Can Latent Alignments Improve Autoregressive Machine Translation?

Adi Haviv∗ Lior Vassertail∗ Omer Levy

The Blavatnik School of Computer Science
Tel Aviv University
{adi.haviv,lior.vassertail,levyomer}@cs.tau.ac.il

Abstract
Latent alignment objectives such as CTC and AXE significantly improve non-autoregressive machine translation models. Can they improve autoregressive models as well? We explore the possibility of training autoregressive machine translation models with latent alignment objectives, and observe that, in practice, this approach results in degenerate models. We provide a theoretical explanation for these empirical results, and prove that latent alignment objectives are incompatible with teacher forcing.

1 Introduction
Latent alignment objectives, such as CTC (Graves et al., 2006) and AXE (Ghazvininejad et al., 2020a), have been recently proposed for training non-autoregressive models for machine translation (Libovický and Helcl, 2018; Saharia et al., 2020). These objectives use a dynamic program to comb the space of monotonic alignments between the “gold” target sequence and the token probabilities the model predicts, thus reducing the loss from positional misalignments and focusing on the original prediction error instead. For example, consider the target sequence “there is a tiny difference between pink and magenta”; if the model’s distribution favors the paraphrase “there is a very small difference between pink and magenta”, substituting one token (“tiny”) with two (“very small”) will cause a misalignment, and result in a disproportionately large cross entropy loss. A latent alignment loss would match the predictions of both “very” and “small” with the target “tiny”, while aligning the rest of the sentence properly and computing a much lower loss that focuses on this particular discrepancy. Could latent alignments also benefit autoregressive models?

We apply CTC and AXE to standard autoregressive machine translation models. We observe that, when trained with teacher forcing (Williams and Zipser, 1989), CTC reduces to the vanilla cross entropy loss because CTC assumes that the prediction sequence is longer than the target and has only one valid alignment when they are equal. We further examine AXE, which does not share this assumption, and find that it yields a degenerate model that almost perfectly fits the training set but completely fails at inference time.

Our analysis reveals that latent alignments and teacher forcing are fundamentally incompatible. We observe that there exists a valid alignment in which the prediction $p_i$ is aligned with the target $y_{i-1}$ for almost every token. Simultaneously, teacher forcing feeds the model with $y_{i-1}$ when computing the prediction $p_i$, encouraging the model to simply predict its input under this alignment. While AXE allows this alignment for equal-length prediction and target sequences, the phenomenon also occurs (theoretically) in CTC if the predictions are longer, and in fact, occurs in any latent alignment objective that can align a prediction $p_j$ with a target $y_i$ where $i < j$.

2 Background: Latent Alignments
A latent alignment objective measures the compatibility between the target sequence $Y$ and the sequence of predicted token probabilities $P$ by considering a subspace of possible mappings between $Y$ and $P$. Latent alignments are typically used in non-autoregressive models for automatic speech recognition, and optical character recognition (Graves et al., 2006), and have recently been introduced to the task of machine translation (Libovický and Helcl, 2018; Ghazvininejad et al., 2020a; Saharia et al., 2020). We describe two such objectives, beginning with an overview of the common notation and framework.

Monotonic Alignments Let $Y = y_1, \ldots, y_n$ be the target sequence of $n$ tokens, and $P =$
These algorithms for summation, as in CTC) or the Viterbi algorithm. Table 1: The subspace of alignment functions that the dynamic program combs through the space of alignments by implicitly constructing every possible sequence: \( A_{i,j} = L^f(Y_{\leq i}, P_{\leq j}) \). The dynamic program combs through the space of alignments by implicitly constructing every possibility using the set of local operators defined in Table 1. The subspace of alignment functions that the program explores is determined by the subspace of operators it employs.

| Operator         | Description                                                                 | Formula                                                                 | CTC | AXE |
|------------------|-----------------------------------------------------------------------------|------------------------------------------------------------------------|-----|-----|
| Align            | Predict the target token \( Y_i \) with the distribution \( P_j \). This is the default alignment, advancing along \( A \)'s diagonal. | \( A_{i,j} = A_{i-1,j-1} \cdot P_j(Y_i) \)                             | ✓   | ✓   |
| Clone Target     | Assuming the target token \( Y_i \) was predicted with the previous distribution \( P_{j-1} \), repredict \( Y_i \) using \( P_j \). | \( A_{i,j} = A_{i,j-1} \cdot P_j(Y_i) \)                             | ✓   |     |
| Clone Prediction | Assuming the previous target token \( Y_{i-1} \), was predicted with the distribution \( P_j \), reuse \( P_j \) to predict the next target token \( Y_i \). | \( A_{i,j} = A_{i-1,j} \cdot P_j(Y_i) \)                             | ✓   |     |
| Delimiter        | Use the distribution \( P_j \) to predict the blank token \( \varepsilon \) instead of the target token \( Y_i \). This operation is akin to inserting \( \varepsilon \) into the target sequence at the \( i \)-th position. | \( A_{i,j} = A_{i,j-1} \cdot P_j(\varepsilon) \)                       | ✓   | ✓   |

Table 1: The set of possible operators in latent alignment dynamic programs. Both AXE and CTC use *align* and *delimiter*, but apply different *clone* operators.

\( p_1, \ldots, p_m \) be the model prediction, a sequence of \( m \) token probability distributions. A monotonic alignment \( \alpha \) is a function that maps every target position \( i \in \{1, \ldots, n\} \) to a set of one or more consecutive prediction positions \( \alpha(i) \subseteq \{1, \ldots, m\} \), such that \( i \leq j \iff \max \alpha(i) \leq \min \alpha(j) \).

**Objective** Given an alignment \( \alpha \), the objective is defined as follows:

\[
L_\alpha(Y,P) = \prod_{i=1}^{n} \prod_{j \in \alpha(i)} p_j(y_i) \tag{1}
\]

Since \( \alpha \) is not provided a priori, it is necessary to aggregate over all the possible alignments (hence *latent* alignments), by either summation (Equation 2) or maximization (Equation 3):

\[
L^{\Sigma}(Y,P) = \sum_{\alpha} L_\alpha(Y,P) \tag{2}
\]

\[
L^{\text{max}}(Y,P) = \max_{\alpha} L_\alpha(Y,P) \tag{3}
\]

In practice, the negative log loss is minimized during training:

\[
\ell(Y,P) = -\log L(Y,P) \tag{4}
\]

**Dynamic Programming** Aggregation can be done efficiently with dynamic programming, using derivations of the forward-backward algorithm (for summation, as in CTC) or the Viterbi algorithm (for maximization, as in AXE). These algorithms create an aggregation matrix \( A \in R^{n \times m} \), where each cell represents the desired aggregation score \( f \) (sum or max) over prefixes of the target and prediction probability sequences: \( A_{i,j} = L^f(Y_{\leq i}, P_{\leq j}) \). The dynamic program combs through the space of alignments by implicitly constructing every possibility using the set of local operators defined in Table 1. The subspace of alignment functions that the program explores is determined by the subspace of operators it employs.

**Connectionist Temporal Classification (CTC)** The CTC objective (Graves et al., 2006) was originally introduced for speech and handwriting recognition, where the prediction sequence \( P \) is typically much longer than the target sequence \( Y \) (\( m \gg n \)). While computing the summation objective (Equation 2), CTC uses only the *align*, *clone target*, and *delimiter* operators. This means that CTC restricts \( \alpha \) to the space of alignments where every item in \( P \) is aligned with at most one item in \( Y \), i.e. \( \alpha(i) \cap \alpha(j) = \emptyset \) for \( i \neq j \).

CTC was used in non-autoregressive machine translation by Libovický and Helcl (2018) and more recently by Saharia et al. (2020). In both cases, the prediction sequence was artificially inflated to be double (or more) the length of the source-language input sequence in order to simulate the \( m \gg n \) condition of speech recognition.

**Aligned Cross Entropy (AXE)** The AXE objective (Ghazvininejad et al., 2020a) is specifically designed for non-autoregressive machine translation. AXE finds the monotonic alignment that minimizes the cross entropy loss (i.e., maximizes the likelihood, Equation 3) in order to focus the penalty on the root errors instead of positional shifts that result from them. AXE uses only the *align*, *clone prediction*, and *delimiter* operators. This combination of operators allows AXE to align prediction and target sequences of any lengths because *clone prediction* inflates the prediction sequence while *delimiter* adds new target tokens. However, since AXE cannot clone target tokens, every target position \( i \) is always aligned to a *single* prediction position, i.e. \( |\alpha(i)| = 1 \). Figure 1 illustrates how AXE aligns the model’s predictions with the target sequence.
**3 Combining CTC with Teacher Forcing Defaults to the Trivial Alignment**

In an autoregressive setting, it is standard practice to use teacher forcing (Williams and Zipser, 1989); i.e., when predicting the $i$-th token, the model takes the prefix of the (gold) target sequence $Y_{\leq i}$ as input. This dictates that the number of predictions is identical to the number of target tokens ($m = |P| = |Y| = n$).

However, CTC assumes that the prediction sequence $P$ is typically much longer than the target sequence $Y$ ($m \gg n$), and can only inflate $Y$ via clone target and delimiter (see Section 2). This leaves only one valid alignment when $m = n$: the trivial alignment $\alpha(i) = \{i\}$. CTC will thus default to the same objective as the standard cross entropy loss.

Unlike CTC, the AXE objective aggregates over multiple alignments even when $m = n$, because it uses both the delimiter operator (which inflates $Y$) as well as the clone prediction operator (which inflates $P$).

**4 Applying AXE to Autoregressive NMT**

To apply AXE to autoregressive machine translation, we use a standard sequence-to-sequence transformer model (Vaswani et al., 2017) trained with teacher forcing, replace the simple cross entropy loss function with AXE, and add the empty token $\varepsilon$ to the vocabulary. We remove the $\varepsilon$ tokens after decoding.

**Experiment Setup** We use fairseq (Ott et al., 2019) to train a transformer encoder-decoder (Vaswani et al., 2017) on the IWSLT’14 DE-EN dataset (Cettolo et al., 2015). The dataset is preprocessed and tokenized into subwords with BPE (Sennrich et al., 2016) using the scripts provided by fairseq. We also use the implementation’s default hyperparameters: 6 layers of encoder/decoder, 512 model dimensions, 1024 hidden dimensions, 4 attention heads. We optimize with Adam (Kingma and Ba, 2015) for 50k steps with early stopping using 4096 tokens per batch. We decode with beam search ($b = 5$) and evaluate performance with BLEU (Papineni et al., 2002).

**Results** We observe two seemingly contradictory behaviors. On the one hand, the model approaches a near-zero training loss within a single epoch, and observes similar results when computing AXE loss on unseen examples in the validation set (Figure 2). Meanwhile, at inference time, the model consistently produces the empty sequence (after removing all instances of $\varepsilon$), scoring 0 BLEU on the test set. This indicates that the model has learned to “game” the AXE objective without actually learning anything useful about machine translation. What shortcut did the model learn?

**5 Analysis**

To understand how the model learns to game the AXE objective, we analyze the optimal alignments chosen by the objective, and find that they allow the model to condition on the target token when trying to predict it. We prove that this is the optimal solution when combining teacher forcing and AXE, and that it holds for any latent alignment objective that allows the model to align future target tokens with the current prediction.
AXE finds a constant alignment  We examine the alignments chosen by AXE’s dynamic program for a sample of training examples, and observe that they all belong to a consistent pattern: delimiter, align, align, ..., clone prediction. In other words, the chosen path skips the first prediction with teacher forcing, it can simply learn to predict its input at each position, and assume that AXE will align the prediction with the previous token (which is identical to the input). For example, \( p_2 \) predicts “thank” with very high probability because teacher forcing uses the previous target \( y_i \) as the decoder’s input in the second position. Notice how the final prediction \( p_n \) is used twice to predict both “.” and EOS.

Inference produces empty sequences  The model essentially learns to produce the blank token \( \varepsilon \) in the first step, and then copy the latest token that is fed into the decoder as input. During training, that input is indeed the target token. At inference, however, it is the model’s prediction from the previous timestep. Since the first prediction is \( \varepsilon \), the model will continue and predict the blank token until the end of the sequence.

This exploit is not unique to AXE  AXE is not the only latent alignment objective that the model can “game” when coupled with teacher forcing. We would see a similar phenomenon if we were to use CTC with a longer prediction sequence; for example, if we doubled the prediction length (Libovický and Helcl, 2018) and applied a version of teacher forcing that feeds each target token twice in a row. In fact, every latent alignment objective that can align a prediction \( p_i \) with a target \( y_i \) where \( i < j \) will be subject to this exploit, and allow a model trained with teacher forcing to glimpse into the future.

Restricting AXE to causal alignments leads to the trivial alignment  We further limit AXE to allow only causal alignments, where a prediction \( p_j \) may only align with a target \( y_i \) if \( i \geq j \). After training with the restricted objective, we observe that AXE selects the trivial alignment \( (i = j) \) in 98% of the validation set sentences, whereas the remaining 2% contain only minor deviations from the trivial alignment, typically one delimiter quickly followed by one clone prediction.

6 Conclusion  This work elaborates why latent alignment objectives are incompatible with autoregressive models.
trained with teacher forcing. That said, teacher forcing might not be the best way to train a machine translation model (Bengio et al., 2015; Lamb et al., 2016; Ghazvininejad et al., 2020b), and perhaps a future alternative could reopen the discussion on applying latent alignment objectives to autoregressive models.

Acknowledgements

This work was supported in part by Len Blavatnik and the Blavatnik Family foundation, the Alon Scholarship, and the Tel Aviv University Data Science Center.

References

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In Advances in Neural Information Processing Systems, volume 28. Curran Associates, Inc.

M. Cettolo, J. Niehues, S. Stüker, L. Bentivogli, and Marcello Federico. 2015. Report on the 11th iwslt evaluation campaign, iwslt 2014.

Marjan Ghazvininejad, Vladimir Karpukhin, Luke Zettlemoyer, and Omer Levy. 2020a. Aligned cross entropy for non-autoregressive machine translation. In Proceedings of the 37th International Conference on Machine Learning, volume 119 of Proceedings of Machine Learning Research, pages 3515–3523. PMLR.

Marjan Ghazvininejad, Omer Levy, and Luke Zettlemoyer. 2020b. Semi-autoregressive training improves mask-predict decoding.

Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. 2006. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In Proceedings of the 23rd International Conference on Machine Learning, ICML ’06, page 369–376, New York, NY, USA. Association for Computing Machinery.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings.

Alex M Lamb, Anirudh Goyal ALIAS PARTH GOYAL, Ying Zhang, Saizheng Zhang, Aaron C Courville, and Yoshua Bengio. 2016. Professor forcing: A new algorithm for training recurrent networks. In Advances in Neural Information Processing Systems, volume 29. Curran Associates, Inc.