TARAS SHEVCHENKO NATIONAL UNIVERSITY OF KYIV

Faculty of Computer Science and Cybernetics

Department of Mathematical Informatics

Samsung R&D Institute Ukraine (SRK)

QUALIFICATION WORK

for obtaining a master's degree

in training direction 122 Computer Science

on the topic:

ZERO-SHOT SKETCH-BASED IMAGE RETRIEVAL

Made by 2nd year student

Korobka Anastasiia Mykolayivna _____

(signature)

Academic adviser:

Professor, Doctor of Physical and Mathematical Sciences

Tereshchenko Vasily Mykolayovych _____

(signature)

Consultant:
Ph.D., Docent:

Radyvonenko Olga Sergiivna _____

(signature)

I certify that in this work there are no borrowings from the works of other authors without the corresponding references Student_____

(signature)

The work was considered and allowed to be defended at the session of The Department of Mathematical Informatics

«___» _____ 2021 y..

Protocol №____
Head of Department

V.M. Tereshchenko _____

(signature)

Kyiv-2021

ABSTRACT

Total page count 57, figure count 25, table count 4, information source count 32.

ZERO-SHOT LEARNING, IMAGE RETRIEVAL, SKETCH, INDEX, OPTIMIZATION, CLASSIFICATION, IMAGE PREPROCESSING, THRESHOLD, SALIENT CONTOURS, RANG METRICS, MEAN AVERAGE PRECISION, PRECISION, DEMO APPLICATION, QUANTIZATION.

Nowadays many processes are using Image Retrieval, Classification, or Clustering as subtasks. Digital information such as image content takes a strong place in communication in social media, business, or culture as digitalization of paintings or real exhibitions. So it becomes more popular to use image search as drawing a sketch on the fly or set another similar image instead of providing a textual description of the content. And at the same time, there are a lot of open challenges in sketch-based image retrieval like a domain gap or inner-class huge variety based on different abstraction levels of drawings.

We have analyzed existing methods of sketch-based image retrieval and classification on ZSL. We have noticed restrictions for input data format as detailed keywords description or sketch-image pairs, which limit the possibility of using arbitrary dataset. In our proposed methodology we suggest a preprocessing method for image-to-sketch transfer to simplify conditions for training models. Also, we have trained existing solutions as MobileNet and ADGPM on sketch-based data and speed up retrieval process with FAISS indexing.

For preprocessing phase we analyze and compare the performance of existing approaches from Computer Vision and Deep Learning: Canny Edge Detector, a style transfer net, Adaptive Threshold, and Salient Contours network. Next, we have implemented some experiments based on different effective nets from classical Computer Vision (MobileNet), classical Zero-Shot (ADGPM), and sketch-based Classification

(Doodle2Search) tasks. Furthermore, we have proposed to use the FAISS library for indexing to improve image search based on output embeddings of nets in an effective in time and accurate way. We have compared our performance based on local and general benchmarks.

Finally, we have implemented a Demo application on sketch-based image retrieval to demonstrate the effectiveness of our methodology and the possibility of using such a solution for searching some photos in the Gallery app on devices. Also, we have optimized our proposed approaches with the quantization process, so we can use the Demo in real time, without tangible delay locally without a huge amount of memory.

CONTENT

ABBREVIATIONS

ZSL – Zero-Shot Learning;

AI – Artificial Intelligence;

SBIR – Sketch-based image retrieval;

CBIR – Content-based image retrieval;

DL – Deep Learning;

ML – Machine Learning;

GZSL – Generalized Zero-Shot Learning;

GAN – Generative Adversarial Network;

VAE – Variational Autoencoder;

CVAE – Conditional Variational Autoencoder;

NN – Neural network;

AT – Adaptive Threshold;

SC – Salient Contours;

IR – Image retrieval;

CNN – Convolutional neural network;

ADGPM – Attentive Dense Graph Propagation Module;

ONNX – Open Neural Network Exchange Format;

P@k – Precision on k samples;

mAP@k – Mean Average Precision on k samples.

INTRODUCTION

Conventional approaches assume that we have a lot of data for all classes and they are available during the training phase. But this assumption is not always practical [5] [6]. Many applications have little or do not have data for some classes at all. As usual, new technologies are made up for a lack of physical resources, in case of ZSL new approaches are needed to perform satisfactorily with the support of new classes that appeared in production. That is why it is becoming a popular field in Computer Science – N-shot Learning [6], where N is the minimal number of samples for one class, the overview is demonstrated in SECTION 2. In some projects, the total number of classes can be huge [19], and obtaining sufficient training data for each of them becomes extremely challenging. Zero-Shot Sketch-Based Image Retrieval allows the algorithms to handle previously unseen classes during the test phase using only labeled data from a small dataset and some external knowledge about their relations, existing methods are considered in SECTION 4 and SECTION 6.

As computers have become more usable and allow huge memory resources to store, image data has become an integral part of virtual communication for users and businesses. Digitalization of paintings, real exhibitions, general photo collections are preferable areas of application in Image Retrieval today [1]. Fast development in digital photography has led to an enormous database of images, as a result, it is required to store and efficiently retrieve objects in time, etc. Now it is common for users to do an image search, instead of classical document retrieval, either by providing a textual description of the content or by proposing another image that is quite similar to the desired one. Sometimes it is indeed easier to provide a visual description as a sketch, if no other image is available, for example, as a pattern for matching images. So users can simply and fastly draw the sketch on their touch-based device.

There are a lot of challenges in ZSL because of the domain gap [3] between images and sketches, which contain only outlines of the desired object, so we have much less information for efficient retrieval. And more, it exists a large intra-class variance in sketches, it is possible because humans draw with different levels of abstraction the same objects [4].

In this work, we have analyzed and compared actual solutions for sketch-based image retrieval in terms of Zero-Shot Learning. We have detected restrictions based on input data format which provoke limitation on using arbitrary dataset such as pair-wise sketch-image data [20], textual description. We have compared different approaches [23] [24] [25] [26] for image automatic transformation to sketch to minimize the requirements of some models for training [15] [28], detailed overview is showed in SECTION 6. So this feature allows us to use straightforward models as ADGPM [15], from the ZSL classification field, and MobileNet [28], from classical classification task, without various losses, for example, triplet-loss in Doodle2Search [20]. Experiments results are demonstrated and analyzed in SECTION 9. Additionally, our methodology speeds up the retrieval process of existing solutions using FAISS technology [27] after obtaining embedding from the model. And finally, we have implemented an application to demonstrate and test our approach in real time.

In the future, we see that our approach can be used as an efficient search method on mobile devices. A lot of people have a large digital photo library on their smartphones [1], and finding the desired picture there becomes difficult as the user accumulates even more photos. Our proposal can be used to solve this problem. It can also be used to search on websites and in general for Internet search.

SECTION 1 IMAGE RETRIEVAL TASK

1.1 Image Retrieval methodology

Image Retrieval is a sub-task of Multimedia Information Retrieval which progresses significantly for the last years [4] [2]. This happens because people replace many of their needs in the virtual world. For example, in general, it can be the case of conversation (social media), business needs (customers on various e-commerce sites), or for a new reason it can be restrictions about COVID-19 to minimize social contacts in reality. As the demand for digital information increases, the need to store and retrieve data is also arising. It exists two main concepts for Image Retrieval: Content-based and Concept-based [2].

A content-based approach is any technology that aims to retrieve or organize image archives based on their visual content. On other hand, concept-based image indexing refers to process text-based information of images, for example, keywords, captions, or natural text [2]. But manual image annotation task for large datasets is very expensive and time-consuming, and more, it may be subjective and incomplete. This motivates researchers to work on content-based image retrieval (CBIR), where retrieval is guided by getting a query image. Figure 1 shows the diagram of the CBIR process.

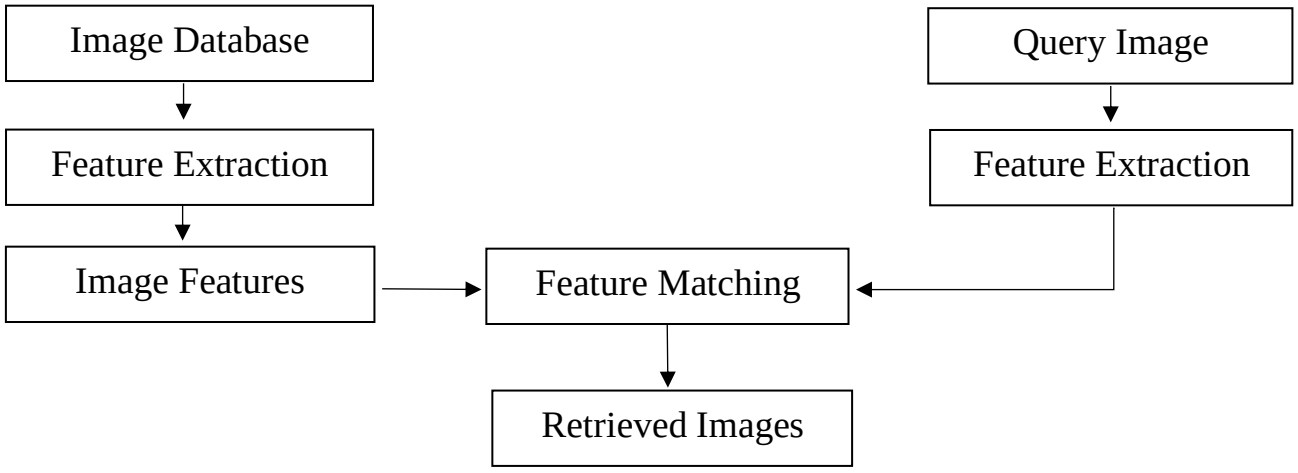

Figure 1 – Diagram of basic SBIR system

1.2. Human-Based CBIR

CBIR methods are used when text annotations are non-existent or incomplete, and more, they can potentially improve retrieved results even with text annotations. To bridge the problem of the semantic gap there are some techniques: human-centered computing, summarization, new media information, new features.

The main idea of human-centered computing is to satisfy or, furthermore, improve the retrieval results by making queries in users' terminology. This gives insight directly into the relationship between a human and a computer.

Understanding user satisfaction is inherently emotional. Due to this fact, efficient computing is good because its main idea is to focus on the understanding user's emotional specter and analyze how and what will be the best reaction. This approach also can be used for the retrieval task. For example, how relevance feedback works: some candidates (images) are shown to the user, who decides whether each image is relevant or not, after that it will modify some spaces: parameter, classification, semantic, feature to reflect the relevant and irrelevant samples. To implement these approaches, we calculate the degree of similarity or some distance of object, shape, or color of the queried images. But the fundamental problem of the CBIR is the semantic gap between computers and humans [3].

The key question is how to match images according to the computational features, patterns. Now the solution was found in merging such fields as computer vision and machine learning with image processing. Thanks to that it became possible to solve such problems as image abstraction, semantic gap [3] [4], dimension reduction, etc. Low-level features are color, texture, or shape, in general – visual features, that can be easily extracted from an image. Emotions, objects like classes, activities, or actions on the photo are so-called high-level features. The low-level concepts cannot describe the full situation; they are very limited relative to high-level ones. Humans naturally recognize objects based on prior knowledge of different things, it can be personal preferences or

experience, or similar situations or contexts. For CBIR system is hard to perform with high-level features, it can only retrieve images using low-level ones, so they can’t work with queries like “give the images of a white horse running on the field”. This process can be characterized as semantic gap [3].

1.3 N-Shot Learning

Another approach to solving the Image Retrieval task is N-Shot Learning. ImageNet database contains multiple examples for machine learning that's not always the case in specific areas like medical issues, where such AI technologies are used for diagnostic or selection of new drugs [5]. Typical architecture in DL relies on substantial data from based datasets like ImageNet, and before accurately retrieve images of new unseen in ImageNet class it should be trained on hundreds of images of this specific class. Furthermore, some datasets are greatly lacking in needed information or classes. In another hand, we need to train a model with millions of parameters all randomly initialized, to learn the classification task for unseen images using limited data sources. For machine learning, it is usual to get scarce data, and this is where N-Shot Learning comes in.

In N-Shot there are N labeled examples for each of the C classes, the total number of samples is equal to N*C, and this is the so-called support set S. We need to classify query set Q, in which each example corresponds to one of the C classes. In general, it exists three major sub-fields for this task: Zero-Shot Learning (ZSL), One-Shot Learning (OSL), Few-Shot Learning (FSL) [4] [5] [6] [7].

SECTION 2 TYPES OF N-SHOT LEARNING

2.1 Few-Shot Learning

It is a version of the approach, where we have more than one training sample per class, in general, the number is two or five examples [5]. For the FSL classification, problem researchers use Prototypical, Matching, and Relation nets [7].

The most popular algorithm is the Prototypical network [5][7]. In few words, the approach is based on an encoder that maps an image to vector in the embedding space, support images help to define a prototype per class, and after we calculate the distance between the encoder query and prototypes, finally, we can classify our request images. We can see that this kind of network does not classify the query directly, instead, this architecture learns how to map an image to the embedding space, where we can calculate distances. So, we do not have original distinguished points, we only calculate the distances between points in metric space. Furthermore, unlike the vectors, points are only a representation of coordinates, so no addition or scalar multiplication is used.

Matching Nets [7] are based on metric learning, which works with large datasets, where query image embedding is compared with support set embedding. In this case, compared to the classical classification problem, the model learns how to distinguish classes, but not specific features per category.

Relation Nets [7] are working with a relation module, that is on the top of embedding. It takes part in the computation process of class prototypes and general embedding. The main case is that the distance function is not defined before training, it is learnable. The input to the relation module is a query image embedding and class prototypes, they will be processed as couples, and after that, we will obtain a relation score for each.

2.2 One-Shot Learning

In OSL we have only one sample per class. The most popular architecture is Siamese Neural Network, which led to good results and then Matching networks [6] [7]. Siamese Nets [6] can be described as two parallel NNs, which have the same parameters and shared weights. These nets take different inputs and after processing, the prediction will be made based on combined outputs. So we work with two images, as inputs, if they are similar, the output will be a value lower than a threshold - a hyperparameter. During the training, we encourage models to output more similar feature vectors for one class example, and at the same time, we penalize them in the case of different categories. Based on this idea Siamese Nets learn a function, that can define the similarity level of input images.

An example of using this approach is passport checks at airports and border gates [6]: it is necessary to decide whether it is the same person in front of us and the document photo. In fundamental understanding, we need to develop such a CV system that can look only at two unseen to this moment images and decide whether they present the same object. In this case, we turn this problem from a classification task to a difference-evaluation challenge.

2.3 Zero-Shot Learning

In ML we utilize the metadata of the image to perform classification task, it is something like features associated with our target object. ZSL refers to a specific field in DL or ML where we want to get the model that can classify data based on a very limited number of samples or even no labeled ones, which means classification on the fly [4]. During training the network to recognize a query image for instances, it has already trained to classify more low-level features like face, legs, and others parts of the body.

So, to perform another task without additional training, we can re-use these feature detectors and tune them according to our requirements and desires. Now we can say that ZSL approaches can use DL models already trained by supervised learning in new ways without additional supervised learning, for example, classification task or information retrieval.

SECTION 3 MAIN PRINCIPLES OF ZERO-SHOT LEARNING

3.1 When using ZSL approaches

In some projects [8] the number of classes can be in thousands and to obtain enough data for training for each class we can spend years, it is a complex task [4] [5]. ZSL works to predict unseen classes during the training phase using only labeled data from some not-so-big set of classes and additional information or knowledge about class relations. So, ZSL approaches do not suffer from the appearance of new classes.

Secondly, to get an effective solution for recognition task images in the dataset should be taken from various angles in different environments (for example, it can be some levels of brightness) to enrich the dataset. Also, we need to work with a fixed set of classes and collect a lot of images for each class in the dataset to increase accuracy.

And it is important to note, in some situations obtaining verified information in a labeling process can only be achieved by an expert in desired areas. For example, in ML and DL fields the fine-grained object classification task is very common [4][8].

3.2 How ZSL works

The main idea of this approach is based on the existence of a marked training set that contains seen classes and for the evaluation phase, we use additional unseen classes. Both types are connected in a high dimensional vector space - semantic space, where knowledge from classes in the training set can be passed on unseen. So, these algorithms study the hidden semantic encoding, their attributes and features, and how to use them to predict new classes during evaluation.

Typically, ZSL fistful encodes instances to intermediate features. Which can be presented as seen classes. After that predicted attributes are started to compare with a set of unseen classes through the knowledge database. So, no additional training samples are required for new classes.

Classical ZSL is based on two main principles: inductive and transductive. In the first type, we can use in training only labeled classes, as so-called seen classes. And in a transductive method, we can train our models on seen labeled and unseen target classes simultaneously.

And what about the evaluation or test phase? In general test images come only from an unseen set of classes, it also is called conventional settings. However, in most practical uses we can define that target images comes from unseen and source classes, this mode is called generalized ZSL or GZSL [8]. And it is important to know, that existing ZSL approaches perform much worse in GZSL than in the classical settings.

3.3 ZSL: Bias problem

During training, a model based on ZSL approaches can converge to the bias problem. What is mean: in a general scenario, the targeted unseen classes are classifying as labeled seen classes and so we will get poor performance results and metrics in the GZSL task [8]. It can happen because processes of collection and labeling training dataset is hard and time-consuming, so it is difficult to get enough diverse information, ex. images, in statistical point of view.

ZSL approaches get their predictions about classes based on building connections between visual and semantic embeddings from source data. But just in this phase of establishing relations, we can get bias, however, to detect this problem we should do an evaluation process. It can happen due to the fact, that during training the visual settings and features from seen classes are usually mapped to some fixed anchor point in hidden

semantic space. And so during testing images of unseen classes are tended to classify as one of the seen source classes, for example, we can see an illustration of this problem in Fig. 2 [8].

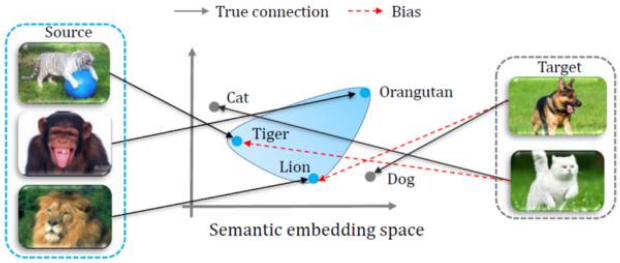

Figure 2 – Semantic space simple visualization

3.4 ZSL: Types of Semantic Spaces

In fundamental view, it is split into two main groups: Engineered and Learned Semantic Spaces. The first one contains Attribute, Lexical, and Text-Keywords Spaces, and the second one – Label-Embedding, Text-Embedding, Image-Representation Spaces [4].

In Attribute Spaces [4] each attribute describes a list of the term that corresponds to different properties of the category and in semantic space, it presents as one dimension. For example, it can be a word or phrase that can help to define an object as a class: color, shape, etc.

In Lexical Spaces [4] we use a dataset that can help us with semantic information. For example, some databases like WordNet, from which we can leverage the hierarchical relation between objects and use to classify our source and target classes together with their ancestors and decenters to form hidden space, where each dimension is one class.

If we decide to extract from text corpus with a description of each class some keywords and work with this kind of information, where each dimension – keyword, we will get Text-Keyword based Space [4]. In this case, we usually use as information

sources some websites, they can be specified on some areas or topic, or it can be simply Wikipedia. For example, in the Information (Document) Retrieval task we can use the TF-IDF approach – term frequency-inverse document frequency [8].

In Label-Embedding Space [4] we mostly work with class label embedding, where words or phrases are encoded to real number vectors. In such type of semantic space similar labels are nearby (their vector representations), and at the same time, dissimilar stay apart from each other.

Text-Embedding Space [4] is quite similar to text-keyword space, due to fact that class encoding comes also from the text description. But the main difference is in how we build our semantic space: in text-embedding, we don't just extract keywords as a dimension but also pass them through some model, that learns how we can them encode to our feature space, as input and the output is a vector that represents a prototype of the class [8].

In Image-Representation Space [4] is used images of one class as input for pre-trained model, usually this is architecture trained, for example, on the ImageNet, and as output, we will have a representation vector that forms a prototype for each class.

3.5 ZSL methods

ZSL methods can be split into two groups: Classifier-Based and Instance-Based Methods [8], a more detailed overview we can see in Appendix A and B [8].

3.6 GZSL methods

GZSL methods can be divided into three main groups: Embedding, Generative and Gating methods [8]. What about the first one [8] [11], this methodology learns how to

unify semantic and visual features into embedding based on similarity measurement. And, as we saw before, it exists a strong and popular problem on this approach – bias, so the anchors on target classes can be projected near seen classes from the training set, and as a result, we have a problem of miss-classification.

The next Generative approach works with own-generated synthetic features, it uses for this purpose such architecture as Generative Adversarial Network (GAN) [9] [12] [13] and Variational Autoencoder (VAE) [10] [12]. So we can say that this algorithm transforms the GZSL task to supervised learning. The generative method shows good performance results, but it still suffers, a bit in this case, from feature confusion problem: during the generation process the synthetic data of unseen classes can be mixed with seen, and finally when we start to classify query data we risk to have similar features but under different labels. So we can check trade-offs between source and targeted classes accuracies to get representative Harmonic Mean results.

In Gating methods [8] [14] to split train and test classes, it is a good practice to use a gating mechanism with two expert models. In this way we can avoid bias and feature confusion problems: we decompose our GZSL approach into two subtasks – classical ZSL and supervised classification, for example, in general, it is binary. But such a method is hard to implement and to learn such classifier since unseen classes are not used in training.

SECTION 4 ZSL FOR IMAGE CLASSIFICATION TASK

4.1 Quasi-fully supervised learning (QFSL)

This transductive method means that seen and unseen classes are used simultaneously in the training phase [11]. It works so that it is less sensitive to bias problem thanks to the shared semantic space and during the test phase it works without any modifications (Fig. 3) [11]. In this approach, the general idea is a combination of the classifier and the multi-layer NN. But according to the last competition results, this approach is less effective than later models.

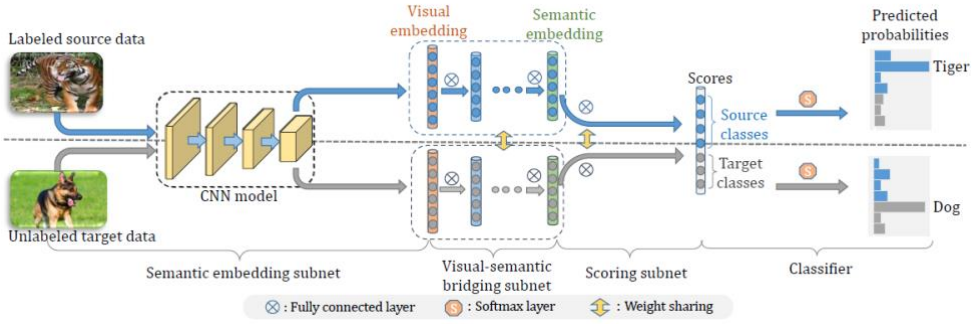

Figure 3 – Overall architecture of QFSL

4.2 Generative approaches

In TF-VAEGAN [12] semantic consistency was enforced at all phases of GZSL, to avoid feature confusing problem. It was proposed to use a feedback loop that iteratively refines new-generated synthetic features from sematic embedding decoder (SED), and then the output will be transformed into discriminative attributes to reduce the effect of ambiguity between seen and unseen classes during testing.

In LsrGAN [13] Semantic Regularised Loss, so-called SR-Loss, was proposed. It controls the generation process of semantic features between target and source classes and in this way helps WGAN to generate visual features.

But to finalize, the main problem in this approaches: it is difficult to determine whether the model will converge, and this process is time and resource expensive [9] [12] [13].

4.3 Boundary Based Out-Of-Distribution (OOD) Classifier

It is important to notice that training such types of architecture is quite hard because of very limited available information on unseen classes. So to avoid such a problem in boundary-based OOD classifier [14] we only use seen classes in the training phase. The algorithm contains a trick based on hyper-sphere that plays the role of shared latent space, where visual and semantic features are arranged class-wisely. For this purpose, Hyper-Spherical Variational Auto-Encoder (SVAE) is used. So next we need to find a center, that can be defined as the mean direction which is predicted by semantic features, and boundary for each class in our set. So each class can be represented by von Mises-Fisher distribution and thanks to that process we can split unseen classes from our training classes (Fig. 4) [14]. Finally, source and target classes are processing separately by two experts model.

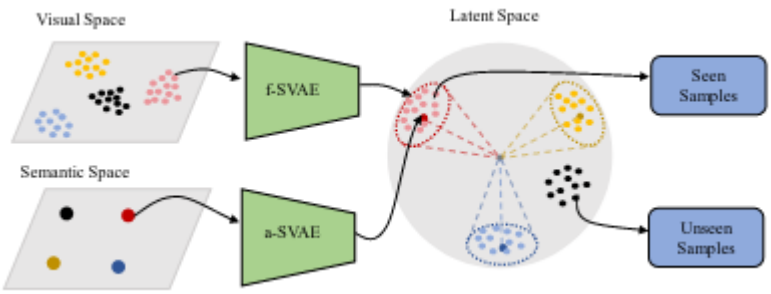

Figure 4 – Boundary Based OOD Classifier scheme

And so, this architecture is composed of two experts: the first one is an encoder of SVAE with linear softmax for seen classes categorization and the second one – CLSWGAN for unseen classes.

4.4 Attentive Dense Graph Propagation Module (ADGPM)

Another approach for the ZSL task is the graph CNN. Its structure allows related concepts to appear statistically related [15]. This enables generalization to unseen classes even in a low-data setting. Modern methods of ZSL use propagation schemes at each layer with Laplacian smoothing to make classification easier with increasing depth, as representations will be more similar. This causes distant knowledge (from remote nodes) to get diluted during propagation.

In ZSL this smoothing makes regressing classifier weights more difficult. To preserve information from distant nodes while utilizing the structure of the graph, we can use ADGPM [15], which allows us to peruse the hierarchical structure of the knowledge graph with more connections. They are added depending on the node's hierarchical relationships with an attention scheme to assign weights to its contribution based on the distance. Fine-tuning of the feature representation improves the results after training ADGPM (Fig. 5) [15].

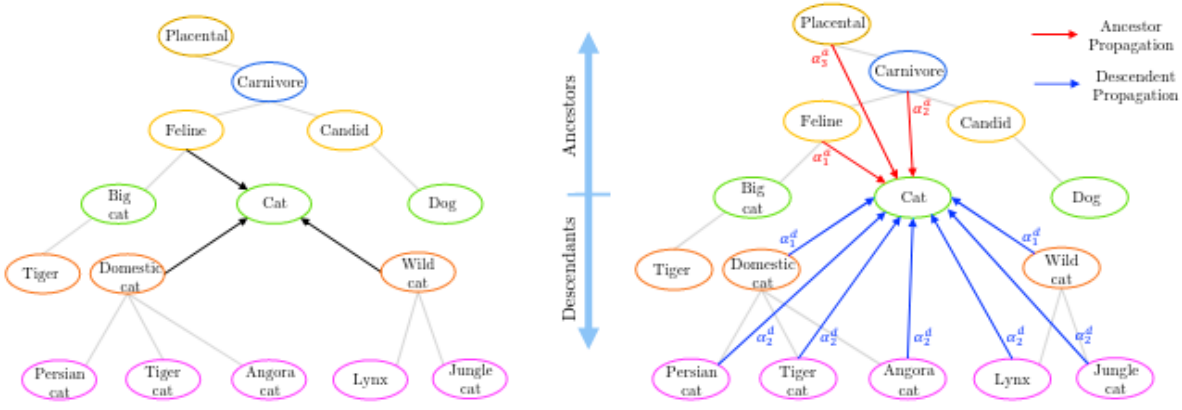

Figure 5 – Difference between Graph Propagation at the left and Dense Graph Propagation at the right side

ADGPM predicts weights for each node of the graph for the classifier. They are extracted from pre-trained ResNet. The graph is built from word embedding representations of each node in the knowledge graph. The network is two-phase (Fig. 6), with a descendent phase where knowledge is propagated from children nodes to their parents and an ancestor phase, where knowledge is propagated from ancestors to their children.

In the testing mode, CNNs are applied to new images to extract features, while the classifiers predicted by the GPM are used to classify them.

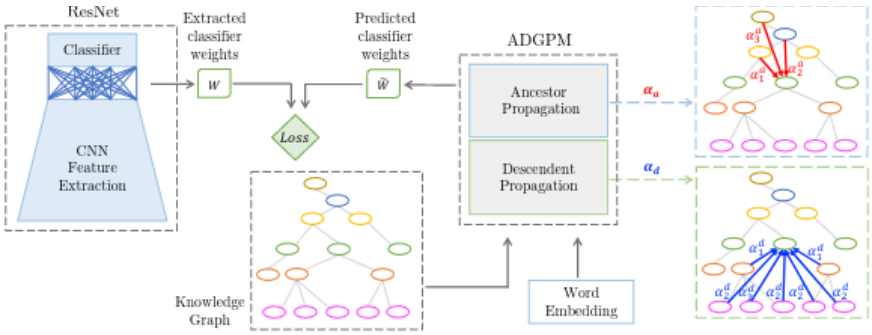

Figure 6 – ADGPM architecture [15]

It was very hard to compare the effectiveness level and performance of all these nets since the authors have used different benchmarks for the evaluation process on different datasets. But we think that the most promising and effective, especially in the context of a large database, will be the ADGPM approach.

SECTION 5 DATASETS FOR SKETCH-BASED ZSL

The first problem in the usual process of project implementation is to find a dataset to satisfy all requirements for future work. In sketch-based tasks, the main bottleneck is data scarcity, in other words, images can be crawled for free easier than drawn for the same needs. So, nowadays SBIR datasets contain in general few hundred categories with less than thousands of sketch samples per class without a big variety inside, for example, people draw the same object. The task of ZS-SBIR is challenging cause of the large domain gap between sketches and photos, and there is a big influence of different levels of abstraction which is a result of variant drawing skills or visual interpretation of the human mind.

Today, the most popular datasets on ZS-SBIR are Sketchy Database [16], TUBerlin [17], QuickDraw [18], and QuickDraw-Extended [18].

Based on our main idea, we will use the most commonly used benchmark dataset for evaluation: Sketchy, and the ImageNet [19] dataset transferred to Sketched-ImageNet to obtain a large variety of samples per class and simulate the Gallery application on phone or another device. And more, TUBerlin dataset is extremely imbalanced and QuickDraw does not contain sketch-image pairs to effectively evaluate models and in general train them according to our needs

5.1 Sketchy Database

It is a big collection of fine-grained sketch-photo pairs: 125 classes, which contains 75471 sketches of 12500 objects. All sketches are highly detailed and less abstract, so some models trained on this dataset can be collapsed and be less efficient in real-life

usage (Fig. 7) [16]. But at the same time, this limitation helps to avoid the domain gap problem during training time.

Figure 7 – Examples from Sketchy Database

5.2 ImageNet

This database [19] is a large set of data of human-annotated photos organized according to the WordNet hierarchical structure, particularly the class labels. It contains more than 20000 classes and it is the most used dataset for evaluation and benchmarks. During the annotation process, Amazon's Mechanical Turk crowdsourcing platform was used. It is important to note, that during the data gathering process, there were images made exactly for this project, so all samples have their copyright holders. Moreover, ImageNet is not distributed directly on the Internet, the access to data is organized only through links to them. One of the challenges of this dataset is the variety of image quality, size, etc. Also, there is the annual ImageNet Competition on Computer Vision from 2010 to 2017– ImageNet Large Scale Visual Recognition Challenge (ILSVRC). It was focused on two challenges: image classification and object detection. So there is image-level annotation in binary format which describes the presence of class on the sample, and object-level as a bounding box with class label respectively.

SECTION 6 SKETCH-BASED ZSL APPROACHES

6.1 Doodle2Search

This NN [20] has been trained on Sketchy [16] and TUBerlin [17] datasets. The main trick is a visually attended triplet ranking model which is commonly used in SBIR. Authors have proposed some new techniques to bridge the domain gap, for example, they use domain-agnostic embedding, which means that Gradient Reversal Layer helps to encoder understand how to get the most principal features from sketches and images. If we go deeper we will understand that only negative samples from the input triplet (sketch, positive image, and negative image) help to encoder network to split information and unite similar classes.

Doodle2Search contains three types of losses (Fig. 8) [20]:

1) triplet loss which helps to learn ranking metric;
2) domain loss helps to create a shared latent semantic space with sketches and images, and helps to avoid domain gap;
3) semantic loss works with word2vec additional information to force embedding quality of each class, especially when a dataset is large.

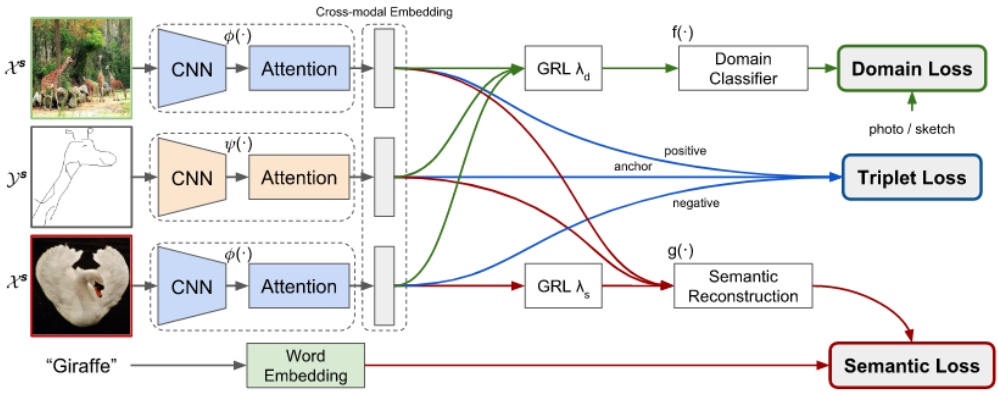

Figure 8 – Illustration of Doodle2Search architecture

6.2 SketchGCN

In this network to effectively collect semantic features and avoid domain gap graph CNN is used and for building semantic space based on visual features CVAE is used, in this case, we improve the generalization ability of NN [21]. In general, we can define three main parts (Fig. 9) [21]:

1) Encoding Net, which embeds sketches and images to common semantic space;
2) Semantic Preserving Net, which gets as input some visual embedded features and additional text information (embedding) to construct their shared category-level graph relations;
3) Semantic Reconstruction Net, which helps to preserve correct and efficient semantic encoding for extracted attributes.

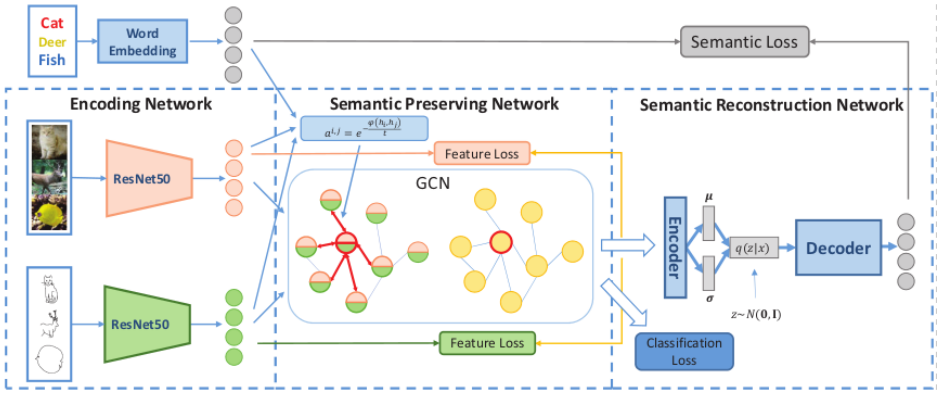

Figure 9 – Illustration of SketchGCN architecture

6.3 Stacked Adversarial Network (SAN)

In this approach, the authors utilize SAN [22] to generate quality samples, because multi-stages GAN, and Siamese Network (SN) replace classical algorithm, nearest neighbor search, to calculate and compare distances in semantic space (Fig. 10) [22]. And more, SN can help to transfer features of synthetic and real data into space where they

can be more discriminative thanks to Contrastive Loss. But this approach is very hard to train in terms of time and convergence.

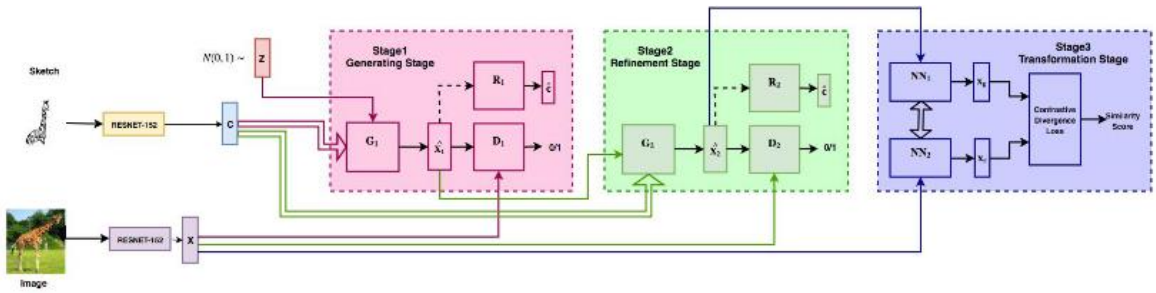

Figure 10 – Illustration of SAN architecture

Based on results demonstrated by authors in papers, we can notice that SketchGCN and Doodle2Search [20] work only with classical ZSL task, and only SAN try to cover both ZSL and GZSL fields.

SketchyGCN declares the best results on Sketchy Database [16], TUBerlin [17], and QuickDraw [18] datasets, but according to our purpose, we can try Doodle2Search in our experiment with nearly the same results in Sketchy.

SECTION 7 PROPOSED METHODOLOGY

In our research, we have noted that a process of training models has a lot of prerequisites for input such as pair-wise data as sketch-image, or text-description as keywords for semantic space. Moreover, it limits the number of datasets to use for this task or complicates the process of fine-tuning pre-trained models or training on specific proprietary datasets. And in general, no effective automatic methods were proposed for sketch-to-image transfer, so the problem of sketch-crawling is critical due to the cost in time and money of human resources. Furthermore, current models, which are used for the image retrieval task, are huge and the evaluation process is time-consuming and this is a problem for our task requirements.

So, in this work we suggest:

1) Sketched ImageNet. We have implemented different approaches to realize sketch-to-image transfer and based on the comparison results we have defined that the most sketch-like samples were obtained with Salient Contours DexiNed [26]. To go deeper in details of the analysis of dataset preprocessing see SECTION 8.
2) We have trained effective net for classical classification tasks such as MobileNet [28] based on an edged dataset – Sketched ImageNet. So in this case we have appreciated the idea of simplifying the process of training, so we can use data as input without complications and difficult structural transformations. And according to our requirements, this model can be used for low-performance devices.
3) We have tested classical ZSL classification approach ADGPM [15] for sketch-based image retrieval task to compare performance with MobileNet [28] trained on the edged dataset and Doodle2Search net [20], which is used specifically for the sketch-based task.

4) We have optimized our approaches with the model quantization process to get a possibility to use them on local servers without a big amount of resources.
5) We have used the FAISS library [27] to speed up the searching process in semantic space based on the outputs of our models.
6) We have developed a functional Demo application to present the full system for sketch-based image retrieval tasks.

Experiment results from points 2-6 are demonstrated in SECTION 9.

SECTION 8 EMPIRICAL PART: DATA PREPROCESSING

We have trained our models on a limited part of the full ImageNet dataset, particularly on 77 different classes. We have preprocessed it to Sketched ImageNet so we have transferred images to sketches format. We have tried this approach in aim to minimize the requirement of datasets to start training, we try to use only sketches during training without semantic embedding based on labels or other textual information, and we decide to try not to use sketch-image pairs.

We have tried various approaches, for example, Canny Edge Detector [23], Adaptive Threshold [24], style transfer technique [25], and Salient Contours model [26].

8.1 Canny Edge Detector

It is a multi-stage algorithm that is used for sketching edges of any image [27], but we need to remember that it works only with grayscale images. It was presented to the world in 1986. Fundamentally, it consists of steps. Starting this process with noise reduction based on Gaussian blur, which is a 5x5 kernel. After that, we use gradient calculations, which help to detect edge intensity, in other words, highlight changes in horizontal and vertical directions (Sobel filter), and directions (arctan). To make sure all edges are thin, we use non-maximum suppression: we go through all points in the matrix and find the pixels with the max value in edge directions, based on the angle from the angle matrix. The next step is the double threshold which detects three types of resulted intermediate pixels:

1) strong pixels which have a high level of intensity comparing to the high threshold, so they will go to the final resulted image;

2) non-relevant pixels with a too-small value of intensity compared to the low threshold, they do not go to the final image;
3) weak pixels whose value is between thresholds.

The final step processes weak pixels and decides if they are closer to strong or non-relevant pixels. This moment is so-called edge tracking by Hysteresis. It transforms a weak pixel to strong only if at least one pixel around the first one is strong.

8.2 Adaptive Threshold

In the general case, we use a global threshold for the image, where if the value is higher than the corner value, so this pixel is going to be white, else – black. But it does not work fine in case if the image has different levels of brightness in various regions. In this situation, researchers use the Adaptive Threshold technique [24] which works based on calculating local thresholds for small image areas. So, it performs better on images with varying illumination. It exists two times for threshold calculation by region:

1) based on the mean of neighborhood region, which defined by kernel size;
2) it is a weighted sum of neighbored values where the Gaussian window corresponds to weights.

The prerequisite for this algorithm is the same as for Canny Edge Detector [23].

8.3 Style transfer

It is a type of Computer Vision technique that transfers the content image to the style of another one, so-called style reference image, so that the first image preserves its core elements, but with patterns of the second. In this case, we use deep NN to effectively in time process this transformation. The basic structure contains two main parts: pre-

trained feature extractor, usually CNN, and transfer net, usually has encoder-decoder architecture [25].

8.4 Salient Contours

NN approaches are proposed to solve the problem of edge detection as a low-level Sobel operator, but in the way to predict edge-map with better performance. We use novel architecture DexiNed [26] which consists of an encoder with six blocks, that output intermediate edge-maps which will be concatenated and processed to obtain in the end a fused edge-map. Figure 11 [26] was illustrated the used approach.

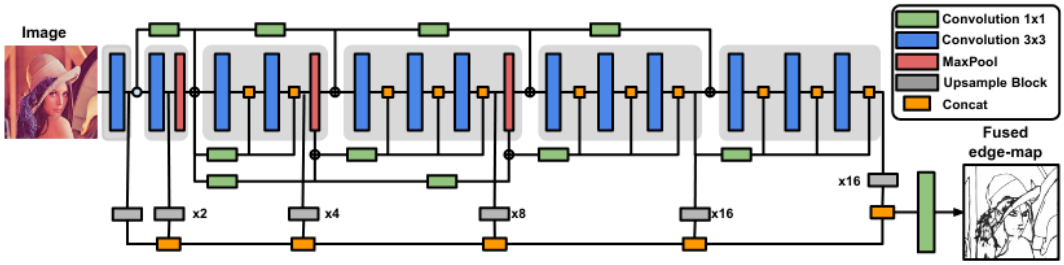

Figure 11 – DexiNed architecture visualization

8.5 Comparison of different methods

As a result, we can define that in the case of Canny Edge Detector [23] and style transfer model in some classes we can find out bad results based on too much simplification of images with too high a brightness level and in other cases too much noise is presented in sketch transformed image. So it can be so-called an external-class transformation problem because it becomes impossible to distinguish classes or main objects.

And what about Adaptive Threshold [24], so with this approach, we have had quite correct images, but it suffers from inner-class transformation to sketch problem. It means that in the general case all classes are successfully passing the image-to-sketch process, but if the image has a lot of objects, for example, a snake in the grass, so we still have a risk to get too noisy images, that cannot be fixed by the implementation of erosion or Morphological Mean (Opening) because of different nature of noise in the full dataset. Some successful examples of image-to-sketch transformation are presented in Figure 12 and a set of failed samples in Figure 13.

Figure 12 – Adaptive Threshold successful examples of transfer image-to-sketch

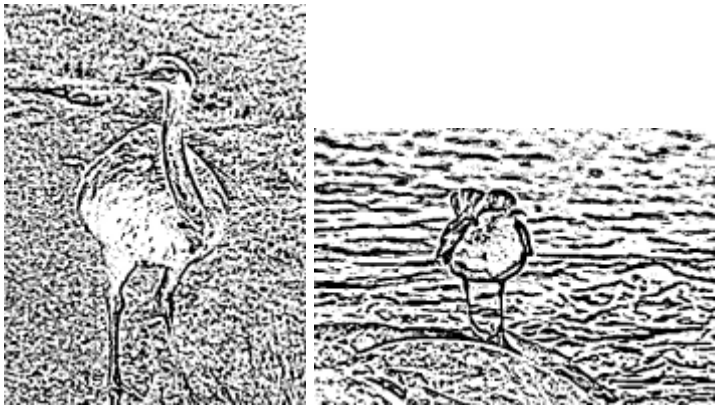

Figure 13 – Adaptive Threshold noised examples of transfer image-to-sketch

The best results are presented based on the Salient Contours Model [26], some examples are demonstrated on Figure 14. But sometimes it suffers from the inner-class problem as in the Adaptive Threshold approach [24], however, in this case, is a smoothing effect or oversimplification (Fig. 15), but it is still recognizable by category.

Figure 14 – Salient Contours method successful examples of transfer image-to-sketch

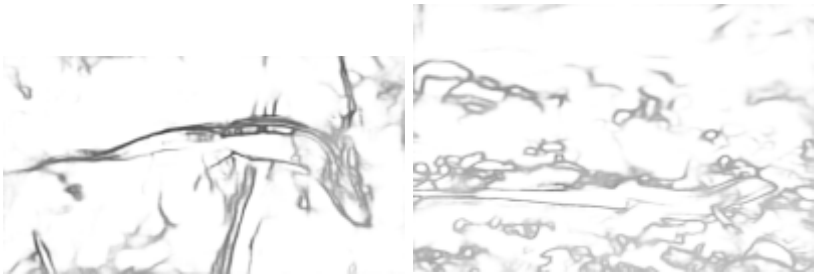

Figure 15 – Salient Contours method too smoothed examples of transfer image-to-sketch

SECTION 9 EMPIRICAL PART: ANALYSIS OF EXPERIMENT RESULTS

9.1 Demo application

We have implemented an application with functionality to be able to work with different sketch data source based on sketch-based image retrieval requirements:

1) opportunity to load files (sketched image or real sketch);
2) draw and edit the sketch yourself on-fly.

The obtained object is processed by NN and the output is embedded image features on which we will search the nearest photos in our database with help of FAISS [27] indexing.

Example of the interface of Demo application on Figure 16.

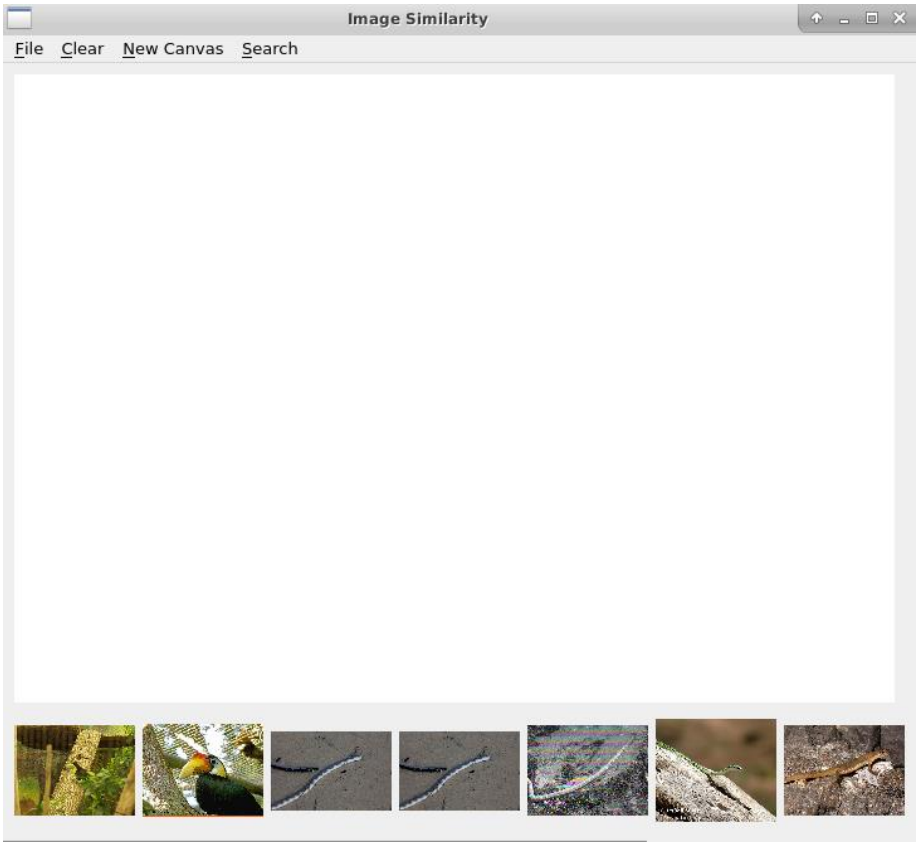

Figure 16 – Demo application UI

In our work, we have tested various NN architectures in ".tflite" format to quantize and facilitate calculations on the local machine.

9.2 FAISS

It is a library for efficient calculation (searching and indexing) of large collections of data based on similarity search and clustering developed by Facebook AI Research [27]. In original it is implemented on C++, but also it is completely wrapped for Python. Some principal features are written on GPU, but it also can accept input from the CPU. In our application, we use FAISS to index our encoded image embeddings. When indexing vectors, FAISS clusters them with a fixed amount of centroids and employs them to find a compact efficient representation of their coordinates relative to the centroids (Fig. 17) [27]. This process is called quantization, as it compresses vectors. The compressed vectors are indexed in a space search tree afterwards. When looking up, FAISS converts the coordinates to the quantized format, looks them up in the compressed search tree index, and converts the search results back to original decompressed coordinates. In our approach, it preserves both vector coordinates and their indices in integer format. This approach allows us to support billions of vectors in memory on one server or machine as we are not required to store full uncompressed vectors. FAISS supports different metrics such as Euclidean (L2), dot, and cosine similarity.

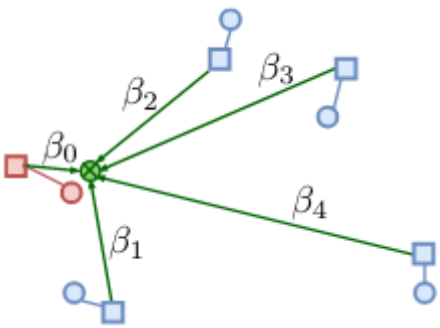

Figure 17 – FAISS approach illustration

9.3 MobileNet with FAISS

First, we try an approach based on the MobileNet [28] model. We have chosen this architecture because this is an efficient classification model which is tailored for mobile devices or another resource, especially memory, limited environments. It uses inverted residual blocks and lightweight depth-wise convolutions, so we remove non-linearity in narrow layers. The difference between classical residual block and inverted is demonstrated on Figure 18 [28], based on it we can note that diagonally hatched layers in the left image are the layers with a big number of channels, and in the right– the bottlenecks.

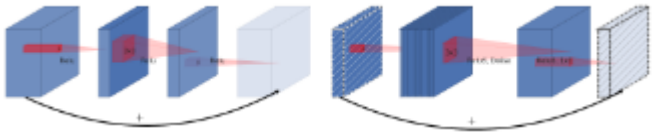

Figure 18 – Visualization of difference between classical and inverted residual blocks

In our approach we add to MobileNet feature extractor, we use pre-final layer to obtain embedding, FAISS indexation. We have trained our approach with Adaptive Threshold and Salient Contours preprocessing on the full test dataset and part, particularly one category. For the training phase, we use early-stopping technology.

As a result, for evaluation on only one concrete class, we have good performance, namely, we can see strong relations between sketched object direction and retrieved images. We demonstrate some examples on Figure 19 and Figure 20. So this experiment was effective on time and image retrieval response, we check the quality by visual user satisfaction on image similarity, because in our dataset (Sketched ImageNet) on just one class we cannot use usual metrics for IR task cause of not representative metrics' output compared to another inter-class evaluation. It is also important that before search we indexed the test set to obtain efficient and accurate search results based on the model's output (embedding), and so in Sketched ImageNet, we have one-to-one pairs, not like in

general ZS-CBIR datasets many-to-one, as a result we will find exactly the same one image, but it will be not representative and competitive for benchmark comparison.

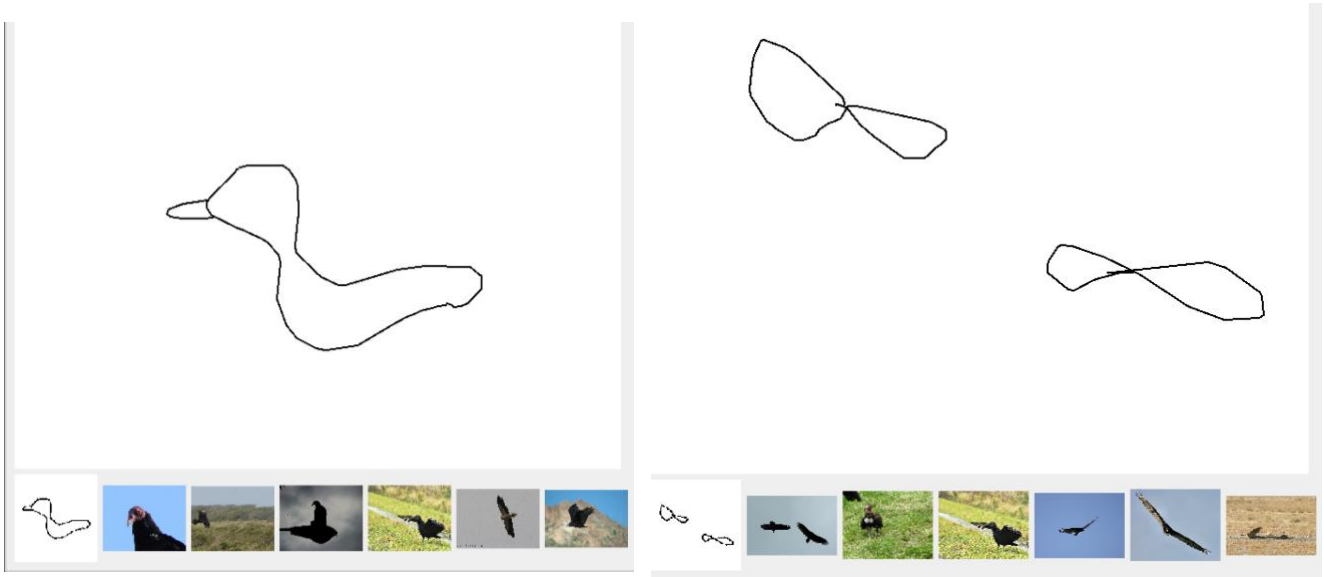

Figure 19 – Inner-class IR results for MobileNet+FAISS based on Adaptive Threshold preprocessing for “eagle” category

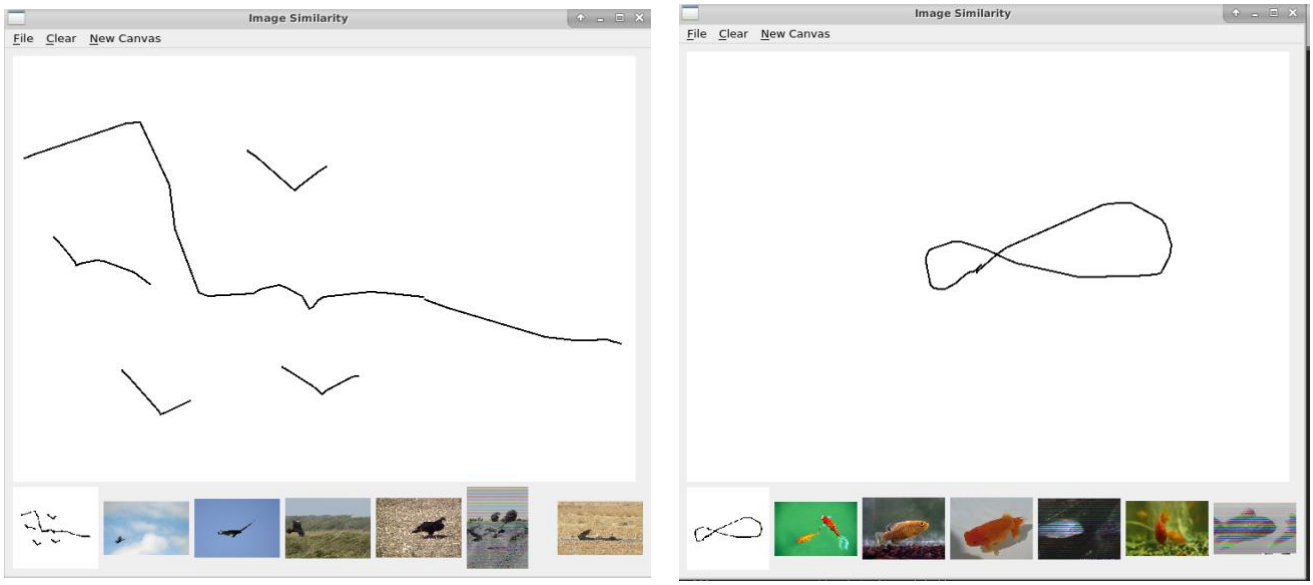

Figure 20 – Inner-class IR results for MobileNet+FAISS based on Salient Contours preprocessing for “eagle” and “fish” categories

So next step was evaluation on the full test set. In this case, the results were very poor and ambiguous. In the Adaptive Threshold [24] preprocessing experiment we can notice, that the model can retrieve images with similar shape description (tendency) of the general object on the sketch (Figure 21), for example, the thin lizard is matched with

another lizard, some flat animal and then some failed images; or rounded fish is similar to turtle's carapace or beard body, etc. Total epoch number for training: 121.

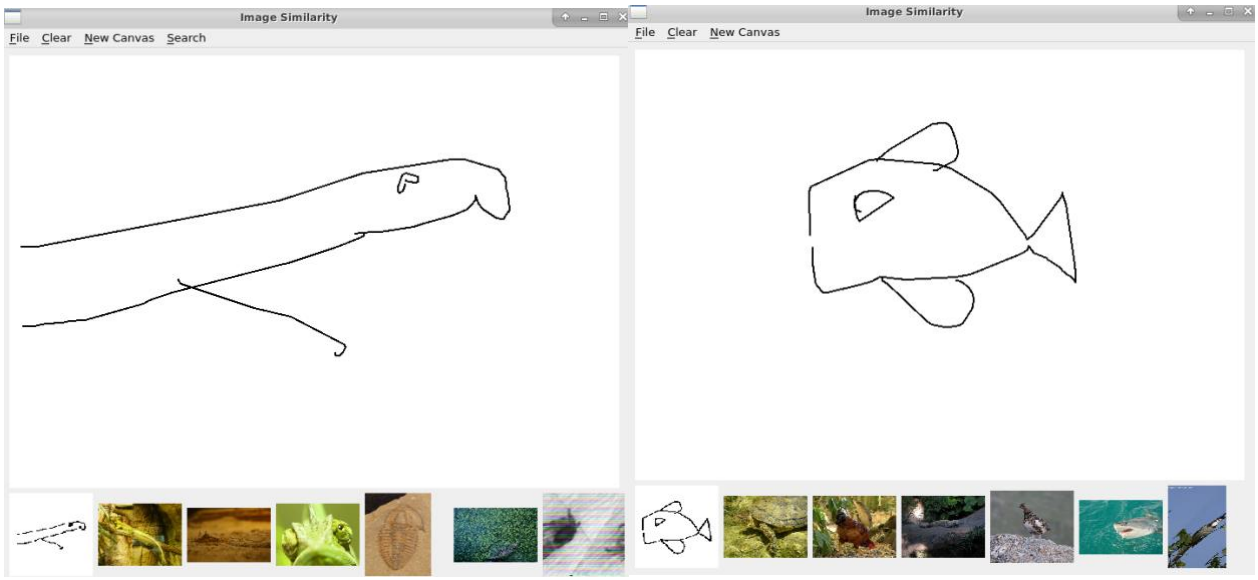

Figure 21 – Inter-class IR results for MobileNet+FAISS based on Adaptive Threshold preprocessing for "lizard" and "fish" categories

The same experiment was repeated with Salient Contours [26] preprocessing and the results were, unfortunately, worse. For example, query sketch simulated fish was retrieved, unexpected, as bird categories (Figure 22). Total epoch number: 107.

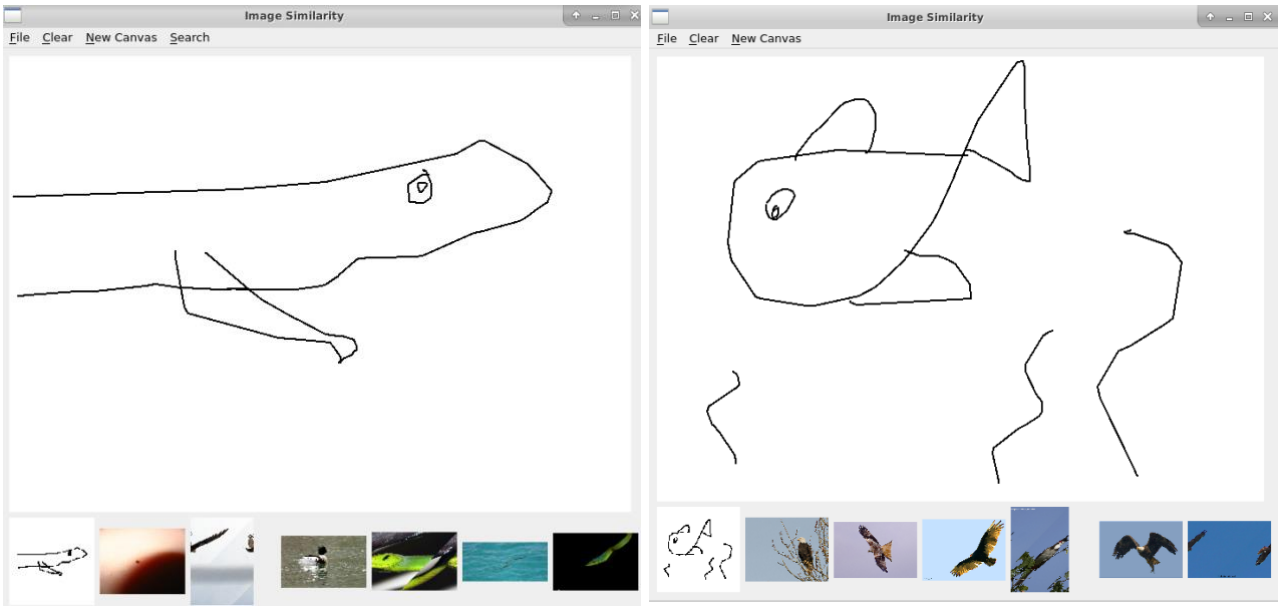

Figure 22 – Inter-class IR results for MobileNet+FAISS based on Salient Contours preprocessing for "lizard" and "fish" categories

9.4 ADGPM with FAISS

We have decided to combine the graph approach [15] on the classical ZSL classification task with FAISS [27] because the first one has demonstrated competitive results between other architectures. The results are demonstrated on Figure 23, as we can see our approach retrieves similar objects based on class and direction or pose. But it is important to notice, that is slower than a typical MobileNet [28] based proposal. Total epoch number for training: 1000.

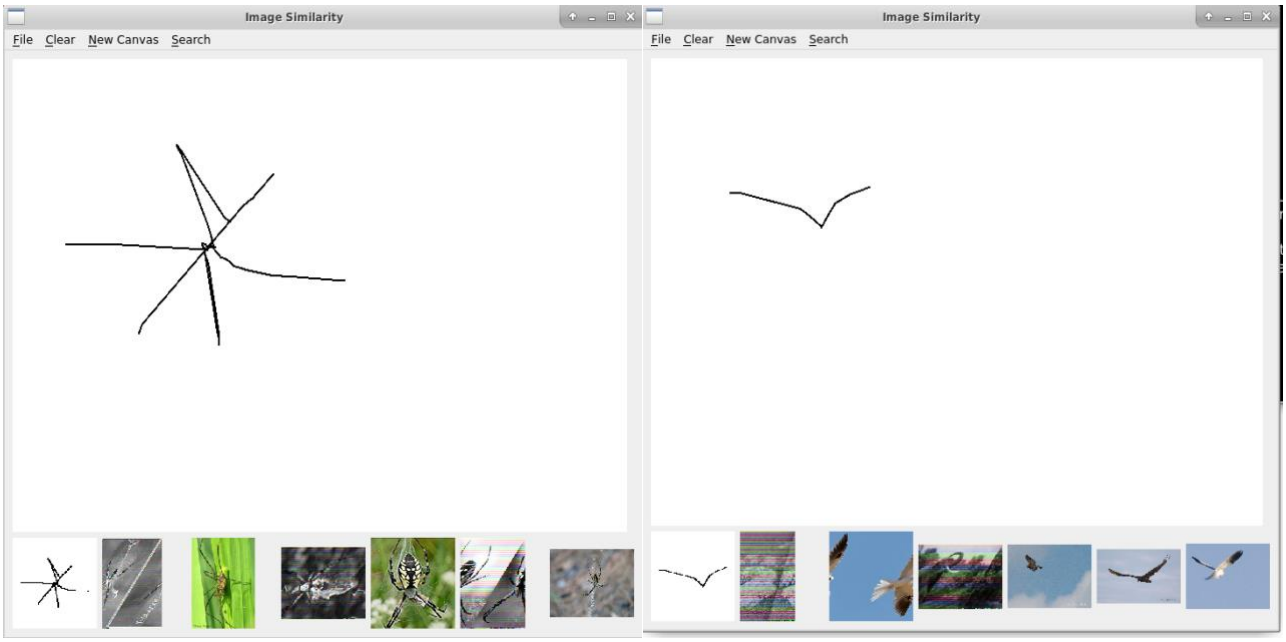

Figure 23 – Inter-class IR results for ADGPM +FAISS based on Salient Contours preprocessing for “spider” and “eagle” categories

9.5 Doodle2Search with FAISS

This experiment aims to try to improve classical Doodle2Search [20] results and to expand the application possibility of this approach also to the GZSL task [4]. Basically, the proposed authors' model is quite slow in the evaluation phase, so we have decided to speed up it by adding FAISS based on embedding outputted by the net.

As we see in Figure 24 the retrieval results have a good generalized ability for different levels of sketch abstraction. And more, this model outputs the first image in retrieval results with influence to a direction or pose of the object on the sketch. But it is important to note, that in this experiment in the net-outputted query in the general case the first image is very similar to the sketch, it is right, but others are from different classes, for example, not only “fish”, but also other strange chosen categories.

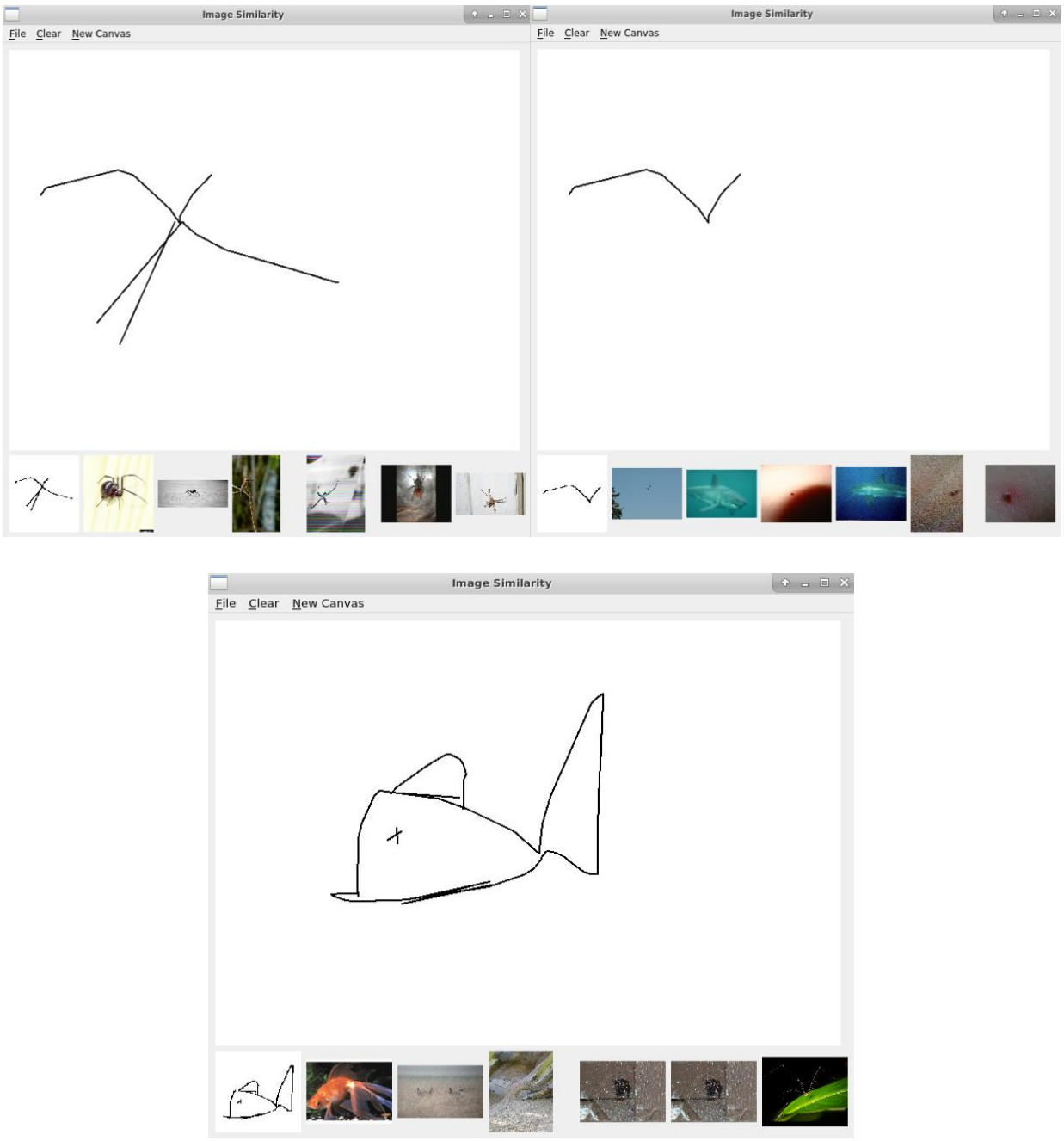

Figure 24 – Inter-class IR results for Dood2Search+FAISS based on Salient Contours preprocessing for “spider”, “eagle” and “fish” categories

So, if you want to draw a sketch and get a query of more similar results, for example, not the same as your art image, but from the same category as all flying birds based on your illustration in Gallery application, it will be better to choose ADGPM +FAISS approach. Total epoch number for training: 27.

9.6 Quantization of model

When we try to implement a net for the application it is important to remember about memory resources on your server and the time that we want to spend waiting for the model's output results. In our case, we use a model trained with the PyTorch framework. But we have faced with problems of model size, when, at first, it was loading with application start. It was a time-consuming process and it needed a lot of memory. So we decided to transfer our models from torch ".pth" to ".tflite" format [29]. Tensorflow Lite is a module of Tensorflow for quick and efficient quantized model inference. But it is a problem since PyTorch does not support direct conversion to TensorFlow Lite. So we would need to transfer for the first time to the ".onnx" format (Open Neural Network Exchange Format) [30], as an intermediate representation, and then to Tensorflow. It was a difficult and multistage process, the full transferring including in-between states:

PyTorch -> ONNX -> Tensorflow -> TFLite.

The first phase PyTorch-to-ONNX is implemented quite easily as the method "torch.onnx.export()". The main goal of this format is to simplify the communication between different AI frameworks and make it possible to support any type of platform. The pre-requisites are the trained model on the evaluation phase and a simple dummy input of the corresponding torch model shape. It needs to remember, if it will be needed to test the newly converted model on ONNX format [30], we need to convert "torch.tensor" to equivalent "np.array". The output file will have a format of graph representation.

So, next, we need to transfer to the Tensorflow format, in this case, we have used the onnx-tensorflow library. The library allows converting a model from ONNX intermediate representation into a Tensorflow saved model. A downside of this conversion process is that it still keeps the model weights in the torch-native NCHW tensor format, which causes some issues later on during model quantization.

This issue occurred when converting the ADGPM [15] model, as some of the Convolutional and Pooling operations used by the ResNet model are implemented in the TFLite only for the NHWC image storage format, which is native for Tensorflow, while the NCHW format used by Torch is unsupported. To bypass this problem, we have decided to convert the model weights from the NCHW format to the NHWC format. This functionality is supported by the openvino2tensorflow library, which has built-in support for transposing necessary model weights during the conversion process. But to use it, we needed to convert the model into the OpenVINO [31] intermediate representation first. OpenVINO toolkit has built-in support for converting ONNX models, thus we have used our ONNX-based model and converted it into the OpenVINO (.xml + .bin) format. So the process of format transferring looks like:

PyTorch -> ONNX -> OpenVINO -> Tensorflow -> TFLite.

This has enabled us to employ the openvino2tensorflow library, to export the OpenVINO model [31] into a Tensorflow saved model which uses the Tensorflow-native NHWC format.

After obtaining the converted Tensorflow model (with correct image format, if necessary), we quantized it into a Tensorflow Lite [29] model with int8 activations and weights for more efficient memory usage and inference. We employed a method called post-training static quantization. It involves rounding down the weights from float32 32-bit floating-point format into the int8 8-bit integer format while employing a calibration dataset, which is a subset of data, which is evaluated by the quantized model. These results are used to estimate the distribution of the input data, to adapt the rounded down

weights to the dataset. This allows us to maximize the achieved accuracy while keeping the model size low and inference fast.

9.7 Results comparison

In this sub-section we will demonstrate results comparison based on numeric metric benchmarks used in the ZSL Image retrieval task: mean Average Precision, Precision, Recall.

Precision@K (P@K) [32] computes the percentage of relevant results in top-K samples and ignores those that are ranked lower than K. After that we divide that number by K. MAP is an Average Precision across multiple queries, it considers the rank position of each relevant image (K1, K2…Kn) in results, so it computes P@K for each K1, K2…Kn. In other words, average precision (AP@K) is equal to the average of P@K, and mAP@K [32] is summing AP@K and divide this by several queries. Precision metrics are calculating the proportion of actually correct positive identified samples on all positive identified, particularly, false positive and true positive. And in the same time Recall identifies the proportion of actual positive samples on a general number of samples of this class – true positive and false negative examples.

We have used Sketchy Database [16] and mAP@100 and @200, and P@100 and P@200 as common for paper and competition, results are demonstrated in Table 3.

Net type/ Metrics	Recall @100	Recall @200	mAP @100	P @100	mAP @200	P @200
MobileNet+FAISS with AT	0.97	0.978	0.257	0.153	0.193	0.113
MobileNet+FAISS with SC	0.983	0.99	0.312	0.197	0.242	0.144
ADGPM +FAISS	0.99	0.99	0.333	0.2	0.25	0.144
Doodle2Search+FAISS	0.999	0.999	0.618	0.514	0.543	0.432

Table 3 – Metric results based on Class evaluation

So, as we can see our approaches have a big gap in results on Sketchy Database relatively Doodle2Search model, but it needs to remember, that it was training on Sketchy and TUBerlin comparing to others. Also, results in Table 3 can be described as a difference in approach in fundamental meaning, so Doudle2Search was training with triplet input and trying to learn how to find especially the same image as in the sketch, so it was based on pairing, and in case of ADGPM and MobileNet we try to implement ZSL approach for image retrieval based only on sketch-similarity. That's why ADGPM and MobileNet can base their prediction on embedding similar images on shape content, for example, so they can be from different categories. Recall metrics show that all approaches are successfully finding needed images on the database, but another equation is on Precision, especially in what position is these results.

Additionally, we have checked model performance based on Recall@1 and @5, mAP@5, and P@5 metrics for the local evaluation for our application based on retrieval query, numerical results are shown in Table 4.

Net type/ Metrics	recall@1	recall@5	mAP@5	P@5
MobileNet+FAISS with AT	0.825	0.895	0.647	0.485
MobileNet+FAISS with SC	0.888	0.936	0.733	0.578
ADGPM+FAISS	0.946	0.967	0.814	0.659
Doodle2Search+FAISS	0.963	0.982	0.869	0.744

Table 4 – Metric results based on Instance evaluation

As we can see from the table above, we have not outperformed the existing approach for ZSL sketch-based image retrieval (Doodle2Search), but our proposed method based on ADGPM [15] shows quite good performance on local evaluation based on Instance, in other words, how good different sketches on one same image are retrieving. But it is important to notice, that Doodle2Search [20] was trained on Sketchy Database [16] and TUBerlin [17], and at the same time ADGPM was trained on Sketched ImageNet. The worst results were demonstrated by MobileNet with AT preprocessing, we can guess this effect can be justified by the presence of too noised images in the dataset.

Finally, we want to show some illustrations on image retrieval results of the ADGPM+FAISS approach on Figure 24. As we can see the Top-20 images are quite

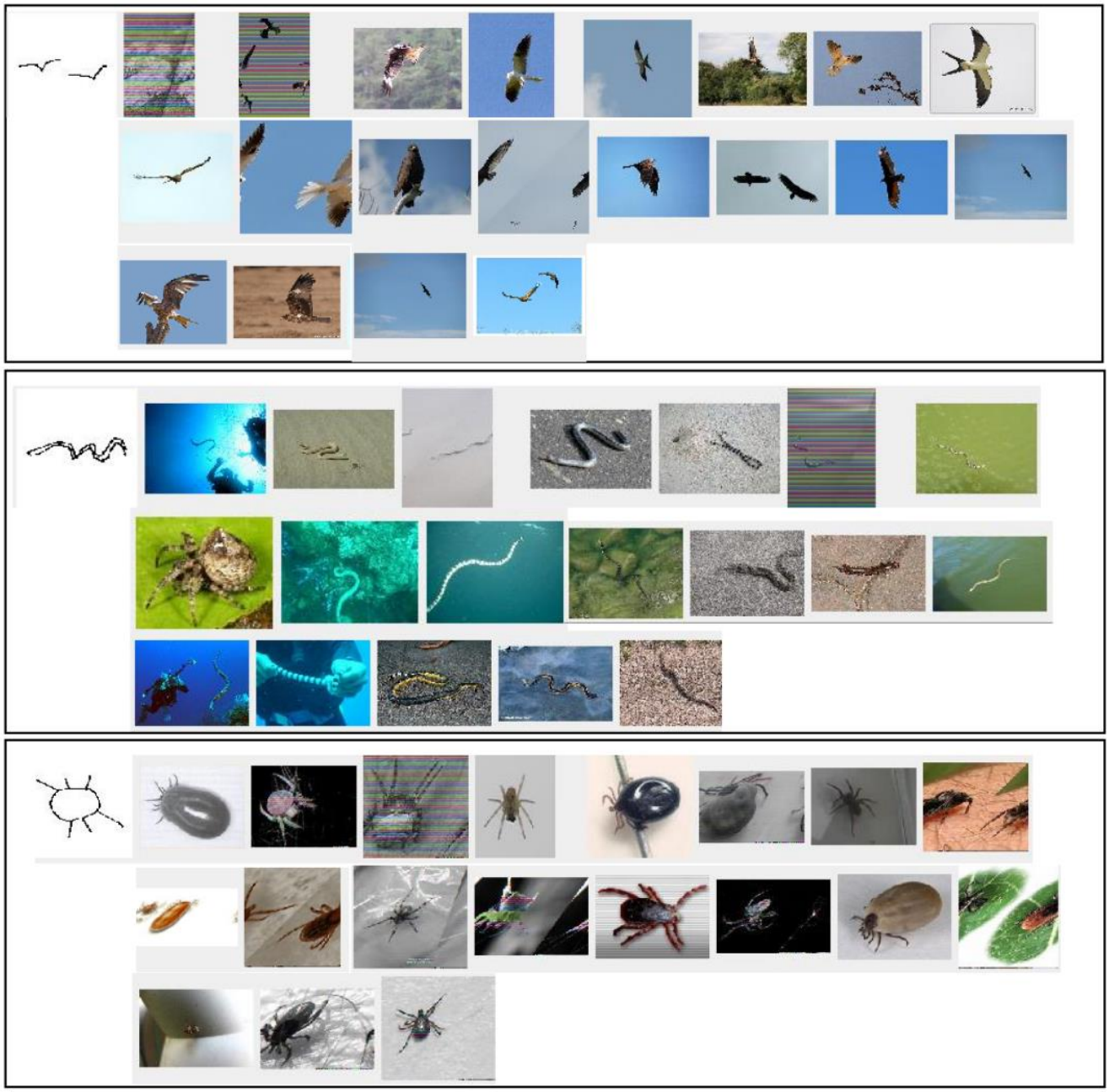

Figure 25 – Top-20 Image Retrieval results based on ADGPM+FAISS on “flying bird”, “snake” and “spider-insect” sketches

CONCLUSION

So, in this work we have analyzed existing methods of classical ZSL classification and sketch-based IR tasks, detailed overview is presented in SECTION 4 and SECTION 6 respectively. Based on this research, we have noticed that actual solutions [20] [21] [22] have strict prerequisites for input data format as sketch-image pairs, keywords descriptions, etc. Due to this fact, the number of suitable datasets [16] [17] [18] is limited, and there are no proposed automatic image-to-sketch transfer methods for use with an arbitrary proprietary dataset. So, our proposed methodology contains such a pre-processing method, experiment results are demonstrated in SECTION 8, and improvements of existing effective architectures [15] [28] such as simplification of input format based on sketch-data and speeding up retrieval process with FAISS [27] indexing, see SECTION 9.

We have trained our approaches on the proposed Sketched ImageNet, for this purpose we have implemented different techniques of image preprocessing to implement the image-to-sketch transfer. Especially in this step, we compare such methods from the conventional threshold and to a deep learning model. The main problem which we faced is too noisy images in results. The causes for this can be the huge variety of image scenes, quality, and size of photos in the classic ImageNet dataset [19]. So, Canny Edge Detector [23] and style-transfer model [25] have demonstrated too noisy images in inter- and inner-class diversity. As a result, for preprocessing phase, we have chosen two approaches for our further experiments: Adaptive Threshold [24] and Salient Contour model DexiNed [26]. For the first one, the problem was still actual in some cases, but in general, it was possible to recognize to which class an image was related. A detailed comparison is demonstrated in SECTION 8.

As a next step, we have tried and compared different approaches from various DL fields for our task, for example, we have used MobileNet feature extractor as an effective

NN for the classical classification task, ADGPM [15] as a NN from classical ZSL classification task and Doodle2Search [20] as an example of a model for ZSL sketch-based image-retrieval task. For the purpose to implement an effective application for sketch-based IR, we have combined these nets with FAISS [27] indexing library. The last works with embeddings from nets and indexes them in a fast and efficient way that allows using search in an image database in real-time without a big delay.

Also, we have faced the problem of model size and time to load it in our demo, so we have implemented a solution to convert the model from PyTorch format to TFLite.

Finally, our models are not outperforming the Doodle2Search model [20], but in our local metrics, the performance of ADGPM [15] is nearby. We are satisfied with obtained results because our approach retrieves quite good images based on an abstract sketch. So if you want to search an image in Gallery and you are remembering only the shape of the object without precise details, our model can give you some images similar to the illustration.

In future work, we plan to concentrate on improving performance based on the ADGPM approach [15]. We believe that training this deep net on full Sketched ImageNet will be a base for finding stronger relations in the hidden graph and it will be useful for semantic space encoding.

REFERENCES

1. Lew, M.S., Sebe, N., Djeraba, C. and Jain, R. – 2006 – Content-based multimedia information retrieval: State of the art and challenges – ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM), 2(1) – p. 1-19.

2. Multimedia Information Retrieval [Electronic resource]: Schmitt, I. – 2006 – Brandenburgische Technische Universität Cottbus, Institut für Informatik –

https://docs.google.com/viewer?url=http://inka.htw-berlin.de/mp06/docs/Schmitt.pdf

3. Datta, R., Li, J. and Wang, J.Z. – 2009 – Exploiting the human–machine gap in image recognition for designing CAPTCHAs – IEEE transactions on information forensics and security, 4(3) – p. 504-518.

4. Wang, W., Zheng, V.W., Yu, H. and Miao, C. – 2019 – A survey of zero-shot learning: Settings, methods, and applications – ACM Transactions on Intelligent Systems and Technology (TIST), 10(2) – p. 1-37.

5. N-shot learning: Learning More with less Data [Electronic resource]: Deep Learning Production – 2019 – https://blog.floydhub.com/n-shot-learning/

6. What is one-shot learning? [Electronic resource]: Ben Dickson – 2020 –

https://bdtechtalks.com/2020/08/12/what-is-one-shot-learning/

7. Understanding Few-Shot Learning in Computer Vision [Electronic resource]: V. Lyashenko – 2021 – https://neptune.ai/blog/understanding-few-shot-learning-in-computer-vision

8. Song, J., Shen, C., Yang, Y., Liu, Y. and Song, M. – 2018 – Transductive unbiased embedding for zero-shot learning – In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition – p. 1024-1033.

9. Generative Adversarial networks and how they work [Electronic resource]: Evergreen Group – 2019 – https://evergreens.com.ua/ua/articles/gan.html

10. How does VAE work? [Electronic resource]: Basic course NeuroHive – 2018 –

https://neurohive.io/ru/osnovy-data-science/variacionnyj-avtojenkoder-vae/

11. From Zero to Hero: Shaking Up The Field of Zero-shot Learning [Electronic resource]: Alibaba tech – 2018 – https://alibabatech.medium.com/from-zero-to-hero-shaking-up-the-field-of-zero-shot-learning-c43208f71332

12. Narayan, S., Gupta, A., Khan, F.S., Snoek, C.G. and Shao, L. – 2020 – Latent embedding feedback and discriminative features for zero-shot classification – arXiv preprint arXiv:2003.07833.

13. Vyas, M.R., Venkateswara, H. and Panchanathan, S. – 2020 – August. Leveraging seen and unseen semantic relationships for generative zero-shot learning – In European Conference on Computer Vision – Springer, Cham – p. 70-86.

14. Chen, X., Lan, X., Sun, F. and Zheng, N. – 2020 – August. A Boundary Based Out-of-Distribution Classifier for Generalized Zero-Shot Learning – In European Conference on Computer Vision – Springer, Cham – p. 572-588.

15. Kampffmeyer, M., Chen, Y., Liang, X., Wang, H., Zhang, Y. and Xing, E.P. – 2019 – Rethinking knowledge graph propagation for zero-shot learning – In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition – p. 11487-11496.

16. Sangkloy, P., Burnell, N., Ham, C. – 2016 – The sketchy database: learning to retrieve badly drawn bunnies – ACM Transactions on Graphics (TOG), 35(4) – p. 1-12.

17. Eitz, M., Hays, J. and Alexa, M. – 2012 – How do humans sketch objects? – ACM Transactions on graphics (TOG), 31(4) – p. 1-10.

18. QuickDraw dataset and QuickDraw-Extended [Electronic resource]: Google Creative Lab – 2017 – https://github.com/googlecreativelab/quickdraw-dataset

19. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K. and Fei-Fei, L. – 2009 – Imagenet: A large-scale hierarchical image database – IEEE conference on computer vision and pattern recognition – p. 248-255.

20. Dey, S., Riba, P., Dutta, A., Llados, J. and Song, Y.Z. – 2019 – Doodle to search: Practical zero-shot sketch-based image retrieval – In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition – p. 2179-2188.

21. Zhang, Z., Zhang, Y., Feng, R., Zhang, T. and Fan, W. – 2020 – April. Zero-shot sketch-based image retrieval via graph convolution network – In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 34, No. 07) – p. 12943-12950.

22. Pandey, A., Mishra, A., Verma, V.K., Mittal, A. and Murthy, H. – 2020 – Stacked adversarial network for zero-shot sketch based image retrieval – In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision – p. 2540-2549.

23. Canny Edge Detection Step by Step in Python [Electronic resource]: Sofiane Sahir – 2019 – https://towardsdatascience.com/canny-edge-detection-step-by-step-in-python-computer-vision-b49c3a2d8123

24. Adaptive Threshold [Electronic resource]: OpenCV Guide – https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_thresholding/py_thresholding.html

25. Jing, Y., Yang, Y., Feng, Z., Ye, J., Yu, Y. and Song, M. – 2019 – Neural style transfer: A review – IEEE transactions on visualization and computer graphics, 26(11) – p. 3365-3385.

26. Poma, X.S., Riba, E. and Sappa, A. – 2020 – Dense extreme inception network: Towards a robust cnn model for edge detection – In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision – p. 1923-1932.

27. Douze, M., Sablayrolles, A. and Jégou, H. – 2018 – Link and code: Fast indexing with graphs and compact regression codes – In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition – p. 3646-3654.

28. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A. and Chen, L.C. – 2018 – Mobilenetv2: Inverted residuals and linear bottlenecks – In Proceedings of the IEEE conference on computer vision and pattern recognition – p. 4510-4520.

29. TensorFlow Lite guide [Electronic resource]: TensorFlow –

https://www.tensorflow.org/lite/guide

30. ONNX: Easily Exchange Deep Learning Models [Electronic resource]: Pier Paolo Ippolito – 2020 – https://towardsdatascience.com/onnx-easily-exchange-deep-learning-models-f3c42100fd77

31. OpenVINO Toolkit Overview [Electronic resource]: OpenVINO –

https://docs.openvinotoolkit.org/latest/index.html

32. Introduction to Information Retrieval [Electronic resource]: Stanford Unversity –

CS 276 / LING 286 – 2019 –

https://web.stanford.edu/class/cs276/handouts/EvaluationNew-handout-1-per.pdf

APPENDIX A COMPARISON TABLE ON ZSL METHODS. CLASSIFIER-BASED METHODS

Method Category		Advantages	Disadvantages
Classifier -based methods	Correspondence metohds	The correspondence between the classifiers and the prototypes is captured through the correspondence function. Such a correspondence function design is simple and efficient.	They do not explicitly model the relationships among different classes, which can be useful for zero-shot learning.
	Relationship methods	The relationships among classes are modeled. In some methods, the classifiers for the seen classes learned in other problems can be directly utilized. In this way, the cost of model learning is reduced.	The relationships among classes in the semantic space is directly transferred to the feature space. The adaption problem from the semantic space to the feature space is hard to solve. The two steps of attribute classifier learning and inferring from the attribute to the class are hard to optimize in a unified process.
	Combination methods	If there are classifiers for the attributes learned in other problems, they can be used directly. Similar to relationship methods, the cost of model learning is reduced.	

APPENDIX B COMPARISON TABLE ON ZSL METHODS. INSTANCE-BASED METHODS

Method Category		Advantages	Disadvantages
Instance-based methods	Projection methods	The choice of projection functions is flexible and we can choose a suitable one according to the characteristics of the problem and the dataset. Specifically, under the CTIT learning setting, various semisupervised learning approaches can be adopted in the projection and the classification steps.	Each unseen class has just one labeled instance (the prototype). Thus, classification methods (especially under the CIII and CTII learning settings) are usually limited to the nearest-neighbor classification or some similar methods.
	Instance-borrowing methods	The number of borrowed labeled instances for unseen classes is relatively large. Thus, various classification models in supervised classification can be used.	The borrowed instances are actually instances belonging to the seen classes. This causes inaccuracy when learning the unseen class classifier.
	Synthesizing methods	The number of synthesized labeled instances for the unseen classes is relatively large and various classification models in supervised classification can be used. This is similar to instance-borrowing methods.	The synthesized instances are usually assumed to follow some distributions (usually Gaussian distribution). This causes bias of the synthesized instances.