Instance Segmentation Episode I

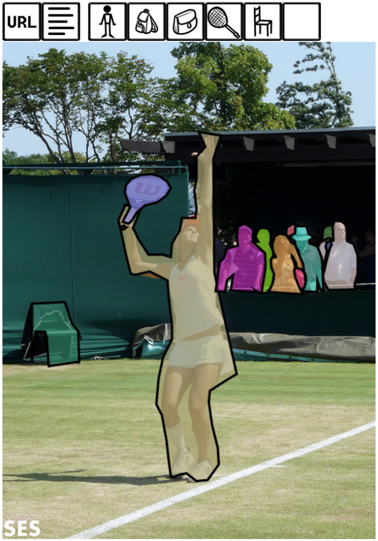
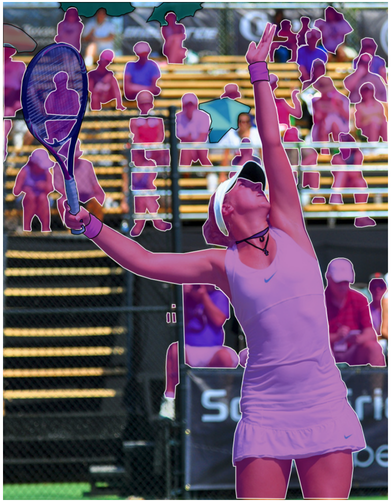
When Bounding Boxes Grow Up

Christopher Bate

September 28, 2021

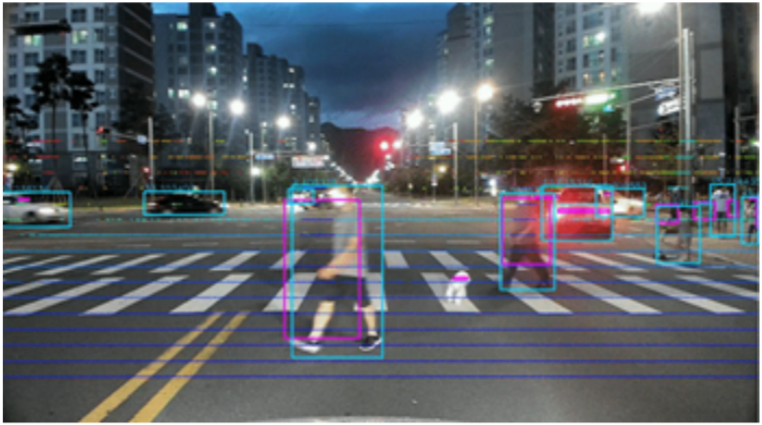
Instance Segmentation - Definition

Instance segmentation combines object detection with a per-pixel labeling task.

COCO	LVIS
	

Instance Segmentation - Motivation

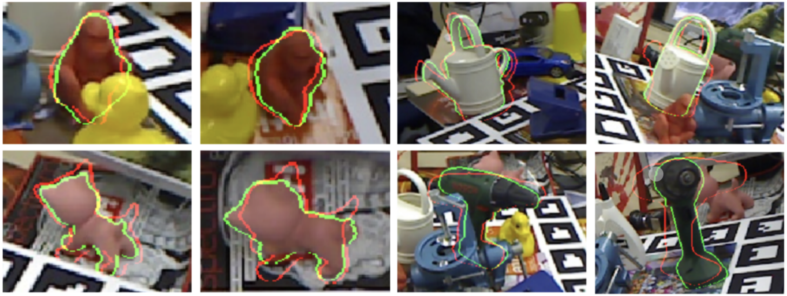
Why segment instances? An example from robotics: Identify all the lidar points which should be associated with each detection.



Credit: *Estimation of Closest In-Path Vehicle (CIPV) by Low-Channel LiDAR and Camera Sensor Fusion for Autonomous Vehicle. Bae et al. (2021)*

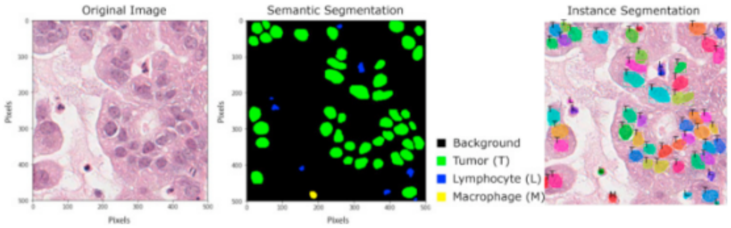
Instance Segmentation - Motivation - Robotics

Segmentation mask of object carries information about possible orientations



Ref: *DeepIM: Deep Iterative Matching for 6D Pose Estimation. Li et al. (2018)*

Instance Segmentation - Motivation - Medical / Biological



Ref: Pathology Image Analysis Using Segmentation Deep Learning Algorithms, Wang et al. (2019)

Masks - Representation

Choosing how to represent masks affects: data storage size, loading times, geometry limitations

Method	Pros	Cons
Per-pixel mask	Represent any mask	Less efficient, even with compression
Geometric representation	More efficient representation	May not be able to represent all masks (.e.g occlusions)

Instance Segmentation - Evaluation Metric I

The most often reported metric for instance detection and segmentation is **"Mean Average Precision" (mAP)**. How to we calculate this quantity?

The following example is loosely based on the one given in *A Comparative Analysis of Object Detection Metrics with a Companion Open-Source Toolkit, Padilla et al, (2021)*.

Instance Segmentation - Average Precision Calculation (VOC Style)

Hypothetical dataset: 5 images, 7 total ground truth objects. For a fixed IoU threshold τ (e.g. 0.5), tabulate all* your detections, order by confidence

Image	Conf	TP	FP	$\sum$TP	$\sum$FP	Prec.	Recall
2	0.95	1	0	1	0	1.0	0.1
1	0.91	1	0	2	0	1.0	0.2
5	0.88	1	0	3	0	1.0	0.3
3	0.80	0	1	3	1	0.75	0.3
4	0.75	1	0	4	1	0.80	0.5
9	0.61	1	0	5	1	0.83	0.7
7	0.54	1	0	6	1	0.85	0.7
10	0.31	0	1	6	2	0.75	0.9
8	0.23	0	1	6	3	0.66	0.9
9	0.06	1	0	7	3	0.70	1.0

Instance Segmentation - Average Precision Calculation (VOC Style) - PR Curve Integration

Image	Conf	TP	FP	$\sum$TP	$\sum$FP	Prec.	Recall
2	0.95	1	0	1	0	1.0	0.1
1	0.91	1	0	2	0	1.0	0.2
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
9	0.06	1	0	7	3	0.70	1.0

Instance Segmentation - Average Precision Calculation (VOC Style) - Final

Image	Conf	TP	FP	$\sum$TP	$\sum$FP	Prec.	Recall
2	0.95	1	0	1	0	1.0	0.1
1	0.91	1	0	2	0	1.0	0.2
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
9	0.06	1	0	7	3	0.70	1.0

1. Create the table, graph, and integrate* for each class using IoU threshold $\tau = 0.5$.
2. Average over all classes to get mAP.

* - Integration done using N interpolation points, "waterfall" the precision to ensure it is monotonically decreasing.

Instance Segmentation - Average Precision Calculation (COCO Style)

Similar to PASCAL VOC-style calculation, except:

1. Create PR curve, integrate, and average over classes (using $N = 101$ recall interpolation points) using linearly spaced IoU thresholds $\tau = \{0.5, 0.55, ..., 0.95\}$.
2. This yields the set of scores $\{AP_{0.5}, ..., AP_{0.95}\}$
3. Take the average of this set to get COCO-style mAP.

Instance Segmentation - What about recall?

Recall is reported as well:

1. The average recall (AR) quantity AR_N is defined as the average recall over all classes when allowing only the top N detection candidates per image (class agnostic).
2. Commonly reported on COCO dataset is $AR_{1,10,100}$.

Datasets - Framework

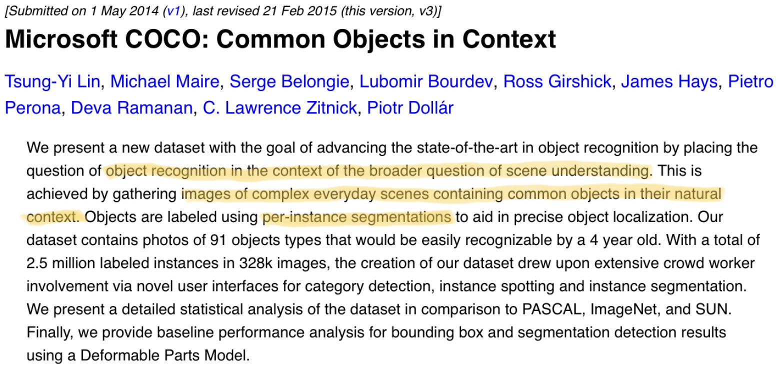
Today: COCO, LVIS

How we'll analyze the datasets:

1. What is the task and why?
2. How do we measure success?
3. How do we get good accuracy (label quality) without breaking the bank and deadline?
4. Can it push progress forward?

Note: All images and plots in the following slides are taken from the respective papers unless otherwise noted.

Common Objects in Context (COCO) - Task Motivation



[Submitted on 1 May 2014 (v1), last revised 21 Feb 2015 (this version, v3)]

Microsoft COCO: Common Objects in Context

Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, Piotr Dollár

We present a new dataset with the goal of advancing the state-of-the-art in object recognition by placing the question of object recognition in the context of the broader question of scene understanding. This is achieved by gathering images of complex everyday scenes containing common objects in their natural context. Objects are labeled using per-instance segmentations to aid in precise object localization. Our dataset contains photos of 91 objects types that would be easily recognizable by a 4 year old. With a total of 2.5 million labeled instances in 328k images, the creation of our dataset drew upon extensive crowd worker involvement via novel user interfaces for category detection, instance spotting and instance segmentation. We present a detailed statistical analysis of the dataset in comparison to PASCAL, ImageNet, and SUN. Finally, we provide baseline performance analysis for bounding box and segmentation detection results using a Deformable Parts Model.

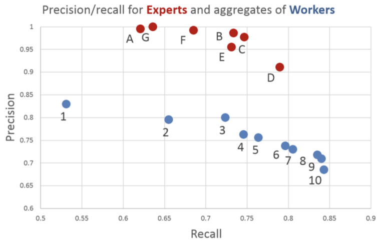
1. 80 classes used, representing common types of objects ("recognizable by a 4 year old")
2. All splits (train, val, test) combined have $> 200,000$ images.
3. Classes are "pairwise distinct." An object is easily recognizable as 1 out of the 80 classes to minimize ambiguous cases.

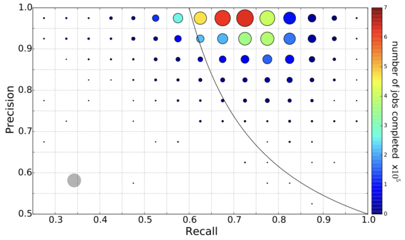
Common Objects in Context (COCO) - Task Motivation

1. Ensure images reflect context and not only close-up, centered shots.
2. Focus on a smaller number of categories (versus ImageNet), with a large number of instances per category.
3. Provide segmentation masks to ensure accurate labeling.

Common Objects in Context (COCO) - How do we measure success?

1. "Non iconic" - images should have several categories, with good centroid distribution.
2. Image mining process - used Flikr, but with semi-automated pruning process.
3. Measure labeler P/R to validate the process (below).
4. Labeler agreement at each stage (next).

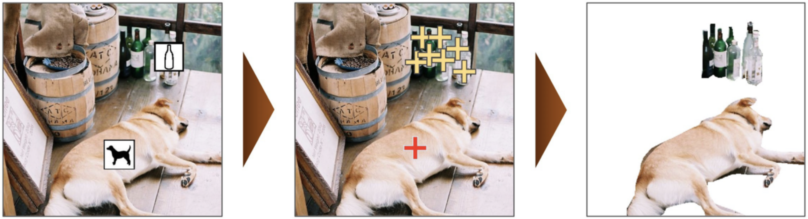




Common Objects in Context (COCO) - How do we get good accuracy?

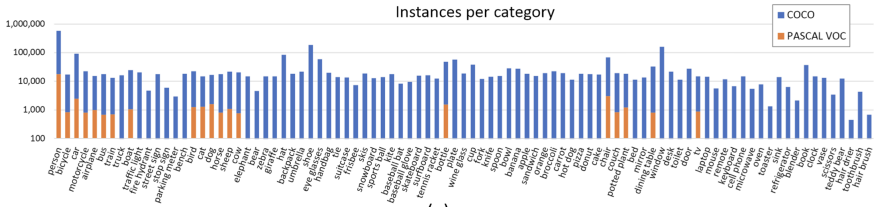
Structured Labeling Pipeline

1. Stage I - Cateogry labeling
2. Stage II - Instance spotting
3. Stage III - Instance segmentation

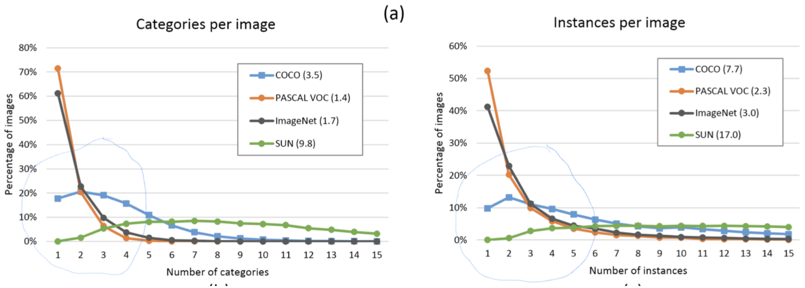


(a) Category labeling (b) Instance spotting (c) Instance segmentation

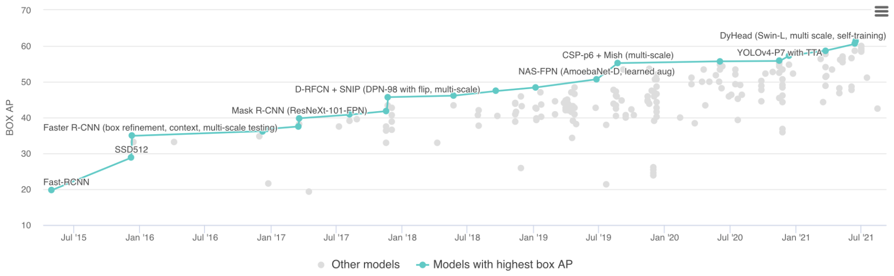
Common Objects in Context (COCO) - Results - Instances per Category



Common Objects in Context (COCO) - Results - Per Image - Instances and Categories

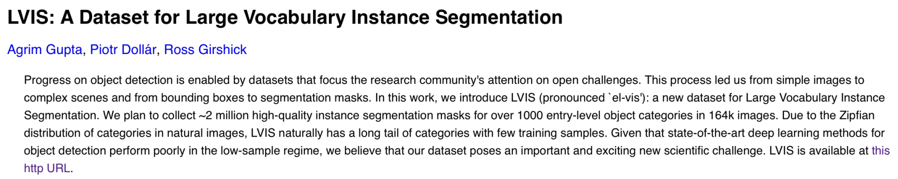


Common Objects in Context (COCO) - Does it push progress forward?



Source: *Papers With Code*

LVIS: A Dataset for Large Vocabulary Instance Segmentation



LVIS: A Dataset for Large Vocabulary Instance Segmentation

Agrim Gupta, Piotr Dollár, Ross Girshick

Progress on object detection is enabled by datasets that focus the research community's attention on open challenges. This process led us from simple images to complex scenes and from bounding boxes to segmentation masks. In this work, we introduce LVIS (pronounced `el-vis'): a new dataset for Large Vocabulary Instance Segmentation. We plan to collect ~2 million high-quality instance segmentation masks for over 1000 entry-level object categories in 164k images. Due to the Zipfian distribution of categories in natural images, LVIS naturally has a long tail of categories with few training samples. Given that state-of-the-art deep learning methods for object detection perform poorly in the low-sample regime, we believe that our dataset poses an important and exciting new scientific challenge. LVIS is available at this http URL.

LVIS - Task and Motivation

1. **Task** - Reuse images from existing MS COCO dataset, but add more categories with more detailed segmentation masks.
2. **Motivation** - The opposite of COCO's instances/category goal: capture the natural distribution object appearance
3. **Motivation** - Current popular models typically perform poorly "in the low sample regime"
4. **Motivation** - "The long tail of rare categories is inescapable; annotating more images simply uncovers previously unseen, rare categories" - A common industry problem
5. **Motivation*** - Solve two simultaneous problems: low examples/category and a non-exhaustive training set

The "Inescapable Long Tail"

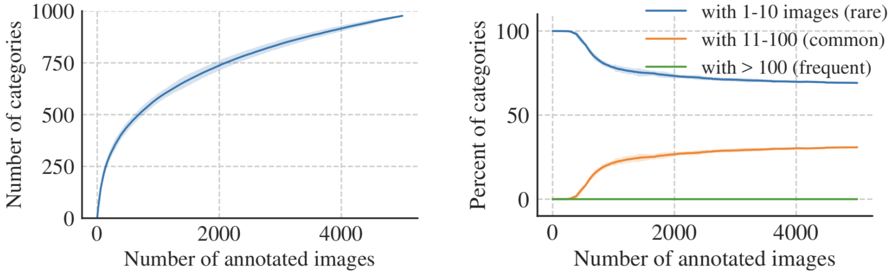
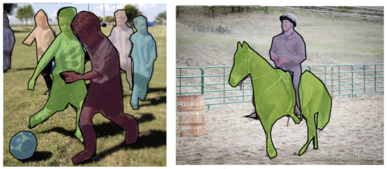


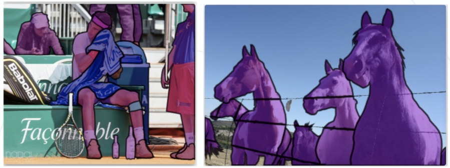
Figure 9. (Left) As more images are annotated, new categories are discovered. (Right) Consequently, the percentage of low-shot categories (blue curve) remains large, decreasing slowly.

LVIS vs. COCO - I



COCO

Coarse polygon annotations

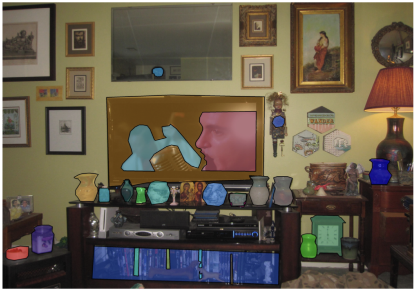


LVIS

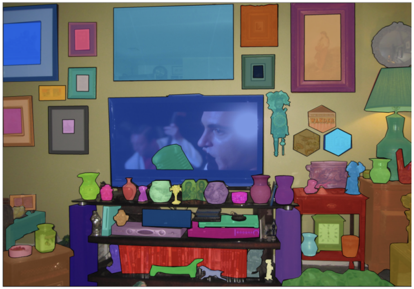
Precise polygon annotations

Source: *A Good Bounding Box is not a Guarantee of a Good Mask. Tan et al. ECCV 2020 LVIS Workshop.*

LVIS vs. COCO - II



COCO



LVIS

Source: *LVIS Introductory Talk. Agrim Gupta. ECCV 2020 LVIS Workshop.*

LVIS - Federated Dataset - I

Problem: Exhaustive labeling of 160k images with 1000 categories in multi-class regime is very difficult. There will be unavoidable ambiguity.

1. Separate into multiple single-class datasets.
2. The global image collection has positive/negative subsets P_c and N_c for each category c.
3. Final dataset $D = \cup_c (P_c \cup N_c)$

Why use a federated dataset?

LVIS - Federated Dataset - II

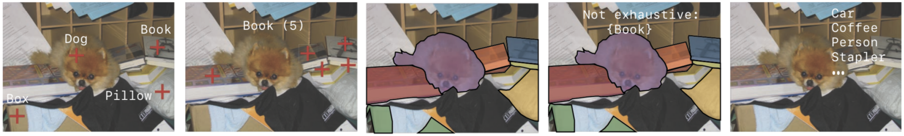
Problem: Multi-class labeling, non-exhaustive labels make a federated dataset difficult to evaluate.

1. Images are manually annotated with binary label e_c indicating whether or not it has been exhaustively labeled for category c. When e_c is false, false positives for category c are not counted.
2. The global image collection has positive/negative subsets P_c and N_c for each category c.

Why use a federated dataset?

LVIS - How do we get good accuracy?

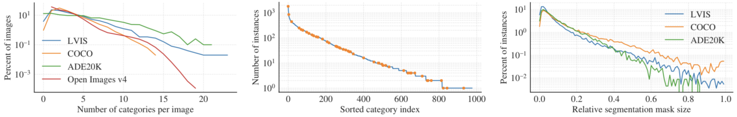
Expanded labeling pipeline vs. COCO:



Stage 1: Object spotting, one instance per category | Stage 2: Exhaustive instance marking of each category | Stage 3 & 4 (back and forth): Segmentation and verification | Stage 5: Exhaustive annotation verification | Stage 6: Negative labels

Figure 4. Our **annotation pipeline** comprises six stages. **Stage 1: Object Spotting** elicits annotators to mark a single instance of many different categories per image. This stage is iterative and causes annotators to discover a long tail of categories. **Stage 2: Exhaustive Instance Marking** extends the stage 1 annotations to cover all instances of each spotted category. Here we show additional instances of *book*. **Stages 3 and 4: Instance Segmentation and Verification** are repeated back and forth until ~99% of all segmentations pass a quality check. **Stage 5: Exhaustive Annotations Verification** checks that all instances are in fact segmented and flags categories that are missing one or more instances. **Stage 6: Negative Labels** are assigned by verifying that a subset of categories do not appear in the image.

LVIS - Analysis I



(a) Distribution of category count per image. LVIS has a heavier tail than COCO and Open Images training set. ADE20K is the most uniform.

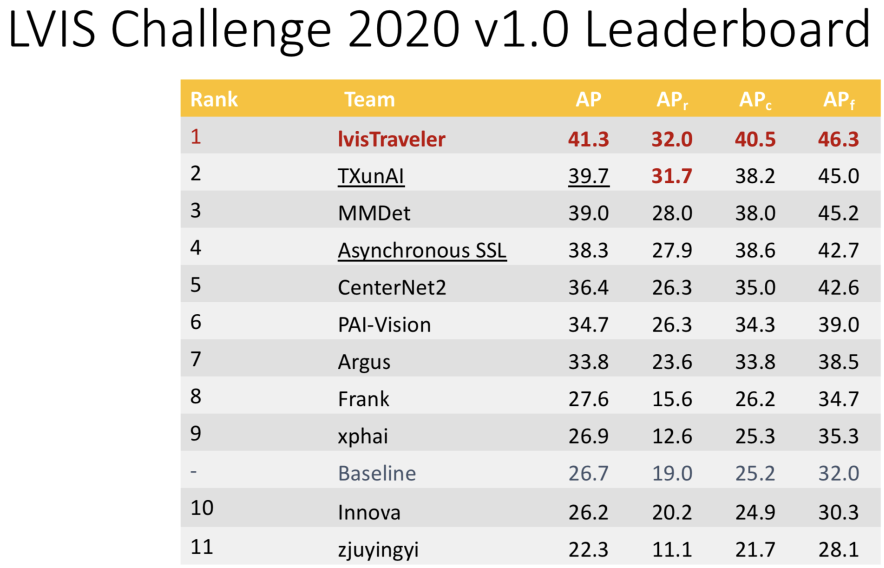
(b) The number of instances per category (on 5k images) reveals the long tail with few examples. Orange dots: categories in common with COCO.

(c) Relative segmentation mask size (square root of mask-area-divided-by-image-area) compared between LVIS, COCO, and ADE20K.

Figure 6. **Dataset statistics**. Best viewed digitally.

LVIS - Does it push progress forward?

LVIS Workshop (ECCV 2020) Results:



LVIS Challenge 2020 v1.0 Leaderboard

Rank	Team	AP	AP_r	AP_c	AP_f
1	**lvisTraveler**	**41.3**	**32.0**	**40.5**	**46.3**
2	<u>TXunAI</u>	<u>39.7</u>	**31.7**	38.2	45.0
3	MMDet	39.0	28.0	38.0	45.2
4	<u>Asynchronous SSL</u>	38.3	27.9	38.6	42.7
5	CenterNet2	36.4	26.3	35.0	42.6
6	PAI-Vision	34.7	26.3	34.3	39.0
7	Argus	33.8	23.6	33.8	38.5
8	Frank	27.6	15.6	26.2	34.7
9	xphai	26.9	12.6	25.3	35.3
-	Baseline	26.7	19.0	25.2	32.0
10	Innova	26.2	20.2	24.9	30.3
11	zjuyingyi	22.3	11.1	21.7	28.1

From Convolution to Self Attention - I

Consider the block matrix data X which is of shape (1×5) (ignoring block parameter). Each x_i is a column vector.

$$X = \begin{bmatrix} x1 & x2 & x3 & x4 & x5 \end{bmatrix} \tag{1}$$

From Convolution to Self Attention - II

Convolution as (block) matrix multiplication, filter W convolved with input X to yield Y:

$$W_1 = \begin{bmatrix} w1 & w2 \end{bmatrix} \tag{2}$$

$$Y = WX^T \tag{3}$$

$$\begin{bmatrix} w2 & 0 & 0 & 0 & 0 \\ w1 & w2 & 0 & 0 & 0 \\ 0 & w1 & w2 & 0 & 0 \\ 0 & 0 & w1 & w2 & 0 \\ 0 & 0 & 0 & w1 & w2 \end{bmatrix} \begin{bmatrix} x1 & x2 & x3 & x4 & x5 \end{bmatrix}^T \tag{4}$$

From Convolution to Self Attention - II

Taking a step back, we have

$$Y = WX^T \tag{5}$$

Currently W is 5×5 (sparse, omitting the block size). There are only learnable vectors of size C: w_1, w_2.
Let's do something slightly arbitrary.

$$Y = QX^T \tag{6}$$

Where Q is now 5×5 *dense*. We now have 25 parameters. This is a super-set of the possible operators vs. the convolution structure.

From Convolution to Self Attention - III

Let's do something a little more interesting, however. The insight we want to have is that *the input parameterizes the transformation*. What does this mean?

Let's make Q a function of X:

$$Q = q(X) \tag{7}$$

$$Y = q(X)X^T \tag{8}$$

But what specific form should q have?

From Convolution to Self Attention - IV

But what specific form should q have?
Let's choose one common type of *factorization*, loosely based on the "Gram matrix" idea: We want a 5x5 matrix, so we can choose a matrix K:

$$q(X) = (XK)^T(XK) \tag{9}$$

The output is of shape (5×5).

From Convolution to Self Attention - IV

But what specific form should q have?
Let's choose one common type of *factorization*, loosely based on the "Gram matrix" $(X^T X)$ idea: We want a 5x5 matrix, so we can choose a matrix K which is $C \times 5$

$$q(X) = (XK)^T (XK) \tag{10}$$

The output is of shape $(5 \times C) \times (C \times 5) = (5 \times 5)$.

From Convolution to Self Attention - V

In summary we now have

$$Y = [(XK)^T(XK)]X^T \tag{11}$$

We can apply a row-wise normalization ρ:

$$Y = \rho((XK)^T(XK)])X^T \tag{12}$$

To get to *actual* "dot product self attention" we increase learnable parameters:

$$Y = \rho((XW_q)^T(XW_k))(W_vX)^T \tag{13}$$

Questions