Multimedia Technology

Lecture 10: Visual Instance Search &
Text to Visual Instance Search

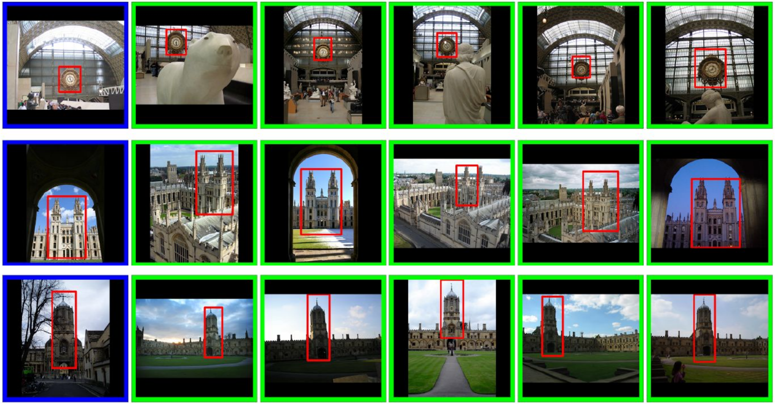
Lecturer: *Dr*. Wan-Lei Zhao

Autumn Semester 2024

Email: wlzhao@xmu.edu.cn, *copyrights are fully reserved by the author.*

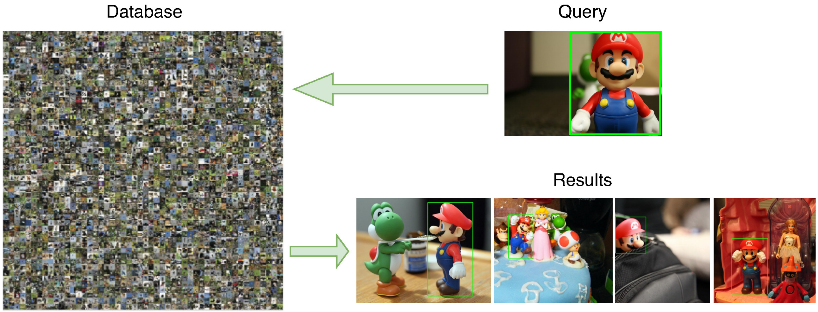
Outline

Overview of Instance Search



1. Search for instances of a specific object, person, or location
2. Localize the instance in the image (given as bounding box)
3. Also known as "sub-image search"

Instance Search: the problem



- Instance search is widely used in various multimedia applications
  - video editing, image hyperlink and online shopping, etc.
  - Instance: <u>any semantically meaningful visual subject</u>

Major Challenges in Instance Search: representation (1)

- Faces similar challenges as Image Search
- But ... even more ...

Figure: Object proposals produced by "edgebox".

1. Global representation does not work
2. Keypoint features are vulnerable to object deformations
3. Bounding boxes produce too many meaningless candidates
4. It requires an object level representation

Major Challenges in Instance Search: indexing structure (2)

- Given 40 instances in one image
  1. Memory consumption is one magnitude higher than image search
  2. The location information should be kept with indexing structure
  3. The speed efficiency is one magnitude slower than image search
  4. Faces similar performance degradation as the scale of problem grows

Overview of Instance Search

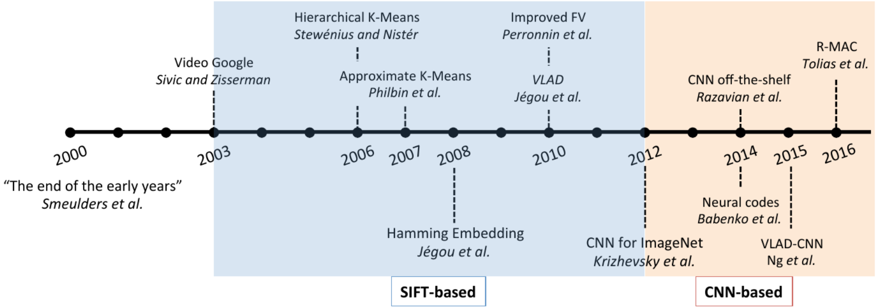


Figure: Milestones of instance search.

- Different kinds of visual features
  1. Local features: SIFT, SURF, BoVW, Fisher Vector, VLAD, etc.
  2. Global features: GIST, HOG, LBP, etc.
  3. Deep features: aggregated or quantized by BoVW or VLAD.

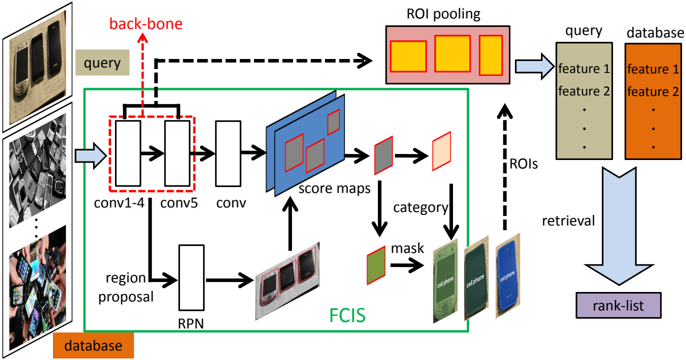
Overview: Instance Search with Deep features (1)

- Image local features are good to describe image local regions
- Advantage
  1. they cover most of the local regions
  2. they are very distinctive
- Disadvantage
  1. they are in big number
  2. they are sensitive to deformations
  3. they are sometimes too distinctive
  4. they do not cover an object exactly

Overview: Instance Search with Deep features (2)

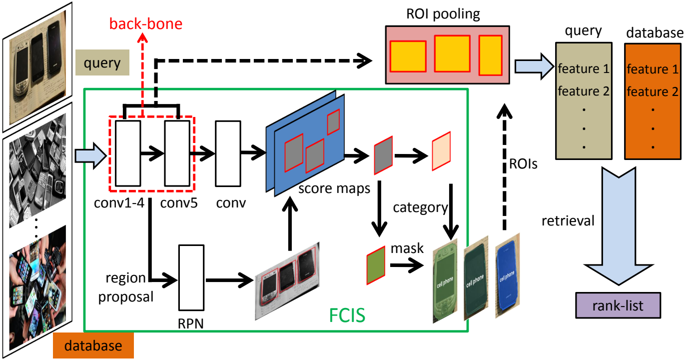
- We are searching for an instance level feature representation
- One feature should cover exactly/approximately one instance
- Challenges and Expectations
    1. Instances are in various shapes and layouts
    2. Instances of the category should be similar
    3. Instances of the same class should be still distinctive to each other

Instance Search with Deep features: the framework (1)



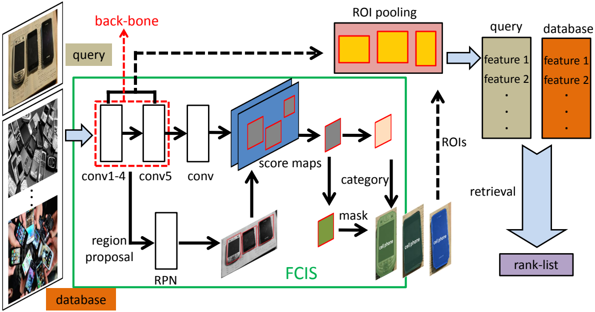
- A full convolutional neural network is trained
- It is originally used for semantic segmentation

Instance Search with Deep features: the framework (2)



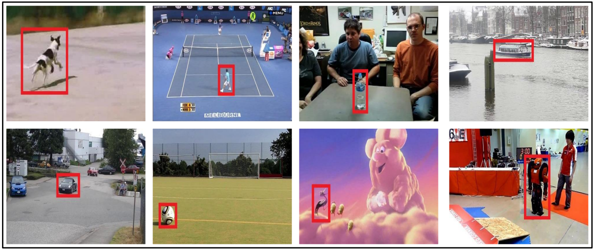
- The backbone network is ResNeXt
- The output are the segmentations of instances

Instance Search with Deep features: the framework (3)



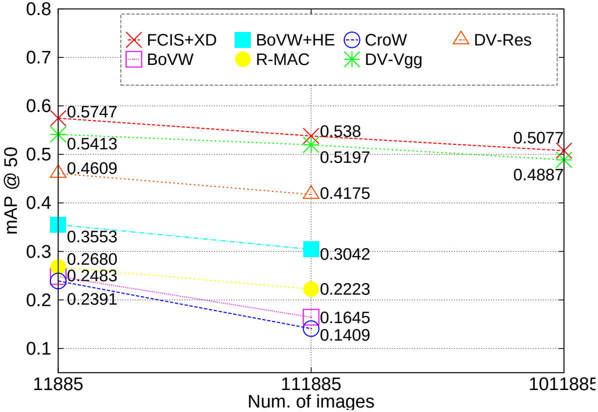
- The output are the segmentations of instances
- ROI pooling is applied on each segmented region
- One instance is finally represented by one feature with uniform length

Dataset: Instance-160



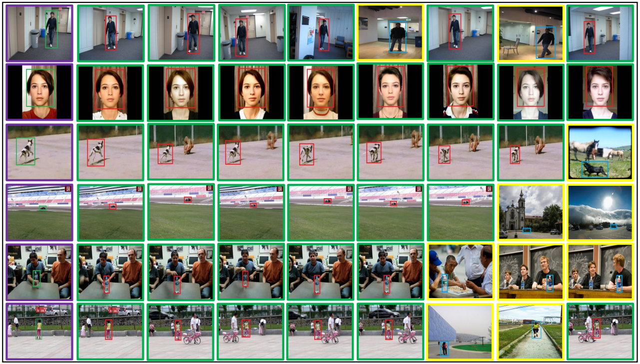
- 160 query instances are collected from 160 object tracking video
- 12,000 images are extracted from the video (dense sampling)

Performance on Instance-160 (1)



- Comparisons are conducted with deep features and image local features

Performance on Instance-160 (2)



- This approach works pretty well
- It requires a well annotated trainning set (pixel level)

Outline

Existing Solutions and Challenges (1)

- Image-search based solutions
  - Features are aggregated from several local regions into a global feature
  - Several weighting strategies are employed to highlight instances
  - e.g., R-MAC, CroW, CAM, BLCF-SalGAN, and Regional Attention
- Advantages
  - Only pre-trained models are required
- Challenges
  - Features are not discriminative for instance search
  - The instance localization are unfeasible

Existing Solutions and Challenges (2)

- Instance-level solutions
  - Instances are localized using object detection or segmentation framework
  - For instance, DeepVision, FCIS+XD and PCL*+SPN
- Advantages
  - Instance-level localizations and features are obtained
- Challenges
  - The training conditions are demanding
  - Generalization to the unseen categories is nearly impossible

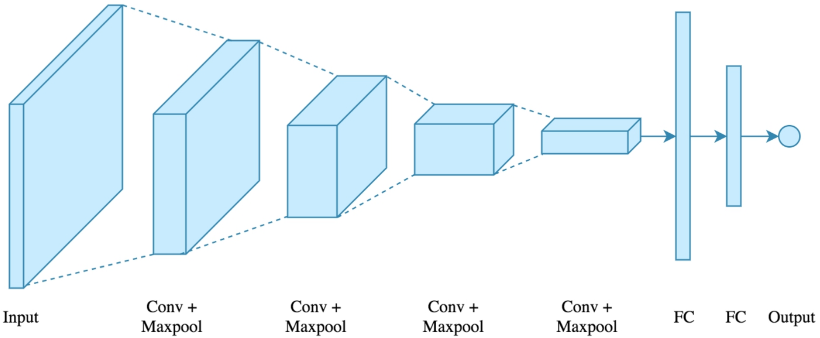
The Aim of our Design

1. Class-agonistic
2. Instance localization
3. High discriminative of the instance level feature

Motivation: the idea

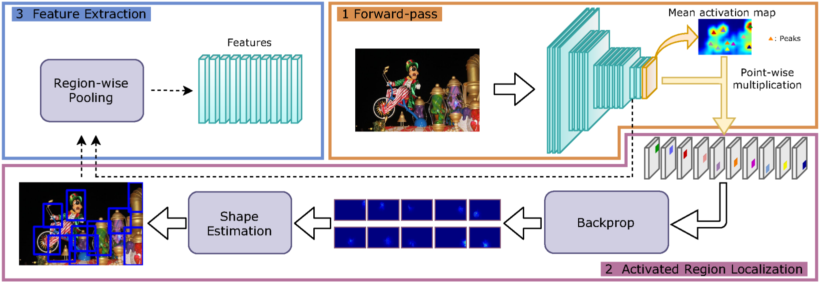
- The last convolution layer preserves class-agnostic clues for latent instances
  - They are not suppressed in the prediction layer yet

The Last Conv. Layer: a recap



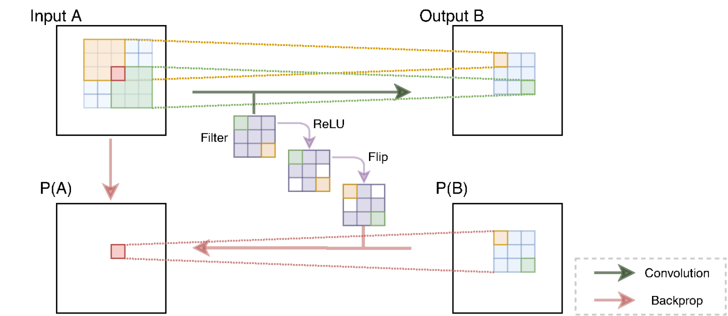
- Objects from both the known and unknown classes are activated
- After FC, the activation on the unknown objects will be suppressed

The Framework



- Peaks in the forward-pass indicate the latent instances (of both known and unknown)
- A back-propagation process is leveraged to highlight instance regions
- Instance-level features are extracted with localization results

Back-propagation in One Layer in Detail

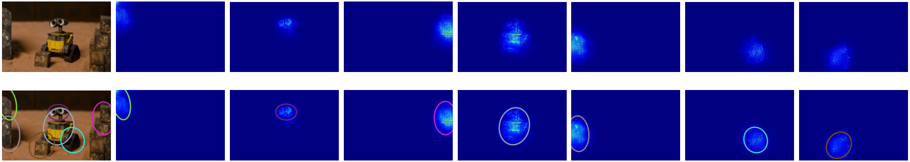


$$P(A_{x,y}) = \sum_{i=x-\frac{W_f}{2}}^{x+\frac{W_f}{2}} \sum_{j=y-\frac{H_f}{2}}^{y+\frac{H_f}{2}} P(A_{x,y}|B_{i,j})P(B_{i,j}) \tag{1}$$

$$P(A_{x,y}|B_{i,j}) = \begin{cases} Z_{i,j}A_{x,y}F_{x-i,y-j}, & \text{if } F_{x-i,y-j} > 0 \\ 0, & \textit{otherwise}. \end{cases} \tag{2}$$

- A top-down probability model is introduced

Instance Localization with Second Moment Matrix



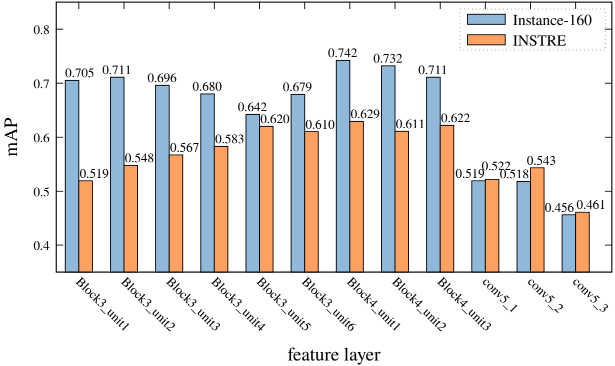
$$\sum_{r(x,y)\geq\tau} \begin{bmatrix} x^2 & x \cdot y \\ x \cdot y & y^2 \end{bmatrix} \tag{3}$$

- The second moment matrix is employed to estimate the instance shape
- The final localizations are the circumscribed rectangles of the estimated ellipses

More Salient region: DASR*

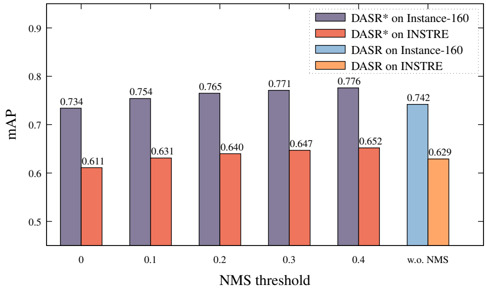
- Remaining issues
  - Different instances share one latent response peak
  - Different peaks indicate nearly the same region
- Solutions
  - More pixels are back-propagated
  - Non-maximum suppression (NMS) is employed to reduce the representation redundancy and select out the most salient regions

Ablation Study (1): layer for feature-pooling



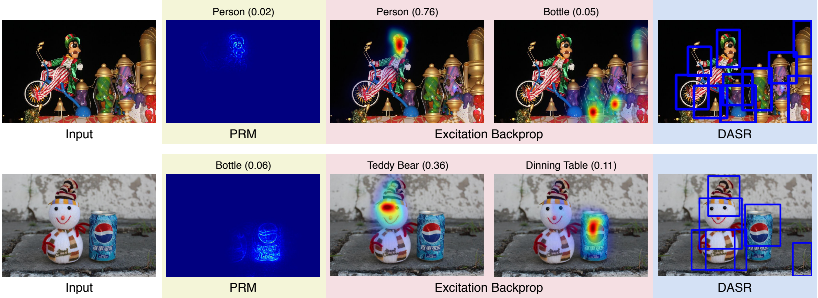
- Experiments are conducted with ResNet-50 and Vgg-16
- Features derived from ResNet-50 are much distinctive

Ablation Study (2): DASR vs. DASR*



- DASR* outperforms DASR when $\beta > 0.1$
- The larger overlapping rate β leads to better performance

How about Back-propagating from the Last Layer



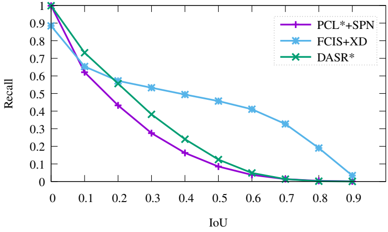
- Comparing with the approaches back-propagated from the last layer, DASR enables to localize class-agnostic instances with bounding boxes.

Instance Search Results on Two Benchmarks

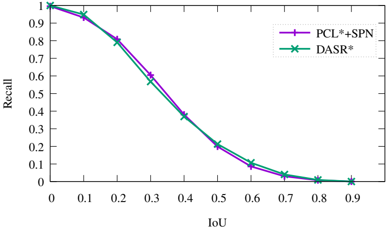
Approach	Model-Type	Loc.	Dim.	Instance-335			INSTRE
				Top-50	Top-100	All	
R-MAC	pre-trained	image	512	0.234	0.315	0.375	0.523
CroW	pre-trained	image	512	0.159	0.225	0.321	0.416
CAM	pre-trained	image	512	0.194	0.263	0.347	0.320
BLCF	pre-trained	image	336	0.246	0.358	0.483	0.636
BLCF-SalGAN	pre-trained	image	336	0.245	0.350	0.469	**0.698**
Regional Attention	pre-trained	image	2,048	0.242	0.351	0.488	0.542
DeepVision	strong	region	512	0.402	0.521	0.620	0.197
FCIS+XD	strong	pixel	1,536	0.403	0.500	0.593	0.067
PCL*+SPN	weak	region	1,024	0.380	0.475	0.580	0.575
DASR	pre-trained	region	2,048	0.419	0.558	0.699	0.629
DASR*	pre-trained	region	2,048	**0.433**	**0.580**	**0.724**	0.647
DASR-m	pre-trained	region	2,048	0.411	0.533	0.662	0.671
DASR-m*	pre-trained	region	2,048	0.428	0.560	0.694	0.692

- DASR outperforms many weakly supervised approaches
- The only pre-trained model that achieves region level localization

Localization Accuracy



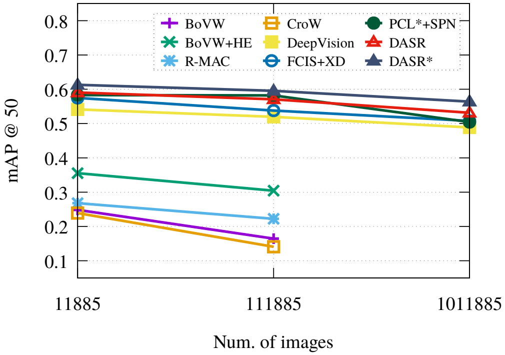
(a) Instance-335



(b) INSTRE

- DASR* shows superior performance compared to weakly supervised model PCL*+SPN

Instance Search Results in Large-scale



- DASR* outperforms all the approaches, including FCIS+XD based on a fully supervised model

Instance Search Samples

- It is meaningful even for false-positive samples
- DASR fails when the object is in small-scale ($< 32 \times 32$ pixels)

DASR for Image Search: the idea

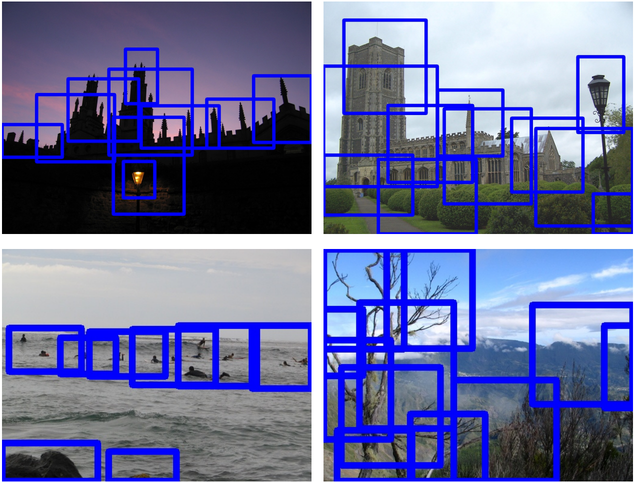
- DASR features are considered as instance level features
- DASR features could be aggregated into image level feature via VLAD

Image Search Results

Method	Dim.	Holidays	Oxford5k	Paris6k
BoVW+HE	65,536	0.742	0.503	0.501
SIFT+VLAD*	8,192	0.664	0.359	0.391
R-MAC	512	-	0.669	0.830
CroW	512	0.851	0.708	0.797
CAM	512	0.785	0.712	0.805
BLCF	336	0.854	0.722	0.798
BLCF-SalGAN	336	0.835	0.746	0.812
Regional Attention	2,048	-	**0.768**	**0.875**
DeepVision	512	-	0.710	0.798
DASR+VLAD	8,192	0.834	0.594	0.690
DASR*+VLAD	8,192	**0.873**	0.613	0.744

- It is competitive to features specfically designed for image-level search
- It becomes possible to integrate instance-level and image-level search under one framework

How DASRs are Distributed in a Natural Image



Summary

- Advantages
  - No additional training data or training stage is required
  - Localization of latent foreground instances is feasible
  - The pipeline can be carried out using any CNN classification network

Outline

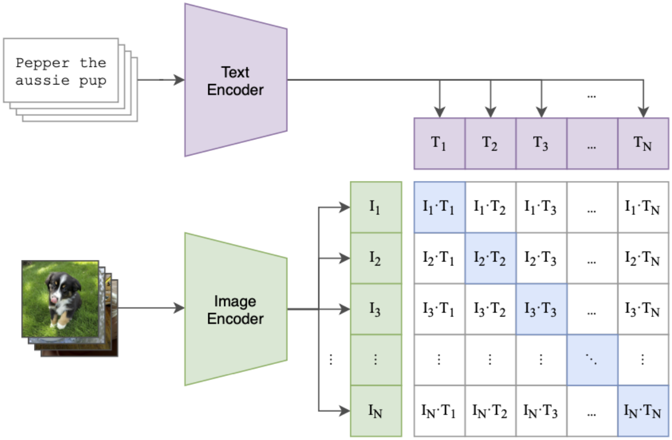
Motivation of Text-to-Image/Instance Search

- Given text query, we want to search for images/visual instance that are semantically relevant
- This is achieved by text descriptions paired with images
- Or "image captioning"
- CLIP fills the semantic gap between image and text

What is CLIP model?

- It is a text-image model
- Mapping text and image into the same feature space
- Support many downstream tasks
  1. Zero-shot object detection
  2. Image Classification
  3. Image generation, e.g. DALLE

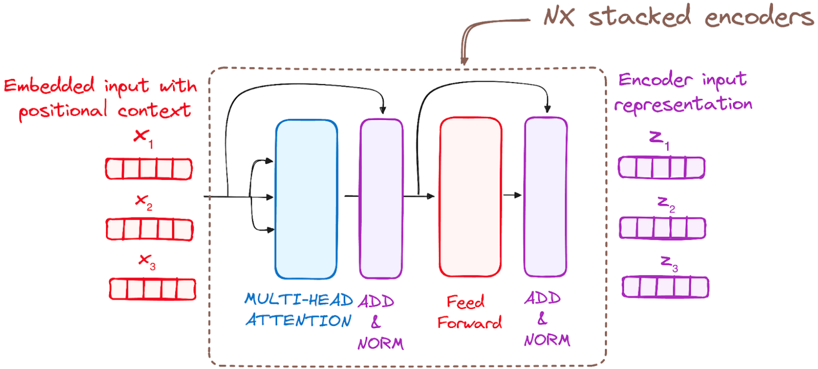
CLIP pre-training framework



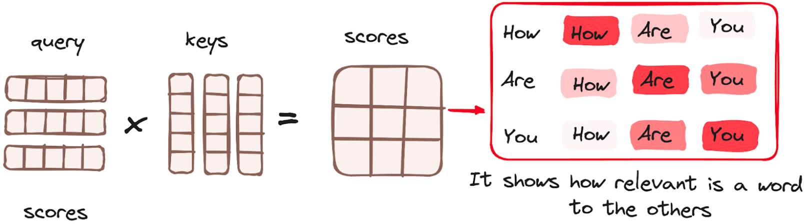
CLIP training code

```
# image_encoder - ResNet or Vision Transformer
# text_encoder - CBOW or Text Transformer
# I[n, h, w, c] - minibatch of aligned images
# T[n, l] - minibatch of aligned texts
# W_i[d_i, d_e] - learned proj of image to embed
# W_t[d_t, d_e] - learned proj of text to embed
# t - learned temperature parameter
# extract feature representations of each modality
I_f = image_encoder(I) #[n, d_i]
T_f = text_encoder(T) #[n, d_t]
# joint multimodal embedding [n, d_e]
I_e = l2_normalize(np.dot(I_f, W_i), axis=1)
T_e = l2_normalize(np.dot(T_f, W_t), axis=1)
# scaled pairwise cosine similarities [n, n]
logits = np.dot(I_e, T_e.T) * np.exp(t)
# symmetric loss function
labels = np.arange(n)
loss_i = cross_entropy_loss(logits, labels, axis=0)
loss_t = cross_entropy_loss(logits, labels, axis=1)
loss = (loss_i + loss_t)/2
```

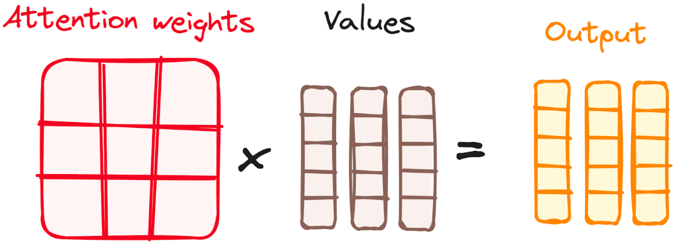
More Details about Encoder



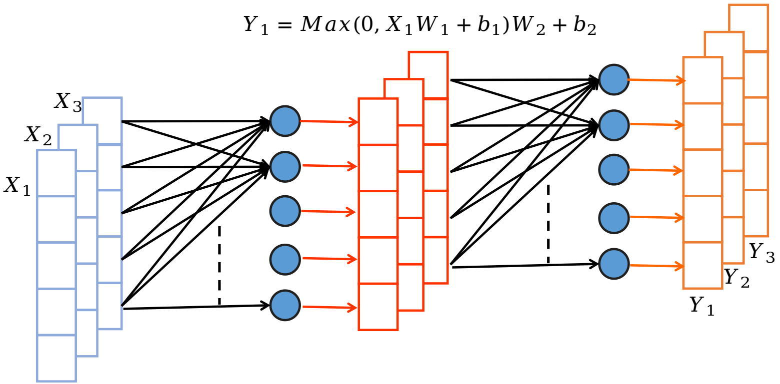
Self Attentions (1)



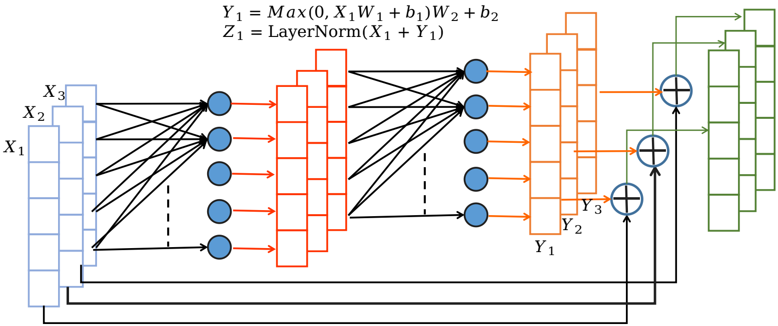
Self Attentions (2)



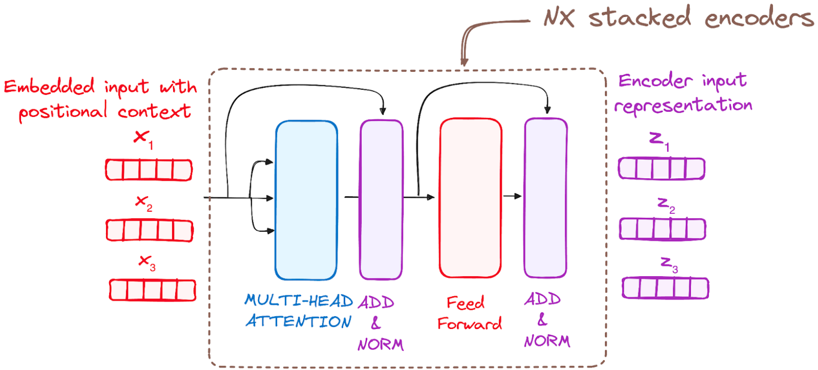
Feed Foward Layers (1)



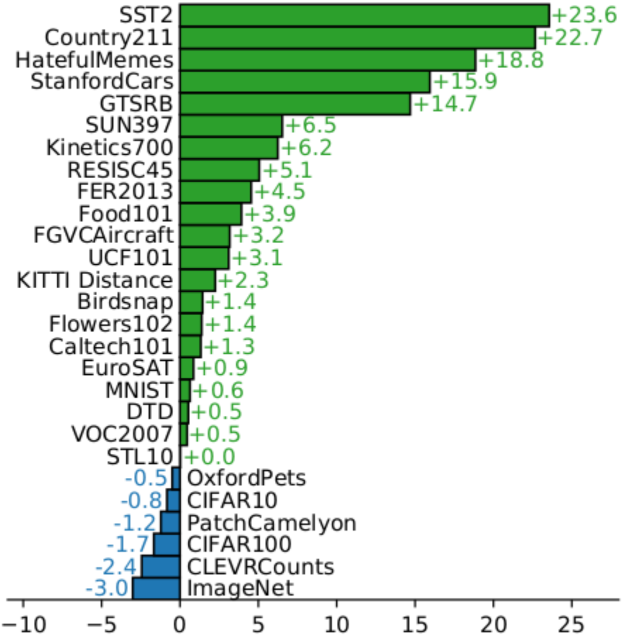
Feed Foward Layers (2)



Review about Encoder



- There is no decoder for the transformer used in CLIP
- The output vectors are either cancatenated or merged into one by sum-pooling/average-pooling/max-pooling


SST2 +23.6
Country211 +22.7
HatefulMemes +18.8
StanfordCars +15.9
GTSRB +14.7
SUN397 +6.5
Kinetics700 +6.2
RESISC45 +5.1
FER2013 +4.5
Food101 +3.9
FGVCAircraft +3.2
UCF101 +3.1
KITTI Distance +2.3
Birdsnap +1.4
Flowers102 +1.4
Caltech101 +1.3
EuroSAT +0.9
MNIST +0.6
DTD +0.5
VOC2007 +0.5
STL10 +0.0
-0.5 OxfordPets
-0.8 CIFAR10
-1.2 PatchCamelyon
-1.7 CIFAR100
-2.4 CLEVRCounts
-3.0 ImageNet
−10
−5
0
5
10
15
20
25

Outline

1. Visualizing and Understanding Convolutional Networks, Matthew D. Zeiler and Rob Fergus, ECCV 2014
2. Deeply Activated Salient Region for Instance Search, ACM TOMM, Hui-Chu Xiao, Wan-Lei Zhao, et. al., 2022
3. Learning Transferable Visual Models From Natural Language Supervision, Alec Radford, Jong Wook Kim, et. al., ICML, 2021

Q & A