to be able to make stronger claims about the suitability of the surrogate losses for gradient-based optimization.

Acknowledgments: We thank the developers of Theano (Bastien et al., 2012) and Blocks (van Merriënboer et al., 2015) for their great work. We thank NSERC, Compute Canada, Canada Research Chairs, CIFAR, Samsung, Yandex, and National Science Center (Poland) for their support. We also thank Faruk Ahmed and David Krueger for valuable feedback.

REFERENCES

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *Proceedings of the ICLR 2015*, 2015a.

Dzmitry Bahdanau, Jan Chorowski, Dmitriy Serdyuk, Philemon Brakel, and Yoshua Bengio. End-to-end attention-based large vocabulary speech recognition. *CoRR*, abs/1508.04395, 2015b. URL `http://arxiv.org/abs/1508.04395`.

Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian J. Goodfellow, Arnaud Bergeron, Nicolas Bouchard, and Yoshua Bengio. Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS Workshop, 2012.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. *arXiv preprint arXiv:1506.03099*, 2015.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pp. 41–48. ACM, 2009.

Christopher Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag New York, 2006.

Jan Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, KyungHyun Cho, and Yoshua Bengio. Attention-based models for speech recognition. *CoRR*, abs/1506.07503, 2015. URL `http://arxiv.org/abs/1506.07503`.

Justin Domke. Generic methods for optimization-based modeling. In *International Conference on Artificial Intelligence and Statistics*, pp. 318–326, 2012.

Jianfeng Gao and Xiaodong He. Training mrf-based phrase translation models using gradient ascent. In *HLT-NAACL*, pp. 450–459, 2013.

Alex Graves and Navdeep Jaitly. Towards end-to-end speech recognition with recurrent neural networks. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pp. 1764–1772, 2014.

Awni Y. Hannun, Carl Case, Jared Casper, Bryan C. Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, and Andrew Y. Ng. Deep speech: Scaling up end-to-end speech recognition. *CoRR*, abs/1412.5567, 2014a.

Awni Y Hannun, Andrew L Maas, Daniel Jurafsky, and Andrew Y Ng. First-pass large vocabulary continuous speech recognition using bi-directional recurrent dnns. *arXiv preprint arXiv:1408.2873*, 2014b.

Tamir Hazan, Joseph Keshet, and David A McAllester. Direct loss minimization for structured prediction. In *Advances in Neural Information Processing Systems*, pp. 1594–1602, 2010.

Xiaodong He, Li Deng, and Wu Chou. Discriminative learning in sequential pattern recognition. *Signal Processing Magazine, IEEE*, 25(5):14–36, September 2008. ISSN 1053-5888. doi: 10.1109/MSP.2008.926652.

Joseph Keshet, Chih-Chieh Cheng, Mark Stoehr, David A McAllester, and LK Saul. Direct error rate minimization of hidden markov models. In *INTERSPEECH*, pp. 449–452, 2011.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States.*, pp. 1106–1114, 2012.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov 1998. ISSN 0018-9219. doi: 10.1109/5.726791.

Yann LeCun, Sumit Chopra, and Raia Hadsell. A tutorial on energy-based learning. 2006.

Yajie Miao, Mohammad Gowayyed, and Florian Metze. Eesen: End-to-end speech recognition using deep rnn models and wfst-based decoding. *arXiv preprint arXiv:1507.08240*, 2015.

Daniel Povey and Philip C Woodland. Minimum phone error and i-smoothing for improved discriminative training. In *Acoustics, Speech, and Signal Processing (ICASSP), 2002 IEEE International Conference on*, volume 1, pp. I–105. IEEE, 2002.

Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer, and Karel Vesely. The kaldi speech recognition toolkit. In *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society, December 2011. IEEE Catalog No.: CFP11SRW-USB.

David A Smith and Jason Eisner. Minimum risk annealing for training log-linear models. In *Proceedings of the COLING/ACL on Main conference poster sessions*, pp. 787–794. Association for Computational Linguistics, 2006.

Veselin Stoyanov, Alexander Ropson, and Jason Eisner. Empirical risk minimization of graphical model parameters given approximate inference, decoding, and model structure. In *International Conference on Artificial Intelligence and Statistics*, pp. 725–733, 2011.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pp. 3104–3112, 2014.

Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. Large margin methods for structured and interdependent output variables. In *Journal of Machine Learning Research*, pp. 1453–1484, 2005.

Bart van Merriënboer, Dzmitry Bahdanau, Vincent Dumoulin, Dmitriy Serdyuk, David Warde-Farley, Jan Chorowski, and Yoshua Bengio. Blocks and fuel: Frameworks for deep learning. *arXiv:1506.00619 [cs, stat]*, June 2015.

Vladimir Naumovich Vapnik. *Statistical learning theory*, volume 1. Wiley New York, 1998.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pp. 3320–3328, 2014.

APPENDIX

Proof for Theorem 1

Proof. As illustrated at Figure 1,

$$L(x,\hat{y}) \leq \begin{cases} F(x,y) + |L(x,\hat{y}) - F(x,\hat{y})|, & \text{if } F(x,\hat{y}) \leq F(x,y), \\ F(x,y) + |L(x,\hat{y}) - F(x,\hat{y})| + F(x,\hat{y}) - F(x,y), & \text{otherwise.} \end{cases}$$

Or simplifying

$$L(x,\hat{y}) \leq F(x,y) + |L(x,\hat{y}) - F(x,\hat{y})| + \max(F(x,\hat{y}) - F(x,y), 0).$$

Finally, after summation over all examples x_i

$$\hat{R}(\alpha) \leq \mathcal{R}_{\text{min,min}} + M(\alpha).$$

□

Proof for Theorem 2

Proof. Let us prove the following inequality

$$\delta_o^{1\dots j} \leq |\delta_\alpha^{1\dots j} - \delta_o^{1\dots j}| + |\delta_\alpha(c_{min}^j, x, y^{1\dots j-1})|, \tag{16}$$

where we denote $\delta_o^{1\dots k} = \delta_o(y^k, x, y^{1\dots k-1})$.

Equation (16) immediately follows from Theorem 1 when we apply it to every step of loss estimation. Then we compute sum over $j = 1 \dots |y|$ in Equation (16), deltas sum to $L_o(x,y) = L(x,y)$, which gives

$$L(x,y) \leq \sum_{j=1}^{|y|} |\delta_\alpha^{1\dots j} - \delta_o^{1\dots j}| + |\delta_\alpha(c_{min}^j, x, y^{1\dots j-1})|, \tag{17}$$

which proves the theorem. □