Employing AI to Help Lost Pets Return to Their Homes

Nikolay Arefyev
University of Oslo
nikolare@uio.no

Outline

1) The Kashtanka.pet project and website

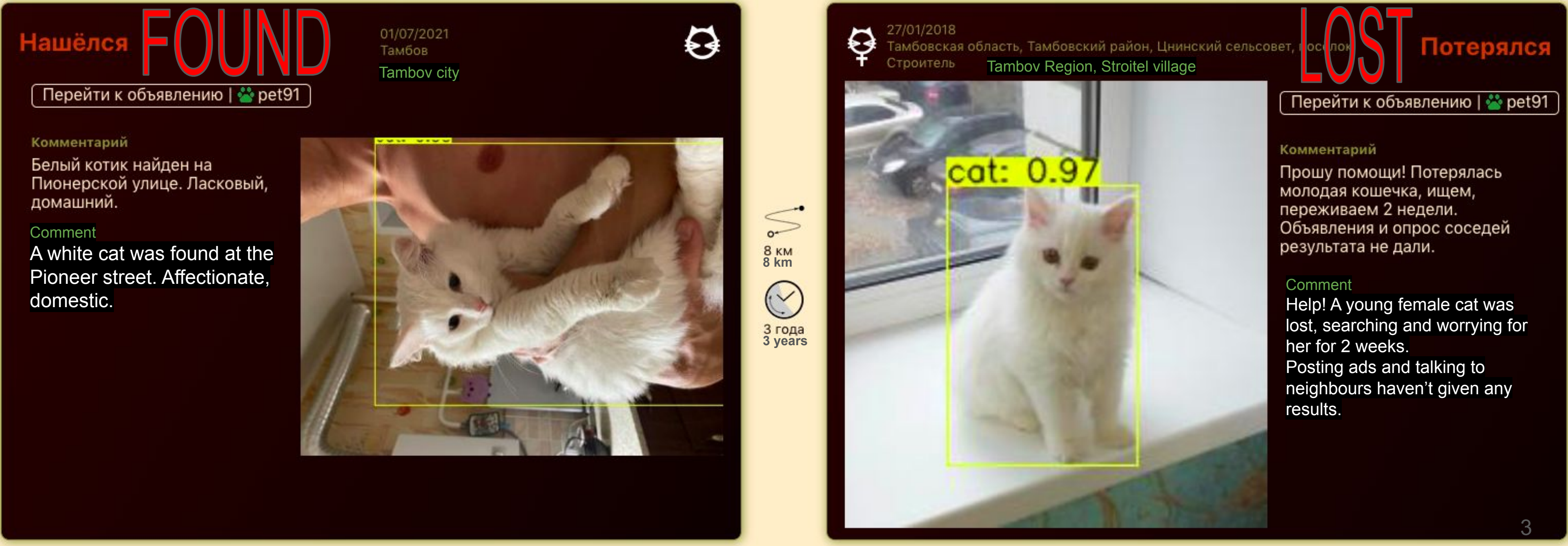
2) Datasets and evaluation

3) The best pet retrieval model (currently deployed)

> Kashtanka.pet project:

Build AI that helps lost pets return to their homes!

Everyday people post advertisements about lost or found pets in numerous websites / groups in social networks. It is almost impossible to find your lost pet among millions of ads posted in different places.😢
We need AI that matches ads about lost and found pets.😎



Kashtanka.pet Team

Mostly researchers and student of the Moscow State University and Higher School of Economics.

Lucy Grechka
Web Developer

Lucy has developed significant part of the Kashtanka web app.

Zhirui Zhou
AI model creator, Researcher

During the masters thesis preparation, Zhirui trained a Swin Transformer based neural network model to extract unique pet identity visual features. The model is used in the production system now.

Vyacheslav Stroev
Researcher

During the PhD thesis preparation Vyacheslav created a dataset for training and evaluation of pet retrieval models. Created evaluation scripts and the leaderboard. Trained BLIP-based models of pet retrieval.

Dmitry Grechka
System architect, Researcher, Developer

Dmitry conceived the system, designed and built it. He also maintains the system.

Nikolay Arefyev
Scientific advisor, Researcher

Nikolay is a scientific advisor of master and PhD students involved in the project. Organizes research talks and curates research directions. Nikolay & Vyacheslav created a dataset for training and evaluation of pet retrieval models. They also created evaluation scripts and the leaderboard.

Tee, Yu Shiang
AI model creator, Researcher

During the masters thesis preparation Tee, Yu Shiang trained the YoloV5 Neural Network model to extract a bounding box of cat and dog heads from photos. The model is used in the production system now.

ME

Maria Eliseeva
Researcher

During the masters thesis preparation Maria organized data annotation process, annotated data for machine learning and carried out data analysis.

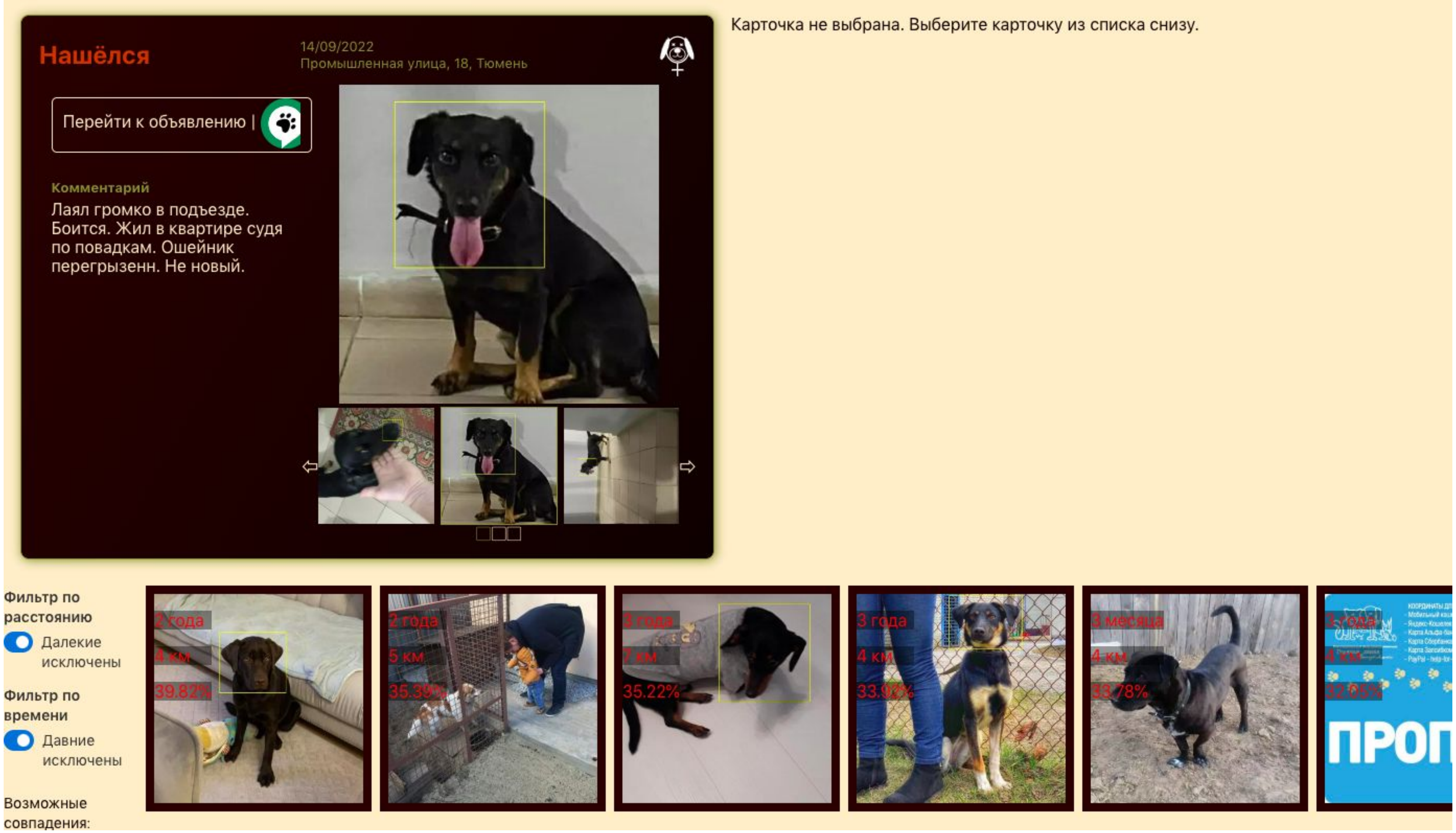
Kashtanka.pet website

Users are volunteers helping to find owners of those pets that were lost and then found by somebody else.

1. New ads about recently found pets are displayed on the main page.

Kashtanka.pet website

2. A volunteer selects an ad about a pet found and gets best matching ads about lost pets.

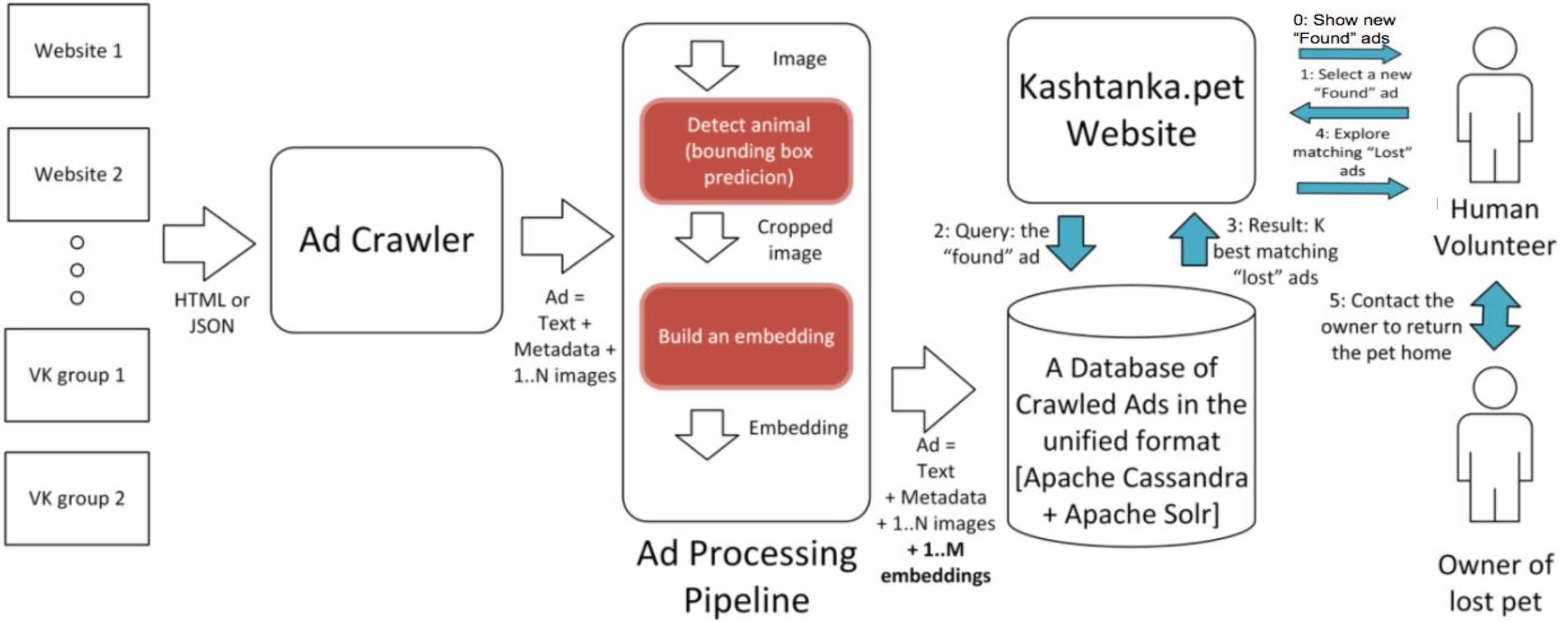


Kashtanka.pet website

3. The volunteer selects one of the matches and can compare the lost and the found pets (several photos for each, textual descriptions, genders if specified, places and dates of these events if specified). If it looks like the same pet, the volunteer contacts the owner.

Kashtanka.pet components and data flow

DL models are in red boxes.



Version 1 was bad - dissimilar dogs of different breeds were proposed as candidates!

Datasets and Evaluation

Before improving quality learn to measure it first!

We created the evaluation setup: formalized the task, proposed metrics, created dev/test sets.
Created tools: the evaluation script and the leaderboard!

The Pet Retrieval Task formalization

Supposed usage scenario. A new “Found” ad arrives. Among 10^5-10^6 “Lost” ads, the model shall select K best candidates AND estimate the chances that one of them is really about the same pet (ads will be shown to the volunteers ordered by these estimated chances, the volunteers will not spend time on unpromising ads).

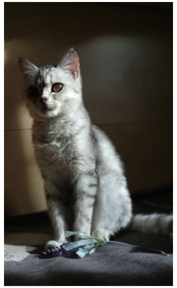
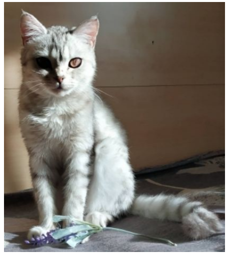
Inputs:
Query: a lost / found ad (several photos + text [+ type,gender,loc.&date])
DB (a.k.a. Answers or Keys): found / lost ads

Output: K best matching found / lost ads ordered by similarity, the probability (or unnormalized score) that there is a match among the returned candidates.

Creating dev/test sets

There is only one textual description, it is about a lost cat. Thus, one of the synthetic ads has empty description. Can we generate it?

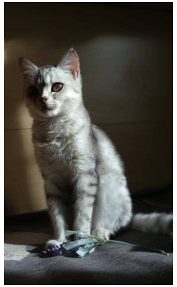
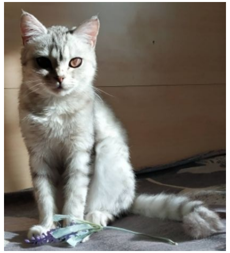
Don’t know which pairs of ads really contain the same animal
⇒ for evaluation generate pairs by splitting photos from a single ad
Random split? Bad idea


Пропал кот, по кличке Барсик , окрас Серый, Белый

(a male cat is lost, his nickname is Barsik, he is gray and white)

Synthetic found (answer)

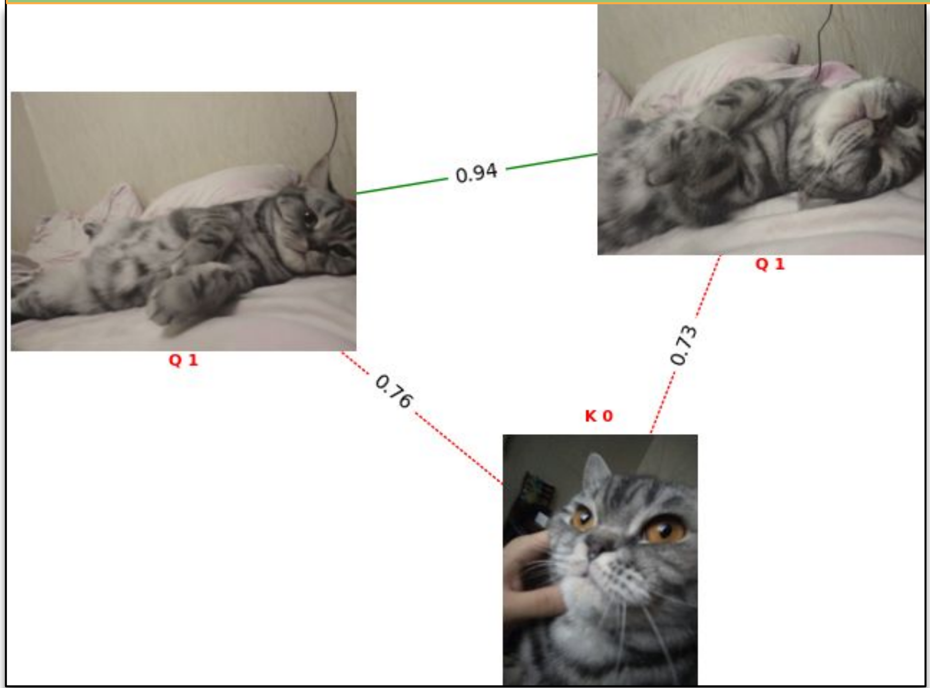
Lost (query)

Пропал кот, по кличке Барсик , окрас Серый, Белый

Ad splitting

1) Build a weighted graph, weights representing similarity between backgrounds (NOW: the cosine similarity between image embeddings from the BLIP model).
2) Binarize edges: **I[weight < threshold]** (red/green edges)
3) Compute the connected components of the graph (different backgrounds/places, numbers in red).

All images from the same component go together to the query ad (Q), or the answer ad (K).
Too large threshold ⇒ some images taken in the same place end in different components, and are distributed between the query ad and the answer ad ⇒ unrealistically simple example, a retrieval model can compare backgrounds and ignore the pet.
Too small threshold ⇒ too few ads with at least 2 components to make a query and an answer.



Same-Place and Same-Animal datasets

Among ads with at least 2 photos, we sampled 400 lost and 400 found ads (stratified by the number of photos). From each ad we took a random pair of photos, then added 200 control pairs consisting of photos from different ads.

Annotators were asked if 2 photos were taken in the same place (after masking out the pet to prevent relying on it), and if they contain the same pet (w/o masking).

Fig. 4: An example of the annotation interface for the same place dataset.

Same-Place and Same-Animal datasets: analysis

1) The pairs of photos from the lost ads are 2x more often taken in several different places ⇒ our main source of test examples.
2) As expected, different ads rarely contain photos of the same animal, and two photos from the same ad rarely contain different animals ⇒ can assume that after our synthetic split a query and an answer extracted from the same ad contain the same pet, and no answer for other queries contain this pet.
3) There are images that do not contain pets ⇒ leave them to check the robustness of the competing models.

	Found ads		Lost ads	
Values / Answers	%	95% Confidence interval	%	95% Confidence interval
Made in the same place	61.41%	± 5.06%	20.33%	± 4.16%
Made in different places	33.80%	± 4.92%	72.70%	± 4.16%
Cannot decide	2.54%	± 1.64%	4.18%	± 2.07%
Photo contains text	2.25%	± 1.54%	2.79%	± 1.70%

	Same ad		Different ads	
Values / Answers	%	95% Confidence interval	%	95% Confidence interval
Same animal	88.44%	± 2.24%	1.53%	± 1.71%
Different animals	2.18%	± 1.02%	91.3%	± 3.94%
Cannot decide	1.41%	± 0.82%	1.02%	± 1.40%
Photo contains text	7.96%	± 1.90%	6.12%	± 3.35%

Fig. 5: Results for **same place** annotation (left table) and **same animal** annotation (right table).

Ad splitting: blip_split_v3, sketch of the method

1) Find images containing pets (“pet” images).
   a) NOW: Contains⇔semantic segmentation (Cascade Mask RCNN with Swin Transformer backbone) returned masks for CAT or DOG classes
   b) TRY: image classifiers / object detectors
2) < 2 images contains pets => random split, useless for evaluation but still left in the dataset to check for robustness
3) split images into clusters: different clusters – different backgrounds
   a) NOW: similarity of BLIP embeddings, thresholding, connected components
      threshold=0.789686, selected on the Same-Place dataset such that
      P(same place | similarity < threshold) < 0.2
   b) TRY: training a pairwise image classifier to find all intersections of the backgrounds, graph clustering
4) < 2 clusters contain at least 1 pet images (< 2 “pet” clusters)
   => ignore clusters, random split of pet images into 2 parts, distribute other images across the parts, “**easy**” ex.
5) random split of pet clusters into 2 parts, distribute other clusters across the parts, “**hard**” ex.

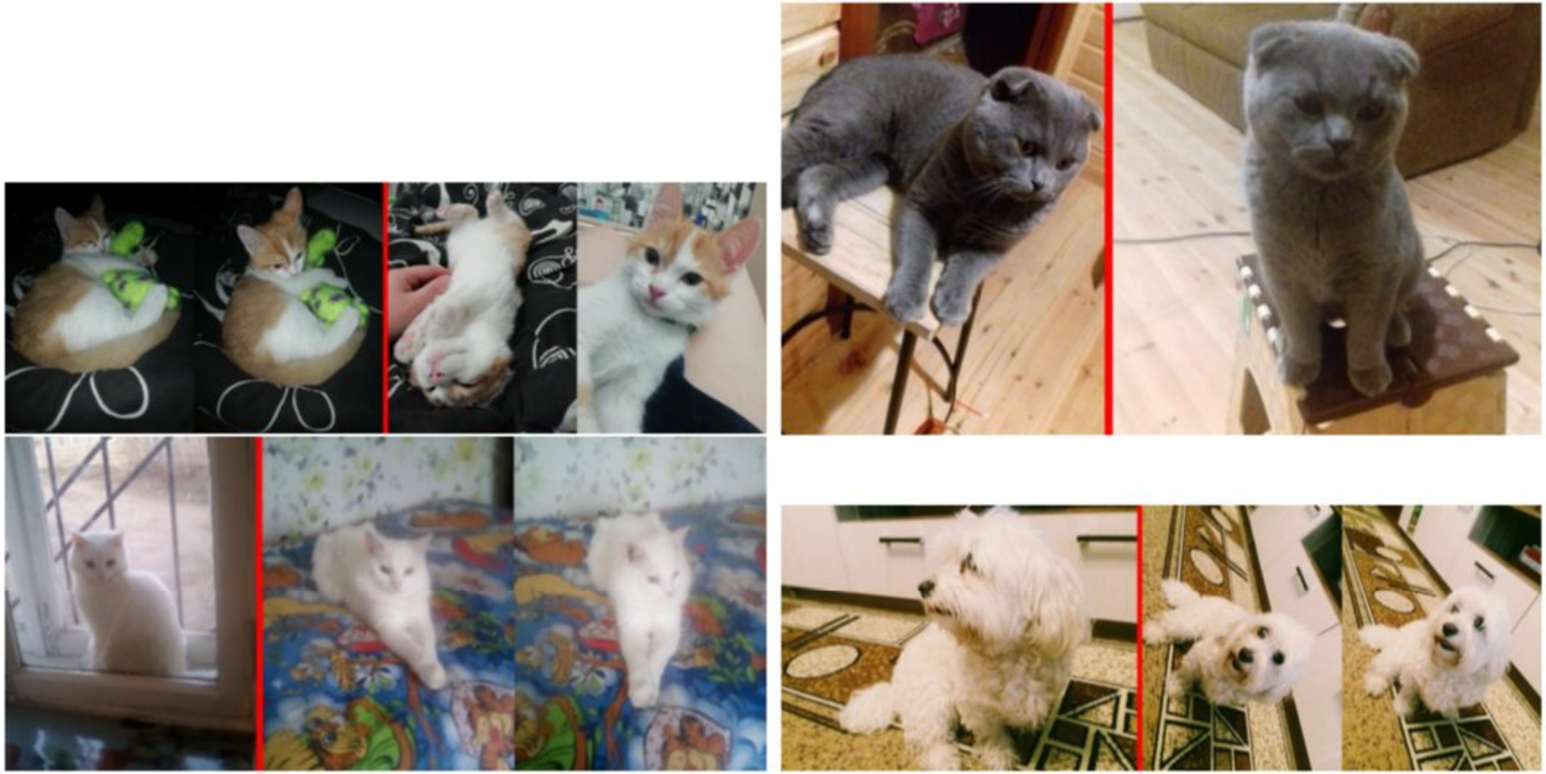


Fig. 7: Two hard (left) and two simple (right) random examples. Queries and answers are separated by red lines.

Evaluation datasets

1) 50k ads (~25% of all ads from kashtanka.pet) were split into dev_small/dev/test (2k/24k/24k) ← split on the pet level
2) ~50% of ads have 1 image only => SKIP
3) for each ad of lost_ads: # similar for found_ads

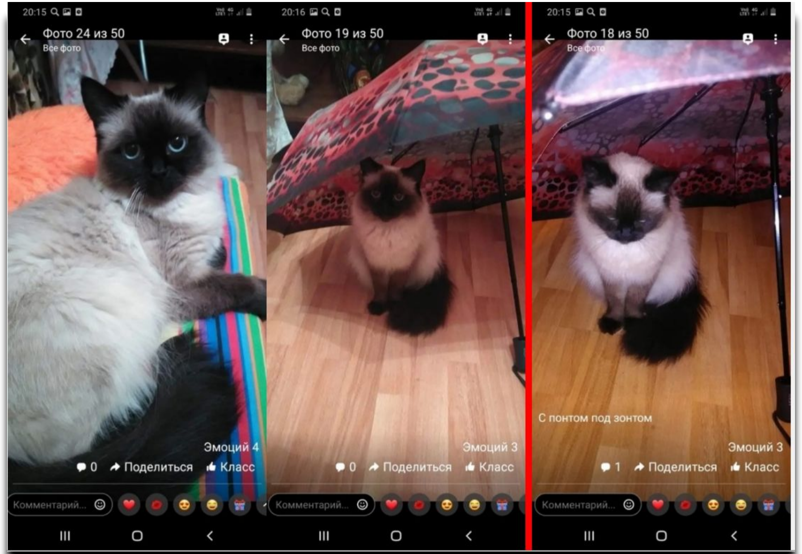
```
Q,K = ad_splitting(ad)
save Q to dataset/lost/lost
with prob of 0.5: save K to dataset/lost/synthetic_found  # matchable ex.
```

Kashtanka.pet dataset

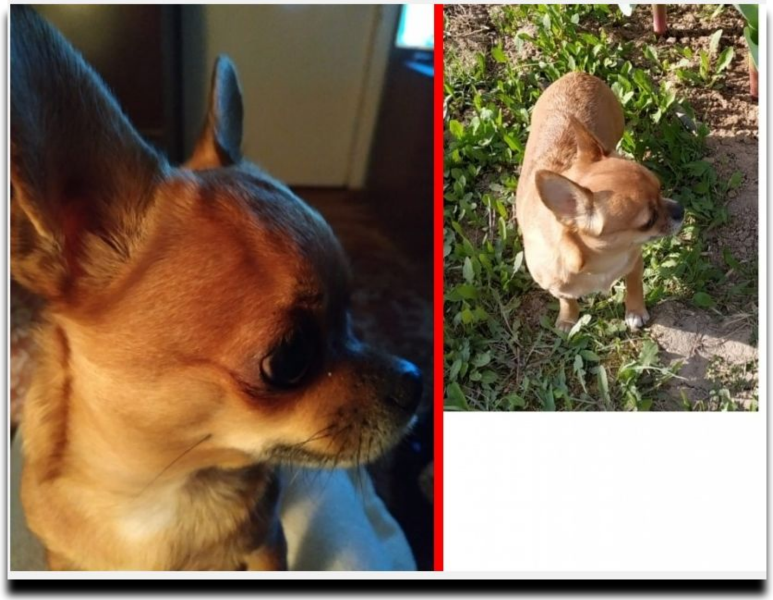
50K ads are used for evaluation, 150K ads left for training.

	dev_small	**dev**	**test**
#lost/found ads	~500/500	~6K/6K	~6K/6K
hard queries	222/102	2131/1005	2084/978
simple queries	78/152	779/1516	805/1508
unmatchable queries	243/243	3198/2574	3163/2686
answers	548/534	5532/5671	5622/5602

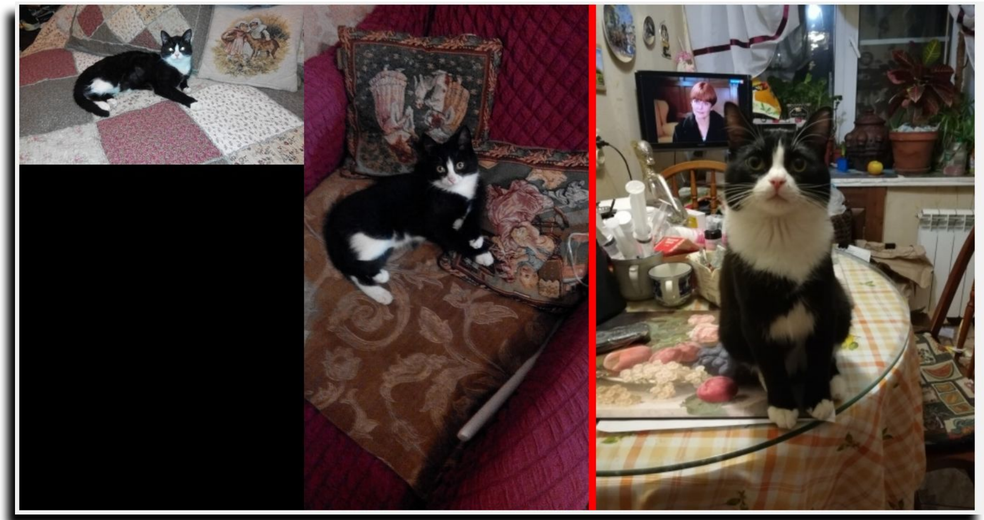
random **simple** ex. (from ads with 2-3 images)



random **hard** ex.



random **hard** ex.



Evaluation metrics

Candidate recall@K: among matchable queries, the proportion of queries having the correct answer among top K best candidates returned by a model.

hit10pred_precision@0.1: take 10% of queries with the highest chances (estimated by the model) that the correct answer is among top 10 candidates returned, calculate the proportion of queries that in fact have the correct answer there.

Currently deployed pipeline

5 Master’s theses were successfully defended

Model comparison (test set, hard lost examples)

Zhirui Zhou. Animal recognition using methods of fine-grained visual analysis. [Master’s thesis, HSE, 2022]

TEE, Yu Shiang. Animal recognition using methods of fine-grained visual analysis. [Master’s thesis, HSE, 2022]

Konstantin Kudelkin. Animal recognition using deep learning based face recognition methods. [Master’s thesis, HSE, 2022]

model	**recall@10**	**recall@100**	**hit10pred90%**
Version 2: Zhirui&Calvin	0.581	0.834	0.192
Konstantin	0.395	0.604	0.005
zero-shot SLIP	0.172	0.423	0.024
zero-shot BLIP	0.113	0.347	0.005
Version 1	0.087	0.459	0.005
random baseline	0.002	0.019	0

Zhirui&Calvin. Step 1: detect and crop head or body

YoloV5m trained on Tsinghua Dogs to detect heads and bodies

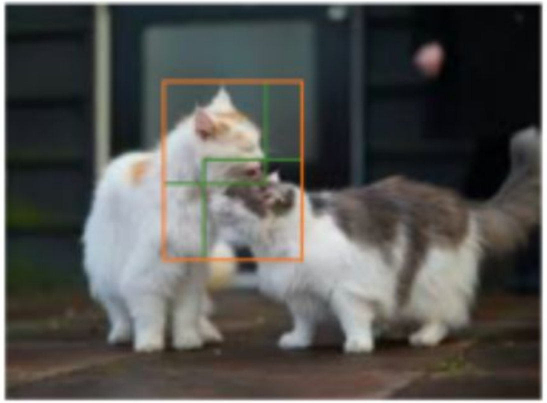
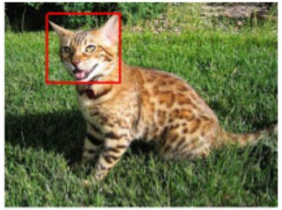
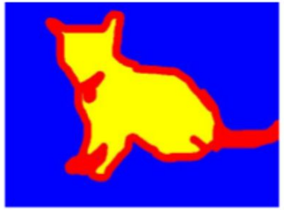


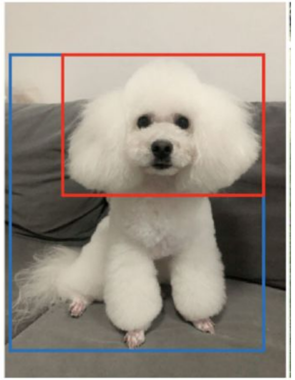
Figure from Zhirui Zhou. Animal recognition using methods of fine-grained visual analysis. [Master’s thesis, HSE, 2022]

Head/body detection for cats&dogs: datasets





1) **Oxford-IIIT Pet**: **3.5K train**, 3.5K test images
   - cats and dogs, head bboxes + body segmentation masks ⇒ body boxes
   - 108 KB on avg.



2) **Tsinghua dogs: 65K train,** 5K valid images
   - dogs only, head and body bboxes
   - many breeds in proportions specific for China, 65% are real-life images from owners
   - low-res. version was used: 37 KB on avg.

3) **Kashtanka:** 150 dogs, 170 cats – test only!
   - cats and dogs, head and body bboxes + 5 landmarks

Images from: Parikh et al. Cats and Dogs, 2012; Zou et al. A new dataset of dog breed images and a benchmark for fine-grained classification, 2020; Konstantin Kudelkin. Landmark annotation guideline, 2022.

Training on **70K dogs** VS. **7K dogs&cats**

Predict on Kashtanka

(Metric)@0.5:0.95		**Cats & Dogs**			**Cats Only**			**Dogs Only**		
	Metric	**All**	**Head**	**Body**	**All**	**Head**	**Body**	**All**	**Head**	**Body**
Trained on Oxford	**mAP@0.5:0.95**	0.524			0.566			0.483		
	AP@0.5:0.95 (Head)		0.566			0.597			0.537	
	AP@0.5:0.95 (Body)			0.483			0.534			0.429
Trained on Tsinghua	**mAP@0.5:0.95**	0.564			0.563			0.564		
	AP@0.5:0.95 (Head)		0.497			0.467			0.521	
	AP@0.5:0.95 (Body)			0.63			0.658			0.606

Better **body** bboxes for both cats and dogs when trained on Tsinghua Dogs

- generalizes well from dogs to cats!

Better head bboxes for both cats and dogs when trained on Oxford Pets

- for dogs the difference is small
- too low res. of head bboxes in low. res. Tsinghua Dogs?

TEE, Yu Shiang. Animal recognition using methods of fine-grained visual analysis. [Master’s thesis, HSE, 2022]

Zhirui&Calvin. Step 2: build embeddings

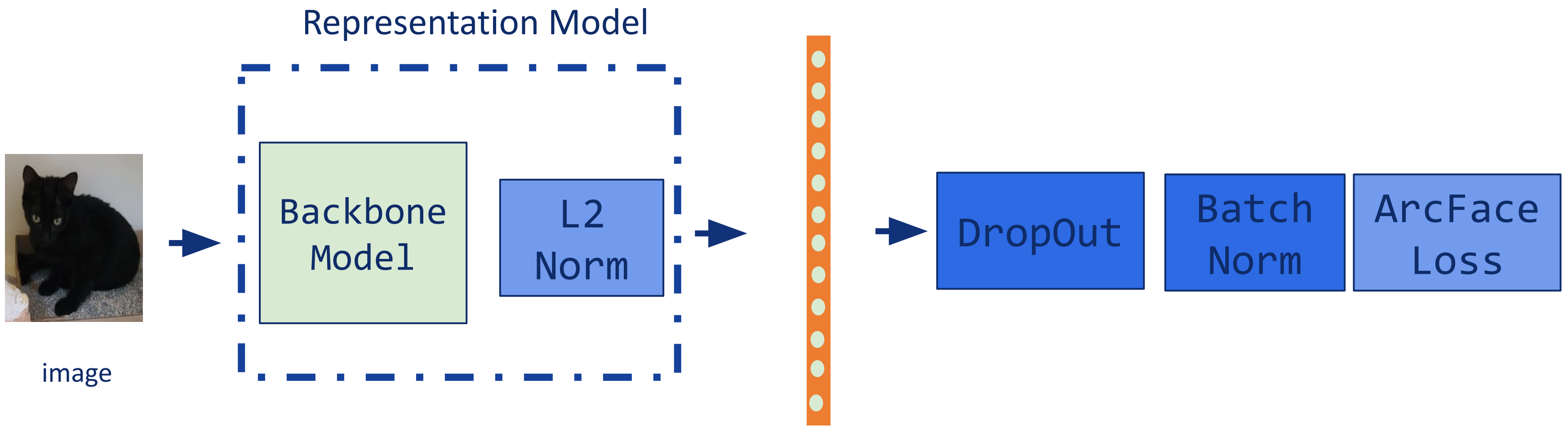


Image representation
for inference

Figure from Zhirui Zhou. Animal recognition using methods of fine-grained visual analysis. [Master's thesis, HSE, 2022]

Backbones and loss functions

Model	recall@1	recall@10
ConvNext-small / CrossEntropy Loss	0.0896	0.2919
ConvNext-small / ArcFace Loss	0.2529	0.5195
EfficientNet-b4 / CrossEntropy Loss	0.0770	0.2515
EfficientNet-b4 / ArcFace Loss	0.0005	0.0117
SwinTransformer-base / CrossEntropy Loss	0.0009	0.0038
SwinTransformer-base /ArcFace Loss	**0.2567**	**0.5275**

Table 4.1. results of different model and loss function, evaluate on dev-hard-lost dataset

Backbones with similar inference speed (~300 fps) are compared.

ArcFace loss gives a huge boost!

Figure from Zhirui Zhou. Animal recognition using methods of fine-grained visual analysis. [Master's thesis, HSE, 2022]

Crops and BNNeck

Model	recall@1	recall@10
SwinTransformer-base /ArcFace Loss	0.2567	0.5275
SwinTransformer-base /ArcFace Loss / BNNeck	0.2656	0.5443
SwinTransformer-base /ArcFace Loss / BNNeck / Body Crop	0.2858	0.5692
SwinTransformer-base /ArcFace Loss / BNNeck / Head Crop	**0.3229**	**0.6021**

Table 4.2. results of model optimizer, evaluate on dev-hard-lost dataset

Head crops are better than body crops, which are better than full image.

BatchNorm before ArcFace loss during training (denoted as BNNeck in the table) helps a bit.

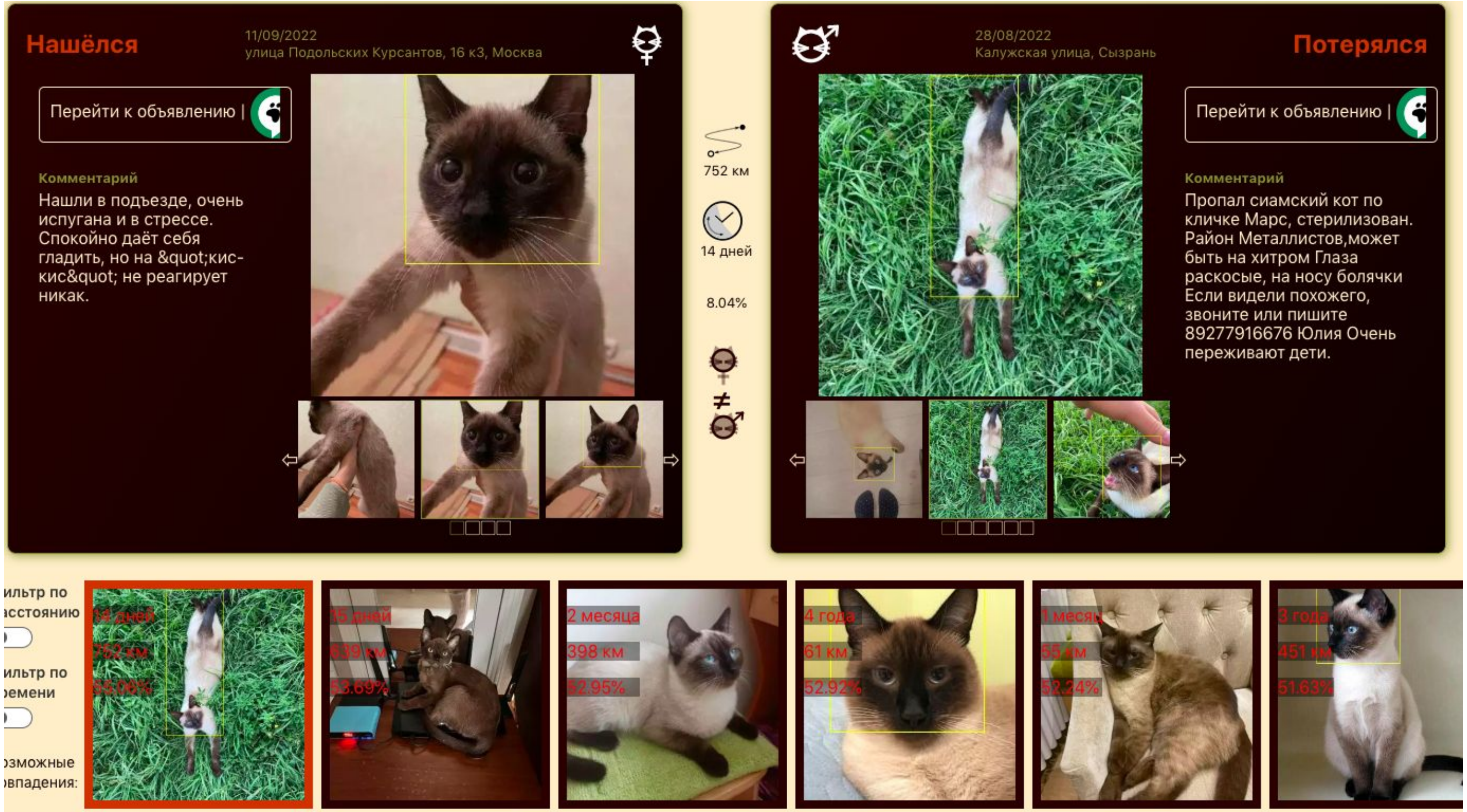
Figure from Zhirui Zhou. Animal recognition using methods of fine-grained visual analysis. [Master's thesis, HSE, 2022]

Zhirui&Calvin. Step 3: find most similar ads (max agg.)

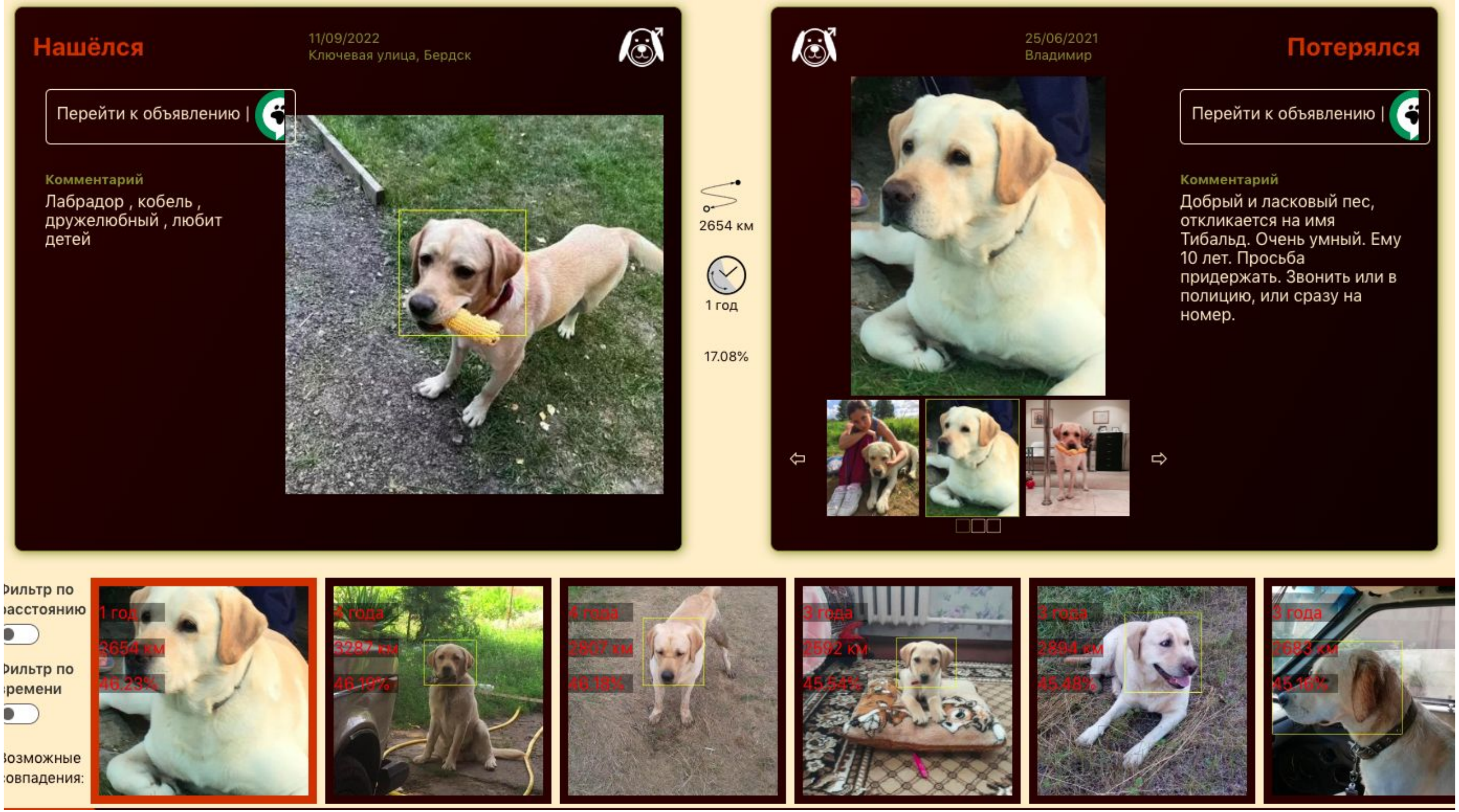
Similarity between ads is the maximum similarity between their images:

$$\mathbf{sim(ad1,ad2) = max\ sim(ad1_i, ad2_j)}$$

Candidates are much better!

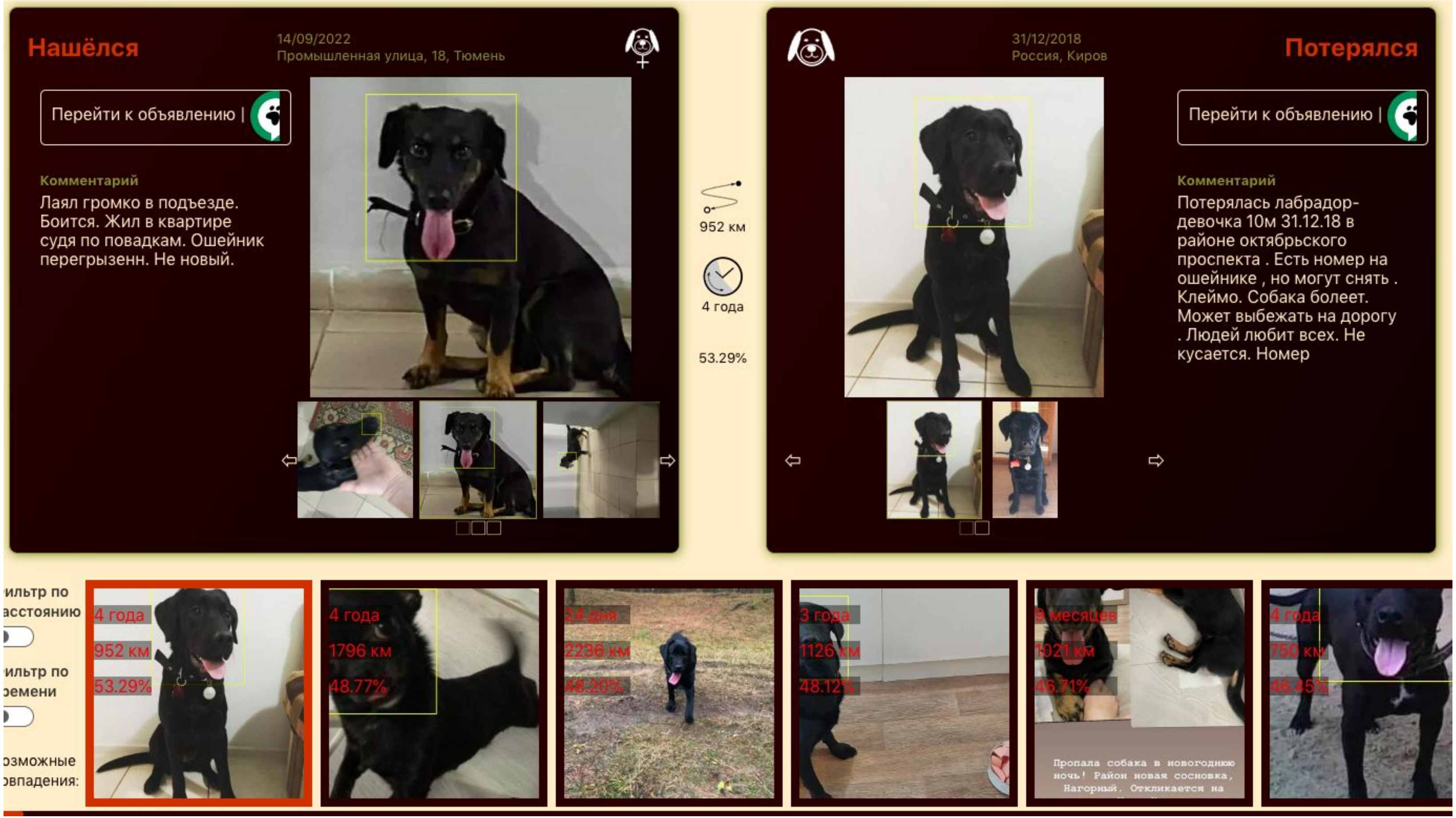


Candidates are much better!



But not ideal (breed!)

But not ideal (color patterns)

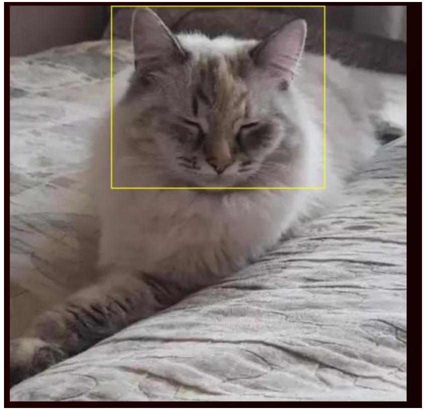


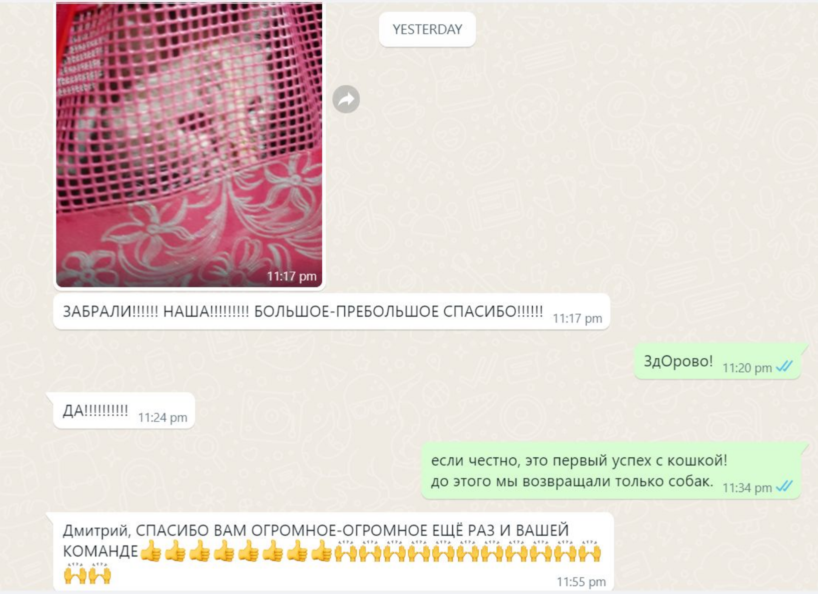
Main results

Pet matches confirmed by pet owners:

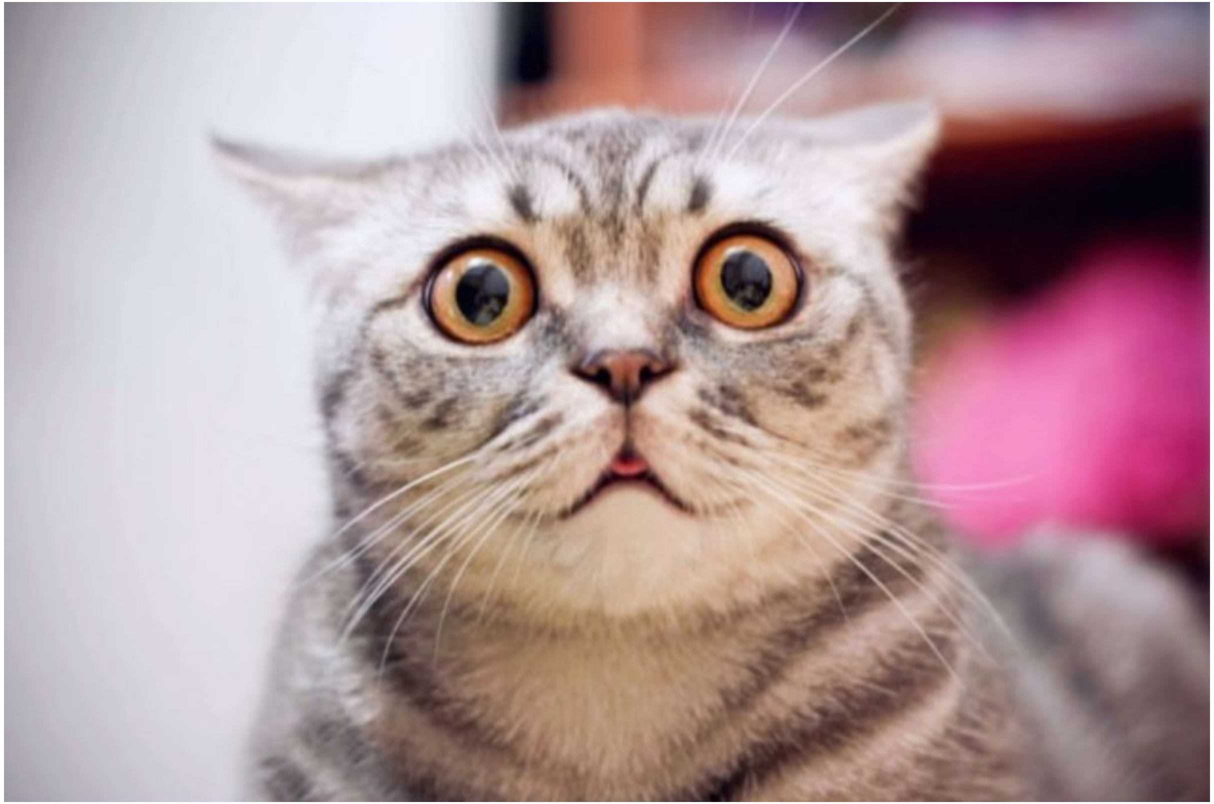
- Version 1: 2
- Version 2: 17

Some of them returned to their homes, others continued living with their new owners who found them.





Thank you for attention!



If you want to join us and help lost pets return home, please write: nick.arefyev@gmail.com

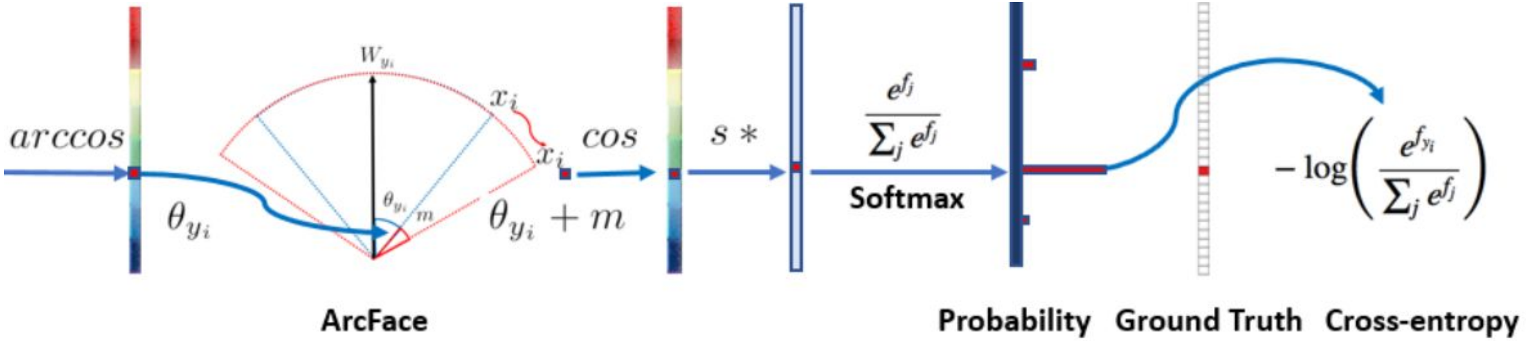
ArcFace loss

Softmax + CE: $L_1 = -\log \frac{e^{W_{y_i}^T x_i + b_{y_i}}}{\sum_{j=1}^N e^{W_j^T x_i + b_j}}$

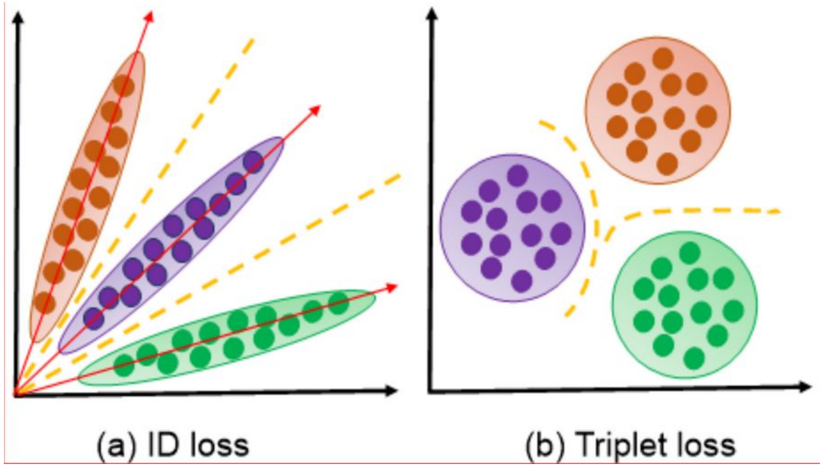
$$W_j^T x_i = \|W_j\| \, \|x_i\| \cos\theta_j$$

Normalize weights and inputs: $L_2 = -\log \frac{e^{s\cos\theta_{y_i}}}{e^{s\cos\theta_{y_i}} + \sum_{j=1, j\neq y_i}^N e^{s\cos\theta_j}}$

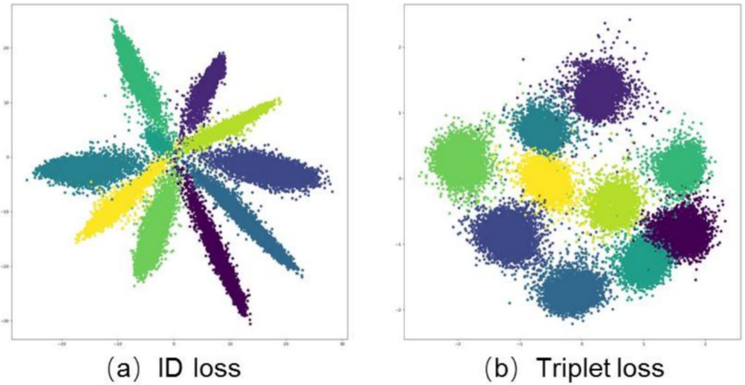
Add an angular margin to the positive class: $L_3 = -\log \frac{e^{s\cos(\theta_{y_i}+m)}}{e^{s\cos(\theta_{y_i}+m)} + \sum_{j=1, j\neq y_i}^N e^{s\cos\theta_j}}$

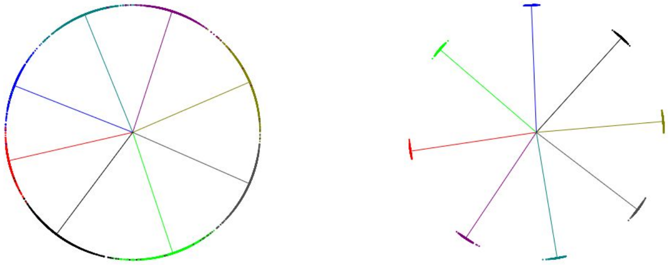


Images from Deng et al. ArcFace: Additive Angular Margin Loss for Deep Face Recognition, 2022



Visualization on MNIST Dataset





(a) Norm-Softmax (b) ArcFace

Fig. 3. Toy examples under the Norm-Softmax and ArcFace loss on 8 identities with 2D features. Dots indicate samples and lines refer to the center direction of each identity. Based on the feature normalization, all face features are pushed to the arc space with a fixed radius. The geodesic distance margin between closest classes becomes evident as the additive angular margin penalty is incorporated.

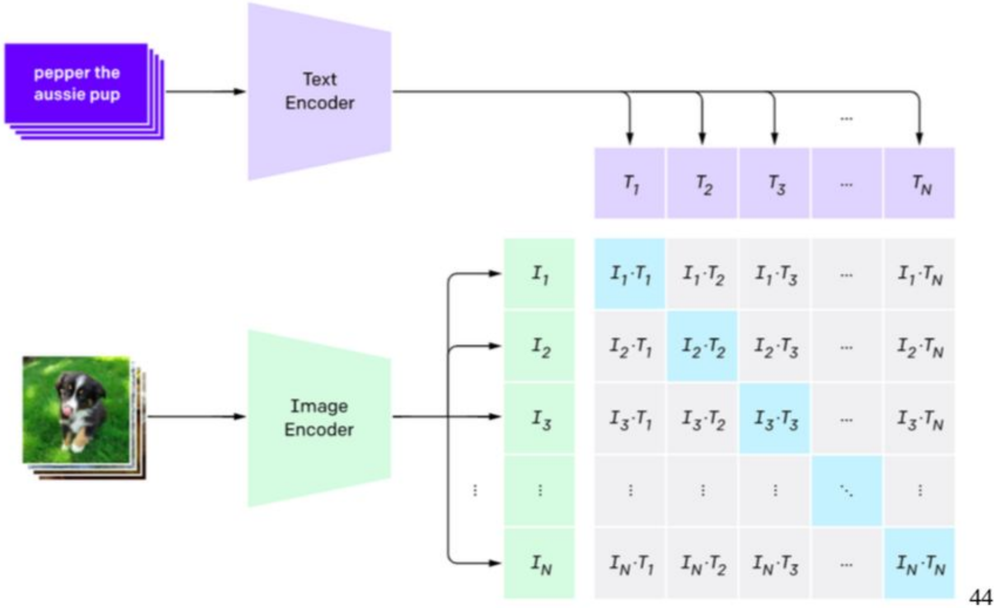
Images from Deng et al. ArcFace: Additive Angular Margin Loss for Deep Face Recognition, 2022 and Luo et al. Bag of Tricks and A Strong Baseline for Person Re-Identification, 2019.

CLIP: contrastive pre-training

- Trained on 400M image-text pairs (compare to 14M in ImageNet)

  Batches of 32K pairs: select text for image and vice versa

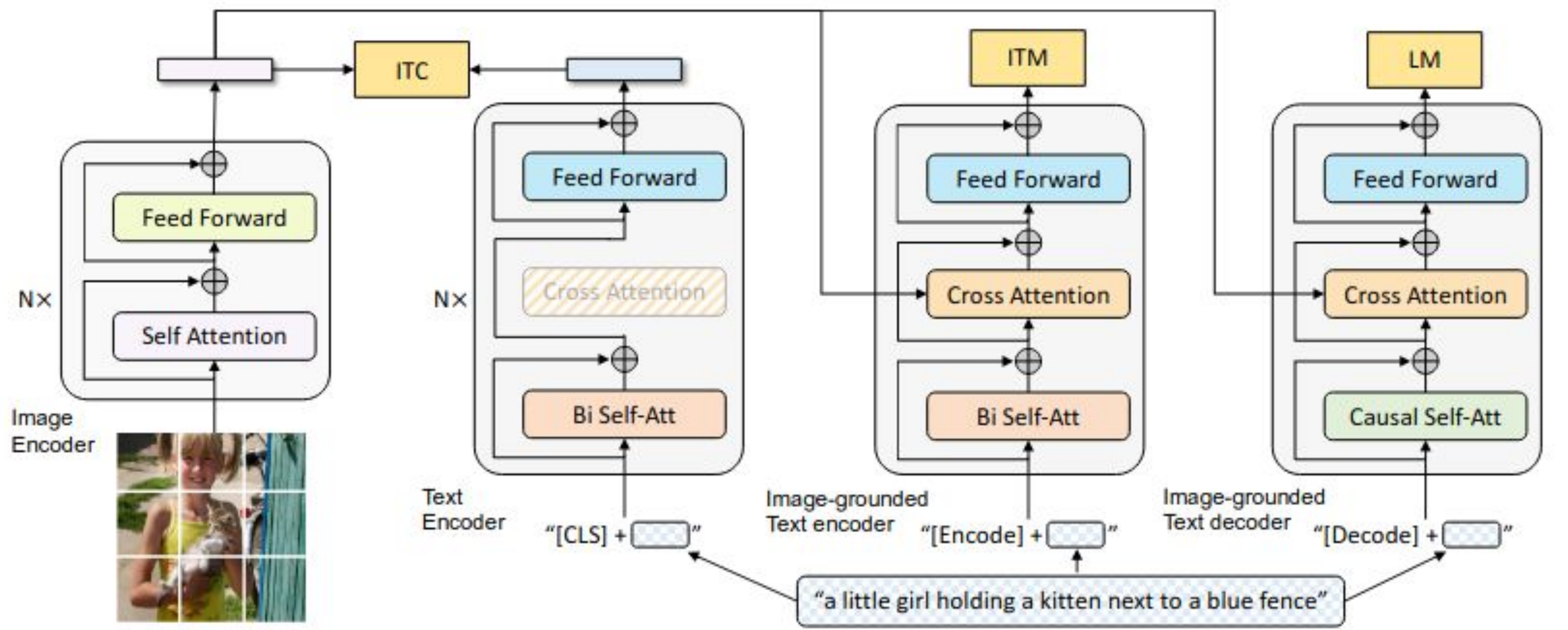
```
# image_encoder - ResNet or Vision Transformer
# text_encoder  - CBOW or Text Transformer
# I[n, h, w, c] - minibatch of aligned images
# T[n, l]       - minibatch of aligned texts
# W_i[d_i, d_e] - learned proj of image to embed
# W_t[d_t, d_e] - learned proj of text to embed
# t             - learned 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
```



Images from https://openai.com/blog/clip/

Psudo-code from Radford et al. Learning Transferable Visual Models From Natural Language Supervision.

BLIP: Bootstrapping Language-Image Pre-training for Unified Vision-Language Understanding and Generation



Li J. et al. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation //arXiv preprint arXiv:2201.12086. – 2022.