Reflection on participation at FB image similarity challenge

BY:

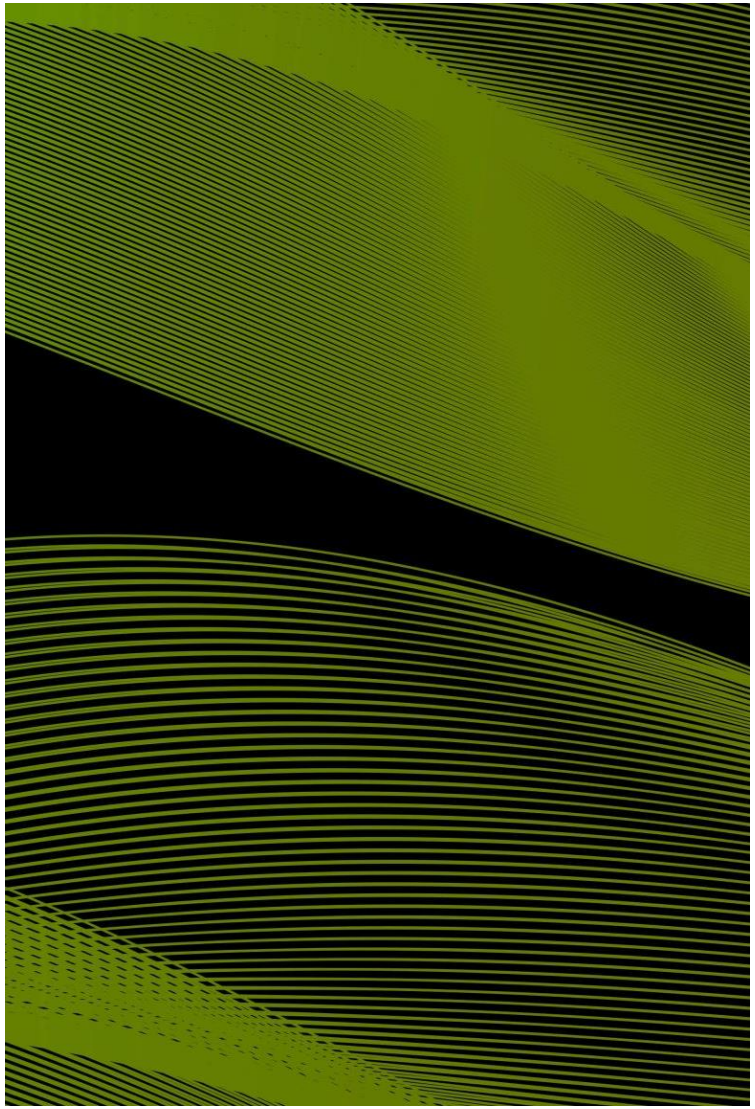
AVISHEK PARAJULI

Presentation objectives

Discuss problems and challenges faced

Thought process and Lessons learned

Discuss winning solutions and ideas



Outline

Competition Description

Base solution, workflow

Supervised Deep Learning method

- Prob: Large data handling
- Prob: Efficient comparison
- Prob: validating results

Lessons summary

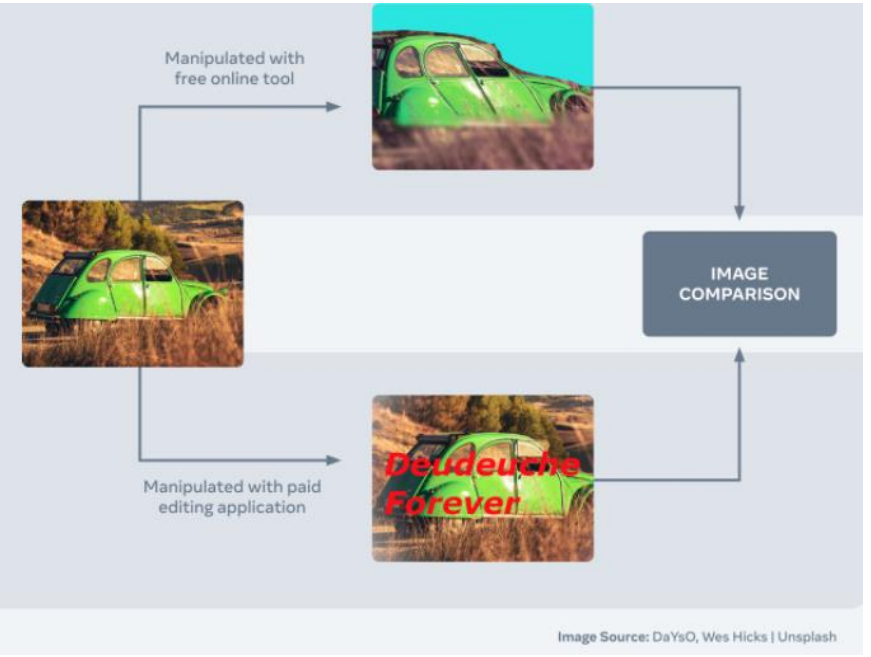
Winning solutions

References

- Images here are not my copyright

- Copyright are for the respective authors, web, Google

Competition description



Task: The task is to devise an algorithm to identify copy detection between images at large scale(million images).

- Learn a metric to identify similar images and dissimilar images

<u>Competition image archive:</u>

- 1 million reference images
- 50K query images, a subset of which were derived from the reference images
- 1 million training images, statistically similar to but distinct from the reference archive

Quote from competition website “determine for each query image whether it originated from one of the reference images and assign a confidence score indicating its similarity to the candidate reference image.”

Why participate?

- Keep upto date with current state-of-the-art
  - SVM->AlexNet ->GoogLeNet->ResNet ->Attention->ViT (VisionTransformers)
- Big learning from opensource projects
  - Learning is fun
  - Confidence booster
- Getting bored with Work at SICK
  - Projects aren't challenging, neither cool

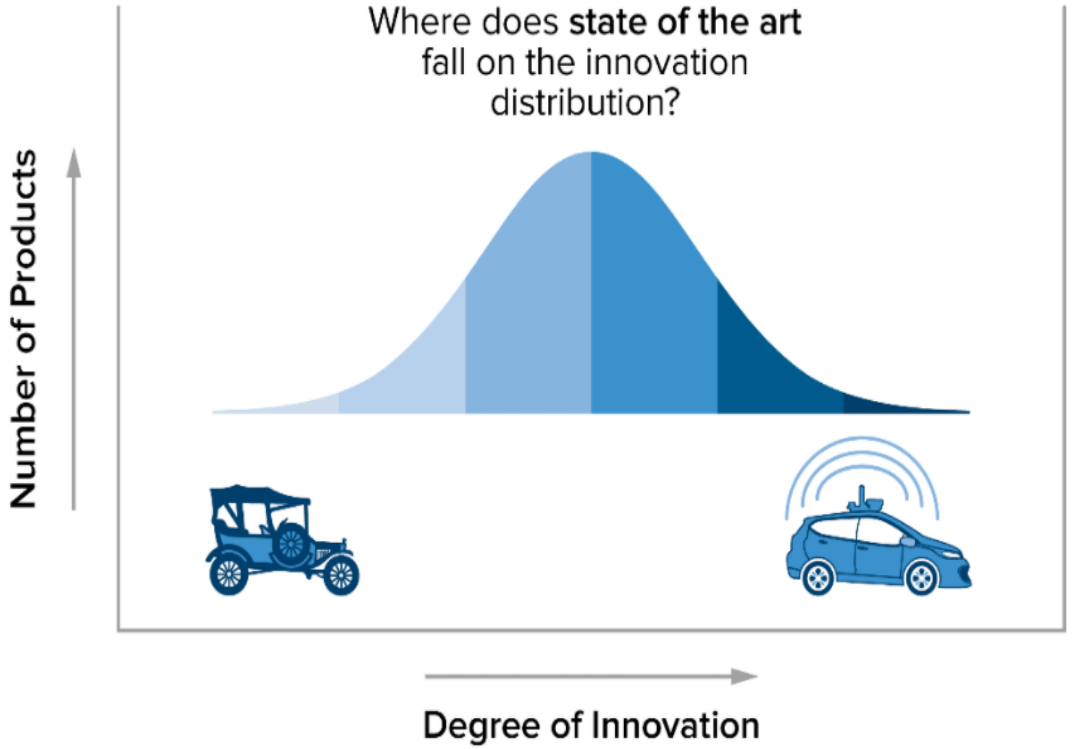
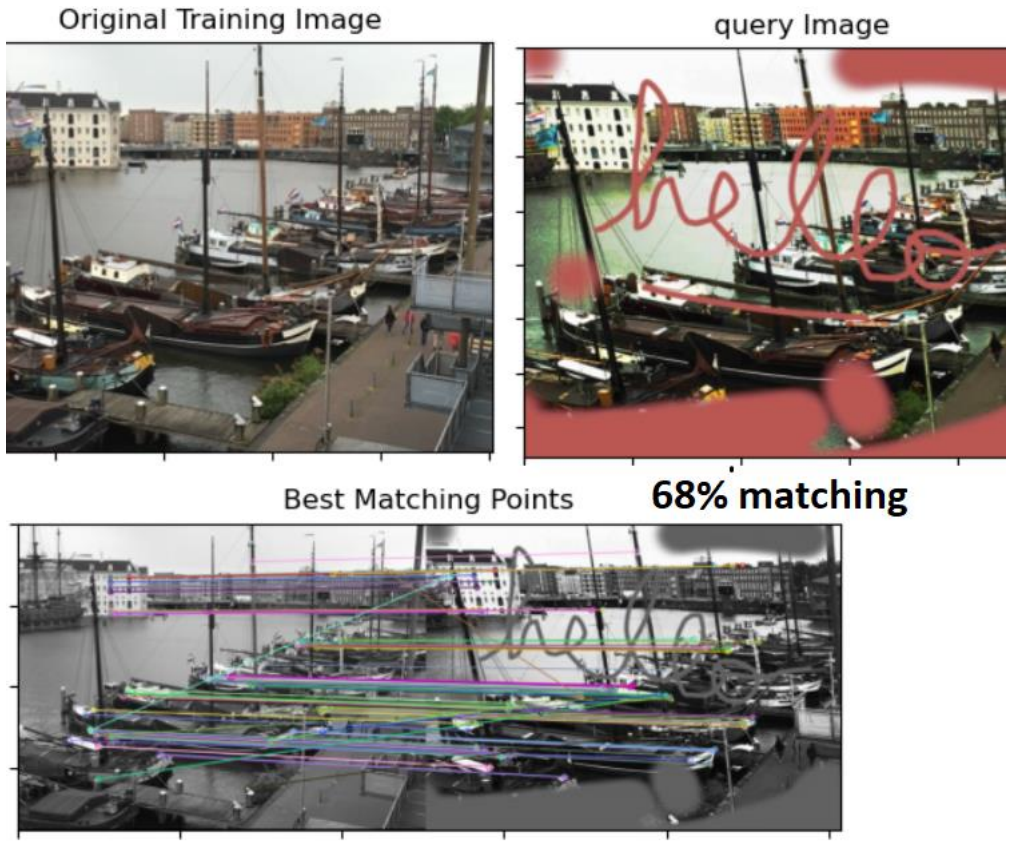


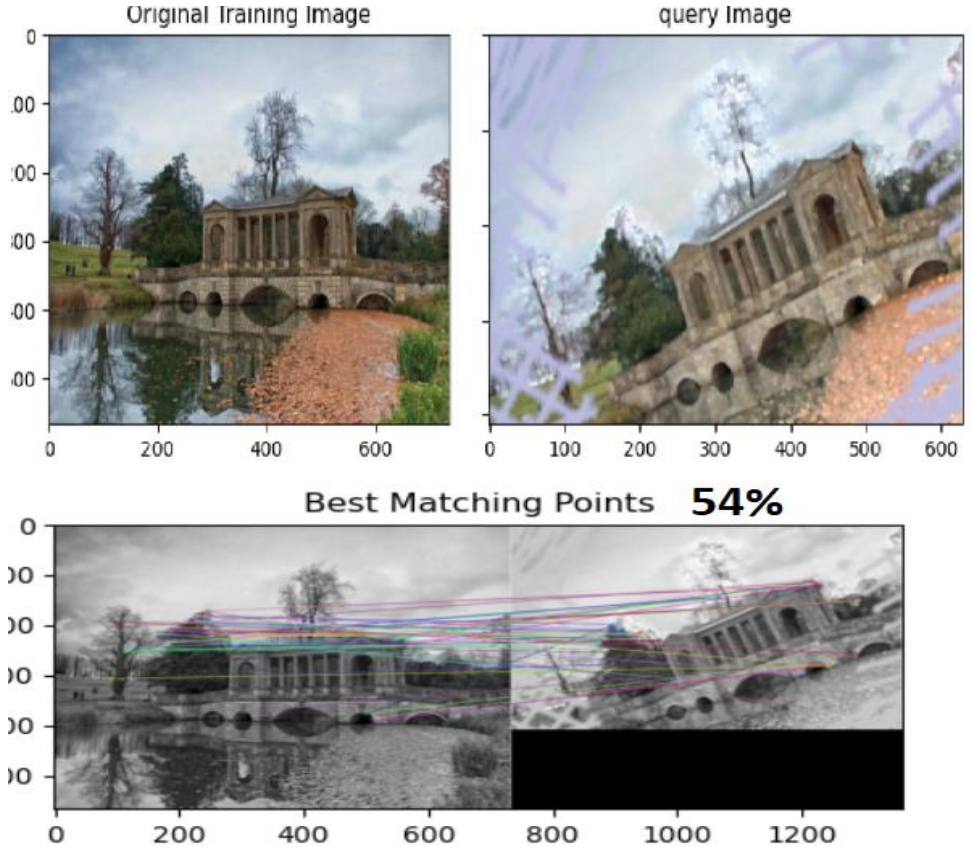
Image Credit: Web

Base Solution

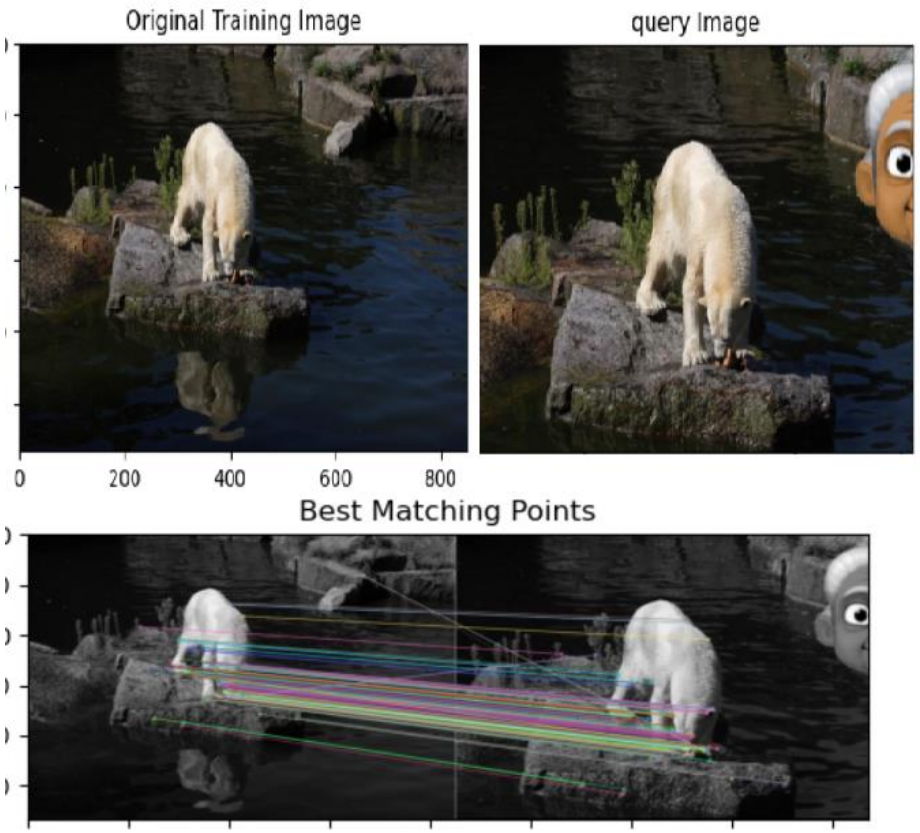
USING CLASSICAL COMPUTER VISION; ORB FEATURE AND MATCHING

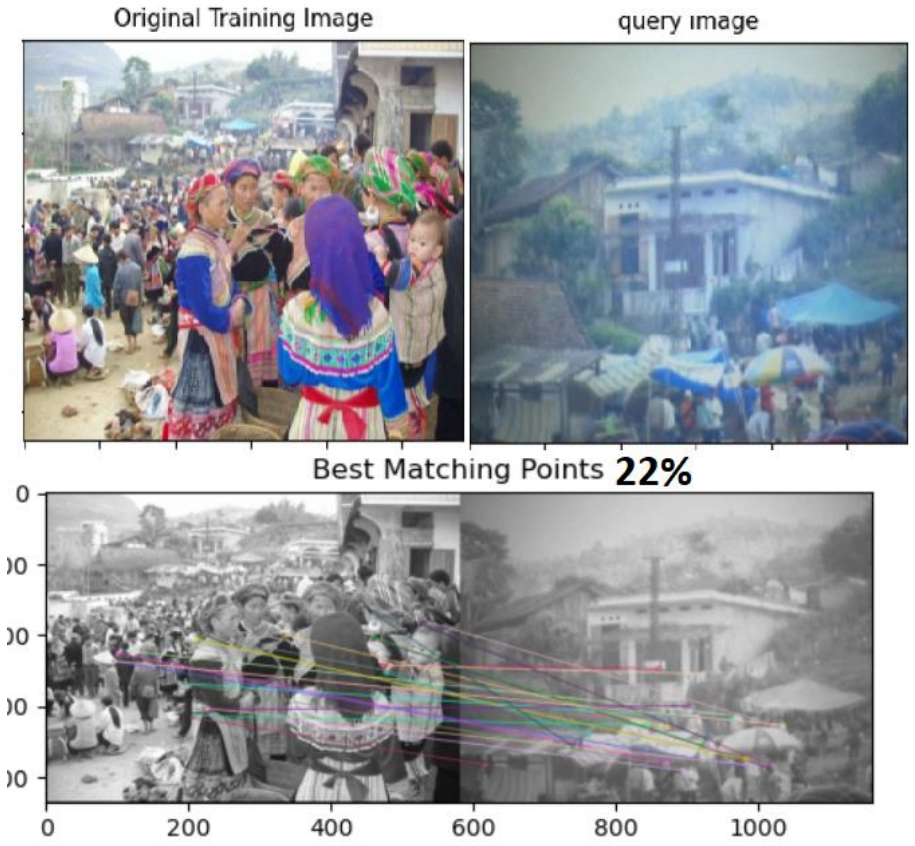
ORB:
Fusion of FAST keypoint detector and BRIEF descriptor + Rotation invariance





ORB: more results…





Approach0: ORB summary

- Slow for large scale

- Too many parameters to tune
  - No of pyramid level(8 used), scale factor
  - Descriptor distance , Threshold

- Too many false positives

- Evaluation
  - Accuracy of 50-60% matching correctly but too many false positives(5k query vs 5k reference)
  - Score of 0.16 from these(better than GIST benchmark from competition)
  - FUTURE enhancement: use RANSAC to filter bad matches; get large features;
  - Maybe combine ORB with Deep Learning models

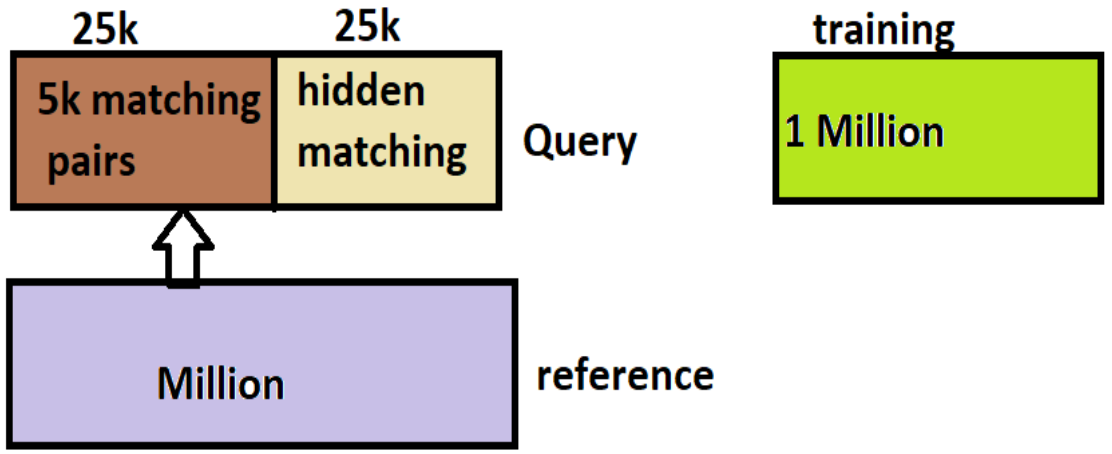
Approach1 introduction

Hardware:

- GTX 1060(6GB, 1280 CUDA cores)
- Moved to Google Colab Pro as training was slow

Approach:

- Review existing literature for metric learning(face recognition)
  - Triplet loss is better
  - Model: pretrained **Resnet50** + fine tuning
  - TensorFlow 2.0
- Dataset
  - Use matching pairs(5k) for training
  - Add augumentaion
  - Evaluate later on 25k Q vs 1Mil Reference



Model Architecture

```python
def complete_model(base_model, alpha =0.2):
    # Create the complete model with three
    # embedding models and minimize the loss
    # between their output embeddings
    input_1 = Input((imsize, imsize, 3))
    input_2 = Input((imsize, imsize, 3))
    input_3 = Input((imsize, imsize, 3))

    A = base_model(input_1)
    P = base_model(input_2)
    N = base_model(input_3)

    loss = Lambda(triplet_loss)([A, P, N])
    model = Model(inputs=[input_1, input_2, input_3], outputs=loss)
    model.compile(loss=identity_loss, optimizer=Adam(LR))
    return model
```

```python
def triplet_loss(x, alpha = 0.2):
    # Triplet Loss function.
    anchor,positive,negative = x
    # distance between the anchor and the positive
    pos_dist = K.sum(K.square(anchor-positive),axis=1)
    # distance between the anchor and the negative
    neg_dist = K.sum(K.square(anchor-negative),axis=1)
    # compute loss
    basic_loss = pos_dist-neg_dist+alpha
    loss = K.maximum(basic_loss,0.0)
    return loss

def my_norm(ip):
    return K.l2_normalize(ip, axis=-1)
```

Approach1: results

- Google Colab training; 5 mins per epochs
  - 25k query, 50K reference images(5k matching +20k distractor6) image as jpg files
  - Embeddings Comparison metric: Distance metric,
  - Training loss = 0; validation loss =1e-3<-->1e-4
  - Accuracy: 31% before training(pretrained resnet50; transfer learning)
  - Accuracy: 87.5% top-2 after training(5k Q vs 5k Ref)
  - Accuracy: 53% top-1 after training(5k Q vs 5k Ref)
  - Accuracy: 33% top-1 after training(5k Q vs 50k Ref); 20% for 5k vs 100K Ref

PROB1: Many False Positives; Accuracy drop when comparing query against large set

PROB2: Comparison of 5k Against 50K takes long time(5k vs 50K comparison)
  - Imagine 50k Q against 1 million Ref images

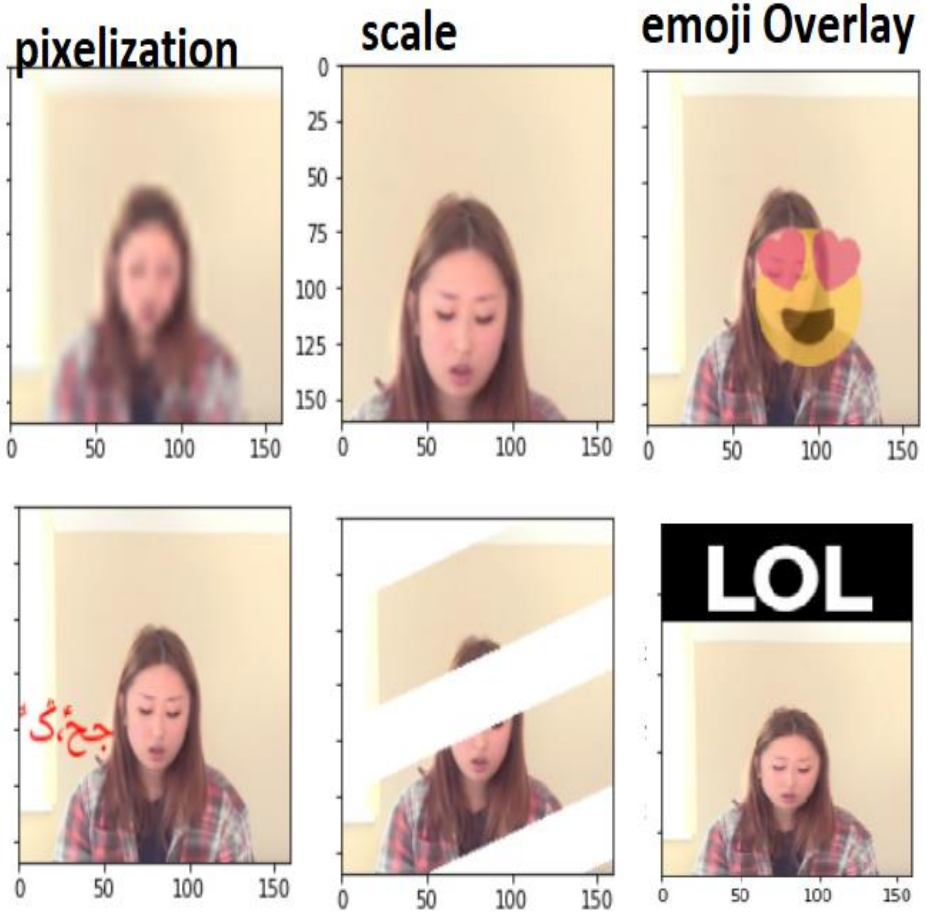
PROB3: Can’t load all images (1 mil ref images, query image) as jpg files in colab

Prob1: Many False positives;

- How to address?
  - Training loss = 0; validation loss =1e-3

- Initial thoughts →overfit
  - Add more augmentation; and train more
  - Train on part of training image (more data to train)
  - Lowered from resnet50 to resnet18

-Results:
  - Augmented data had only Slight improvement
  - Lowering also didn't help much
  - High augmentation intensity in start made worse
    - Future Action: progressively add augmentation intensity



Albumentations
Augly(facebook)

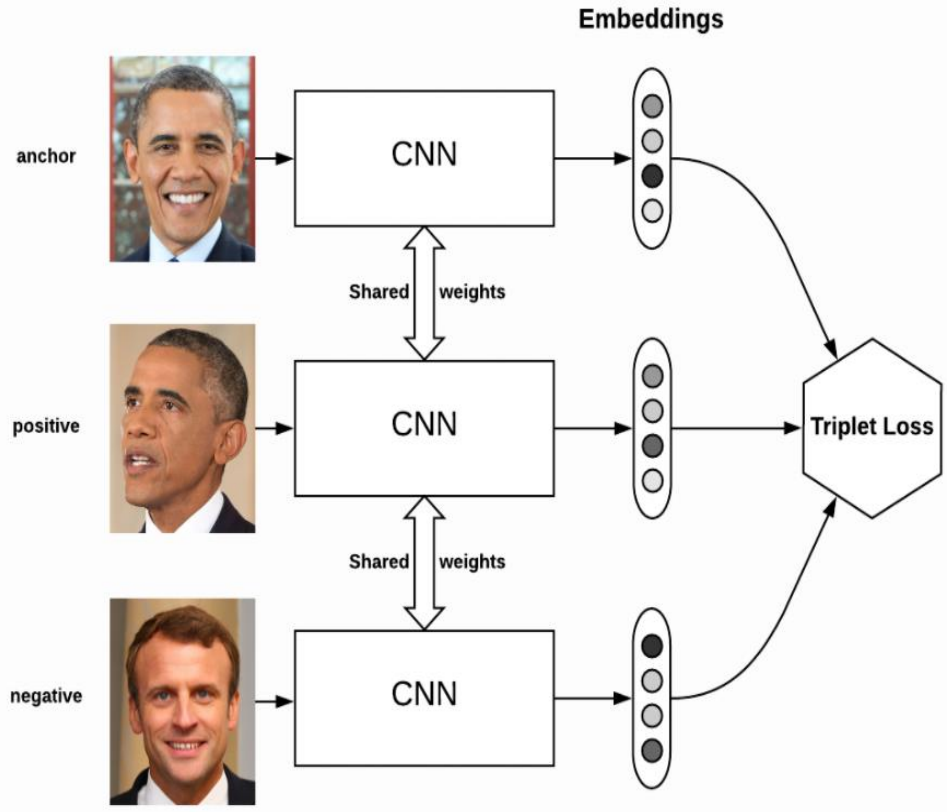
Blur, HorzFlip,
RandomBrightnes
Contrast, scale,
Shift, GaussNoise,
ColorJitter
Flip(horz, vert),
RandomCrop,
Rotate,

Triplet Loss: Crash course

Triplet Loss:

- Same image have closer embeddings
- Different images have far embeddings

$$\mathcal{L} = max(d(a,p) - d(a,n) + margin, 0)$$



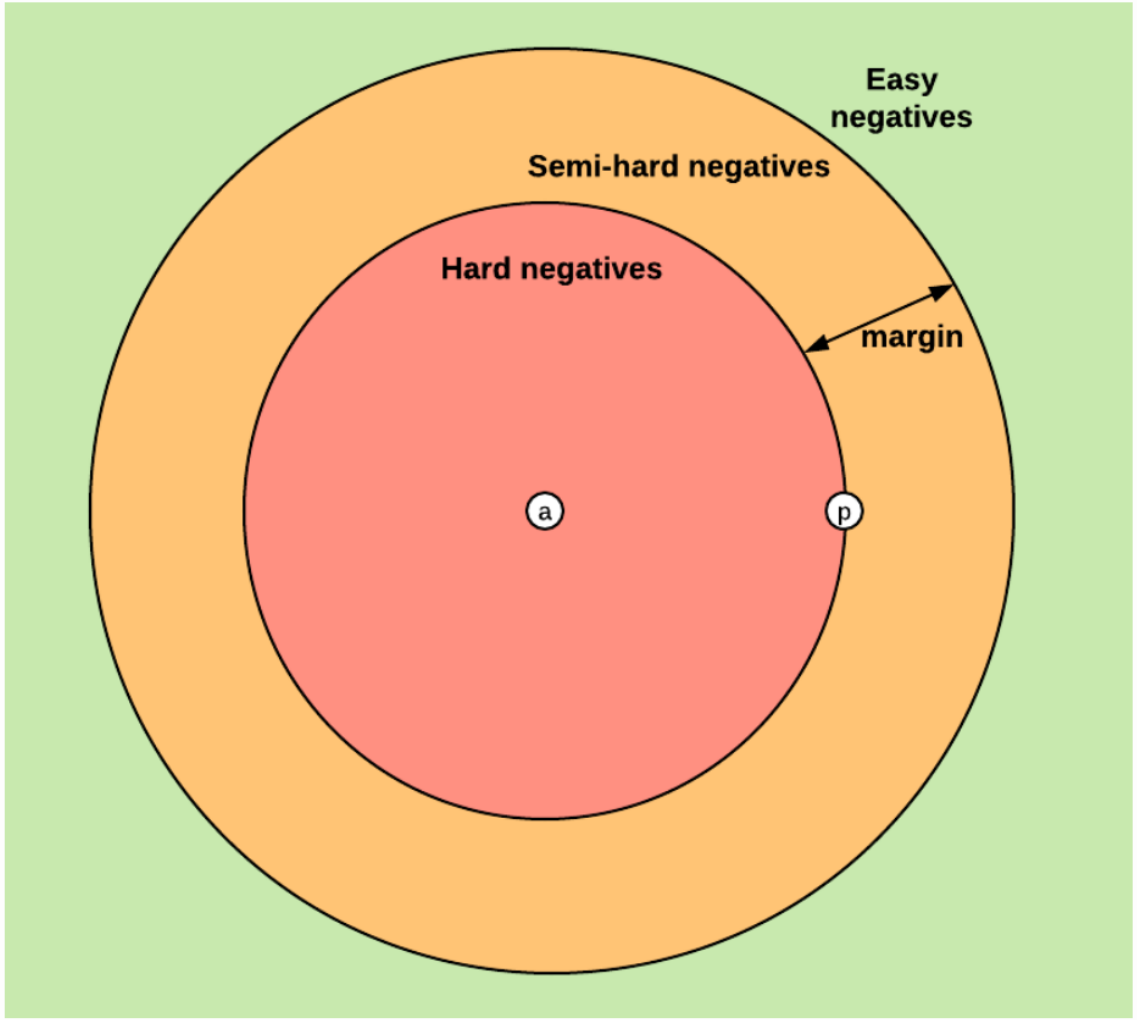
Case1:
d(a,p) = 0.5; d(a,n) = 1.2, margin =0.2
Loss= max(0.5 -1.2 +0.2 ,0) ~ 0

Case2: (margin=0.2)
d(a,p) = 0.5; d(a,n) = 0.6,
Loss= max(0.5 -0.6 +0.2 ,0) ~ 0.1

Improved loss

$$L1 = max(d(a,p) - d(a,n) + margin, 0)$$
$$L1 = max(d(a,p) - d(p,n) + 0.5 * margin, 0)$$

L = L1 +L2



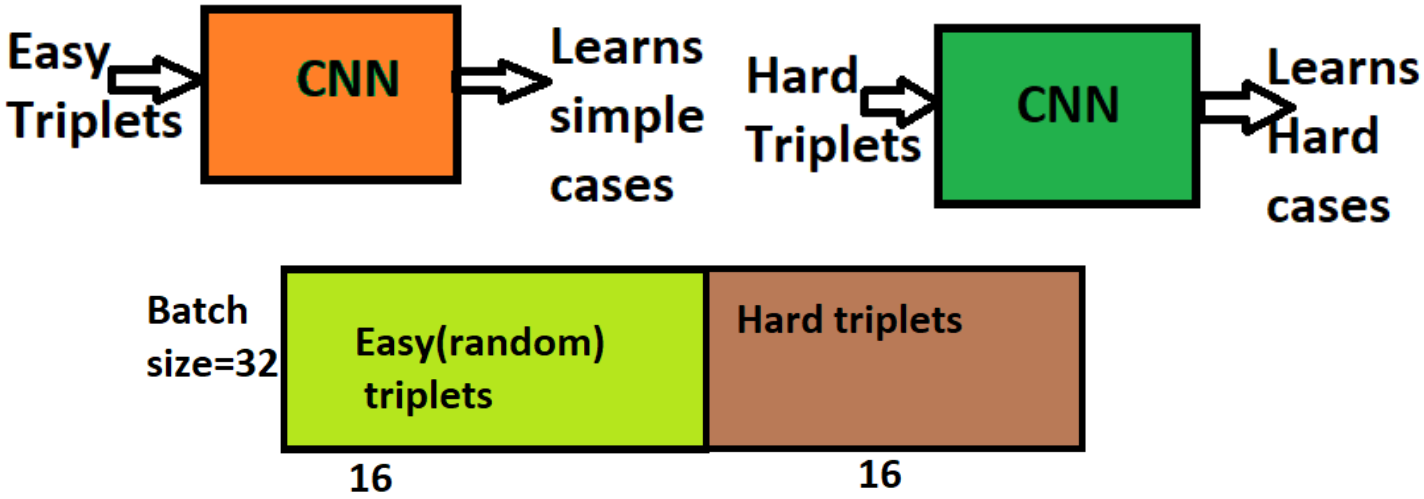
Triplet Loss: crash course cont'd..

- Visualizing Embeddings can help(t-SNE)
  - Easy triplets: d(a, p)+margin < d(a, n)
  - Hard triplets: d(a, n)< d(a, p)
  - Semi-hard triplets: d(a, p)< d(a, n)<d(a, p)+margin
- How to find these Hard Negatives?
- Online triplet mining(on the fly)
  - For a given anchor, find the triplet pair that are hard;
  - Given Batch B, there are maximum of B^3 embeddings
  - From batch of 100, pick the 16 most difficult ones
- Offline triplet mining(create list from evaulation)

Triplet Loss cont'd..

Recommendation:

- Start Semi-Hard
- Then train hard triplets



PROB4: Model forgets:

Quick Train results

- Accuracy: 73% top-1 after training(5k Q vs 5k Ref)
- Accuracy: 70% top-1 after training(5k Q vs 50k Ref);

PROB 5: Training too Slow for online hard mining

PROB6: Embedding generation for 1mil Ref and 50K Q took 12+ hrs

Problems and solutions

PROB 5: Training too Slow for online hard(choose 16 hard from batch of 200)

- Model training bottleneck is File I/O; (accessing training images from large folder)
- SSD slightly helps
- Solution5: Hdf5 file format
  - Pack files as single file in binary format(Single File I/O)
  - Can choose contiguous memory for faster access
  - For large very large file chunking feature is available(chunk size=100)
  - Cut training per epoch (25k Q and 55kRef)for simple case in colab from 3-5 min to 45 sec

PROB6: Embedding generation for 1mil Ref and 50K Q took 12+ hrs

- Image saved in Hdf5 file helped; can save Embeddings in 100 mins at home pc
- Colab was much faster at 45mins

PROB3: Can’t load all images (1 mil ref images, query image) as jpg files in colab

- 100GB Google1; Dropbox; Hdf5 file helped(zip crashed for ziping)

Problem and solution....

PROB2: Comparison of 5k Against 50K takes long time(5k vs 50K comparison)

- Score submission: 50k Q against 1 million Ref images
- For each query embedding, we need to find distance wrt 1mil Ref
- Choose smallest distance, dmin;
- Score= -dmin
- Insane amount of time and memory consumption

Solution:

- Use Faiss Library; fast metric comparison

- Can generate submission file in <2-3mins from embeddings

Evaluation metric

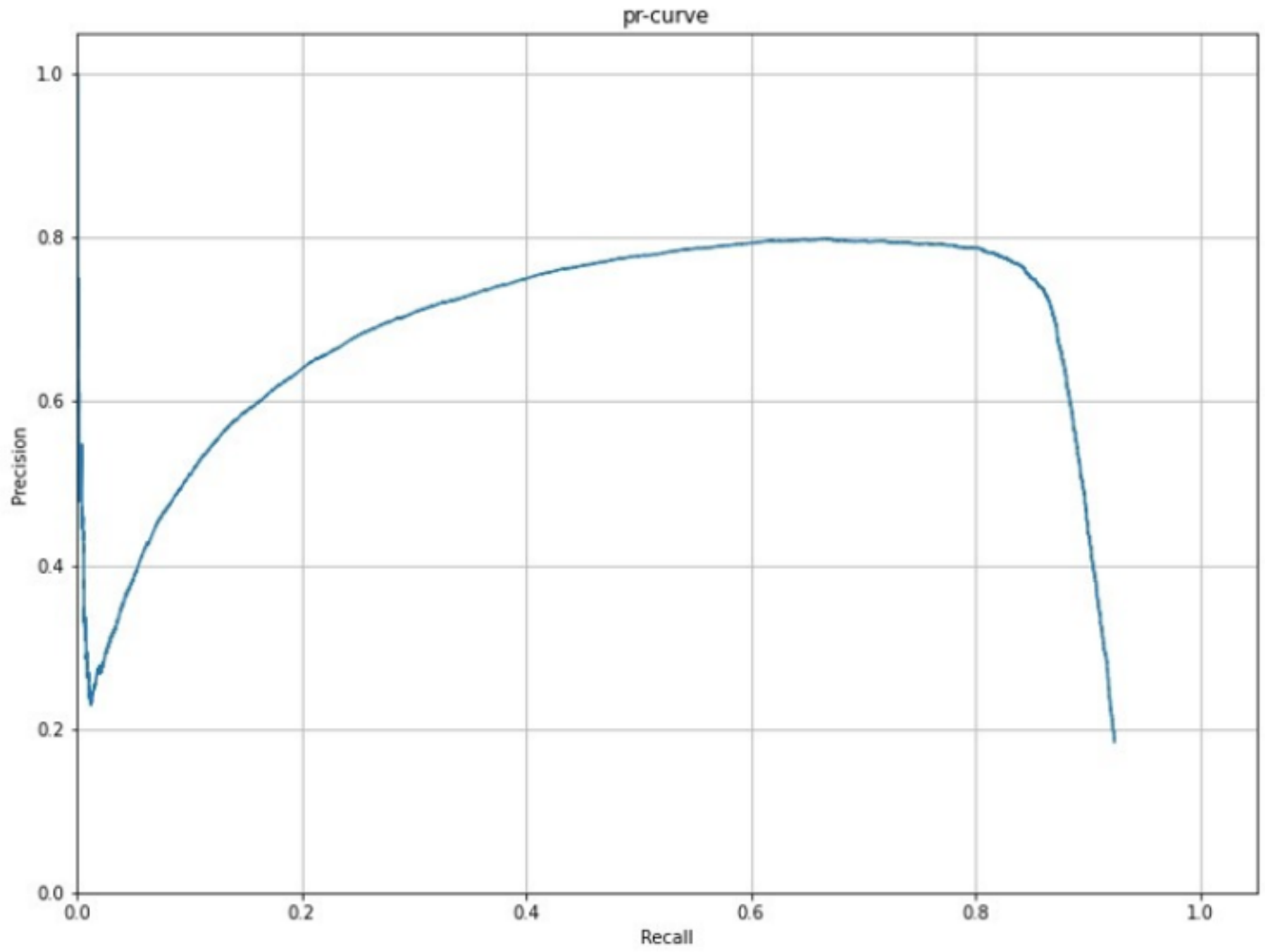
- Area under the curve(Precision vs Recall)

- Score: Micro –AveragePrecision(uAP) with ordered precision

- Accuracy=89%=> uAP=0.35

- Accuracy= 80%; uAP = 0.55

- FalsePositives with high confidence hurts most

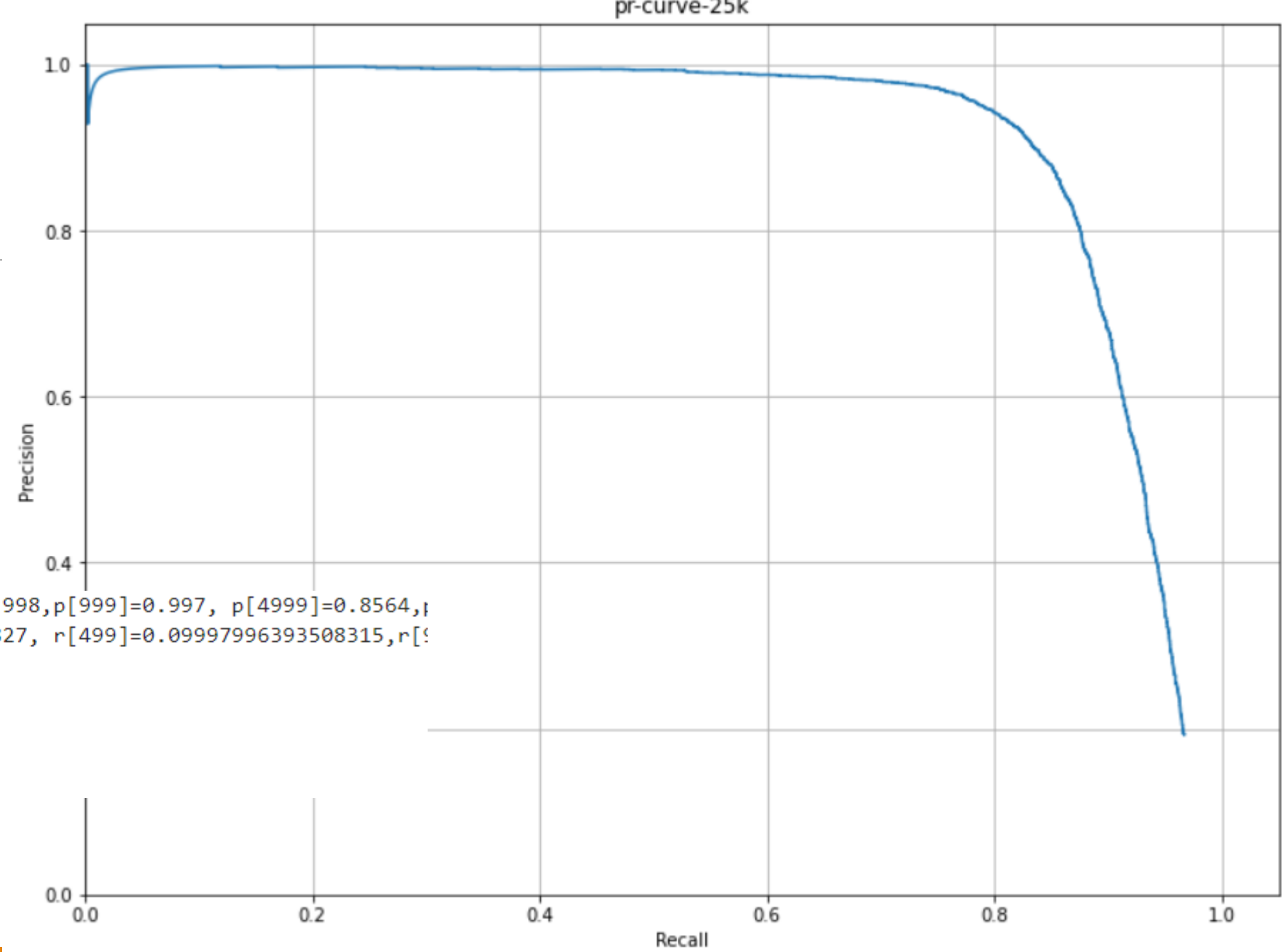


Evaluation

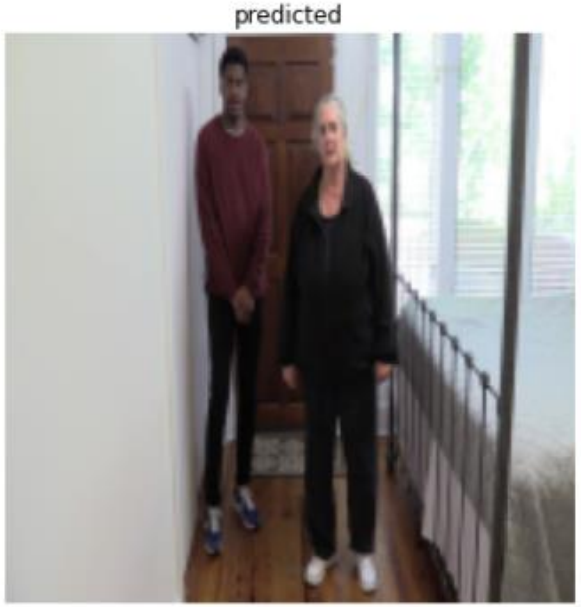
- Multiple tricks
  - Offline training+ online
  - Loss enhancement
  - Fine tuning at 1e-7 rate

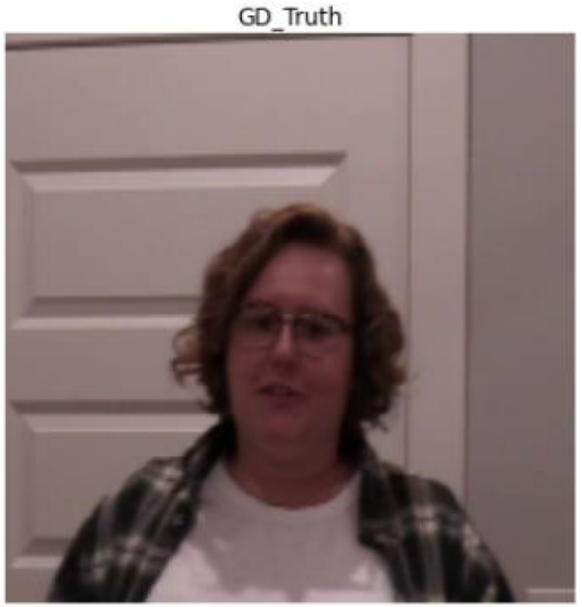
Public: 0.9(overfit)
Official: 0.46

```
precision at p[99]=0.99, p[499]=0.998,p[999]=0.997, p[4999]=0.8564,
recall at r[99]=0.019835704267681827, r[499]=0.09997996393508315,r[
Average Precision: 0.90769
Recall at P90    : 0.83510
Threshold at P90 : -0.177394
Recall at rank 1:  0.96694
Recall at rank 10: 0.96694
```

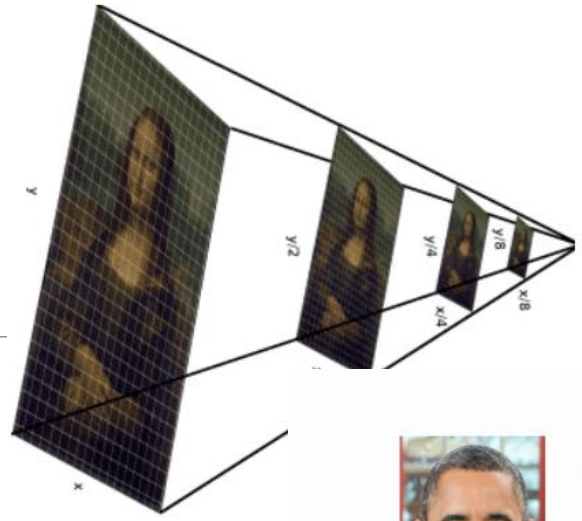


False positives

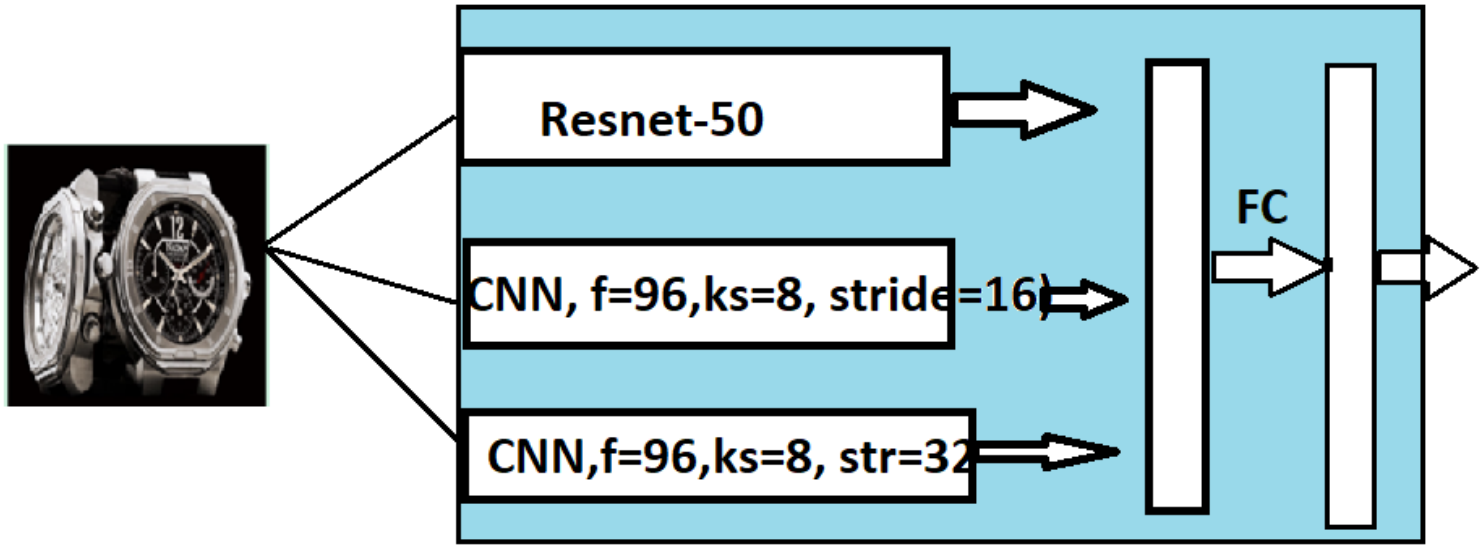


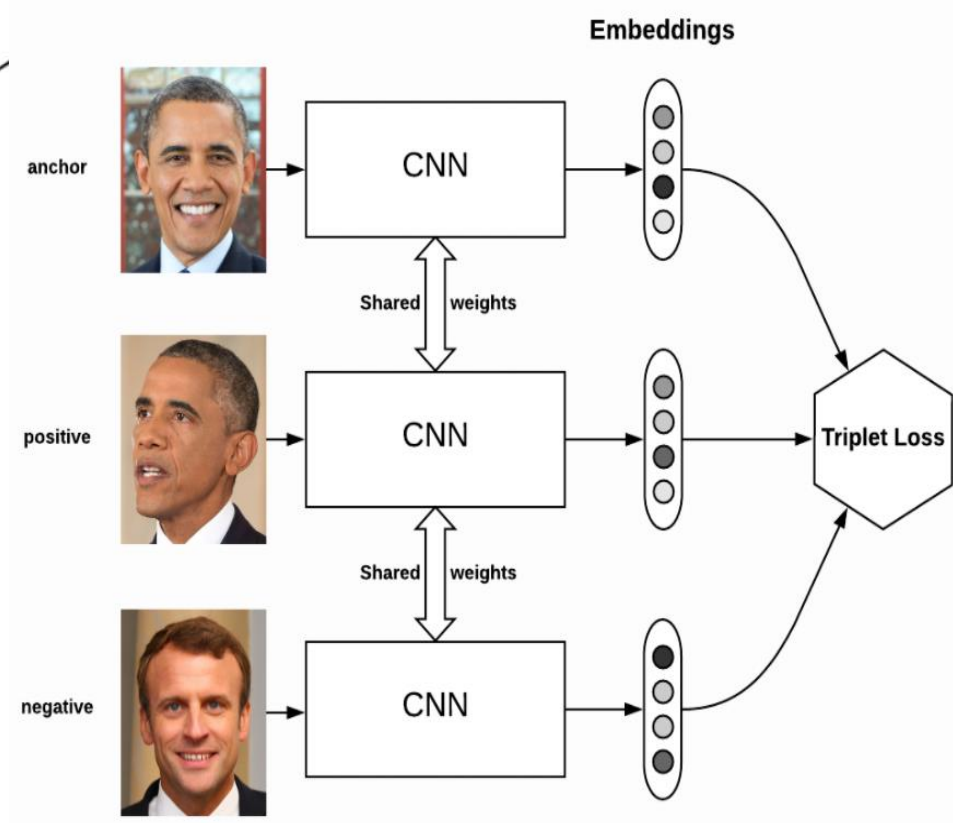


Multi-scale processing



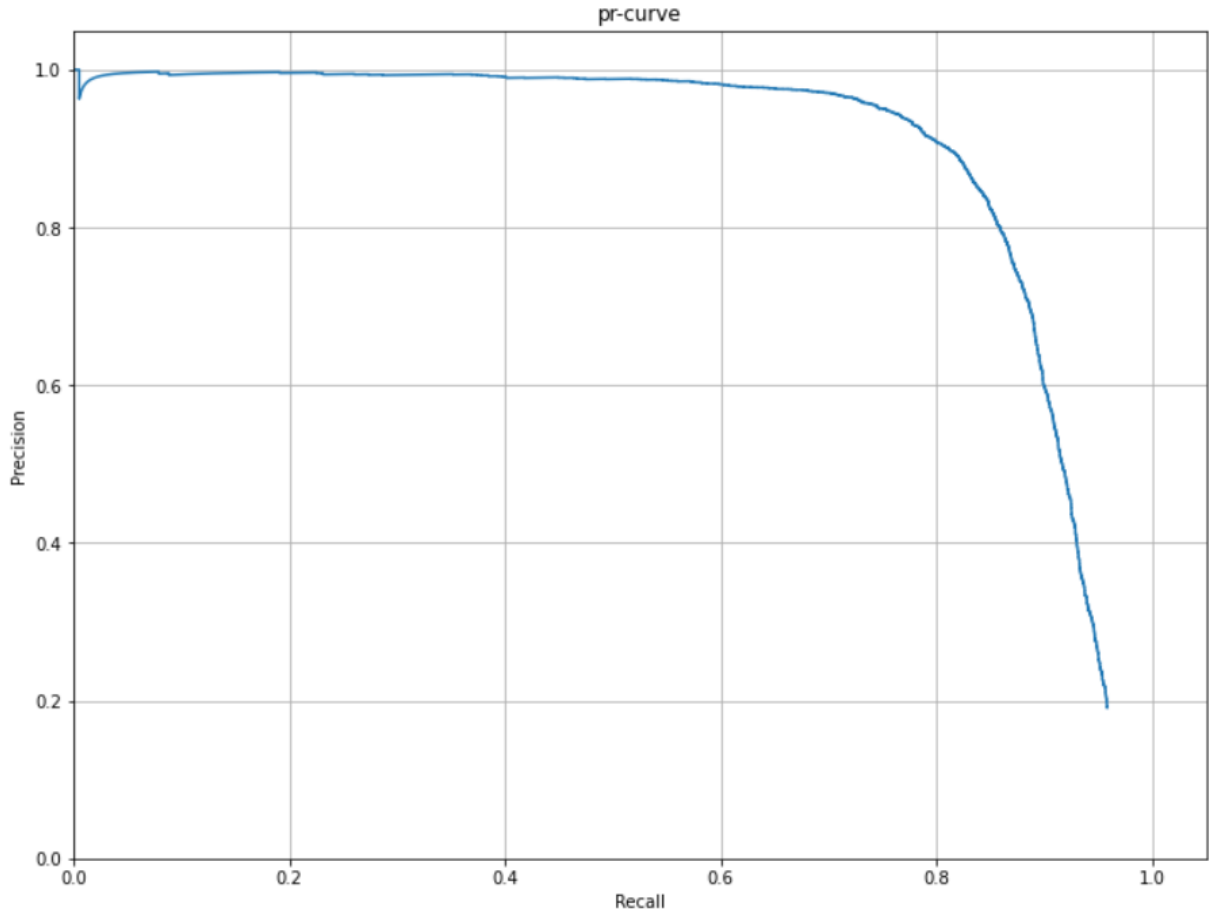
- Classical computer vision
  - Image pyramid
  - Combine ORB/SIFT for these multi scale
- Deeprank model:
  - 3 Parallel branch
  - rapid strides,down sampling in 2nd and 3 branches





Results

- 0.81(public dataset)
- Colab crashed while computing the submission
  - Large embedding dims



Analyzing false matches

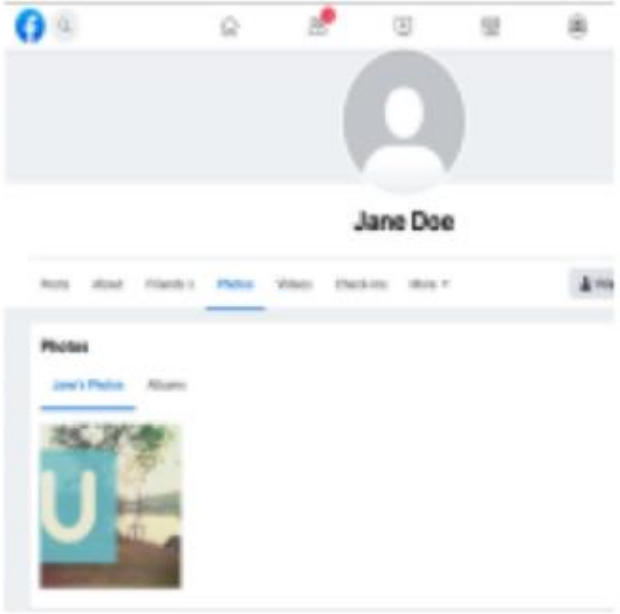
query

predicted

GD_Truth

Analyzing false matches

query

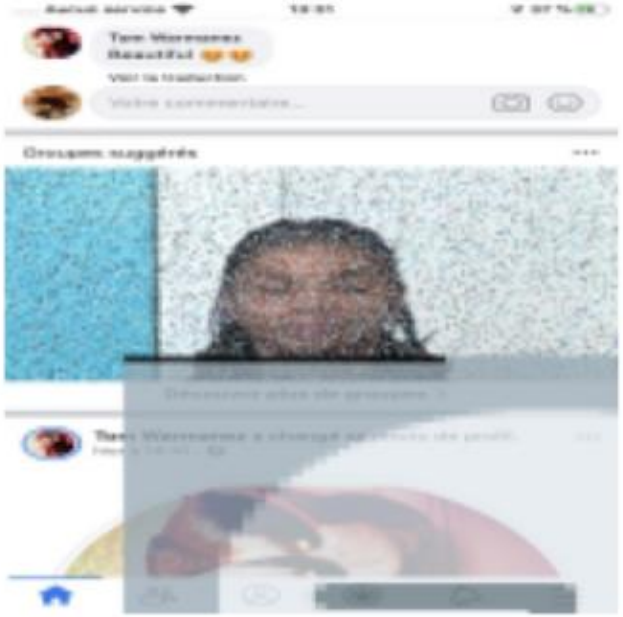


predicted

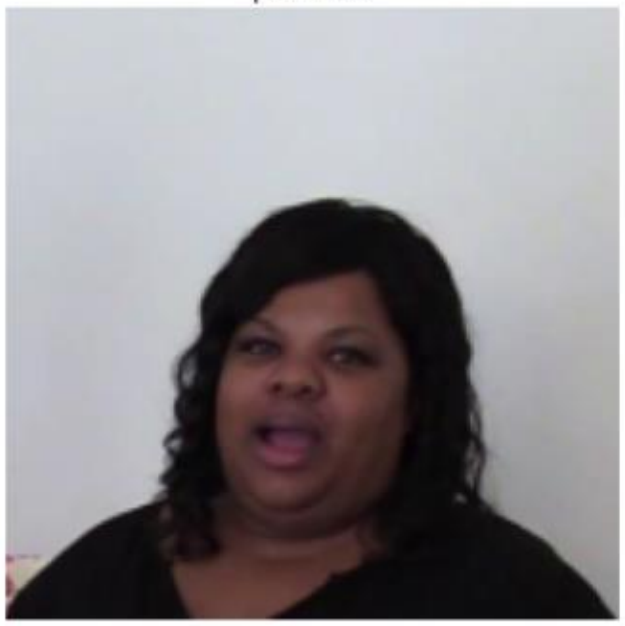
GD_Truth

Analyzing false matches

query



predicted



GD_Truth

Winning Solutions

Common themes

Winning solutions varied in their approaches, but some common themes were apparent.

- **Self-supervision:** All winning solutions took a self-supervised approach to learning (SSL), which is perhaps unsurprising given that many SSL techniques are in effect optimizing for a task like copy detection.
- **Diverse image augmentations:** Winning solutions used a battery of image augmentations, and many relied heavily on the new AugLy library from FB AI. In addition to the standard set of image augmentations (scaling, cropping, flipping, etc), winning solutions tended to use augmentations like screenshot, text and emoji overlays that are more prevalent in this competition's data set and social media.
- **Varying image scales and augmentation intensities:** Multiple winning solutions trained or extracted embeddings at several different image scales. Variation in image augmentation intensity, including some strong augmentations, was also a common theme in the top results, and some participants needed to figure out how to progressively add complexity to their model as training progressed.
- **Local-global comparison:** Most winning solutions compared partial and global views of both query and reference images. These approaches were designed to handle common cases such as query images being cropped from reference images, or reference images being overlayed on distractor images.
- **Descriptor normalization:** For the Descriptor track, all winning solutions used some form of post-processing to normalize the final embeddings relative to nearby training image descriptors. Participants reported significant score improvements with these transformations, for which they coined new terms like "descriptor stretching" and "escaping the sphere."

credit: https://www.drivendata.co/blog/image-similarity-winners/

Credit/Ref: [D2LV: A Data-Driven and Local-Verification Approach for Image Copy Detection]

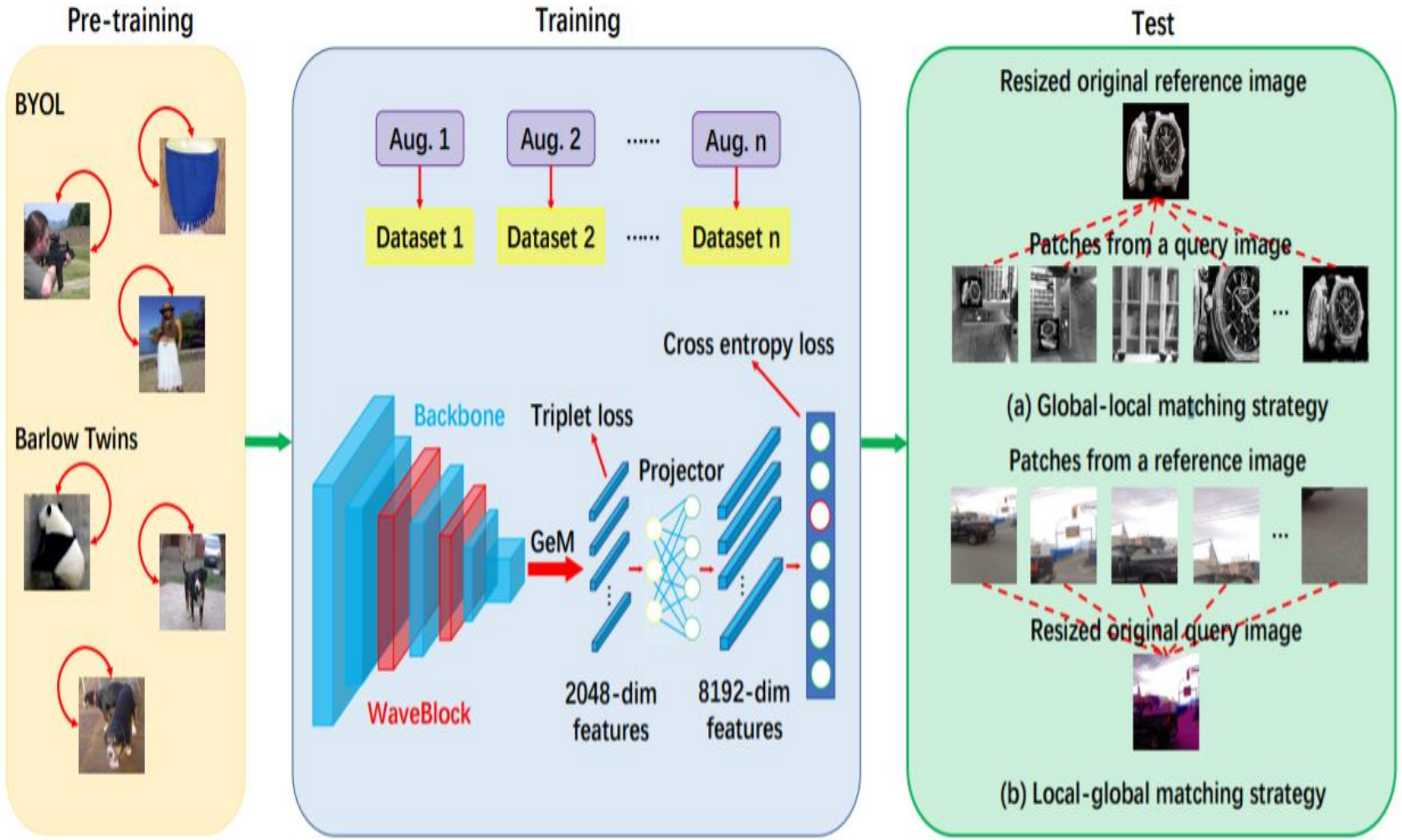
1st place winner

Approach Summary:

1) 3 Pretrained Models on ImageNet
   - ResNet-50 pretrained used from BarlowTwins project
   - ResNet-152 and ResNet-50-IBN (pretraining: 2 wks for R-152, 1 wk forR-50-IBN on 8 V100 GPUs)

2) Generate 11 datasets(100k each), and train 3 models on each; =>33 models

3) Test: multi-scale, multi-model, multi-part testing and ensemble all scores



SSL(Self supervised learning): BYOL

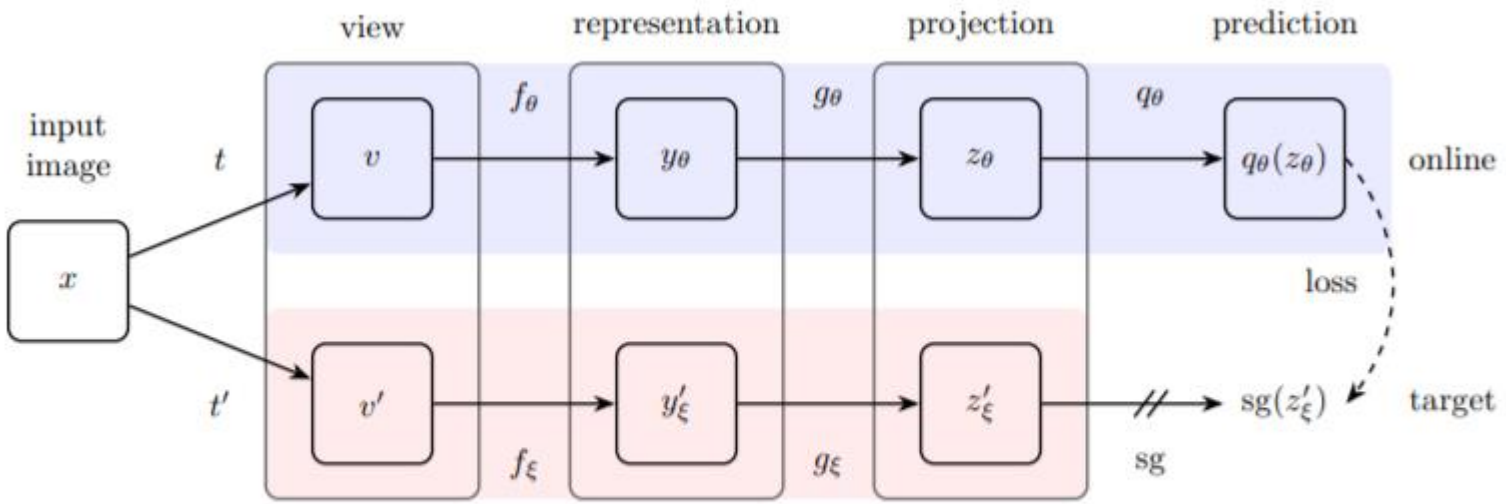
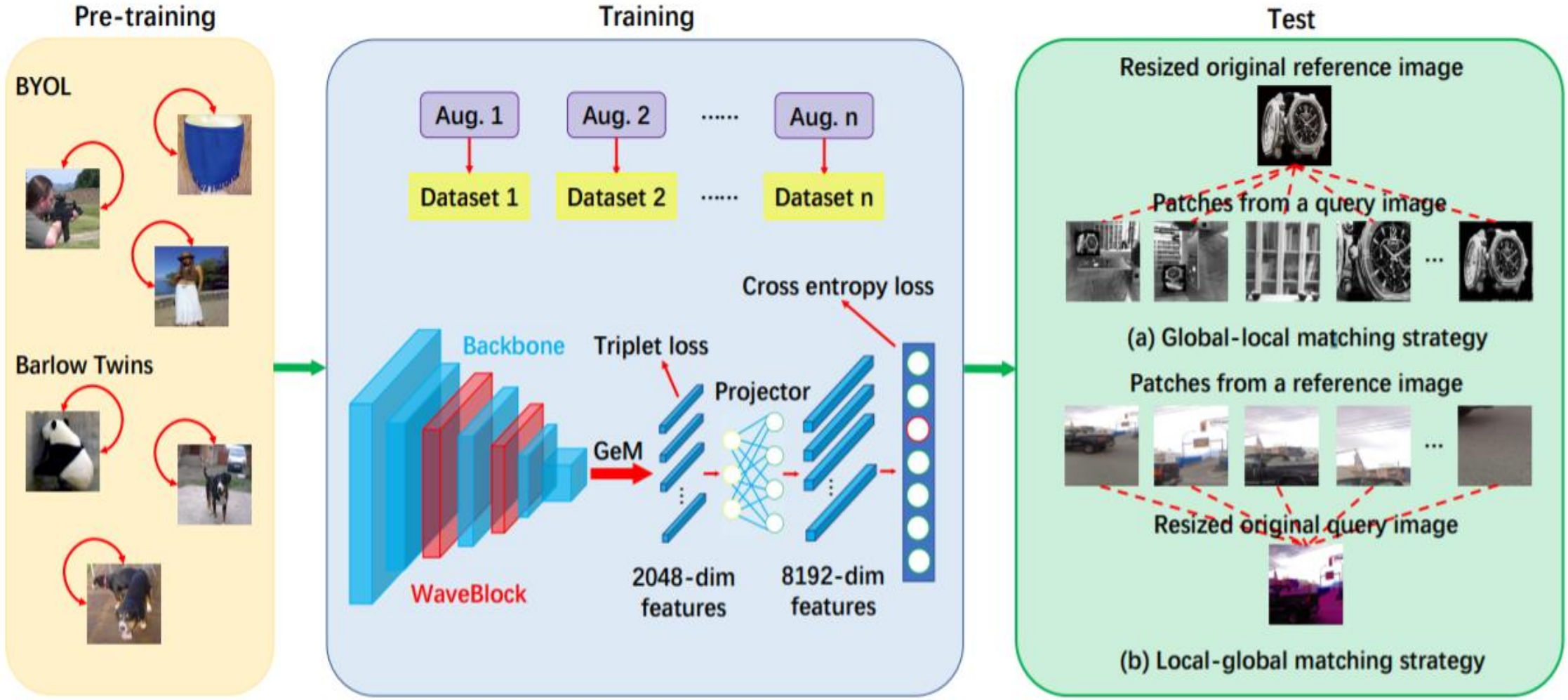


Figure 2: BYOL's architecture. BYOL minimizes a similarity loss between $q_\theta(z_\theta)$ and $\mathrm{sg}(z'_\xi)$, where θ are the trained weights, ξ are an exponential moving average of θ and sg means stop-gradient. At the end of training, everything but f_θ is discarded, and y_θ is used as the image representation.

$$\mathcal{L}_{\theta,\xi} \triangleq \left\| \overline{q_\theta}(z_\theta) - \overline{z}'_\xi \right\|_2^2 =$$

Ref: https://arxiv.org/abs/2006.07733

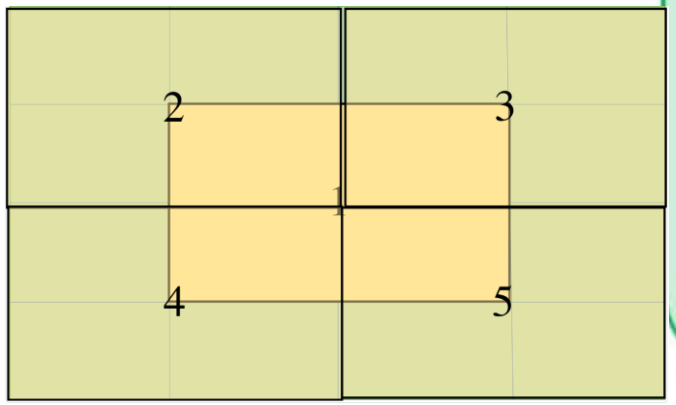
Training and test

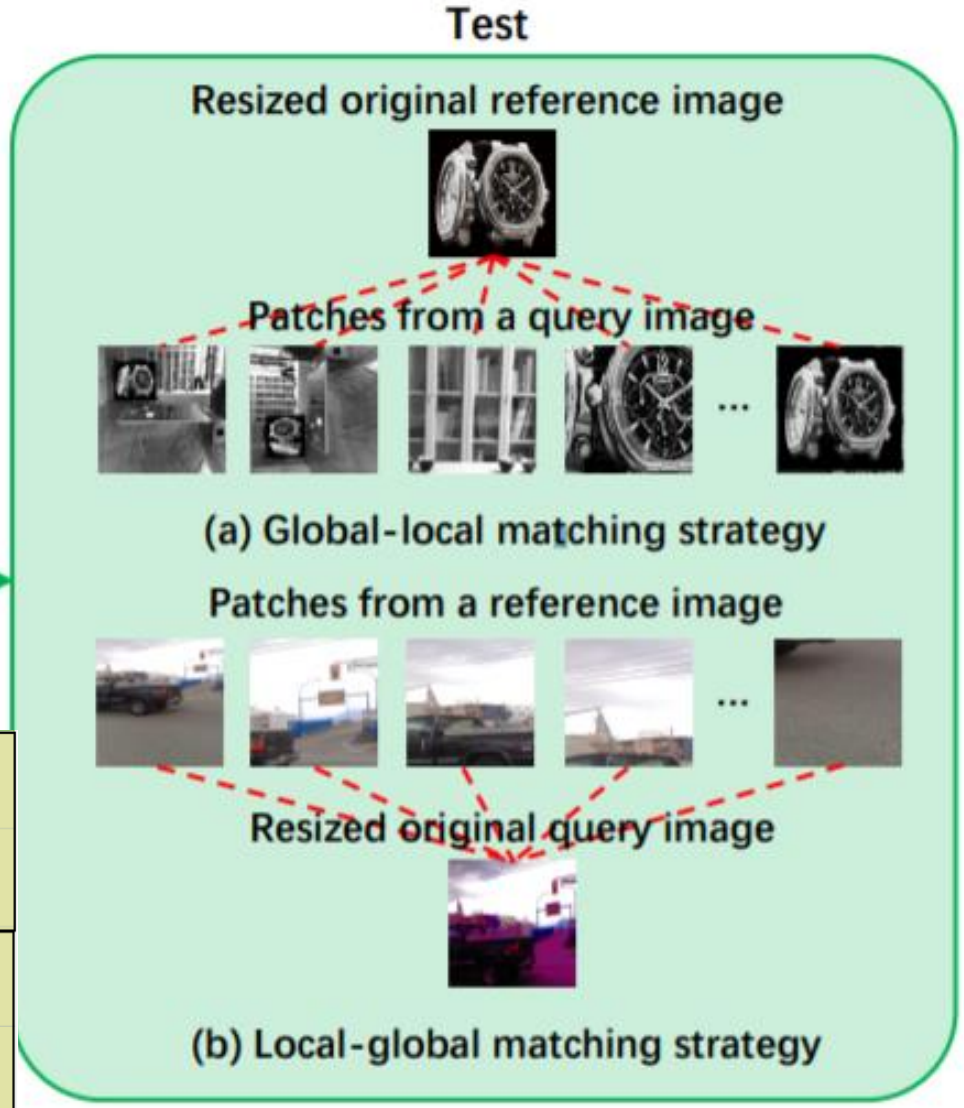


Credit/Ref: [D2LV: A Data-Driven and Local-Verification Approach for Image Copy Detection]

Test startetegy

- Global-local strategy: (AQ + OR:)
  - Original Reference:
  - Augumented Query(1->1k+)
    - Rotation(90,180,270)->3
    - Center crop ->5
    - Selective search for interesting parts- >990
    - Detection: Yolo-v5-> 1+
- Local Global strategy (OQ + AR)
  - Augumented Reference: 1->19
    - Divide Reference in parts





Credit/Ref: [D2LV: A Data-Driven and Local-Verification Approach for Image Copy Detection]

Ensemble scores

Confidence criteria

A-> patch from query; B-> patch from Ref image; f, g-> two models; α, β -> score threshold

$$score = \begin{cases} max(f(A,B), g(A,B)), \ if \ f(A,B) > \alpha \ and \ g(A,B) > \beta, \\ None, \ others. \end{cases}$$

Completeness criteria

$$score = \begin{cases} max(f_0(A,B), f_1(A,B), ..., f_n(A,B)), \ if \ f_i(A,B) > \alpha_i, \\ None, \ others, \end{cases} \quad (4)$$

and

$$score = max(f_0(A,B), f_1(A,B), ..., f_n(A,B)), \quad (5)$$

where: i is from 0 to n, $f_0, f_1, ..., f_n$ denotes n different models, and $\alpha_0, \alpha_1, ..., \alpha_n$ denotes n different score thresholds.

Ensemble score

$$score = max \begin{pmatrix} max(f(A_0,B), f(A_1,B), ..., f(A_l,B)), \\ max(f(A,B_0), f(A,B_1), ..., f(A,B_m)) \end{pmatrix}$$

Credit/Ref: D2LV: A Data-Driven and Local-Verification Approach for Image Copy Detection

Computation time

For ResNet-152 and ResNet-50-IBN, we use the official codes of Momentum2-teacher. We only change the backbone to ResNet-152 and ResNet-50-IBN. It takes about 2 weeks to pre-train the ResNet-152, and 1 week to pre-train the ResNet-50-IBN on 8 V100 GPUs. To be convenient, we supply the whole pre-training codes in the `Pretrain` folder.

Training

For training, we generate 11 datasets. For each dataset, 3 models v
takes about/less than 1 day on 4 V100 GPUs (bigger backbone tak
whole training codes, including how to generate training datasets
the `Training` folder. For more details, please refer to the readme

During testing, for global-local matching, we test the 33 trained models for three scales, *i.e.* 200×200, 256×256, 320×320, then ensemble them to get the final results. The time for extracting all query patches' features using 33 NVIDIA Tesla V100 32GB GPUs is about six hours. For local-global matching, we only test the three models trained on the basic augmentations for one scale, *i.e.* 256×256. The time for extracting all $19,000,000$ reference patches' features using 3 NVIDIA Tesla A100 40GB GPUs on a DGX-server is less than one day. We apply a PCA dimensionality reduction to decrease dimension from 8192 to 1500 with

Credit/Ref: D2LV: A Data-Driven and Local-Verification Approach for Image Copy Detection

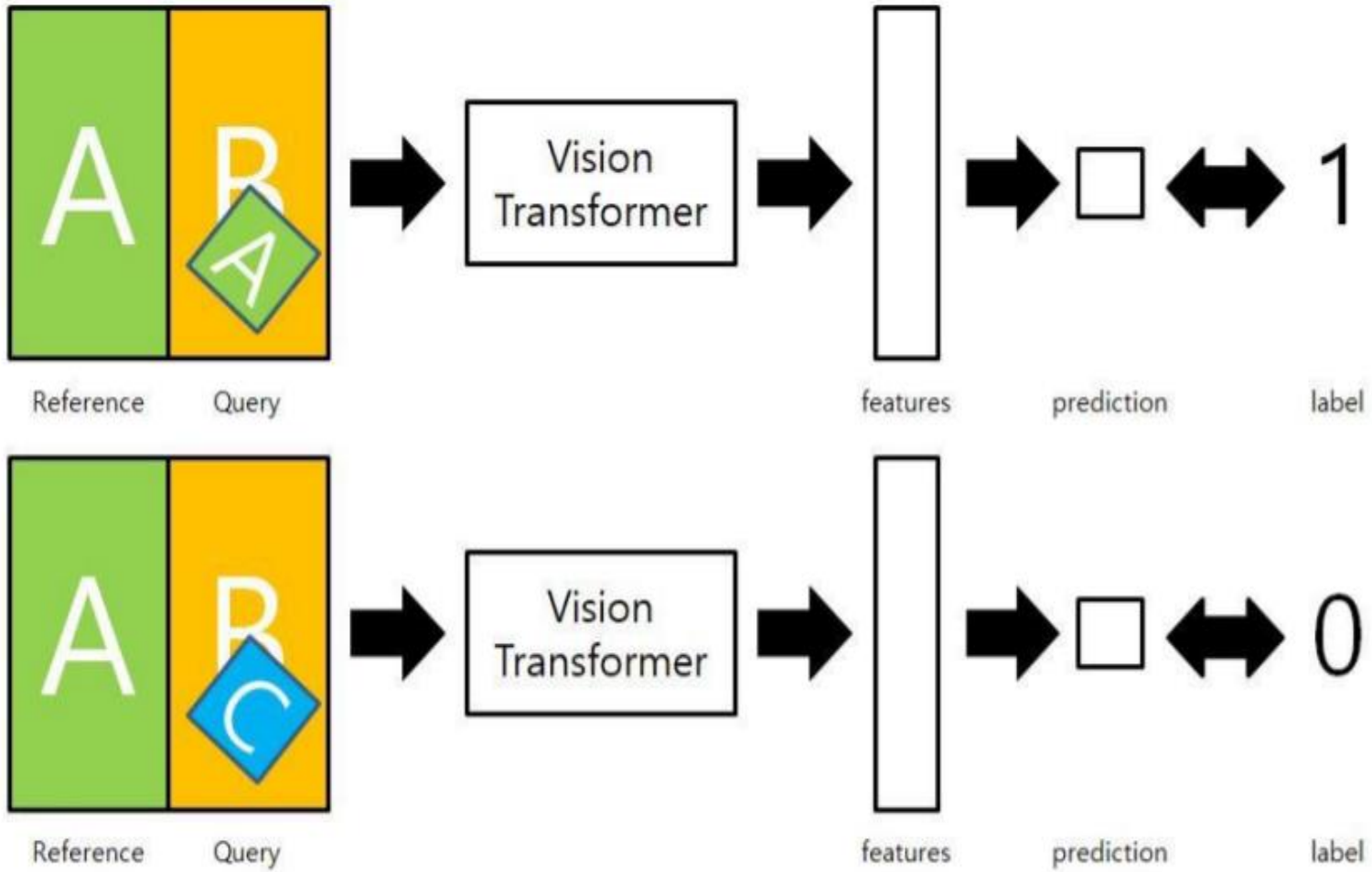
Credit/Ref: [2nd Place Solution to Facebook AI Image Similarity Challenge Matching Track]

2nd place winner

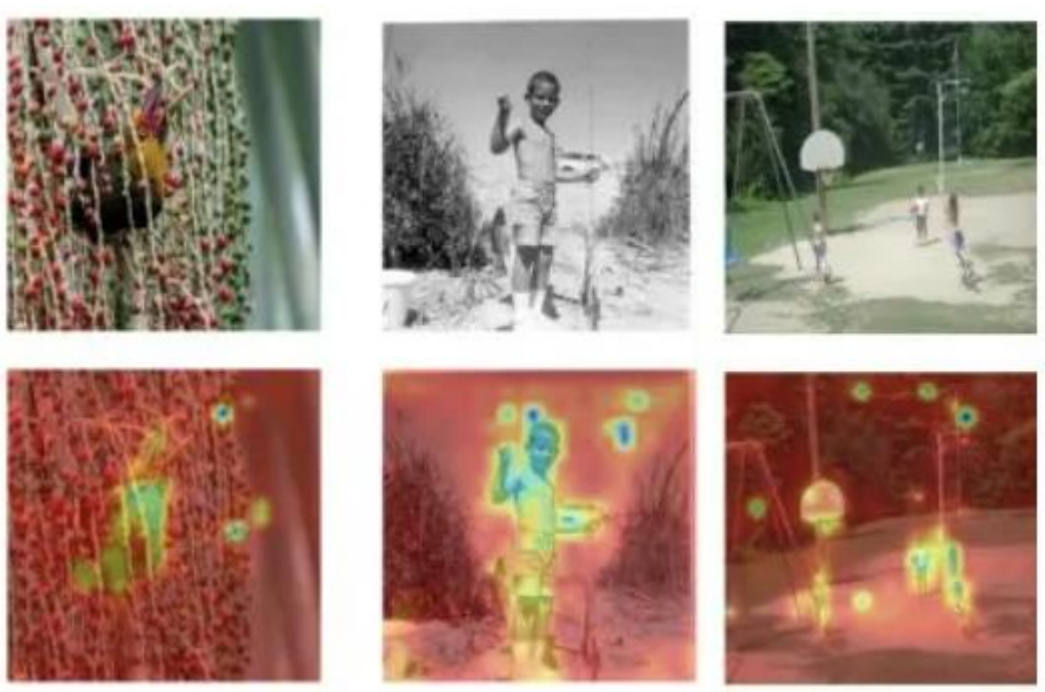
Approach Summary:

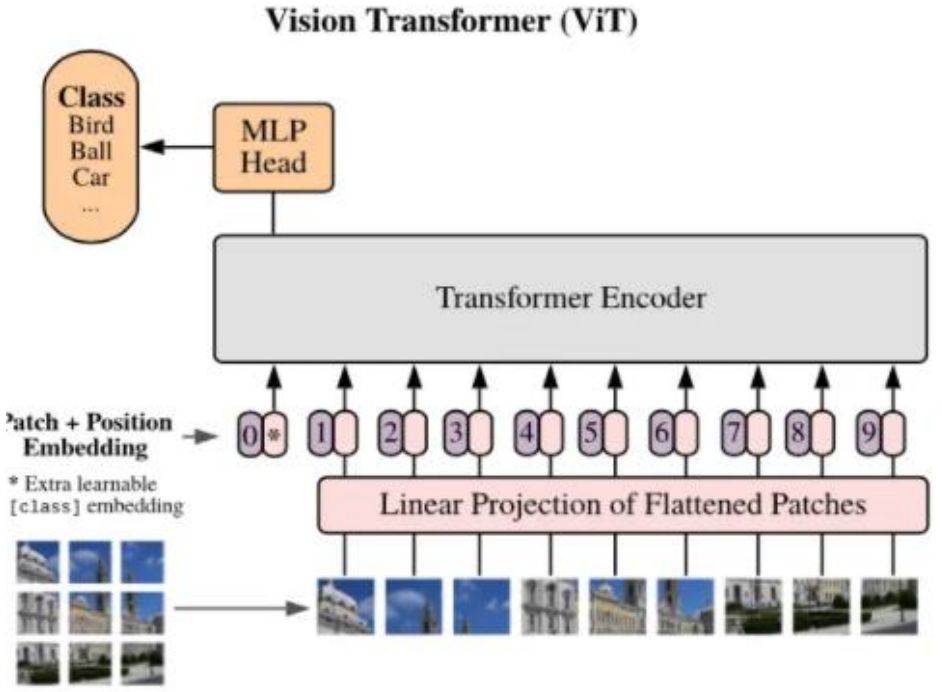
1) Combine Query and reference image

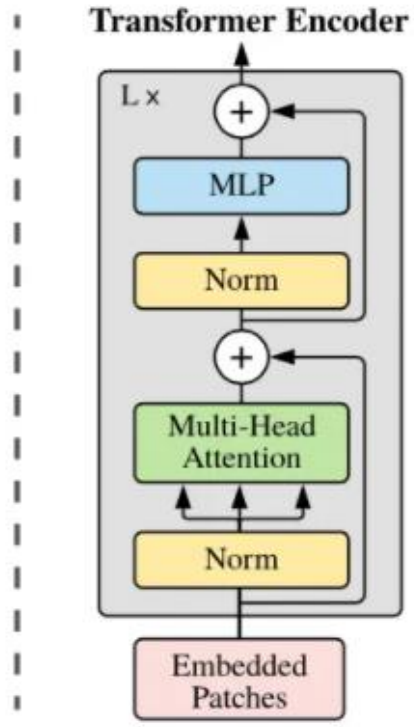
2) Use ViT (Vision Transformer) instead of CNN



Transformers







2nd Place: Testing and summary

Testing:

- Used cosine distance between 256-d embeddings d(Q,R)

- Compare between and get max score
  - Full Query and full Reference image
  - Partial Query, and Full reference
  - Full Query and partial reference

Team	Score	
	Micro-average Precision	Recall@Precision 90
Ours	**0.8329**	**0.7309**
separate	0.8291	0.7917
imgFp	0.7682	0.6715
forthedream	0.7667	0.7218

Credit/Ref: 2nd Place Solution to Facebook AI Image Similarity Challenge Matching Track

3rd place winner

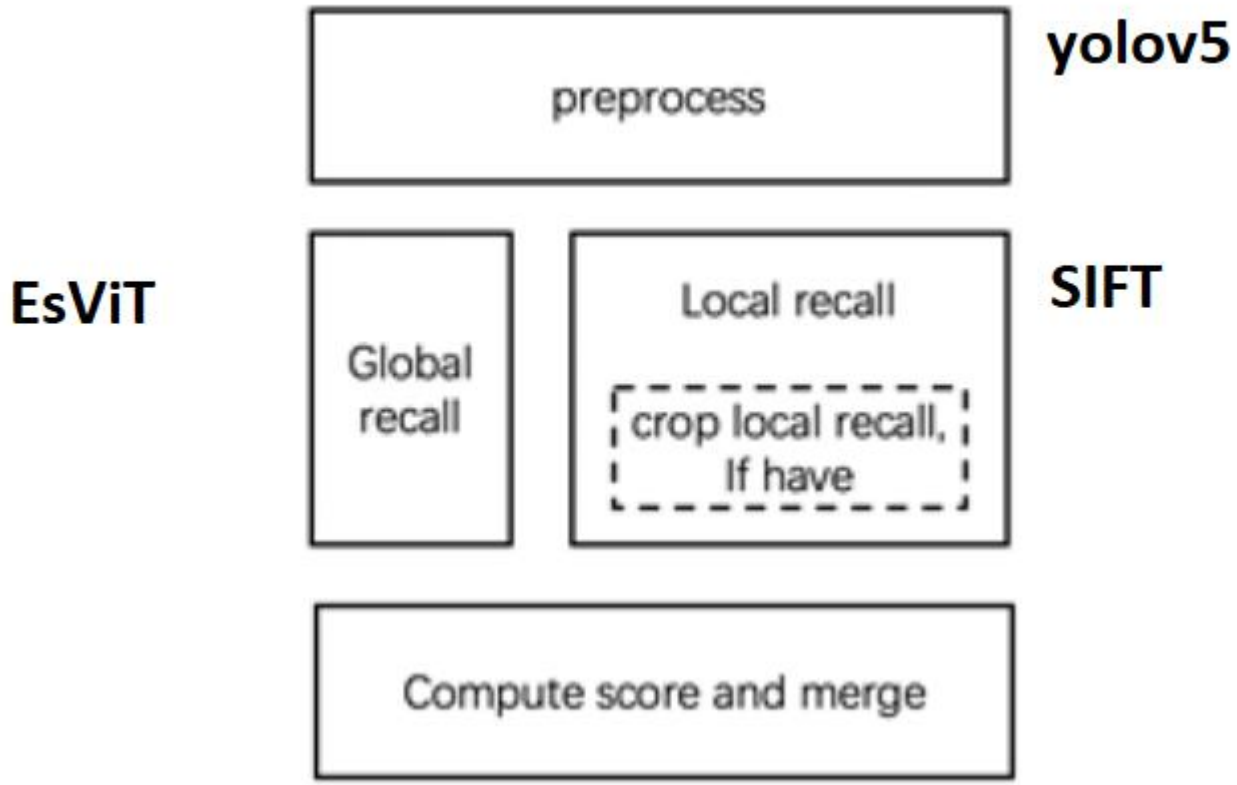
Global+local recall method

1) EsViT as global model

2) Trained detection model for detecting overlay augument

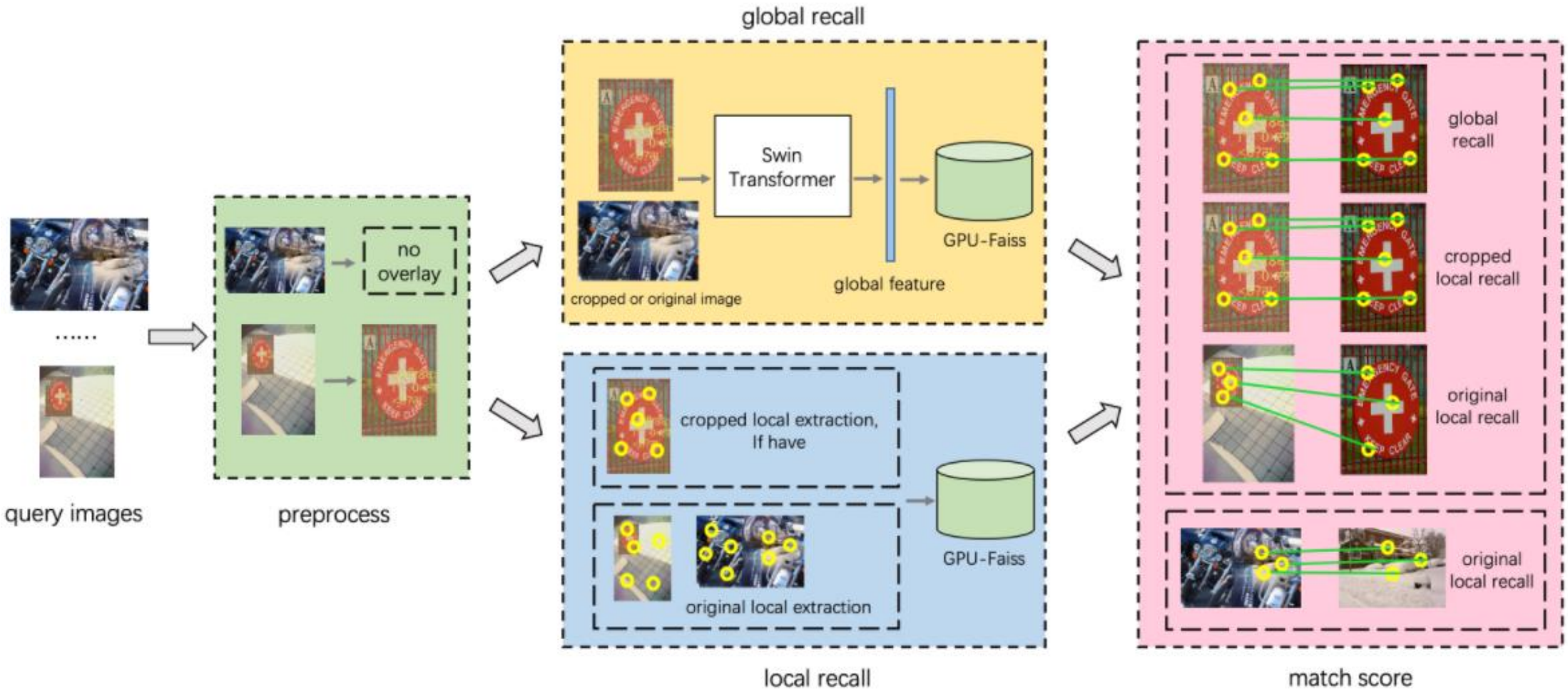
3) SIFT as local recall

4)



Credit/Ref: [3rd Place: A Global and Local Dual Retrieval Solution to Facebook AI Image Similarity Challenge]

3rd place


global recall
Swin Transformer
GPU-Faiss
cropped or original image
global feature
query images
.....
no overlay
preprocess
cropped local extraction, If have
original local extraction
GPU-Faiss
local recall
global recall
cropped local recall
original local recall
original local recall
match score

Reflection

- Didn’t participate to win

- Learning from problems/solutions was huge

- Winner solutions/approach opened whole new dimension

- Hindsight:
  - Might have done better with extra test processing(split images into 4 parts, ensemble test scores)
  - Requirements analysis:
    - Computation speed wasn’t requirement
    - Matching track vs descriptor track
  - Used only 200K training, 50K Q and 1 Mil ref
  - Overfit on public dataset: trained on 25k Qvs 100k(matching +neg)

Thanks