Pix2Seq:

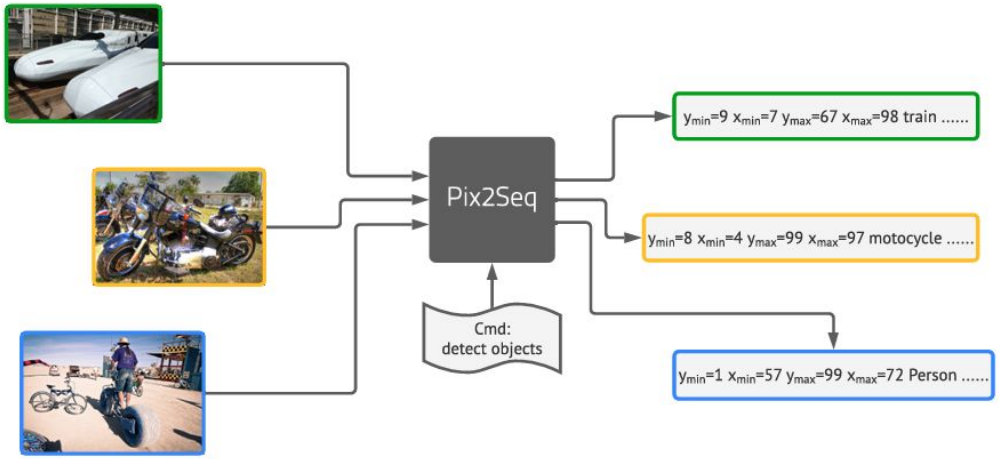
A LANGUAGE MODELING FRAMEWORK FOR OBJECT DETECTION

Ting Chen, Saurabh Saxena, Lala Li, David J. Fleet, Geoffrey Hinton

Presented by: Dohhyun, Nathan, Rodrigo

Pix2Seq: What is it?

- Treats object detection like a language task based on pixel inputs.
- “Based mainly on the intuition that if a neural network knows about where and what the objects are, we just need to teach it how to read them out.”
- Takes an image and generates a series of tokens for the objects in the image.
    - Think image captioning



Pix2Seq: How does it work?

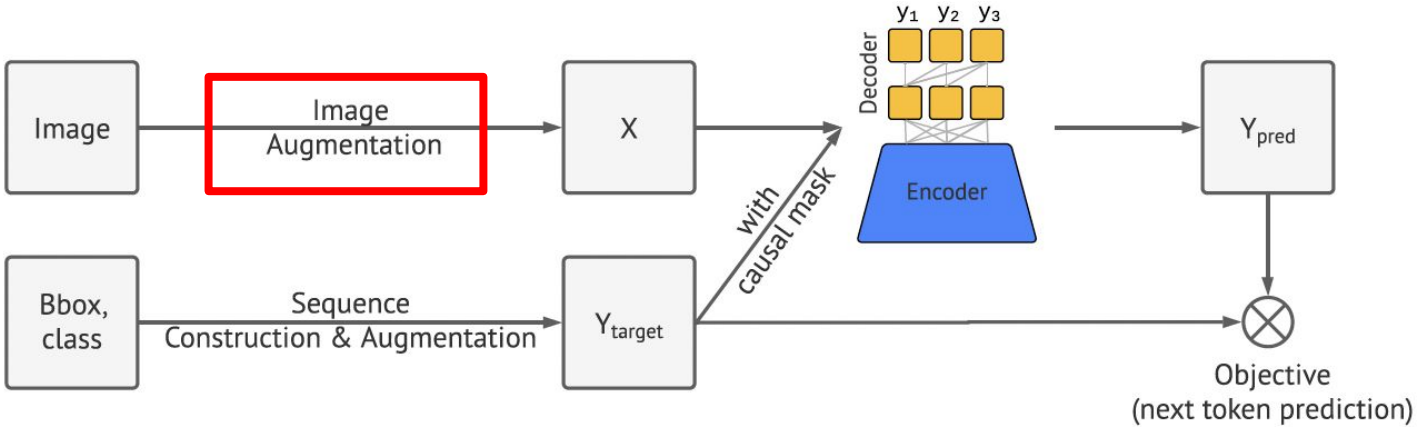


Figure 2: Major components of the Pix2Seq learning framework.

- Image augmentation:
  a. we use image augmentations to enrich a fixed set of training examples (e.g., with random scaling and crops).

Pix2Seq: How does it work?

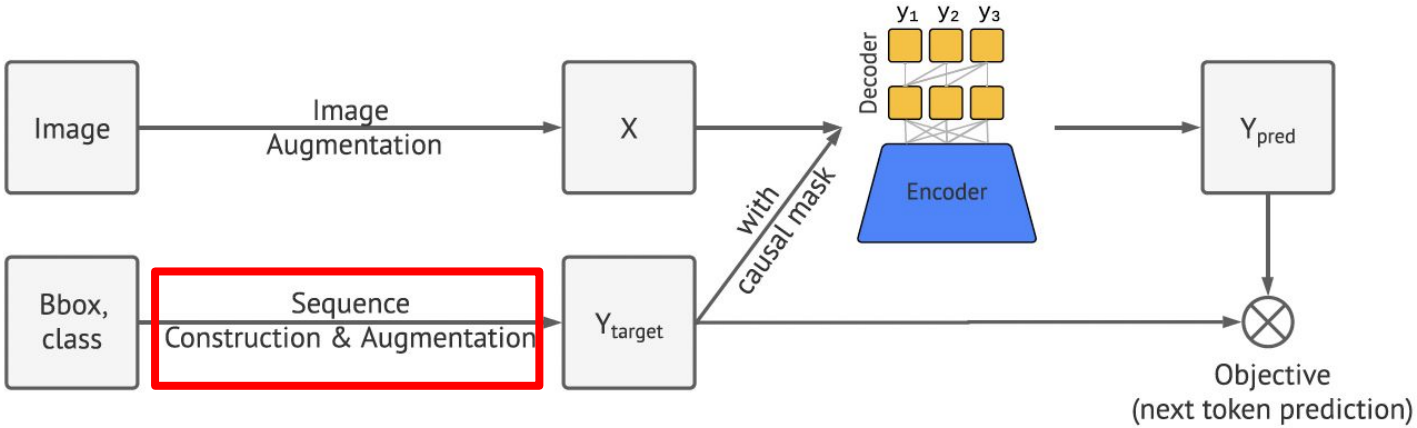


Figure 2: Major components of the Pix2Seq learning framework.

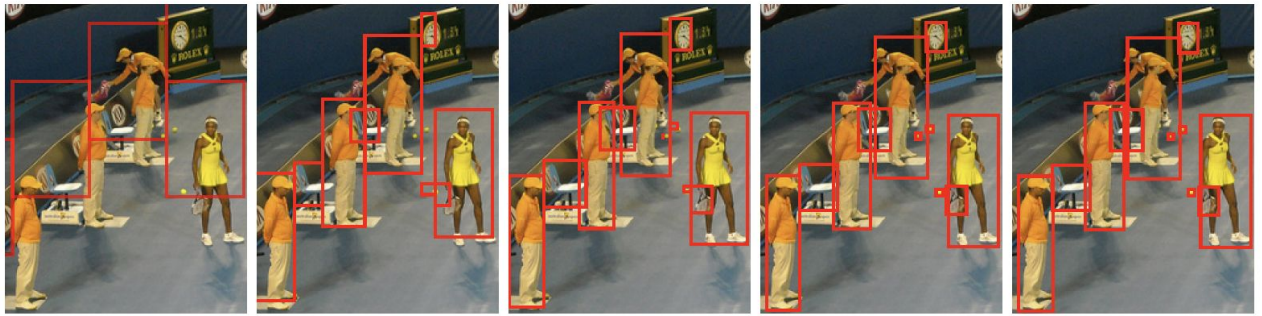
- Sequence construction & augmentation:
  a. As object annotations for an image are usually represented as a set of bounding boxes and class labels, we convert them into a sequence of discrete tokens

How we represent an object

An object is represented as a sequence of five discrete tokens,

i.e. [ymin; xmin; ymax; xmax; c],

where each of the continuous corner coordinates is uniformly discretized into an integer between [1; nbins], and c is the class index.



(a) $n_{\text{bins}} = 10$ (b) $n_{\text{bins}} = 50$ (c) $n_{\text{bins}} = 100$ (d) $n_{\text{bins}} = 500$ (e) Original

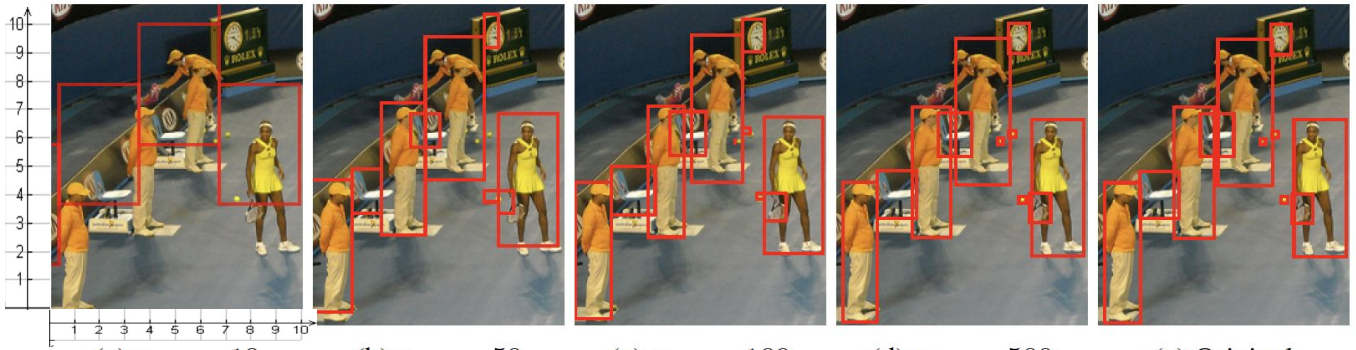
Figure 3: Applying the proposed discritization of bounding box on an image of 480×640. Only a quarter of the image is shown for better clarity. With a small number of bins, such as 500 bins ($\sim$1 pixel/bin), it achieves high precision even for small objects.

Quantization scheme

Shared vocabulary for all tokens, so

vocab_size = n_bins + n_classes

For example, a 600x600 image requires only 600 bins to achieve zero quantization error.



(a) $n_{\text{bins}} = 10$ (b) $n_{\text{bins}} = 50$ (c) $n_{\text{bins}} = 100$ (d) $n_{\text{bins}} = 500$ (e) Original

Figure 3: Applying the proposed discritization of bounding box on an image of 480×640. Only a quarter of the image is shown for better clarity. With a small number of bins, such as 500 bins ($\sim$1 pixel/bin), it achieves high precision even for small objects.

Serializing of objects in image

- Done randomly, since order of objects does not matter for the detection task.
- Different images, will have different number of object, so the length of the generated sequences will have different lengths, so we introduce an EOS (end-of-sequence) token.

Random ordering (multiple samples):

327 370 653 444 1001 544 135 987 338 1004 508 518 805 892 1004 0

544 135 987 338 1004 327 370 653 444 1001 508 518 805 892 1004 0

508 518 805 892 1004 544 135 987 338 1004 327 370 653 444 1001 0

Area ordering:

544 135 987 338 1004 508 518 805 892 1004 327 370 653 444 1001 0

Dist2ori ordering:

544 135 987 338 1004 327 370 653 444 1001 508 518 805 892 1004 0

Figure 4: Examples of sequence construction with $n_{\text{bins}} = 1000$, and 0 is EOS token.

Altered Sequence Construction

We first create synthetic noise objects to augment input sequences in the following two ways:

1) adding noise to existing ground-truth objects (e.g., random scaling or shifting their bounding boxes), and
2) generating completely random boxes (with randomly associated class labels).

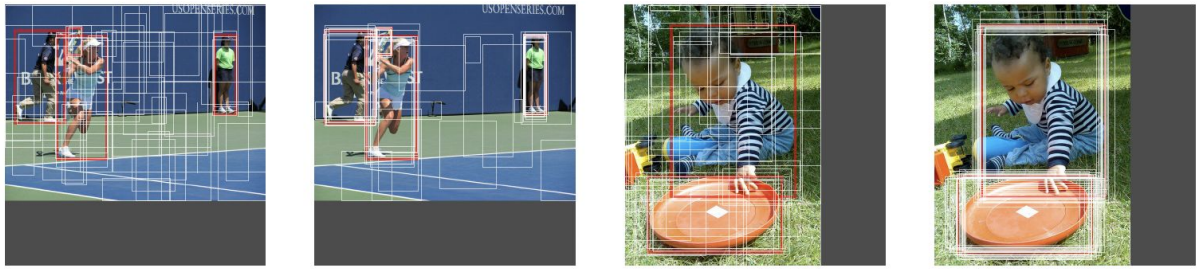
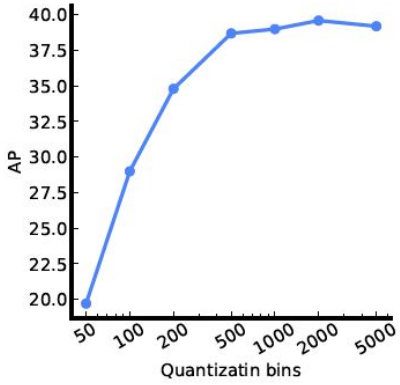
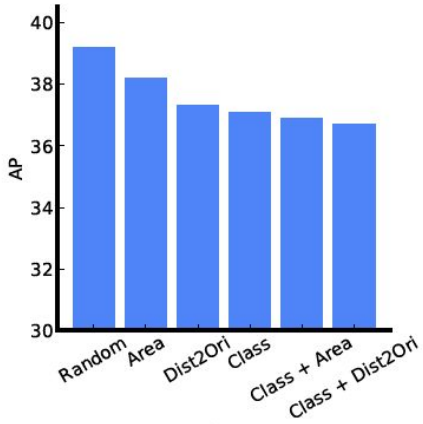


Figure 6: Illustrations of randomly sampled noise objects (in white), vs. ground-truth objects (in red).

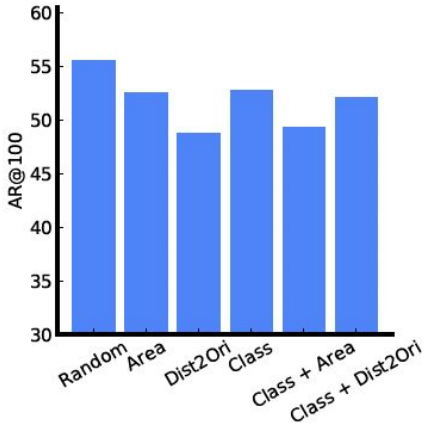
Ablation on Sequence Construction



(a)



(b)



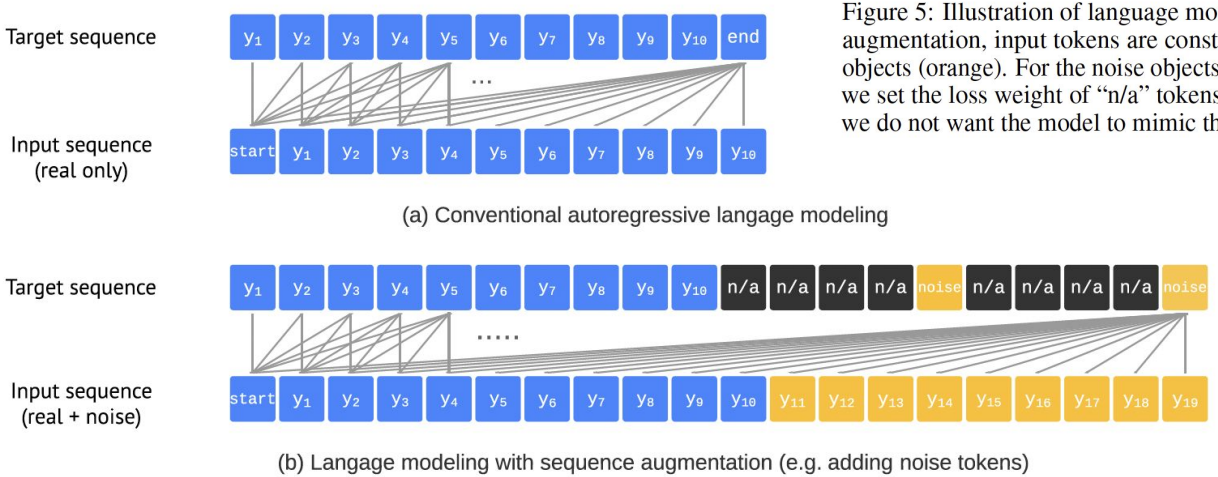
(c)

a) **Quantization:** This plot shows why the researchers chose 500 bins for 640 pixel images:
   - Plateau in performance with subsequent increases
b) **Serialization:** This graph shows average precision across different serialization techniques
c) **Serialization:** This graph shows top-100 average recall across different serialization techniques

Sequence Augmentation

We augment input sequences during training to include both real and synthetic noise tokens.

We also modify target sequences so that the model can learn to identify the noise tokens rather than mimic them.



(a) Conventional autoregressive langage modeling

(b) Langage modeling with sequence augmentation (e.g. adding noise tokens)

Figure 5: Illustration of language modeling with / without sequence augmentation. With sequence augmentation, input tokens are constructed to include both real objects (blue) and synthetic noise objects (orange). For the noise objects, the model is trained to identify them as the "noise" class, and we set the loss weight of "n/a" tokens (corresponding to coordinates of noise objects) to zero since we do not want the model to mimic them.

Ablation on Sequence Augmentation

Recall is significantly worse without Sequence Augmentation, AP slightly improved with its inclusion

- SeqAug mainly effective during fine-tuning

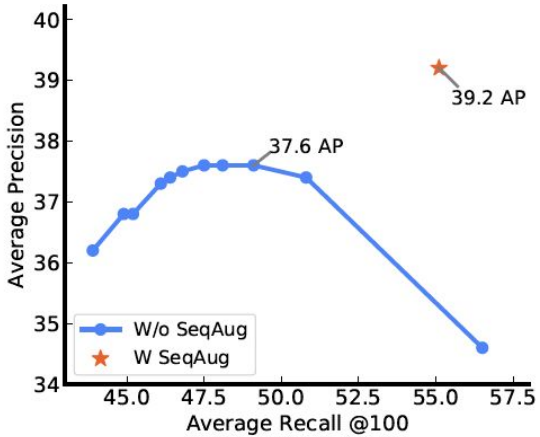


Figure 8: Impact of sequence augmentation on when training from scratch on COCO.

SeqAug in Pretrain	SeqAug in Finetune	AP	AR@100
✗	✗	43.7	55.4
✗	✓	44.5	61.6
✓	✓	44.7	61.7

Table 3: Impact of sequence augmentation when pretraining on Objects365 and finetuning on COCO. Sequence augmentation has a major impact on average recall (@100) but a smaller influence on AP. Most improvements can be achieved during fine-tuning.

Pix2Seq: How does it work?

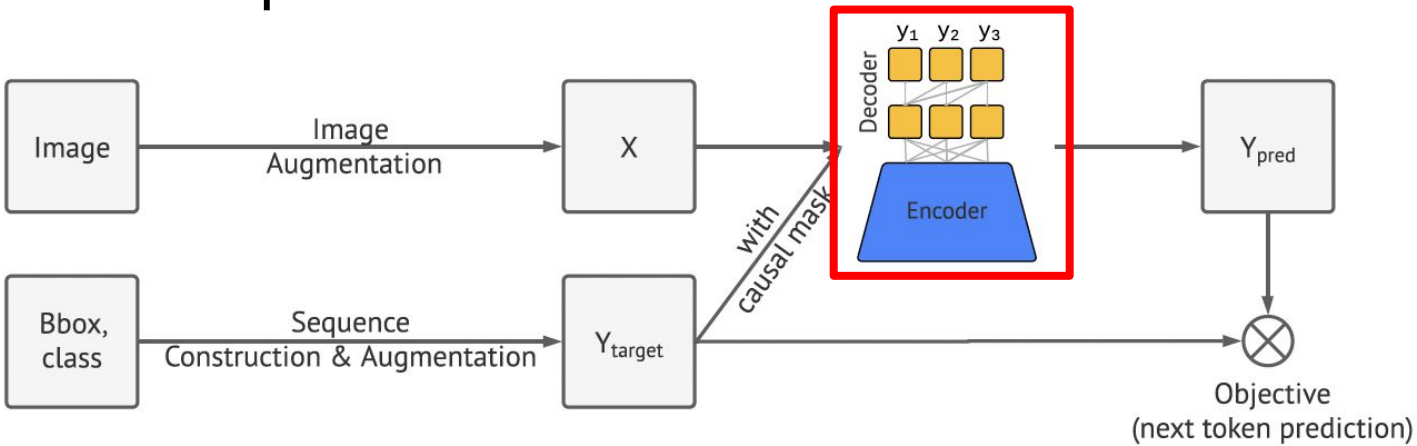


Figure 2: Major components of the Pix2Seq learning framework.

- Architecture:
  - We use an encoder-decoder model, where the encoder perceives pixel inputs, and the decoder generates the target sequence (one token at a time).

Encoder and Decoder

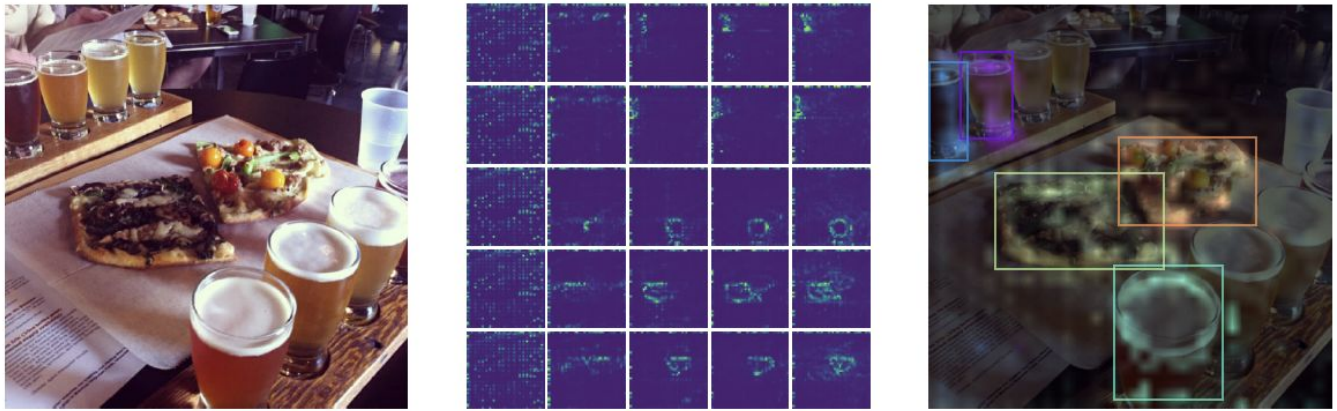
Encoder:

- General image encoder that perceives pixels and encodes them into hidden representations.
- Can be a CNN, Transformer, or hybrid.

Decoder:

- Transformer decoder (commonly used in NLP-tasks).
- Generates one token at a time, conditioned on the preceding tokens and the encoded image representation.
    - Goal is to maximize the likelihood of tokens

Decoder’s Cross Attention Map



(a) (b) (c)

a) Input Image
b) 5x5 grid visualization of the decoder generating 25 new token predictions. 1 row per object
c) Overlay of cross attention map with the input image, showing accuracy of predictions

Pix2Seq: How does it work?

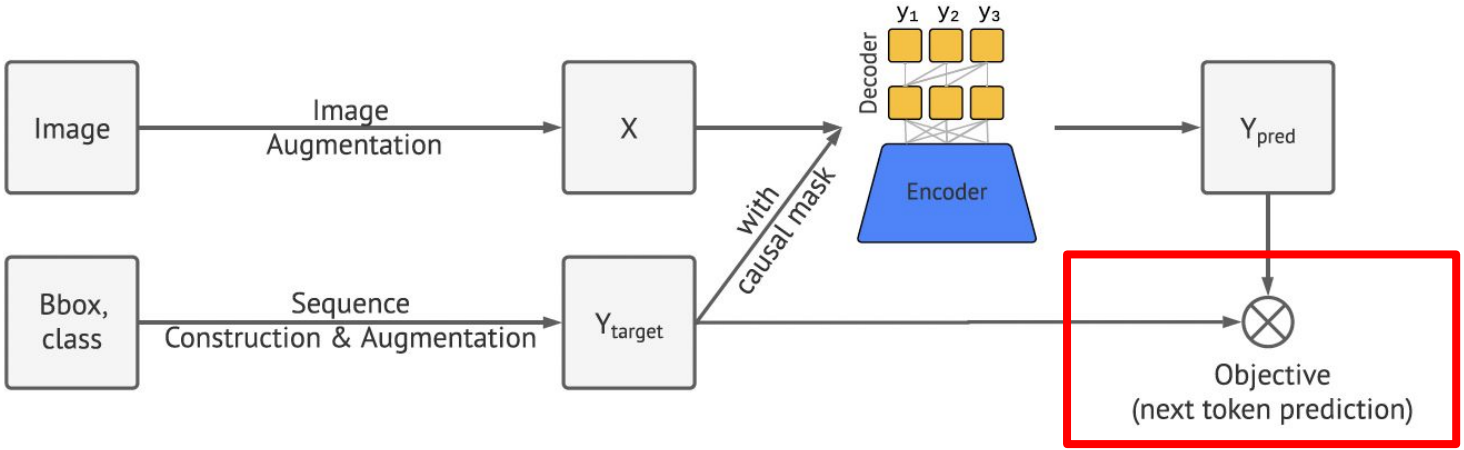


Figure 2: Major components of the Pix2Seq learning framework.

- Objective/loss function:
  - The model is trained to maximize the log likelihood of tokens conditioned on the image and the preceding tokens (with a softmax cross-entropy loss).

Objective/Loss function

Pix2Seq is driven by a maximum likelihood loss function that allows it to predict tokens given an image and preceding tokens

- Like how language models use it

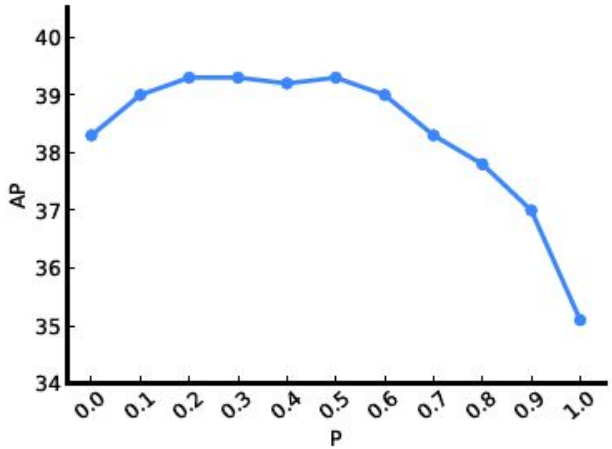
$$\text{maximize} \sum_{j=1}^{L} \boldsymbol{w}_j \log P(\tilde{\boldsymbol{y}}_j | \boldsymbol{x}, \boldsymbol{y}_{1:j-1}) \,, \tag{1}$$

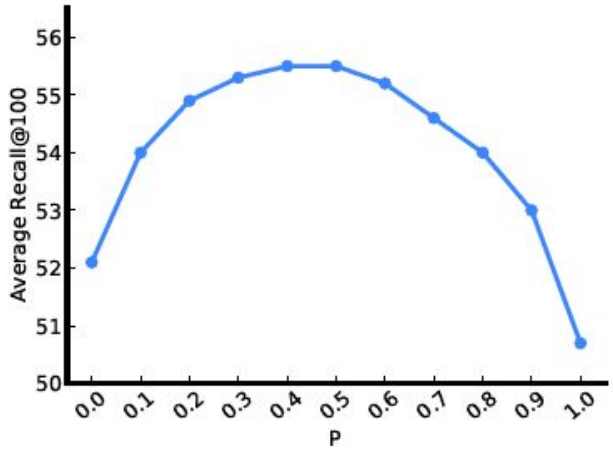
where $\boldsymbol{x}$ is a given image, $\boldsymbol{y}$ and $\tilde{\boldsymbol{y}}$ are input and target sequences associated with $\boldsymbol{x}$, and L is the target sequence length. $\boldsymbol{y}$ and $\tilde{\boldsymbol{y}}$ are identical in the standard language modeling setup, but they can also be different (as in our later augmented sequence construction). Also, $\boldsymbol{w}_j$ is a pre-assigned weight for j-th token in the sequence. We set $\boldsymbol{w}_j = 1, \forall j$, however it would be possible to weight tokens by their types (e.g., coordinate vs class tokens), or by the size of the corresponding object.

Inference

To make inferences, the model samples tokens from the model likelihood function using a method called nucleus sampling.

- reduces duplication
- increases diversity in generated samples





Ablation: P=0 is arg max sampling, P>0 is nucleus sampling

Altered Inference

Technical challenge: Early on, Pix2Seq tended to generate the EOS token prematurely (before predicting all objects)

Solution: Sequence Augmentation

Implementation: Altered Inference- after extracting the bounding boxes and class labels from generated sequences, replace the “noise” class label with the real class label that has the highest likelihood.

Experimental Setup

To test Pix2Seq, it is evaluated on MS-COCO 2017 detection dataset

- 118k training images
- 5k validation images

To compare with DETR and Faster R-CNN, report average precision (AP)

Two training strategies:

- training from scratch on COCO in order to compare fairly with the baselines
- Pretraining+finetuning
  - pretrain the Pix2Seq model on a larger object detection dataset called Objects365
  - finetune the model on COCO

Hypothesis: Pretraining+Finetuning will be superior since the Pix2Seq approach incorporates zero inductive bias / prior knowledge of the object detection task

Table 1: Comparison of average precision, over multiple thresholds and object sizes, on COCO validation set. Each section compares different methods of the similar ResNet "backbone". Our models achieve competitive results to both Faster R-CNN and DETR baselines.

Method	Backbone	#params	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Faster R-CNN	R50-FPN	42M	40.2	61.0	43.8	24.2	43.5	52.0
Faster R-CNN+	R50-FPN	42M	42.0	62.1	45.5	26.6	45.4	53.4
DETR	R50	41M	42.0	62.4	44.2	20.5	45.8	61.1
Pix2seq (Ours)	R50	37M	**43.0**	61.0	45.6	25.1	46.9	59.4
Faster R-CNN	R101-FPN	60M	42.0	62.5	45.9	25.2	45.6	54.6
Faster R-CNN+	R101-FPN	60M	44.0	63.9	47.8	27.2	48.1	56.0
DETR	R101	60M	43.5	63.8	46.4	21.9	48.0	61.8
Pix2seq (Ours)	R101	56M	**44.5**	62.8	47.5	26.0	48.2	60.3
Faster R-CNN	R50-DC5	166M	39.0	60.5	42.3	21.4	43.5	52.5
Faster R-CNN+	R50-DC5	166M	41.1	61.4	44.3	22.9	45.9	55.0
DETR	R50-DC5	41M	**43.3**	63.1	45.9	22.5	47.3	61.1
Pix2seq (Ours)	R50-DC5	38M	**43.2**	61.0	46.1	26.6	47.0	58.6
DETR	R101-DC5	60M	**44.9**	64.7	47.7	23.7	49.5	62.3
Pix2seq (Ours)	R101-DC5	57M	**45.0**	63.2	48.6	28.2	48.9	60.4

Table 2: Average precision of finetuned Pix2seq models on COCO with different backbone architectures and image sizes. All models are pretrained on Objects365 dataset. As a comparison, our best model without pretraining obtains 45.0 AP (in Table 1) with image size of 1333×1333. The pretraining is with 640×640 image size while fine-tuning (a few epochs) can use larger image sizes.

Backbone	# params	Image size during finetuning		
		640×640	1024×1024	1333×1333
R50	37M	39.1	41.7	42.6
R50-C4	85M	44.7	46.9	47.3
ViT-B	115M	44.2	46.5	47.1
ViT-L	341M	47.6	49.0	50.0