A Unified Sequence Interface for Vision Tasks

Anonymous Author(s)
Affiliation
Address
email

Abstract

While language tasks are naturally expressed in a single, unified, modeling framework, i.e., generating sequences of tokens, this has not been the case in computer vision. As a result, there is a proliferation of distinct architectures and loss functions for different vision tasks. In this work we show that a diverse set of "core" computer vision tasks can also be unified if formulated in terms of a shared pixel-to-sequence interface. We focus on four tasks, namely, object detection, instance segmentation, keypoint detection, and image captioning, all with diverse types of outputs, e.g., bounding boxes or dense masks. Despite that, by formulating the output of each task as a sequence of discrete tokens with a unified interface, we show that one can train a neural network with a single model architecture and loss function on all these tasks, with no task-specific customization. To solve a specific task, we use a short prompt as task description, and the sequence output adapts to the prompt so it can produce task-specific output. We show that such a model can achieve competitive performance compared to well-established task-specific models.

1 Introduction

Training a single neural network model capable of performing myriad tasks is a major step towards artificial general intelligence. In recent years, with the rise of language modeling [30, 31, 2] using Transformers [36], many different language and related tasks are unified under a single modeling framework, where a language model is trained to predict the solution (in text tokens) given a prompt of a task description (also in text tokens). This is only possible because these tasks (both task description and solution) can be expressed in the same, rich language interface.

This can be naturally extended to some vision tasks such as image captioning or visual question answering where the solution is given in natural language, but the majority of "core" computer vision tasks have diverse outputs that are not readily expressed in terms of natural language. The object detection task produces a set of bounding boxes and their corresponding class labels, often associated with scores for ranking. The output for instance segmentation is a set of segmentation masks corresponding to image regions. The output of keypoint detection is a set of keypoints in an image. As such, existing methods [12, 32, 14, 26, 4, 14] have developed specialized architectures and sophisticated loss functions for each of these complex tasks.

An ambitious goal, in the pursuit of artificial general intelligence, is a simple interface that allows one to express seemingly disparate vision tasks in a unified framework. This would simplify the design of architectures and loss functions for new tasks. It would enable greater degrees of feature/representation sharing across many different tasks, thereby avoiding the need for a sophisticated output head for each task. It would also facilitate adapting of existing models to new tasks, and potentially unlock new capabilities with zero or few demonstrations.

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

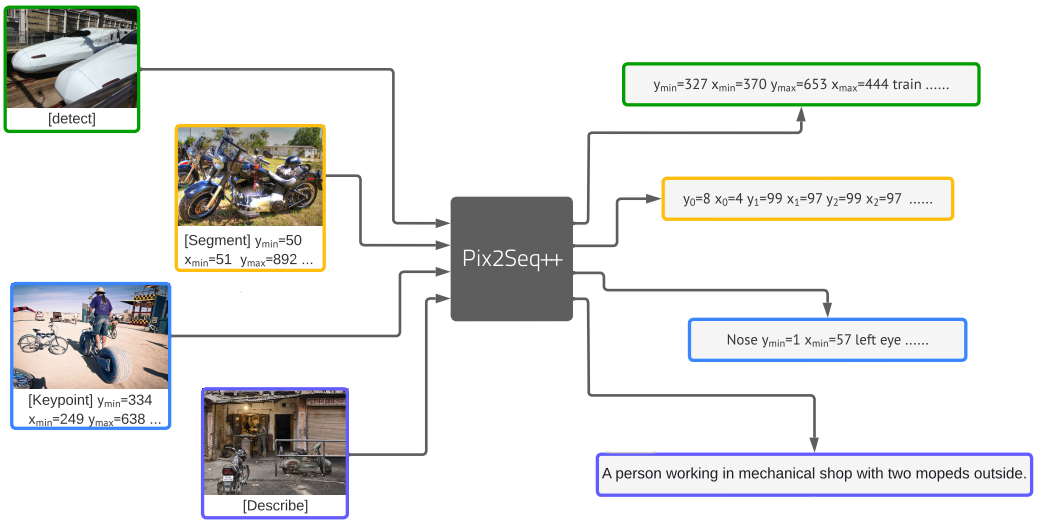


Figure 1: An illustration of the proposed framework. An image and a sequence of task prompt is given, the model produce a sequence of discrete tokens corresponding to the desired output.

To this end, we propose an approach to unify four seemingly different vision tasks in a single pixel-to-sequence interface. In effect, this is an extension of pix2seq [6] for object detection to a broader set of tasks. As a proof of concept, we focus on four core vision tasks, namely, object detection, instance segmentation, human keypoint detection, and image captioning. We first show how to unify these tasks into a single shared interface, and then train a neural network with a shared architecture and objective function. To solve a specific task, instead of using a specific head for that task, we use a prompt to specify the task, and the sequence output adapts to the prompt so it can produce task-specific output given the task description. This makes multi-task learning more efficient and scalable. We conduct experiments on the challenging COCO dataset, and show that it can simultaneously solve all four tasks well, without specialized architectures or loss functions.

2 Approach

In our approach, we cast computer vision tasks as one of translating pixel inputs (along with some descriptions of the task) into sequences of discrete tokens (see Figure 1). As a proof of concept, we focus on four core vision tasks: object detection, instance segmentation, keypoint detection, and image captioning; but we believe it is relatively straightforward to include many more tasks.

2.1 A unified interface with tokenization

The vision tasks we consider are diverse, and traditionally have been formulated quite differently. Object detection requires the model to produce bounding boxes for all objects without duplication. Instance segmentation requires the model to produce a dense pixel-wise mask for each identified object instance. Human keypoint detection requires the model to generate points corresponding to specific positions of landmarks on body parts for person instances (e.g., head, eyes). Image captioning requires the model to produce a sequence of words corresponding to a natural language description of the image. Given the significant differences in the form of the outputs, customized models with specialized architectures and loss functions are designed for each.

To solve these tasks using a single model, we advocate the transformation/tokenization of task inputs and outputs into a unified interface. In this work, we propose a sequence interface for the purpose, where both task descriptions and outputs are expressed as sequences of discrete tokens:

- For object detection, we follow [6] and convert bounding boxes and object descriptions into a sequence of discrete tokens by quantizing the continuous image coordinates.

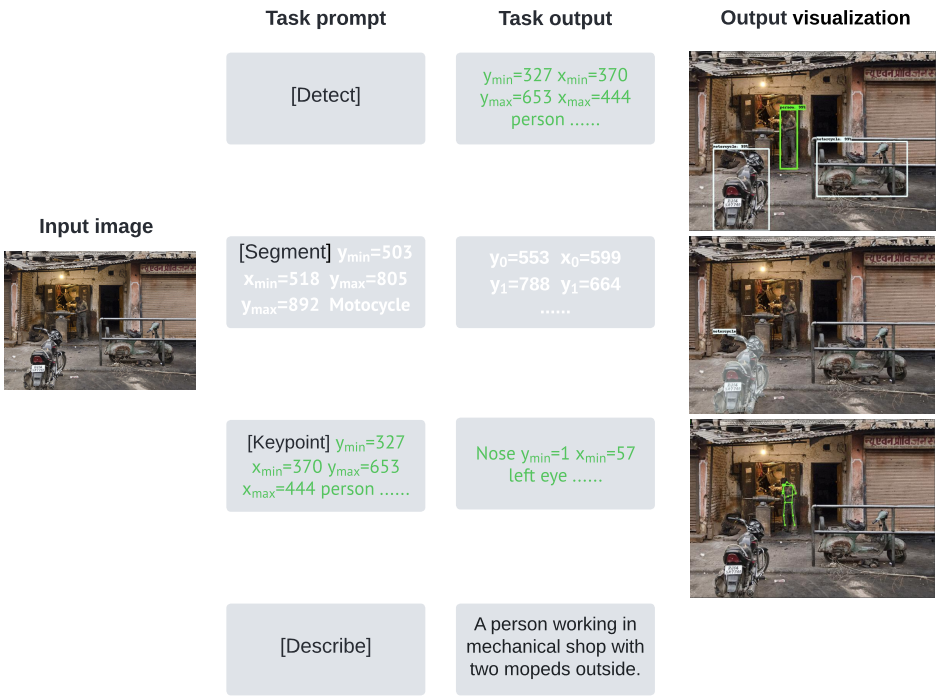


Figure 2: An illustration of sequence interface for four tasks. The model takes input image, task prompt tokens and produce task output tokens, which can be decoded/detokenized into required task output for visualization.

- For instance segmentation, instead of per-pixel masks, we predict the polygon of the masks as a sequence of image coordinates conditioned on a given object instance. Again, we quantize the coordinates into discrete tokens.
- For keypoint prediction, we predict keypoints as sequence of quantized image coordinates conditioned on a given person instance.
- For captioning, we directly predict text tokens.

It is worth noting that all four tasks share a single vocabulary. The specific prompts and output sequences are illustrated in Figure 2.

2.2 Unified architecture and objective function

We need a flexible and expressive architecture that can deal with image input and sequence output with complex semantics. Thus we follow [6] and use an encoder-decoder architecture, with an image encoder and sequence decoder. The image encoder perceives pixels and maps them into hidden representations, which can be a ConvNet [21, 20, 13], Transformer [36, 10], or their combination [4]. The Transformers-based sequence decoder, widely used in modern language modeling [36, 29, 31], generates one token at a time, conditioned on the preceding tokens and the encoded image representation. This removes the complexity and customization in architectures of modern neural networks for these vision tasks (such as per-task specific heads or necks).

Unlike [6] where the decoder only produces output tokens directly, here it also conditions on a task prompt so that the model can produce outputs adapted to the task of interest. During training, we concatenate both prompt and desired output into a single sequence, but leverage a token weighting scheme to ensure that the decoder is only trained to predict the desired output but not the prompt tokens. During inference, the prompt is given and fixed, so the decoder only needs to produce the rest of the sequence. Similar to [6], the training objective is to maximize the likelihood of tokens

conditioned on an image and preceding tokens, i.e.,

$$\text{maximize} \sum_{j=1}^{L} \boldsymbol{w}_j \log P(\boldsymbol{y}_j | \boldsymbol{x}, \boldsymbol{y}_{1:j-1}) \ , \tag{1}$$

where $\boldsymbol{x}$ is the input image, $\boldsymbol{y}$ is a length-L sequence associated with $\boldsymbol{x}$. As mentioned, the initial part of the sequence $\boldsymbol{y}$ is a prompt, for which we set the weight $\boldsymbol{w}_j$ to zero so it is not included in the loss.

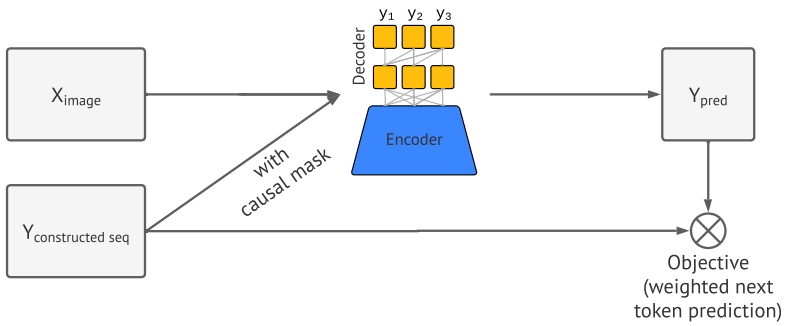


Figure 3: An illustration of our architecture and training objective. Note that $Y_{\text{constructed seq}}$ encapsulates both task prompt tokens and task output tokens. Token weights are set to zero if the target token is within the prompt so the model is only trained to predict desired output tokens.

2.3 Training

Each task has its own paired image-sequence training data. There are two ways one can combine tasks and perform the joint training.

Data mixing. We can create a dataset with mixed image-sequence pairs drawn from different tasks, balanced to account for different dataset sizes and task difficulties. This construction is extremely simple conceptually, but image augmentations can be difficult to incorporate as they may also require a change to their associated sequences in non-trivial ways.

Batch mixing. For each batch, we can sample images with annotations for a single task, perform image augmentations appropriate for this task, and convert the augmented data into image-sequence pairs. The model computes the loss and gradient for each task separately, and we can combine gradients from task-specific batches with an appropriate weighting.

Algorithm 1 Training based on data mixing

1: Tokenize annotation into sequences of tokens,
2: Mixing images and sequences from all tasks,
3: Sample a batch, compute the loss, and update the model.

Algorithm 2 Training based on batch mixing

1: Sample batches of data from all tasks,
2: Tokenize annotation into sequences of tokens,
3: Compute the loss for each task, aggregate their gradients, and update the model.

Algorithm 1 and 2 give summaries of both training strategies. We use batch mixing strategy in this work as it simplifies handling of image augmentations, but we are hopeful that data mixing can be employed in the future to further simplify the pipeline and allow more tasks to be added straightforwardly.

Both data mixing and batch mixing require that we specify the portion or weighting for each task. This is an empirical matter, and we use a greedy strategy by adding one task at a time. Every time when we add a task, we adjust the weighting of the new task while keeping the relative weighting among the existing task fixed. We fix the sum of the weights across all tasks to be one.

2.4 Inference and de-tokenization

At inference time, we sample tokens from the model likelihood, given a prompt at the start of the sequence, i.e., $P(\boldsymbol{y}_j|\boldsymbol{x}, \boldsymbol{y}_{1:j-1})$. We currently use nucleus sampling [15] but other techniques such as beam search could also be used. Once the tokens are generated, they can be decoded for each task. In the same way that different tasks require specific tokenization schemes to generate token sequences, the decoding (de-tokenization) process is also specific to each task. A more detailed description of inference decoding for each task is given below.

- For bounding boxes, following [6], we split predicted sequences into tuples of 5 tokens to get coordinate tokens and a class token, and dequantize coordinate tokens to get the bounding boxes.
- For instance segmentation, we dequantize the coordinate tokens corresponding to each polygon, and then convert them into dense masks. The model is not trained with any geometry-specific regularizers per se, and as such the output polygonal masks can be somewhat noisy. To reduce the noise we find it helpful to sample multiple sequences and then average the masks, followed by a simple threshold to obtain a single binary mask.
- For keypoint detection, we directly dequantize the image coordinate tokens of the keypoints.
- For captioning, we directly map the predicted discrete tokens into text.

3 Experiments

3.1 Settings

We evaluate the proposed method on the widely used MS-COCO 2017 dataset [24], containing 118k training images and 5k validation images, spanning the four tasks we consider. An image in the dataset typically has annotations for object bounding boxes, segmentation masks for object instances, keypoint for person instances, and a few captions. Following [6], we use a Vision Transformer (base) encoder [10, 36], and a Transformer autoregressive decoder [36]. This model has a total of 132M parameters. To initialize the model, we use a pretrained checkpoint trained on the object detection task with the Objects365 dataset [34]; this is useful as COCO is relatively small and our model has less task-specific prior. It is also worth noting that our initialization does not include image-text pretraining on a large-scale dataset, but we expect that would be beneficial, especially for image captioning performance.

For training on COCO, we use a batch size of 128 images, a learning rate of 1e-4, and we train the model for 100 epochs. We use a single vocabulary of 35K, with 32K text tokens, 1K coordinate quantization bins, and a few other class labels. We use a maximum sequence length of 512. It is well-known that larger image size helps with the performance even with the same model size. We use 640$\times$640 for our main results due to some short-term compute constraint, but also present the 1024$\times$1024 in the appendix. This is smaller than the image size used in some some baselines (such as 1333$\times$1333 in DETR). (See the Appendix for training details.)

We compare with a few well-known task-specific baselines. For object detection we compare with a strong 2-stage detector, Faster R-CNN [32], and a more recent Transformer-based detector, DETR [4]. Both Faster R-CNN and DETR use task-specific priors in their design, such as non-maximum suppression in Faster R-CNN and bipartite graph matching with generalized intersection-over-union in DETR. Due to their customized architectures and loss functions, extending them to a wider spectrum of tasks is non-trivial and may require a new model design. Mask R-CNN [14] advocates a design to extend Faster R-CNN to incorporate segmentation masks and keypoints. While Mask R-CNN is able to perform three out of our four tasks, it still requires the same set of task-based customizations as in Faster R-CNN. We also consider an improved version of Mask R-CNN with non-local architectures [38] which incorporates an attention mechanism, similar to Transformers. The above methods cannot do image captioning, so we train a Transformer-based caption model [35, 28] which is specialized for the task. This model is similar to the proposed method trained for caption single task but it is using self-supervised pretrained visual encoder [5] with a high dropout rate.

3.2 Quantitative results

Our main results are summarized in Table 1, where we report baselines and two variants of our model: (1) single task models where the model is trained on a single task (still with the same architecture and

objective function), so each task has its own network weights; and (2) a multi-task model, where a single set of network weights is used for all four tasks. We can see that despite without task-specific priors in architecture and loss function, our model can still achieve competitive results for each individual task compared to strong specialized baselines (even with a smaller image size). When we train a single model on all tasks, our model is able to address these tasks relatively well, despite the model size is kept the same.

Table 1: COCO results for object detection, instance segmentation and keypoint detection are expressed in terms of AP. For Image Captioning we report BLEU score. Single task results for instance segmentation and keypoint detection are based on detected bounding boxes from single task detection model. - indicates the model is not able to solve the task without modifications.

	Object det.	Instance seg.	Keypoint det.	Captioning
Faster R-CNN [32]	42.0	-	-	-
Faster R-CNN+ [32]	44.0	-	-	-
DETR [4]	44.9	-	-	-
Mask R-CNN [14]	39.8	37.1	63.1	-
Mask R-CNN (non-local) [38]	45.0	40.3	66.5	-
Transformer-based captioner	-	-	-	34.3
Pix2Seq++ single task (640×640)	43.8	37.3	68.0	33.9
Pix2Seq++ multi-tasks (640×640)	44.2	36.9	65.0	34.3

Finally, we also perform several ablations to help understand the effectiveness of our model and some of the different design choices (such as task weighting, image sizes, inference and decoding hyper-parameters). More results can be found in the appendix.

3.3 Qualitative results

To demonstrate the capability and performance of our model in a more visual and intuitive way, we show the outputs from our multi-task model on selected images from the COCO validation set, for each of the four tasks, i.e., object detection, instance segmentation, keypoint detection, and image captioning. Figure 4 shows results for the object detection task. The model successfully detects objects different sizes in cluttered scenes with significant occlusion. Empirical results on instance segmentation and keypoint detection are shown in Figures 5 and 6. For both tasks, the multi-task model produces well localized and accurate predictions. We also demonstrate some captions generated by the model in Table 2. With these results, we note that our model has not been pre-trained using large-scale image-text datasets, which would be expected to significantly improve the captioning performance of the model.

4 Related work

Decoding visual concepts: Image understanding involves extracting visual concepts from images. The formats of these concepts vary according to the given task. Image captioning uses a sequence of words to describe an image [8]. Object detection, on the other hand, represents objects with labels and bounding boxes. Depending on the granularity of localization, visual concepts can be expressed as boxes, pixel segmentation, or keypoints [24]. Decoding localized visual concepts often requires tailored methods. For example, image segmentation uses per-pixel classification. Object detection uses sliding window with non-maximum suppression to detect boxes. Person keypoint detection uses part models to assemble detected parts into whole body [3].

Recently, DETR [4] is proposed as an end-to-end object detection approach based on a Transformer decoding scheme (removing complexity on bounding box proposal and non-maximum suppression). MaskFormer [9] further shows that object detection and segmentation can share the same decoding scheme. Pix2seq [6] demonstrates that boxes and labels can be treated as a sequence of discrete tokens, thereby sharing the same training and decoding interface as language models [29, 31]. In our work, we push the envelope further in the unification of language and different visual localization tasks to share the same interface, architecture and training objective.

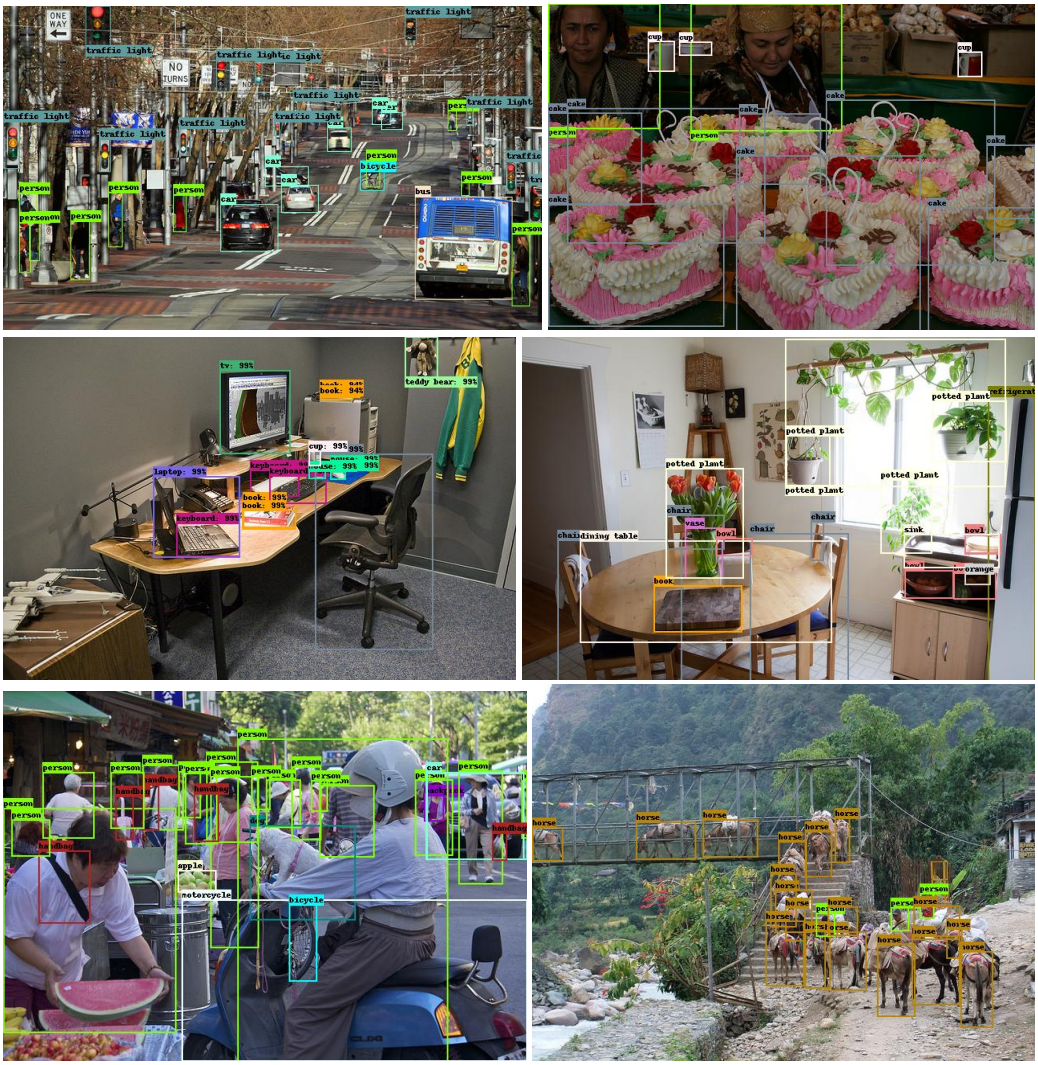


Figure 4: Object detection examples, with predicted bounding boxes on the input images.

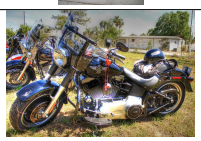
Generalist vision models: Learning a generalist model capable of performing multiple tasks is widely assumed to be a path toward general intelligence. In visual recognition, multi-task learning has shown great success by sharing a backbone model, followed by multiple independent heads [14, 18]. Models with a shared backbone can learn general features which are transferable across tasks when scaling up with training tasks, model capacity and data [19, 22, 11]. Nevertheless, often the task specific backbone models are designed carefully, particularly for tasks that require accurate localization [33, 25].

With the invention of Transformers [36], recently being adopted for image classification [10], the research community has seized on the opportunity to unify the backbone design for vision tasks [27, 7, 23]. Perceivers [17, 16] and OFA [37] demonstrate an architecture for multi-task and multi-modal across vision and language. Notably, OFA designs a unified sequence-to-sequence decoding architecture for both language and object detection tasks. Flamingo [1] and related methods also focus on an universal API that produces a natural language output for a variety of tasks given image input. This line of work shares a common motivation to our work in this paper, however they focus on higher level tasks for which natural language is inherently the desired output. In this paper we demonstrate that one can also express a wide variety of core computer vision tasks with a universal interface if one formulates the appropriate mapping from task specification to token sequences.

Figure 5: Instance segmentation examples, with predicted semantic masks overlaid on the input images.

Figure 6: Human keypoint detection examples, with predicted stick figures on the input images.

Table 2: Image captioning examples.

	A group of teddy bears sitting next to each other. Three teddy bears sitting on a blanket. A group of teddy bears sitting on a blanket with bowls of food.
	A herd of elephants standing inside of a fenced in area. A group of elephants standing in a fenced area. A herd of elephants standing behind a fence.
	A man riding a skateboard over a block of cement. A man doing a trick on a skateboard in the street. A man flying through the air while riding a skateboard.
	A row of motorcycles parked on a grass covered field. A motorcycle with a helmet on the side of it. A motorcycle parked in a grassy area with other motorcycles.

5 Conclusion

In this work, we explore a unified sequence interface for tackling a diverse set of "core" vision tasks, where both the task description (prompt) and task output are expressed as discrete sequences of tokens. This is a significant departure from conventional norms of multi-task vision models in that both architecture and loss functions are shared among the tasks. We show that such a model can achieve competitive performance compared to well-established task-specific models.

Our work is not without limitations. One potential limitation is the inference speed can be slower compared to the specialized systems as our approach is based on autoregressive modeling. However, we believe this could be addressed in future work. Due to the significant departure from conventional approaches, we believe both architectures and other training techniques can be further improved to challenge the state-of-the-art of specialized systems. We also believe our model can significantly benefit from scaling up, both in pretraining on larger (image-text) datasets and/or using larger model sizes.

References

[1] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob Menick, Sebastian Borgeaud, Andrew Brock, Aida Nematzadeh, Sahand Sharifzadeh, Mikolaj Binkowski, Ricardo Barreira, Oriol Vinyals, Andrew Zisserman, and Karen Simonyan. Flamingo: a visual language model for few-shot learning. *arXiv preprint arXiv:2204.14198*, 2022.

[2] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

[3] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7291–7299, 2017.

[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*, pages 213–229. Springer, 2020.

[5] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International Conference on Machine Learning*, pages 1597–1607. PMLR, 2020.

[6] Ting Chen, Saurabh Saxena, Lala Li, David J Fleet, and Geoffrey Hinton. Pix2seq: A language modeling framework for object detection. *arXiv preprint arXiv:2109.10852*, 2021.

[7] Wuyang Chen, Xianzhi Du, Fan Yang, Lucas Beyer, Xiaohua Zhai, Tsung-Yi Lin, Huizhong Chen, Jing Li, Xiaodan Song, Zhangyang Wang, et al. A simple single-scale vision transformer for object localization and instance segmentation. *arXiv preprint arXiv:2112.09747*, 2021.

[8] Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco captions: Data collection and evaluation server. *arXiv preprint arXiv:1504.00325*, 2015.

[9] Bowen Cheng, Alex Schwing, and Alexander Kirillov. Per-pixel classification is not all you need for semantic segmentation. *Advances in Neural Information Processing Systems*, 34, 2021.

[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. In *International Conference on Learning Representations*, 2020.

[11] Golnaz Ghiasi, Barret Zoph, Ekin D Cubuk, Quoc V Le, and Tsung-Yi Lin. Multi-task self-training for learning general representations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8856–8865, 2021.

[12] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1440–1448, 2015.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[14] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2961–2969, 2017.

[15] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*, 2019.

[16] Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, et al. Perceiver io: A general architecture for structured inputs & outputs. *arXiv preprint arXiv:2107.14795*, 2021.

[17] Andrew Jaegle, Felix Gimeno, Andy Brock, Oriol Vinyals, Andrew Zisserman, and Joao Carreira. Perceiver: General perception with iterative attention. In *International Conference on Machine Learning*, pages 4651–4664. PMLR, 2021.

[18] Iasonas Kokkinos. Ubernet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6129–6138, 2017.

[19] Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, and Neil Houlsby. Big transfer (bit): General visual representation learning. In *European conference on computer vision*, pages 491–507. Springer, 2020.

[20] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 25:1097–1105, 2012.

[21] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1 (4):541–551, 1989.

[22] Yanghao Li, Saining Xie, Xinlei Chen, Piotr Dollar, Kaiming He, and Ross Girshick. Benchmarking detection transfer learning with vision transformers. *arXiv preprint arXiv:2111.11429*, 2021.

[23] Yanghao Li, Hanzi Mao, Ross Girshick, and Kaiming He. Exploring plain vision transformer backbones for object detection. *arXiv preprint arXiv:2203.16527*, 2022.

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

[25] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and pattern recognition*, pages 2117–2125, 2017.

[26] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2980–2988, 2017.

[27] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022, 2021.

[28] Yingwei Pan, Ting Yao, Yehao Li, and Tao Mei. X-linear attention networks for image captioning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10971–10980, 2020.

[29] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018.

[30] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[31] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*, 2019.

[32] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in Neural Information Processing Systems*, 28:91–99, 2015.

[33] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[34] Shuai Shao, Zeming Li, Tianyuan Zhang, Chao Peng, Gang Yu, Xiangyu Zhang, Jing Li, and Jian Sun. Objects365: A large-scale, high-quality dataset for object detection. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 8430–8439, 2019.

[35] Piyush Sharma, Nan Ding, Sebastian Goodman, and Radu Soricut. Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2556–2565, 2018.

[36] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

[37] Peng Wang, An Yang, Rui Men, Junyang Lin, Shuai Bai, Zhikang Li, Jianxin Ma, Chang Zhou, Jingren Zhou, and Hongxia Yang. Unifying architectures, tasks, and modalities through a simple sequence-to-sequence learning framework. *arXiv preprint arXiv:2202.03052*, 2022.

[38] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7794–7803, 2018.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Conclusion section
   (c) Did you discuss any potential negative societal impacts of your work? [No] Our work at its current form does not incur any negative social impacts than those existing specialized systems do.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]

2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] We include training settings and details, and also plan to opensource our code.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] Some of the experiments are expensive to run multiple times, and the standard error from non-systemic study is small
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] It's trained on 32-128 Cloud TPUs. Depending on architectures, and tasks, generally takes 4-12 hours.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [No] It is pretty obvious from the dataset website, and it's a well known dataset.
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]