Non-autoregressive Sequence-to-Sequence Vision-Language Models

Kunyu Shi Qi Dong Luis Goncalves Zhuowen Tu Stefano Soatto
AWS AI Labs
{kunyus, qdon, luisgonc, ztu, soattos}@amazon.com

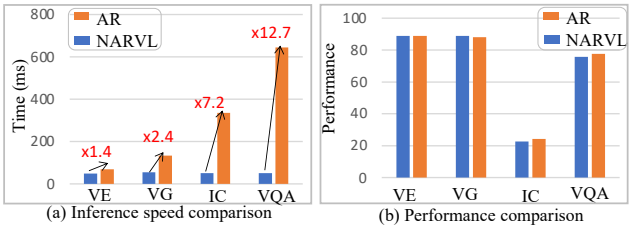
Abstract

Sequence-to-sequence vision-language models are showing promise, but their applicability is limited by their inference latency due to their autoregressive way of generating predictions. We propose a parallel decoding sequence-to-sequence vision-language model, trained with a Query-CTC loss, that marginalizes over multiple inference paths in the decoder. This allows us to model the joint distribution of tokens, rather than restricting to conditional distribution as in an autoregressive model. The resulting model, NARVL, achieves performance on-par with its state-of-the-art autoregressive counterpart, but is faster at inference time, reducing from the linear complexity associated with the sequential generation of tokens to a paradigm of constant time joint inference.

1. Introduction

Sequence-to-sequence autoregressive Transformers [12, 36, 44] are deep neural network architectures that map a sequence of tokens, each representing a segment of text as a vector, onto another sequence, typically representing the same sequence shifted forward by one. Such models can handle a variety of tasks [25, 35, 36], whereby the input (query) text could be a sentence in natural language, and the output (target) the same sentence in a different language (translation), or the answer to a question expressed in the input (question-answering, QA), the name of an entity or class, etc. The Transformer architecture's versatile and unified design has led to the development of all-in-one (AIO) models, such that multiple tasks can be approached as a sequence-to-sequence translation problem.

Vision-Language AIO Models [31, 46, 50], including sequence-to-sequence, have proven successful at mapping multimodal inputs, typically images and strings of text, to textual outputs that encode tasks expressible as a string of text, such as visual question answering (VQA), visual grounding (VG), visual entailment (VE), and image captioning (IC). These auto-regressive sequence-to-sequence models face the inference cost issue, since they tend to be unwieldy and need to be executed T times to generate an output sequence of length T.



(a) Inference speed comparison (b) Performance comparison

Figure 1. Comparison of inference speed and performance between NARVL (non-autoregressive) and its autoregressive counterpart on four vision-language tasks: Visual entailment (VE), Visual grounding (VE), Visual Question Answering (VQA), and Image captioning (IC). From (a), we see that NARVL speeds up the inference of AR by a factor between 1.4 and 12.7, while achieving on-par performance.

Non-autoregressive methods are proposed in some recent Visual-language AIO models [23], which formulate sequence-to-sequence mapping as a bipartite matching problem. This approach excels in tasks where visual information is key, such as object grounding and detection. However, it's less effective of handling language-focused tasks like Visual Question Answering and Image Captioning. This discrepancy may stem from the nature of the tasks: in object detection/grounding, tokens are orderless and each token correlates to distinct objects or boxes, leading to a weaker inter-object correlation compared to the stronger inter-word correlation in sentences where tokens are ordered. Consequently, the set-to-set, order-independent translation method is more suitable for visual tasks than for language-oriented ones.

Main hypothesis: We hypothesize that a transformer-based architecture could leverage the homogeneity of the input and output spaces, while enabling more flexible output spaces. In particular, we are interested in the possibility of performing *joint decoding* of a sequence in one step, rather than step-by-step. We test whether such an architecture could achieve performance comparable to the autoregressive baseline at significantly reduced inference cost.

To test this hypothesis, we develop a new Visual-language AIO model, turning a Transformer-based autoregressive one-step prediction model into a *joint predictor* of the target tokens, as explained in Sect. 3. In Sect. 4 we show that such a model, which we name NARVL, can be used for the multiple visual-language tasks of interest (VQA, captioning, entailment, grounding). As shown in Fig 1, NARVL achieves comparable performance to a state-of-the-art autoregressive model, with significant speed advantage ranging from 1.4 to 12.7 times.

NARVL is made possible by re-purposing of the decoder of an autoregressive Transformer model, and the model has a layer of *learnable query tokens* (LQT) that are fixed at inference time and learned during fine-tuning. NARVL is enabled by Query-CTC (Q-CTC) loss, a variant of the CTC loss used in audio and language [15] but never applied to the visual domain, where the ordinary empirical cross-entropy loss (CE) is marginalized with respect to generative variability in the prediction. Whereas in the language domain the multiple decoding hypotheses stem from the output of the encoder, in vision this is limiting, since input and output spaces are heterogeneous. Therefore, we modify the CTC loss to marginalize not with respect to decoding paths, but with respect to paths from the *sequential* learnable query tokens of order indexes to the predicted tokens.

Our **key contributions** can therefore be summarized as follows: (i) we propose a new sequence-to-sequence *non-autoregressive* all-in-one vision language model, that generates sequences in-parallel. (ii) We introduce Query-CTC loss to train this architecture, inspired by the CTC loss used in audio recognition and language, that leverages the sequential learnable query tokens to generate multiple generative paths, and marginalizes the resulting population in the ordinary cross-entropy loss. We show that (iii) the resulting architecture is competitive with state-of-the-art autoregressive architecture in multiple vision-language tasks, at a significantly reduced inference time, since the model is executed once at inference time, rather than sequentially for as many steps as tokens in the output layer.

2. Related Work

Sequence to Sequence Generation. Many NLP tasks share a common problem setting where the input consists of sequences of words with the output being targeted sequences. Therefore, the sequence to sequence formulation becomes a prototypical setting for many tasks in NLP [43], which can be readily solved by an autoregressive (AR) model. Beyond recurrent neural networks (RNNs), the decoder in Transformers [44] also adopts the autoregressive strategy in both training and prediction. On the vision side, considering an image as a sequence of tokens was popularized by the Vision Transformer (ViT) [10]. Transformers [44] based approaches pix2seq [5, 6] formulate object detection, instance segmentation, and human poses as sequence generation problem, which differs a lot from approaches designed specifically for individual tasks [4, 9, 24]. Furthermore, [46, 54] unify more tasks with seq2seq models. Recently, vision-language tasks have received increasing attention [32], including visual questioning and answering [2], and visual grounding [51] that have also been tackled by AR models for sequence to sequence generation.

Non-autoregressive Sequence Generation. [17] proposes non-autoregressive (NAR) Transformer model for machine translation that generates the translated sequence in parallel. The main challenge in NAR is to capture the inter-token dependency [16] since the predictions are made conditionally independent. To improve inter-token dependency, in [18, 40–42, 45, 49], the model architectures are modified to conduct local NAR only, or light AR layers are added at the end of the decoder. In [1, 3, 22, 37, 39], hand-crafted or learnable latent variables are incorporated. Knowledge distillation [55] has proven effective to reduce complexity of training and increase robustness. In [14, 30], curriculum learning has been applied to simplify the learning task.

Set-to-Set Prediction. In the seminal Detection Transformers (DETR) [4], a set of object queries are turned into a set of detected objects. A key component is the Hungarian matching step that deterministically assigns object queries to the ground-truth objects in training. Broadly speaking, we can also view the decoder in DETR as performing non-autoregressive set generation, which works well for order-less objects but may not be directly applicable to text sequence generation in vision-language models.

Non-autogressive Vision-Language Models. Inspired by [17], we develop NARVL, a non-autogressive vision-language model and illustrate the effectiveness of NARVL on a recent autogressive model, OFA [46]. The Connectionist Temporal Classification (CTC) loss [15] and knowledge distillation [55] have been adopted in NARVL to implement a NAR solution. CTC is designed to align the two sequences with different lengths, and it marginalizes over all possible monotonic alignments. It assumes output always longer than target, and the final output is decoded by collapsing repetitive tokens. The variant of the CTC loss we introduce is formalized in Eq. 1, and NARVL described in Sect. 3.

3. Method

We co-opt a pre-trained sequence-to-sequence autoregressive model (OFA [46]) and turn it from a one-step predictor (AR) encoding of $p(y_{t+1}|x_1, \dots x_T, y_1, \dots y_t)$ to a flexible task-dependent joint encoding of the target query given heterogeneous inputs $p(y_1, \dots, y_N|x_1, \dots x_T; q)$, where x_i are token embeddings of images and text and y are output token embeddings. The overview of the proposed NARVL is shown in Figure 2.

Specifically, we embed an image with a convolutional

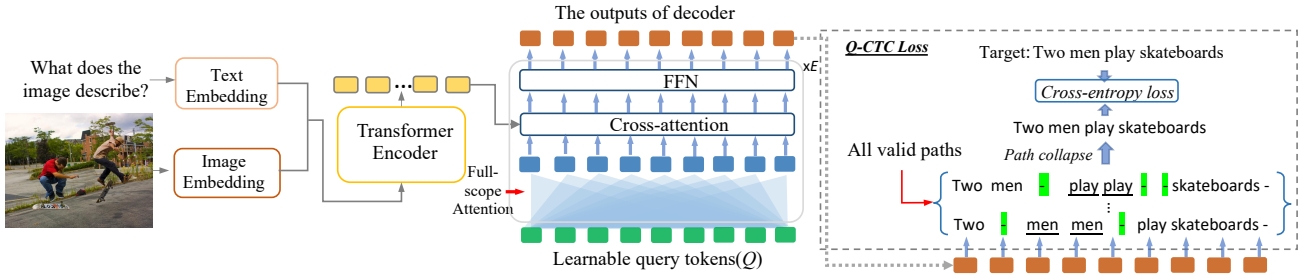


Figure 2. **The overview of NARVL.** NARVL borrows the encoder from OFA [46], where the embedding sequence of input text and image CNN (ResNet) feature are concatenated in the input token sequence. Unlike the standard transformer decoder that generates outputs sequentially, conditioning on the generated sequence, our non-autoregressive decoder takes a sequence of tokens that are learnable weights, and generates outputs for all tokens in parallel. As the output sequence length is unknown, we set the number of of learnable query tokens to a value (hyperparameter) larger than the largest target sequence length. The loss used, Q-CTC, is described in Eq. 1.

backbone and obtain an activation map with D channels, which we represent as $W \times H$ tokens of dimension D, each representing one among the $W \times H$ pixels, along with a positional encoding. We concatenate visual tokens with textual tokens, obtained from the input string with BPE tokenization [38], to form the input to the encoder, which is identical to [46]. The decoder, however, is different from OFA [46], although it shares the overall structure, thus enabling us to leverage its pre-trained weights to fine-tune.

3.1. Parallel Transformer Decoder

The OFA decoder is an auto-regressive predictor that takes as input the output of the encoder and a sequence of T consecutive input tokens, and produces as output the same sequence shifted forward by one, limiting the hypothesis space to the range of the $T + 1$ token, and forcing repeated execution T times at inference time, to produce the single last token (see Figure 3(a)). Our NARVL decoder also takes as input the output of the encoder, but it replaces the input sequence *with a fixed-length and constant sequence of tokens,* representing the task-specific hypothesis space implicit in the training data (see Figure 3(c)).

This constant layer of *Learnable Query Tokens* (LQT), is chosen during training as the output of a module with learnable parameters, including the number of tokens, as a hyperparameter, bounded from below by a function of the length of the ground truth output sequence (target tokens) in the training set.

Discussion The design of NARVL decoder is different from the existing non-autoregressive Transformer decoder proposed in Natural Language processing [16–18] (see Figure 3(b)), which utilizes the outputs of the encoder as the inputs to the decoder. However, unlike translation, the sequence lengths of input and output sequences are not strongly correlated in vision-language tasks. The NARVL encoder-decoder design is also different from set-to-set decoders [4, 23], and our learnable queries for the decoder is a sequence rather than a set. For each learnable query, we add absolute position embeddings in the bias of attention layers, to force the structure of output sequences.

3.2. NARVL with Query Connectionist Temporal Classification Loss

NARVL is trained with a variant of the CTC loss [15] used in audio and language, which consists of a cross-entropy marginalized over a distribution of predictions. In our case, the distribution of prediction corresponds to multiple inference paths in the decoder from the redundant learnable query tokens, to the smaller sequence of output tokens.

3.2.1 Q-CTC for Visual-Language Translation

Suppose we have a training set of n image-text pairs $\mathcal{D} = \{(\mathbf{I}_i, \mathbf{T}_i, \mathbf{Y}_i)\}_i^n$ where $\mathbf{I}$, $\mathbf{T}$, $\mathbf{Y}$ specify the input images, input texts and target outputs, respectively. Note that the output sequences can be text sequences or location sequences. The model generates $\mathbf{Y}$ according to the inputs $\mathbf{I}$ and $\mathbf{T}$. In order to formalize the expression of the Q-CTC loss,we call the Learnable Query Tokens (LQT) $q \in \mathbb{R}^{D \times N}$, the outputs of the encoder $x \in \mathbb{R}^{D \times L}$, the output tokens are $z \in \mathbb{R}^{D \times N}$, and the ground truth tokens $y \in \mathbb{R}^{D \times T}$, where $N > T$. Let the decoder vocabulary denote D_d, containing d possible target tokens including the whole valid vocabulary tokens and one blank token $-$.

Let $\hat{z}_i(x; q)$ denote the "path" from the query set to the output token, meaning the value of z as a (deterministic) function of the encoding x for a given set of LQTs q. We denote the ensemble of paths $p(z|x; q) = \delta(z - \hat{z}(x; q))$, where δ is Dirac function and q are learnable parameters. The ordinary cross-entropy loss would be $L_{CE}(\theta) = -\sum_i \log \frac{e^{f_{y_i}(z_i)}}{\sum_j e^{f_{y_j}(z_i)}}$ where $f_{y_j}(\cdot)$ computes the logits of x_i on target token class y_i, and θ are the learnable parameters

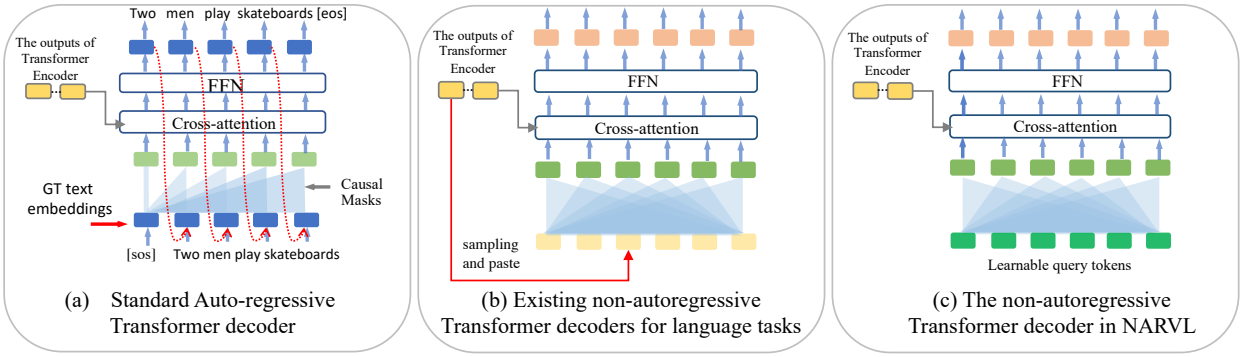


Figure 3. Comparison of various design of Transformer decoder. (a) Standard Auto-regressive Transformer decoder; (b) The existing non-autoregressive Transformer decoders for audio and language tasks; (c) The proposed non-autoregressive Transformer decoder in NARVL. During training, an AR decoder (a) uses teacher forcing with causal masks, where tokens can only attend to previous tokens, while all tokens can attend to each other in the decoders of (b) and (c). NARVL decoder has dedicated query tokens inputs, instead of using the outputs of the encoder as inputs in (b). This design avoids the large latency of the decoder due to the long output sequence from the encoder.

of the encoder that produces x, and of the decoder that produces z given x.

Given the vocabulary size d and output sequence length N, there are N^d possible output sequences. Among all output sequences, the Q-CTC selects the valid output sequences, and maximizes the probability over all valid paths. Collapse operation, denoted as $\mathcal{B}$, removes all blank tokens $-$, and merge continuous repetitive tokens between two blank tokens. For example, $\mathcal{B}(-\text{ a bag on a table}) = \mathcal{B}(\text{a} - \text{bag bag} - \text{on a a table} -) = \text{a bag on a table}$. There are many possible sequences that can be collapsed to the correct target sequence, which are valid alignments. When we calculate the loss, we marginalize over the set of valid paths. We denote the collapsed valid sequences as $\tilde{p}(z|x;q)$.

Before x is instantiated at inference time, z_i are random variables, functions of x, which we need to marginalize in the loss, obtaining

$$L_{\text{Q-CTC}}(\theta, q) = -\sum_{k} \sum_{z_i \sim \tilde{p}(z|x_k;q)} \log \frac{e^{f_{y_i}(z_i(x_k;q))}}{\sum_j e^{f_{y_j}(z_i(x_k;q))}} \tag{1}$$

, where k is the number of all training samples. Note that this loss is different from the CTC loss, where the marginalization only depends on x, not q. Our loss is a function of θ *as well as the learnable parameters* used to produce q, in addition to any other hyperparameters shared with the ordinary autoregressive model.

3.2.2 Knowledge Distillation

Optimising Q-CTC is more challenging in visual-language domain than language domain, because the image-text input sequence has less structure than pure language inputs, and the model is required to generate well-structured sequences from less order sequences. Due to the large solution space and the lack of inter-token dependency between decoder tokens, non-autoregressive models can have lower performance compared to auto-regressive models. To reduce the solution space in model training, we exploit two simple knowledge distillation mechanisms[20] in training. Specifically, we train a standard Transformer with an auto-regressive loss, and set this model as the teacher model, which has significantly less variations in ground-truth and makes learning targets more deterministic. We observe that this knowledge distillation benefits the task of long sequence outputs such as image captioning, and the detailed discussion is in Section 4.3. We also observe that the model initialization weights are non-trivial for training Q-CTC loss, and we use OFA pretrained weights to initialize NARVL and finetune it in various down-stream tasks.

3.2.3 Model inference

During the inference, for each output token, we use the text token in the vocabulary with the largest probability as prediction. We follow the same path collapse rules used in training to remove - and repetitive tokens to obtain the final predicted sequence.

4. Experiments

We perform experiments on various vision-language tasks, and make performance and speed comparisons to the state-of-the-art models to show the effectiveness of the proposed method. Figure 4 summarizes the tasks we experiment on.

4.1. Implementation details

We implement NARVL starting from the official OFA[46] code, which is written in the fairseq library [34]. To make a

Task	Model Inputs		Model Outputs
	Image	*Language*	
VQA		How many people are in the image?	2
VG		The skateboarder who wears a hat.	[x, y, w, h]
IC		What does the image describe?	Two men are playing skateboards.
VE		Three people are running.	Contradiction

Figure 4. We test the proposed NARVL on various vison-language tasks, including Visual Question Answering (VQA), Visual grounding (VG), Image Captioning (IC) and Visual Entailment (VE). The inputs and outputs of each tasks are illustrated here, and all types outputs are unified within the sequence formulation.

fair comparison to the autoregressive OFA model, we keep most hyper-parameters and training schedule the same and this helps us to understand the effect of switching decoder paradigm. Some OFA models are reproduced to get the weights for speed benchmarking purpose, and are noted in the result tables. We use the same OFA encoder task prompts, but don't use decoder prompts, as the NARVL decoder doesn't do conditional token generation.

Q-CTC loss is used in the final model of all tasks, and knowledge distillation is only used in Image Captioning and VQA tasks. We follow the model size settings used in [46]. We benchmark our model speed on Tesla V100-SXM2-16GB GPU with a batch size of 1, and take the average over the test samples. As the GPU needs to ramp up, the first image tends to be much slower than other images, and we remove the first sample inference time in all calculations. Gradient accumulation is used at training time to increase the batch size of the below experiments.

Referring Expression Grounding RefCOCO[51], RefCOCO+[51], RefCOCOg[33] datasets are created based from the COCO dataset, where a piece of text (referring expression) that describes a unique object in the image is given and the model is asked to find the object.

In both RefCOCO and RefCOCO+, testA only contains people and testB only contains non-people objects. We feed the referring expression text sequence and image to the encoder, and let the decoder predict the fixed length output sequence of bounding box position tokens.

When benchmarking the model speed, we take the average inference time on the entire validation or test set. On RefCOCO and RefCOCOplus, we average the inference time of val, testA and test B subsets and only report the average inference time for simplicity. Similarly we average over val and test subsets for RefCOCOg dataset. We train NARVL for 10 epochs with effective batch size of 128, and we set the number of decoder learnable query input tokens to 5.

Visual Entailment SNLI-VE dataset is built off SNLI and Flickr30K datasets, and the task requires the model to reason relationship (entailment, contradiction and neutral) between an image premise and text sentence hypothesis. Image premise and text sequence hypothesis are fed into the NARVL encoder, and the decoder predicts the sequence of one token that is one of the relationship words. We use a batch size of 256 and train our model for 5 epochs, and the number of decoder input learnable query tokens is set to 2. Knowledge distillation is not adopted given the simplicity of the output sequence of this task.

Visual Question Answering Visual Question Answering is a task that requires cross-modal reasoning that the model is asked to answer question by looking at image. The number of decoder input tokens is set to 6, and the model is trained with 10 epochs with a batch size of 512. We don't use any candidate answer set based constraint on the generated answer sequence. We found the default OFA model that uses all-candidate inference is quite slow, and we additionally benchmark and report the faster beam-search version that was released in the OFA github repository. Speed is benchmarked on the validation set, and the accuracy numbers are obtained from the official evaluation server.

MSCOCO Image Captioning Image captioning requires models to generate a fluent and meaningful natural language sentence that describes the image. We use batch size of 128 and train 5 epochs with decoder input sequence length of 20. Each image in the caption dataset has 5 captions written independently from 5 annotators, which increases the difficulty for NARVL training which might merge possible captions and generate captions that are not fluent, and knowledge distillation is adopted to simplify the training captions.

4.2. NAR vs AR

We compare accuracy and speed between AR (autoregressive) and our proposed NAR (non-autoregressive) models on various vision- language tasks and results are shown in Table 1. It shows that the NAR model consistently outperforms the AR model on visual grounding and has significantly higher execution speed (with 2.4 to 12.7 times speed up on various tasks). As visual entailment requires shorter output sequences, the speedup is smaller than for other tasks. The significant speedups are on VQA and Captioning datasets, because the length of output sequence is longer than that of the grounding task. We will analyse the accuracy and inference speed for each downstream tasks in next section.

4.3. Benchmark Performance

RefCOCO, RefCOCO+, RefCOCOg results. Following the metric used previous works, we report Acc@0.5 numbers. We compare NARVL to other methods in Table 2. Our proposed NARVL model achieves state-of-the art per-

Table 1. Accuracy and speed comparisons of AR (autoregressive) and our proposed NAR (non-autoregressive) models on various vision-language tasks. Our NAR model is trained under the exact same settings of model size, parameters and training schedule as the AR model for fair comparison. The NAR model consistently outperforms the AR model on visual grounding and has significantly higher execution speed. As visual entailment requires shorter output sequences, the speedup is smaller than for other tasks. We observe significant speedups on VQA and Captioning datasets, but with a measurable performance drop. Beam search is used in all AR models and greedy decoding is used in NAR models (beam search can be applied and are studied in 10, but not adopted due to its sequential nature.). All models reported here are in base size. Inference wall clock time is measured in ms.

	(a) Visual Grounding										
	RefCOCO				RefCOCO+				RefCOCOg		
Method	Val	TestA	TestB	Time	Val	TestA	TestB	Time	Val	Test	Time
AR	88.15	90.08	83.45	133.6/1×	81.67	86.40	74.49	133.1/1×	81.92	82.02	132.8/1×
NAR	**88.78**	**90.63**	**84.67**	**54.8 / 2.4**×	**82.35**	**87.15**	**74.74**	**54.6/2.4**×	**82.27**	**82.69**	**54.8/2.4**×

	(b) Visual Entailment			(c) Visual Question Answer			(d) Image Captioning				
Method	Dev	Test	Time	Test-dev	Test-std	Time	BLEU@4	METEOR	CIDEr	SPICE	Time
AR	89.0	89.0	68.0/1×	**77.48**	**77.58**	645.1/1×	**41.0**	**30.9**	**138.2**	**24.2**	366.0/1×
NAR	89.0	89.0	**48.7/1.4**×	75.69	75.75	**50.7/12.7**×	36.4	28.7	123.1	22.5	**51.2/7.2**×

Table 2. Results on visual grouding datasets: RefCOCO, RefCOCO+ and RefCOCOg, and comparisons to previous works. * Weights are not released and the model was reproduced by us using the official released training scripts. Difference to the reported results in [46] might due to randomness in checkpoint picking. We do not perform knowledge distillation for this task.

	RefCOCO			
Method	Val	TestA	TestB	Speed (ms)
UNITER[7]	81.41	87.04	74.17	-
VILLA[13]	82.39	87.48	74.84	-
MDETR[23]	86.75	89.58	81.41	73.5
UNICORN[50]	88.29	90.42	83.06	266.4
OFA*[46]	91.24	93.41	87.16	284.9
NARVL$_{tiny}$(ours)	80.40	84.64	73.8	**30.7**
NARVL$_{base}$(ours)	88.78	90.63	84.67	54.9
NARVL$_{huge}$(ours)	**91.8**	**94.24**	**88.01**	150.8

	RefCOCO+			RefCOCOg	
Method	Val	TestA	TestB	Val	Test
UNITER[7]	75.90	81.45	66.70	74.86	75.77
VILLA[13]	76.17	81.54	66.84	76.18	76.71
MDETR[23]	79.52	84.09	70.62	81.64	80.89
UNICORN[50]	80.30	85.05	71.88	83.44	83.93
OFA* [46]	86.93	91.37	80.51	86.38	87.70
NARVL$_{base}$(ours)	82.35	87.15	74.74	82.27	82.69
NARVL$_{huge}$(ours)	**87.90**	**92.18**	**81.2**	**87.7**	**88.42**

formance on all subsets of RefCOCO, RefCOCO+ and RefCOCOg datasets. As shown in Table 1, our NAR model consistently outperforms the AR model on all subsets of the three datasets and has significantly faster speed. Take RefCOCO dataset as an example, our NAR model has on average 0.83 higher accuracy compared to the AR model, and is 2.4 times faster (54.8 ms vs 133.6 ms). The speedup is introduced by the parallel nature of NARVL decoder, and we argue the accuracy improvements come from the bidirectional attention of our parallel decoder, as opposed to uni-directional attention in autoregressive decoder, where the later generated coordinate tokens are not available in attention operation of the early token generation.

Table 3. SNLI-VE visual entailment results. NARVL shows on par performance to previous state-of-the-art. Knowledge distillation is not used in this task.

Method	dev	test
UNITER[7]	73.8	74.0
VinVL[53]	76.5	76.6
UNIMO[27]	75.0	75.3
ALBEF[26]	75.8	76.0
METER[11]	77.7	77.6
VLMo[47]	79.9	80.0
SimVLM[48]	80.0	80.3
Florence[52]	80.2	80.4
OFA [46]	91.0	**91.2**
NARVL$_{base}$(ours)	89.0	89.0
NARVL$_{huge}$(ours)	**91.1**	91.1

SNLI-VE results We compare our NAR model and AR model in Table 3. Our model has average inference time of 48.73 ms, which is 1.4 times faster than the AR model with 68.03 ms. The effective target sequence excluding EOS has only 1 token, which means decoder attention in essence is the same for NAR and AR model, and we see exactly the same performance. As the output sequence (label-EOS) for this dataset is shorter than other dataest, the relative speedup from NAR model is smaller. Our NARVL shows on par performance as the current state-of-art OFA model on both validation and test set, as shown in Table 3 along with comparisons to other previous methods.

VQA results Comparisons to the AR model on VQA are shown in 1. NAR model performance is on average 1.81 lower than the AR model, while it has gigantic speed up of 12.7 times (50.7 ms vs 645.1 ms). Our model shows competitive results and more comparisons to previous methods can be found in Table 4. The default OFA model uses a slow

all-candidate decoding method, and we additionally report the results with beam search decoding. It is noteworthy that OFA-VQA was trained with trie-based auto-regressive decoding that predicts next node on a Trie in each step, and the speedup would be smaller over standard AR models. In this task, one question might have multiple valid answers e.g. "What's the color of the shirt?" can be answered with "Red and White" or "White and Red".This might create confusions for NARVL, which leads to incorrect predictions like "White and White", and here we use knowledge distillation from an AR model of the same size to make the answers used in training more deterministic.

Table 4. Results on VQA. We report base version of NARVL with knowledge distillation. *beam search version is released in the official OFA github that is much faster than the original allcnd version reported in the OFA paper, and we present results from both models here.

Method	test-dev	test-std	Speed (ms)
UNITER[7]	73.8	74.0	-
UNIMO[27]	75.0	75.3	-
ALBEF[26]	75.8	76.0	-
METER[11]	77.7	77.6	-
VLMo[47]	79.9	80.0	-
SimVLM[48]	80.0	80.3	-
Florence[52]	80.2	80.4	-
OFA$_{huge}$ Allcan[46]	**82.0**	**82.0**	-
OFA$_{base}$ Beam Search*[46]	77.48	77.58	645.1
OFA$_{base}$ Allcan[46]	78.0	78.1	15415.3
NARVL-KD$_{base}$ (ours)	75.59	75.75	**50.76**
NARVL-KD$_{huge}$ (ours)	79.59	79.39	81.53

MSCOCO Image Captioning results. Comparisons in Table 9 demonstrate the superiority of the Q-CTC loss function in the context of NARVL, where we observe a huge improvement on all metrics with Q-CTC loss. On top of the Q-CTC version of model, we utilize knowledge distillation from AR model and obtained another set of massive performance boost, as shown in Tab 9. Comparison results of our NAR model to the AR model are shown in Tab 1, and we see huge speed advantage of the NAR model and performance advantage of the AR model. More comparisons to previous methods are shown in Table 5, where all models shown are trained without CIDEr reinforcement learning optimization, and NARVL shows competitive results.

5. Ablation experiments

Sequential learnable queries in decoder We proposed the learnable query token as the decoder input, which is shown in Figure 3. This design differs from the existing non-autoregressive Transformer for sequence generation, which uses the outputs of the encoder as the inputs for the decoder [16]. We compare these two designs on RefCOCO dataset and MSCOCO Image Captioning dataset, and the results are shown in Table 6 and Table 7, respectively. We observe that our learnable query token design has both accuracy and speed advantage over the encoder output design.

Table 5. Results on MSCOCO Image Captioning Karpathy test split. All the models are trained without CIDER reinforcement learning.

Method	BLEU@4	METEOR	CIDEr	SPICE	Speed (ms)
VL-T5[8]	34.5	28.7	116.5	21.9	-
OSCAR[28]	37.4	30.7	127.8	23.5	-
UNICORN[50]	35.8	28.4	119.1	21.5	-
VinVL[53]	38.5	30.4	130.8	23.4	-
LEMON[21]	41.5	30.8	139.1	24.1	-
SimVLM[48]	40.6	**33.7**	143.3	**25.4**	-
OFA[46]	**43.9**	31.8	**145.3**	24.8	545.1
NARVL-KD$_{base}$ (ours)	36.4	28.7	123.1	22.47	**51.2**
NARVL-KD$_{huge}$ (ours)	40.1	30.6	136.7	24.3	130.6

Table 6. Comparisons of two types of decoder input design on RefCOCO dataset. The learnble query token design is faster and more accurate.

Decoder Input	Val	TestA	TestB	Speed (ms)
Output of Encoder	87.78	89.89	83.24	82.8
Learnable Query Tokens(ours)	**88.78**	**90.63**	**84.67**	**54.9**

Table 7. Comparisons of two types of decoder input design on MSCOCO Image Captioning dataset. The learnble query token design has equal or better performance with faster inference speed.

Decoder Input	BLEU@4	METEOR	CIDEr	SPICE	Speed (ms)
Output of Encoder	36.4	28.6	121.8	22.0	58.7
Learnable Query Tokens(ours)	36.4	**28.7**	**123.1**	**22.5**	**51.2**

Q-CTC loss vs Cross-entropy loss We compare Q-CTC loss and standard cross-entropy loss (CE) in NARVL, and the results are in Table 8. With cross entropy loss, the first n input queries are supervised to predict the output sequence, where n is the number of tokens in the target sequence. Q-CTC loss performs significantly better than CE loss as it assigns proper penalty to the model, while CE penalizes the model severely even just one token position shift.

Table 8. Comparing Q-CTC loss and CE loss with NARVL on Captioning. The base model is used in the experiments and knowledge distillation is not used for both losses.

Loss	BLEU@4	METEOR	CIDEr	SPICE	Speed (ms)
CE	17.54	18.57	66.85	11.89	**52.05**
Q-CTC	**26.69**	**24.06**	**93.24**	**17.38**	54.03
Difference	+9.15	+5.49	+26.39	+5.49	+2.0

Knowledge distillation We study the effect of knowledge distillation on VQA datasets, and found it improves the performance by 1.48/1.35 on test-dev/test-std splits, as shown in Tab 9.

The various lengths of query tokens We study the effect on performance and speed of our method when we change the number of input queries to the decoder and decoding

Table 9. Effect of knowledge distillation on VQA and Captioning datasets. The models with "KD" are the distilled models.

Method	BLEU@4	METEOR	CIDEr	SPICE	Speed (ms)
NARVL	26.7	24.1	93.2	17.4	54.0
NARVL-KD	**36.4**	**28.7**	**123.1**	**22.5**	**51.2**
Difference	+9.7	+4.6	+29.9	+5.1	-2.8

(a) MSCOCO Image Captioning

Method	Test-dev	Test-std	Speed (ms)
NARVL	74.21	74.4	51.52
NARVL-KD	**75.69**	**75.75**	**50.76**
Difference	+1.48	+1.35	-0.76

(b) VQA

methods. Ablation experiments are done on MSCOCO Image Captioning dataset, and the results are shown in Table 10. If the input length is smaller than the target sequence, the decoder won't be able to generate complete sequence, which leads to reduced performance for number of values that are too small, while too large number of input queries makes it harder for the model to decide the input output token correspondence. Naturally larger number of queries leads to increased inference time, while increasing the number of input queries from 10 to 1000 only leads to an increase from 51.32 ms to 60.40 ms, demonstrating the amazing ability of long sequence scalability.

Table 10. Ablation experiments for the number of queries on Image Captioning dataset. All experiments are done with base size model. The models used in query number ablations only and 1 epoch training)

Decoder sequence length ablations					
Number of Queries	BLEU@4	METEOR	CIDEr	SPICE	Speed (ms)
10	29.43	24.89	96.90	18.51	51.32
20	33.18	27.10	110.35	20.48	51.72
100	32.47	26.98	108.76	20.42	51.96
500	32.36	26.79	107.34	19.94	53.82
1000	31.91	26.60	105.90	19.89	60.40

Beam search in NARVL We also experiment with greedy and beam search decoding, and observe slightly better performance from beam search. Due to the increased inference time and autoregressive nature of beam search decoding, it's not adopted in previous experiments.

Table 11. Ablation experiments of Beam Search in NARVL. Test it on Image Captioning dataset. All experiments are done with base size model.

Decoding method experiments					
Decoding Method	BLEU@4	METEOR	CIDEr	SPICE	Speed (ms)
Greedy	36.38	28.70	123.15	22.46	51.18
Beam Search 5	36.85	28.74	124.03	22.51	68.95
Beam Search 20	36.91	28.71	124.15	22.51	70.22
Beam Search 100	37.00	28.71	124.25	22.51	122.91

The model scale We investigate the performance and inference speed of AR and NAR models of different model sizes, and the comparison is illustrated in Figure 5. We observe that the inference latency increases when the model becomes large on both AR and NAR model, but AR model has larger latency change compared to AR models. It is worthy to note that NAR_huge has the similar the similar performance as AR_huge, meanwhile its inference speed is closed to the AR_base model.

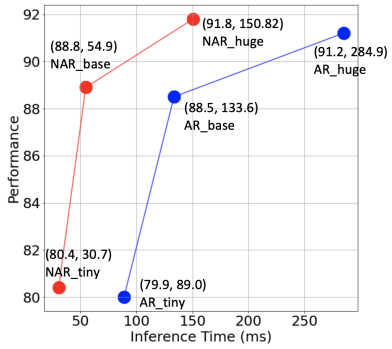


Figure 5. The comparison of accuracy and inference speed with NAR (non-autoregressive) and AR (autoregressive) models for varying model sizes: Tiny, Base and Huge. Speed is measured in wall clock time. NAR models significantly outperform their AR counterparts under the same inference time budget on the RefCOCO validation set.

6. Limitations

Optimization of the NAR models is difficult, and the best performance we get on Image Captioning and VQA datasets rely on distillation from an autoregressive model, which is inconvenient in practice as one needs to train two models. The most common failure case comes from the conditional independence nature between the decoder tokens, and the model might end up merging multiple possible valid sequences into a incorrect output sequence.

7. Conclusion

We have introduced NARVL, an All-in-One non-autoregressive model for various visual-language tasks, and the proposed NARVL repurposes the autoregressive decoder into a more flexible encoding that can be tailored to different hypothesis spaces using a layer of learnable query tokens (LQTs). These tokens are, in turn, used to define Query-CTC loss, akin to losses used in language modeling, but augmented to incorporate LQTs. This innovation is key to enabling the flexibility of letting the task drive the design of the hypothesis space, which we deem critical for heterogeneous input and output spaces as expected in the visual domain but absent in language.

References

[1] Nader Akoury, Kalpesh Krishna, and Mohit Iyyer. Syntactically supervised transformers for faster neural machine translation. *ACL*, 2019. 2

[2] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. Vqa: Visual question answering. In *Proceedings of the IEEE international conference on computer vision*, pages 2425–2433, 2015. 2

[3] Yu Bao, Hao Zhou, Shujian Huang, Dongqi Wang, Lihua Qian, Xinyu Dai, Jiajun Chen, and Lei Li. latent-glat: Glancing at latent variables for parallel text generation. *CoRR*, 2022. 2

[4] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European conference on computer vision*, 2020. 2, 3

[5] Ting Chen, Saurabh Saxena, Lala Li, David J Fleet, and Geoffrey Hinton. Pix2seq: A language modeling framework for object detection. *International Conference on Learning Representations*, 2022. 2

[6] Ting Chen, Saurabh Saxena, Lala Li, Tsung-Yi Lin, David J Fleet, and Geoffrey Hinton. A unified sequence interface for vision tasks. *arXiv preprint arXiv:2206.07669*, 2022. 2

[7] Yen-Chun Chen, Linjie Li, Licheng Yu, Ahmed El Kholy, Faisal Ahmed, Zhe Gan, Yu Cheng, and Jingjing Liu. Uniter: Universal image-text representation learning. In *European conference on computer vision*, 2020. 6, 7

[8] Jaemin Cho, Jie Lei, Hao Tan, and Mohit Bansal. Unifying vision-and-language tasks via text generation. In *International Conference on Machine Learning*, pages 1931–1942. PMLR, 2021. 7

[9] Qi Dong, Zhuowen Tu, Haofu Liao, Yuting Zhang, Vijay Mahadevan, and Stefano Soatto. Visual relationship detection using part-and-sum transformers with composite queries. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3550–3559, 2021. 2

[10] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 2

[11] Zi-Yi Dou, Yichong Xu, Zhe Gan, Jianfeng Wang, Shuohang Wang, Lijuan Wang, Chenguang Zhu, Pengchuan Zhang, Lu Yuan, Nanyun Peng, et al. An empirical study of training end-to-end vision-and-language transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022. 6, 7

[12] Luciano Floridi and Massimo Chiriatti. Gpt-3: Its nature, scope, limits, and consequences. *Minds and Machines*, 2020. 1

[13] Zhe Gan, Yen-Chun Chen, Linjie Li, Chen Zhu, Yu Cheng, and Jingjing Liu. Large-scale adversarial training for vision-and-language representation learning. *Advances in Neural Information Processing Systems*, 2020. 6

[14] Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. Mask-predict: Parallel decoding of conditional masked language models. *EMNLP-IJCNLP*, 2019. 2

[15] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pages 369–376, 2006. 2, 3, 1

[16] Jiatao Gu and Xiang Kong. Fully non-autoregressive neural machine translation: Tricks of the trade. *ACL*, 2021. 2, 3, 7

[17] Jiatao Gu, James Bradbury, Caiming Xiong, Victor OK Li, and Richard Socher. Non-autoregressive neural machine translation. *ICLR*, 2018. 2

[18] Jiatao Gu, Changhan Wang, and Junbo Zhao. Levenshtein transformer. *Advances in Neural Information Processing Systems*, 2019. 2, 3

[19] Awni Hannun. Sequence modeling with ctc. *Distill*, 2017. https://distill.pub/2017/ctc. 1

[20] Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. Distilling the knowledge in a neural network. *arXiv*, 2015. 4

[21] Xiaowei Hu, Zhe Gan, Jianfeng Wang, Zhengyuan Yang, Zicheng Liu, Yumao Lu, and Lijuan Wang. Scaling up vision-language pre-training for image captioning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022. 7

[22] Lukasz Kaiser, Samy Bengio, Aurko Roy, Ashish Vaswani, Niki Parmar, Jakob Uszkoreit, and Noam Shazeer. Fast decoding in sequence models using discrete latent variables. In *International Conference on Machine Learning*. PMLR, 2018. 2

[23] Aishwarya Kamath, Mannat Singh, Yann LeCun, Gabriel Synnaeve, Ishan Misra, and Nicolas Carion. Mdetr-modulated detection for end-to-end multi-modal understanding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021. 1, 3, 6

[24] Justin Lazarow, Kwonjoon Lee, Kunyu Shi, and Zhuowen Tu. Learning instance occlusion for panoptic segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10720–10729, 2020. 2

[25] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*, 2019. 1

[26] Junnan Li, Ramprasaath Selvaraju, Akhilesh Gotmare, Shafiq Joty, Caiming Xiong, and Steven Chu Hong Hoi. Align before fuse: Vision and language representation learning with momentum distillation. *Advances in neural information processing systems*, 2021. 6, 7

[27] Wei Li, Can Gao, Guocheng Niu, Xinyan Xiao, Hao Liu, Jiachen Liu, Hua Wu, and Haifeng Wang. Unimo: Towards unified-modal understanding and generation via cross-modal contrastive learning. *ACL/IJCNLP*, 2021. 6, 7

[28] Xiujun Li, Xi Yin, Chunyuan Li, Pengchuan Zhang, Xiaowei Hu, Lei Zhang, Lijuan Wang, Houdong Hu, Li Dong, Furu Wei, et al. Oscar: Object-semantics aligned pre-training for vision-language tasks. In *European Conference on Computer Vision*, 2020. 7

[29] Jindřich Libovickỳ and Jindřich Helcl. End-to-end non-autoregressive neural machine translation with connectionist temporal classification. *arXiv preprint arXiv:1811.04719*, 2018. 1

[30] Jinglin Liu, Yi Ren, Xu Tan, Chen Zhang, Tao Qin, Zhou Zhao, and Tie-Yan Liu. Task-level curriculum learning for non-autoregressive neural machine translation. *IJCAI*, 2021. 2

[31] Jiasen Lu, Christopher Clark, Rowan Zellers, Roozbeh Mottaghi, and Aniruddha Kembhavi. Unified-io: A unified model for vision, language, and multi-modal tasks. *arXiv preprint arXiv:2206.08916*, 2022. 1

[32] Junhua Mao, Wei Xu, Yi Yang, Jiang Wang, Zhiheng Huang, and Alan Yuille. Deep captioning with multimodal recurrent neural networks (m-rnn). *arXiv preprint arXiv:1412.6632*, 2014. 2

[33] Junhua Mao, Jonathan Huang, Alexander Toshev, Oana Camburu, Alan L Yuille, and Kevin Murphy. Generation and comprehension of unambiguous object descriptions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 11–20, 2016. 5

[34] Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*, 2019. 4

[35] Giovanni Paolini, Ben Athiwaratkun, Jason Krone, Jie Ma, Alessandro Achille, Rishita Anubhai, Cicero Nogueira dos Santos, Bing Xiang, and Stefano Soatto. Structured prediction as translation between augmented natural languages. *arXiv preprint arXiv:2101.05779*, 2021. 1

[36] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 2020. 1

[37] Qiu Ran, Yankai Lin, Peng Li, and Jie Zhou. Guiding non-autoregressive neural machine translation decoding with reordering information. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021. 2

[38] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. *arXiv*, 2015. 3

[39] Raphael Shu, Jason Lee, Hideki Nakayama, and Kyunghyun Cho. Latent-variable non-autoregressive neural machine translation with deterministic inference using a delta posterior. In *Proceedings of the aaai conference on artificial intelligence*, 2020. 2

[40] Mitchell Stern, Noam Shazeer, and Jakob Uszkoreit. Blockwise parallel decoding for deep autoregressive models. *Advances in Neural Information Processing Systems*, 2018. 2

[41] Mitchell Stern, William Chan, Jamie Kiros, and Jakob Uszkoreit. Insertion transformer: Flexible sequence generation via insertion operations. In *International Conference on Machine Learning*. PMLR, 2019.

[42] Zhiqing Sun, Zhuohan Li, Haoqing Wang, Di He, Zi Lin, and Zhihong Deng. Fast structured decoding for sequence models. *Advances in Neural Information Processing Systems*, 2019. 2

[43] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27, 2014. 2

[44] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 2017. 1, 2

[45] Chunqi Wang, Ji Zhang, and Haiqing Chen. Semi-autoregressive neural machine translation. *EMNLP*, 2018. 2

[46] Peng Wang, An Yang, Rui Men, Junyang Lin, Shuai Bai, Zhikang Li, Jianxin Ma, Chang Zhou, Jingren Zhou, and Hongxia Yang. Ofa: Unifying architectures, tasks, and modalities through a simple sequence-to-sequence learning framework. In *International Conference on Machine Learning*, pages 23318–23340. PMLR, 2022. 1, 2, 3, 4, 5, 6, 7

[47] Wenhui Wang, Hangbo Bao, Li Dong, and Furu Wei. Vlmo: Unified vision-language pre-training with mixture-of-modality-experts. *arXiv*, 2021. 6, 7

[48] Zirui Wang, Jiahui Yu, Adams Wei Yu, Zihang Dai, Yulia Tsvetkov, and Yuan Cao. Simvlm: Simple visual language model pretraining with weak supervision. *arXiv*, 2021. 6, 7

[49] Yisheng Xiao, Lijun Wu, Junliang Guo, Juntao Li, Min Zhang, Tao Qin, and Tie-yan Liu. A survey on non-autoregressive generation for neural machine translation and beyond. *arXiv*, 2022. 2

[50] Zhengyuan Yang, Zhe Gan, Jianfeng Wang, Xiaowei Hu, Faisal Ahmed, Zicheng Liu, Yumao Lu, and Lijuan Wang. Crossing the format boundary of text and boxes: Towards unified vision-language modeling. *arXiv*, 2021. 1, 6, 7

[51] Licheng Yu, Patrick Poirson, Shan Yang, Alexander C Berg, and Tamara L Berg. Modeling context in referring expressions. In *European Conference on Computer Vision*, pages 69–85. Springer, 2016. 2, 5

[52] Lu Yuan, Dongdong Chen, Yi-Ling Chen, Noel Codella, Xiyang Dai, Jianfeng Gao, Houdong Hu, Xuedong Huang, Boxin Li, Chunyuan Li, et al. Florence: A new foundation model for computer vision. *arXiv*, 2021. 6, 7

[53] Pengchuan Zhang, Xiujun Li, Xiaowei Hu, Jianwei Yang, Lei Zhang, Lijuan Wang, Yejin Choi, and Jianfeng Gao. Vinvl: Revisiting visual representations in vision-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021. 6, 7

[54] Zhaoyang Zhang, Yantao Shen, Kunyu Shi, Zhaowei Cai, Jun Fang, Siqi Deng, Hao Yang, Davide Modolo, Zhuowen Tu, and Stefano Soatto. Musketeer (all for one, and one for all): A generalist vision-language model with task explanation prompts, 2023. 2

[55] Chunting Zhou, Graham Neubig, and Jiatao Gu. Understanding knowledge distillation in non-autoregressive machine translation. *ICLR*, 2019. 2

Non-autoregressive Sequence-to-Sequence Vision-Language Models

Supplementary Material

A. Model Designs Analysis

We provide some additional discussions for objective function design and simplified training targets.

A.1. Objective Functions Comparison

Cross-Entropy Loss Cross-entropy loss can be used as the objective function for non-autoregressive sequence generation, and it classifies each token independently. Token level cross entropy loss introduces strict prediction and GT alignment, and a slight misalignment could intrigue severe penalty to the sequences close to the correct predictions. This strict alignment confuses the model learning, particularly, on target sequences of high diversity, for example, there exist multiple valid captions for an image. Lack of inter-token dependency modeling causes a large solution space (L^D, where L is the target sequence length and D is the vocabulary size), and makes the optimization process hard.

CTC Loss Connectionist Temporal Classification Loss (CTC) was originally proposed in [15] for Recurrent neural networks (RNNs) to classify speech signal, where sequence alignment is difficult to form between the unsegmented waveform and the sequence of phonemes, constructed by spikes and blanks separating them, and CTC is also adopted in machine translation [29] to handle the length discrepancy between source sentence and the translated sentence. The following URL [19] points to a website that provides a nice illustration about the motivation and formulation of the CTC loss, applied to sequence modeling in speech recognition. Unlike in the standard cross-entropy loss where **each token** corresponds to one (or multiple) ground truth results and the CE loss is trying to encourage the individual tokens to make the correct prediction by matching with the ground-truth output. So the cross-entropy loss is an element-wise loss for each token, if we know it's ground-truth output. In the Connectionist Temporal Classification Loss (CTC) case, we **no longer** have a known **token-wise ground-truth** output. Instead, we only know **sequence-level ground-truth** output. A solution that is fairly close to the ground-truth in the sequence level might just have one position shifted to the right for each token after a certain location; if a strict CE loss would have been used, its loss can be large. The CTC loss deals with the alignment issue by marginalizing predictions that are only slightly off to the ground-truth by a shift to assign a faith loss. The CTC loss is:

$$L_{\text{CTC}}(\theta) = -\sum_{k} \sum_{z_i \sim \tilde{p}(z|x_k)} \log \frac{f_{y_i}(z_i(x_k))}{\sum_j e^{f_{y_j}(z_i(x_k))}} \quad (2)$$

, where θ denotes the learnable parameters of the model, $\tilde{p}(z|x_k)$ denotes the collapsed valid sequences, $f_{y_i}(\cdot)$ computes the logits of x_i on target token class y_i, k is the number of all training samples. $z_i(x_k)$ denotes the "path" from the input tokens to the output tokens, meaning the value of z_i as a (deterministic) function of the encoding x_k.

When implementing the CTC loss, there are certain techniques that have been adopted. For example, since there can be stretches for the same character output or multiple consecutive same character (or sub-words if the basic output element is subword), a blank token is introduced to differentiate between the two cases. A post-process is done to remove blank tokens to produce the sequence output. Some illustration can be found at [19].

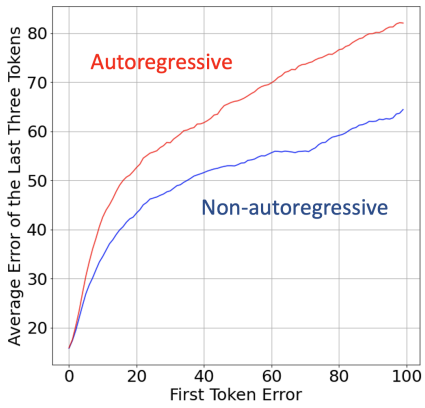


Figure 6. A Plot to show the effect of the first token error on the remaining sequence tokens. We use RefCOCO dataset here and the sequence is x1, y1, x2, y2 representing the target object bbox. The x-axis "First Token Error" is the difference between the predicted x1 coordinate and the ground truth x1 coordinate. The y-axis "Average Error of the Last Three Tokens" is calculated similarly on y1, x2, y2 for samples with higher corresponding first token error value and then we take the average. The errors made on the first token tend to have larger effect on the remaining sequence in AR model, and the gap to NAR model grows as the severity of the first token error increases.

A.2. Simplified Training Targets with Knowledge Distillation

In vision-language tasks, the target sequences often have high freedom, such as multiple captions for a single image, which confuses the model training (see Table 12). This issue is avoided in auto-regressive model learning, because of the teacher-forcing training schema. However, this is-

Images	Captions
	• **Simplified Caption**: A giraffe statue in a room with plants. • **GT-1**: A fake giraffe standing beside a bunch of trees. • **GT-2**: A forest like place to go and eat food. • **GT-3**: A fake giraffe that is hanging on the side of a wall. • **GT-4**: A large giraffe standing in the middle of a rainforest cafe. • **GT-5**: A giraffe bust hanging by a rain forest cafe sign.
	• **Simplified Caption**: Two men sitting at a table eating pizza. • **GT-1**: Two guys in a bar eating pizza and drinking beer. • **GT-2**: Two men are sitting side by side as they are eating and smiling, they both are cutting their food with a knife. • **GT-3**: Two men sitting at a table eating pizza. • **GT-4**: Two young men sitting next to each other sharing a meal. • **GT-5**: Two man sit at a table in a restaurant.

Table 12. Simplified Captions vs GT Captions. We run the autoregressive model on the training dataset to get simplified captions. Each image is annotated by 5 annotators thus we have GT-1 to GT-5 captions. We see that the diversity of original GT output sequences is higher than simplified target sequences.

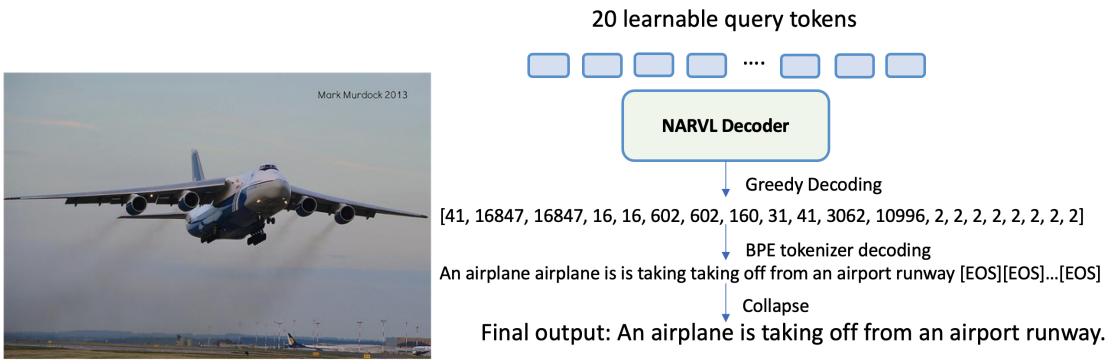


Figure 7. An example to illustrate the whole decoder inference process from learnable query tokens to the final output sequence.

sue becomes more serious in NARVL learning. To solve this problem, we use knowledge distillation to reduce the freedom of target sequences. Specifically, we propose to set the sequences predicted by an auto-regressive model as the targets, which are more deterministic compared to the sequences generated by human. Table 12 shows the comparisons of ground truth captions vs the captions generated by an auto-regressive model.

We also show qualitative result comparisons between the model trained with ground truth and simplified sequences in Table 13. It can be seen that the model trained with ground truth captions struggles to figure out a mode among all possible valid sequences. In the second example in Table 13, repetitive description of the same object "airplane jet plane" might be the consequence of the confusion of training the model with high-freedom target sequences. Such problem is greatly addressed in the model trained with knowledge distillation, and the generated captions are more fluent and cohesive.

B. NARVL Sequence Decoding Analysis

In this section, we illustrate sequence generation process in NARVL and study the features of non-autogressive sequences compared to auto-regressive sequences.

B.1. Sequence generation process

We illustrate the sequence generation procedure in NARVL. Figure 7 contains an example of sequence generation in Im-

Images	Captions
	• **CE**: A baseball of players a. • **Q-CTC**: A baseball of players a on a baseball. • **Q-CTC+KD**: A group of baseball players on a baseball field. • **GT**: Four baseball players pitching balls in the middle of a baseball field.
	• **CE**: A large jet is the. • **Q-CTC**: A large airplane jet plane is taking off from the runway. • **Q-CTC+KD**: An airplane is taking off from an airport runway. • **GT**: A large passenger jet taking off from an airport.
	• **CE**: A stop sign a. • **Q-CTC**: A stop sign a field a cloudy. • **Q-CTC + KD**: A stop sign in the middle of a field. • **GT**: The sky is cloudy over a stop sign.
	• **CE**: A man and a pizza. • **Q-CTC**: A man and a table with a pizza. • **Q-CTC+KD**: A man and a little girl sitting at a table with a pizza. • **GT**: A man and a kids at a table with pizza.
	• **CE**: A living room with a and. • **Q-CTC**: A living room with a and a room. • **Q-CTC+KD**: A living room with a couch and a table. • **GT**: A living room has a couch, a table, and a small television.

Table 13. Qualitative Comparisons of three models on MSCOCO Image Captioning dataset. CE: The first model is trained with cross entropy loss. Q-CTC: The second model is trained with Q-CTC loss. Q-CTC + KD: The third model is trained with Q-CTC loss with knowledge distillation.)

age Captioning. The output sequence from NARVL is fixed length and contains repetitive tokens. According the valid path selection rule in Q-CTC loss, we remove the repetitive tokens and output the final sequence.

B.2. Error Propagation

The training of autoregressive model utilizes ground truth tokens as previous tokens, which however are not available at inference time. During inference, the model has to generate the sequence conditioning on previously predicted tokens. The quality of the next predicted tokens depends on the correctness of previous predicted tokens, and the errors might accumulate and propagate via iterations. We study how the previous token errors affect later sequence quality in both AR and NAR models on RefCOCO dataset. We use the value difference between the GT token coordinates and the predicted token coordinates as a measure of the incorrectness (the smaller value difference, the higher correctness), and as shown in Figure 6, the first token error in the autoregressive model has higher negative impact on the rest of the sequence.