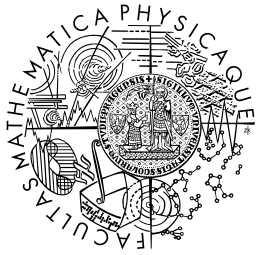

FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

BACHELOR THESIS

Tomáš Souček

Known-Item Search in Image Datasets Using Automatically Detected Keywords

Department of Software Engineering

Supervisor of the bachelor thesis: RNDr. Jakub Lokoč, Ph.D.

Study programme: Computer Science

Study branch: General Computer Science

Prague 2018

I declare that I carried out this bachelor thesis independently, and only with the cited sources, literature and other professional sources.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In Prague, July 18, 2018

Hereby, I would like to thank my supervisor RNDr. Jakub Lokoč, Ph.D. for his valuable advice and suggestions. I am grateful he enabled me to present the work at an international conference.

Furthermore, I thank Gregor Kovalčík for processing our application's logs from Video Browser Showdown competition.

Title: Known-Item Search in Image Datasets Using Automatically Detected Keywords

Author: Tomáš Souček

Department: Department of Software Engineering

Supervisor: RNDr. Jakub Lokoč, Ph.D., Department of Software Engineering

Abstract: Known-item search represents a scenario, where a user searches for one particular image in a given collection but does not know where it is located. The thesis focuses on the design and evaluation of a keyword retrieval model for known-item search in image collections. We use a deep neural network trained on a custom dataset to annotate the images. We design complex yet easy-to-use query interface for fast image retrieval. We use/design several types of artificial users to estimate the model's performance in an interactive setting. We also discuss our successful participation at two international competitions.

Keywords: Artificial neural networks, known-item search, image retrieval.

Contents

Introduction

With the advent of affordable portable electronics, especially mobile phones, the world's digital space has been flooded with user-generated content, specifically photos and videos. Together with cheaper storage options, we are facing a so-called explosion of multimedia. Greater than ever collections pose significant challenges for users to find whatever they search for. Challenges arise not only in personal videos. Miniaturization enabled new types of collections such as medical surgery or police body camera recordings.

The growing amount of data requires new effective and efficient approaches to image and video retrieval. There are two backbones of recent progress. The first is cheaper and more powerful GPUs. The second is a creation of large-scale datasets, such as ImageNet [1] and TRECVid [2], that enabled researchers to experiment with more sophisticated, but also more computationally demanding, models. Those and many other advances led to recent breakthroughs in machine learning, especially artificial neural networks. It gave birth to new applications in fields of computer vision, speech recognition, text translation and countless others.

Nowadays in computer vision, commercial services such as Google Photos need not rely on human labeling either on a web page or by an external worker. Latest neural networks can, on some datasets, even exceed human performance [3] thus one could assume, for specific domains, it is a solved problem. Yet there are not many, if any, non-task-specific tools the author is aware of, that provide a user an interface to organize and search in their collections for whatever their need is. One could argue that Google Photos and YouTube have all the user's content three clicks away, but their focus is different from helping a user to find a tank in thousands of images or videos from dashboard cameras which is difficult in either one of those services.

Such a scenario where a user searches for an exact known scene in a database is called known-item search (KIS). KIS tasks can be further divided based on the knowledge a user has about the searched scene. The first, and usually easier to solve, KIS task is visual. This means a user has already seen the searched scene before, but, for example, needs to find it in the collection as a proof. The other type of KIS task is textual, the harder, but maybe more important, counterpart to the visual task. One of the possible solutions to a KIS task is to find a similar object, for example in Google Images, and use the object's visual and semantic features as a query in the task's collection. A popular approach for extracting such features is by employing deep neural networks trained on some non-related image task [4]. These networks learn highly non-linear transformations from image space to arbitrary feature space in which common metrics can be used to find semantically similar images. To eliminate the need for an example object, that can be hard to come by, many low-level features were proposed, such as motion [5], edge [6] or color [7] sketches. The third approach winning the Video Browser Showdown (VBS) [8] in 2014 and 2015, a competition where state-of-the-art interactive video retrieval tools demonstrate their performance on an in advance known dataset (in 2018 the TRECVid IACC.3 dataset with 600 hours of video content was used).

The textual task presents much bigger challenges since, in most cases, visual features are not available to initiate the search. One way to solve the lack of features is to improve browsing. Some work has been done on an online approximation of relevance of individual images [9] when a user has already seen a part of a database. Others try to overcome the drawbacks of presenting image thumbnails in a two-dimensional grid by proposing to use 3D interfaces [10], but none of this aims to solve the absence of usable features for query initialization. Fortunately, the advent of neural networks and deep learning brought new methods for dealing with automated multimedia annotation. Nowadays there are architectures capable of not only assigning images to classes but also describing them in sentences [11]. Other neural networks can extract useful information from multiple subsequent frames of videos [12], such as tracking moving object in time. However, so far the VBS tools integrating more complex textual annotation systems for KIS tasks did not gain any significant advantage in textual tasks on the actual VBS dataset [13].

We try to examine this gap and summarize possible approaches towards textual annotation. Then we select a viable model and propose modifications to mitigate its shortcomings. Later on, we try to estimate the performance of the model by artificial users. By such simulations, we try to identify promising search strategies and investigate expectations when using the keyword-based retrieval model in a video retrieval tool on a given collection. We also present ideas of an award-winning tool[1] for known-item search jointly developed with Dr. Jakub Lokoč and Bc. Gregor Kovalčík. For that reason, the application prototype on the attached CD will only demonstrate the main ideas of this thesis that are solely the author's work. Some of the ideas were also published at international conferences in the following papers:

1. **Revisiting SIRET Video Retrieval Tool** [14]
   Demo paper describing our retrieval tool.

2. **Using an Interactive Video Retrieval Tool for LifeLog Data** [15]
   Demo paper introducing new ideas added to the tool to better deal with monotonous lifelog domain.

with the first being accepted to Video Browser Showdown at International Conference on MultiMedia Modeling 2018 and the second accepted to Lifelog Search Challenge workshop at International Conference on Multimedia Retrieval 2018.

[1]The tool competed at Video Browser Showdown [8] and outperformed other participating tools in the overall score. In textual tasks, the tool performed even better.

1. Preliminaries

In this chapter, we present the mathematical background that will be further needed to describe our work. We summarize gradient descent algorithm and neural networks, and we outline how to train them. Some of this chapter's definitions we took from [16] which we also highly suggest as a source of further information for a reader more interested in the subject.

1.1 Multivariable Calculus

Let $f\colon \mathbb{R}^n \to \mathbb{R}$ be a function with all partial derivatives existing and being continuous. The *gradient* of f is

$$\nabla f(\boldsymbol{a}) = \left(\frac{\partial f(\boldsymbol{a})}{\partial x_1}, \ldots, \frac{\partial f(\boldsymbol{a})}{\partial x_n} \right).$$

We write $\nabla_{\boldsymbol{x}} f$ or simply ∇f. To compute the derivative of a composition of two or more functions *chain rule* is used. Suppose $g\colon \mathbb{R}^m \to \mathbb{R}^n$ and $z = f(g(\boldsymbol{x}))$, then

$$\frac{\partial z}{\partial x_i} = \sum_{j=1}^{n} \frac{\partial z}{\partial g(\boldsymbol{x})_j} \frac{\partial g(\boldsymbol{x})_j}{\partial x_i} \quad \text{or in vector form} \quad \nabla_{\boldsymbol{x}} z = \left(\frac{\partial g(\boldsymbol{x})}{\partial \boldsymbol{x}} \right)^{\top} \nabla_{g(\boldsymbol{x})} z$$

where $\frac{\partial g(\boldsymbol{x})}{\partial \boldsymbol{x}}$ is the Jacobian matrix.

Given a function $f\colon \mathbb{R}^n \to \mathbb{R}$ we may ask where are its local (or global) minima. For some functions, it can be computed analytically, for the most of them, however, the analytical solution is impossible, and thus a numerical method called *gradient descent* is often used. We start with arbitrary point $\boldsymbol{x} \in \mathbb{R}^n$. Since gradient tells us how to change $\boldsymbol{x}$ in order to make a small improvement in $f(\boldsymbol{x})$, to minimize $f(\boldsymbol{x})$ we make a small change to $\boldsymbol{x}$ in the opposite direction:

$$\boldsymbol{x} = \boldsymbol{x} - \alpha \nabla_{\boldsymbol{x}} f(\boldsymbol{x})$$

where α is the step size also called the learning rate. The step is repeated until convergence or for a certain number of iterations. Note that the algorithm does not have to converge even for convex functions such as x^2 since it can 'jump' around $x = 0$ indefinitely due to a bad value of α. Also if the algorithm converges we are not guaranteed to find the global minimum, we can even be arbitrary far from it. However, in machine learning, this algorithm with minor modifications is widely used usually producing good-enough results.

1.2 Neural Networks

Artificial feedforward neural network is a machine learning model capable of learning its parameters $\boldsymbol{\theta}$ such that given input $\boldsymbol{x}$ it predicts output $\boldsymbol{y} = f(\boldsymbol{x}; \boldsymbol{\theta})$. The network is composed of smaller units called neurons defined as

$$h_i = g\left(\sum_j w_{i,j} x_j \right)$$

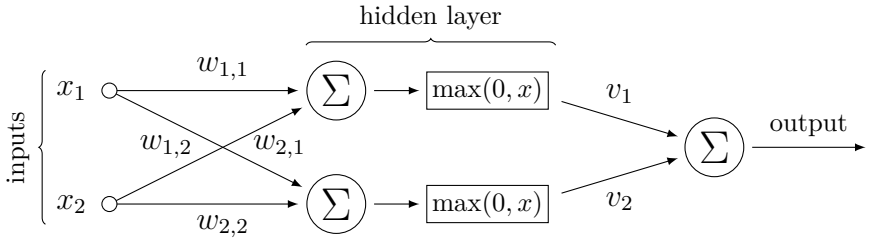

Figure 1.1: A simple neural network with one hidden layer. With weights $w_{1,*} = (1, -1)$, $w_{2,*} = (-1, 1)$ and $v = (1, 1)$ the network calculates linearly inseparable XOR function which posed a problem for single layer 'neural networks' called perceptrons.

where h_i is the output of i^{th} neuron, x_j are its inputs and g is a nonlinear function such as $1/(1+e^{-x})$ or $\max(0, x)$. Usually, we group neurons into layers and therefore we write $\boldsymbol{h} = g(W\boldsymbol{x})$ where $W \in \mathbb{R}^{\dim(\boldsymbol{h})\times\dim(\boldsymbol{x})}$ and g is applied componentwise. Simple feedforward neural network with one 'hidden' layer can be seen in Figure 1.1. Such basic network can approximate any continuous functions if there are enough neurons in the hidden layer [17]. However, the number of neurons can be arbitrarily large and empirically networks with the same number of neurons but with more hidden layers usually achieve better results.

Given training data – inputs $\boldsymbol{X} = \{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(n)}\}$ and predictions $\boldsymbol{Y} = \{\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(n)}\}$, we want the network to learn a mapping f from $\boldsymbol{X}$ to Y by minimizing the prediction error $\mathcal{L}$ (also called loss). One of the first error functions used is mean square error defined as

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\theta}) = \sum_i \left(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}) - \boldsymbol{y}^{(i)}\right)^2$$

The learning is done by computing gradients with respect to the network's parameters $\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{X}, \boldsymbol{Y}; \boldsymbol{\theta})$ and by using gradient descent to minimize the loss. However, even for moderately sized data calculating loss over the whole training data can be extremely computationally demanding. We thus approximate the loss in each step using only some fixed number of randomly selected examples. Such an approach is called (minibatch) stochastic gradient descent (SGD).

For structured data such as images, we might observe that data patterns are independent of position and distant data points do not depend on each other. Instead of connecting each neuron with all neurons in a previous layer we introduce a notion of convolution. Mathematically, the convolution of two vectors in 1D can be expressed as $(\boldsymbol{x} * \boldsymbol{w})_t = \sum_i x_i w_{t-i}$ with easy generalization into higher dimensions. If $\boldsymbol{x}$ is an input, we consider $\boldsymbol{w}$ (called kernel) to have only handful nonzero values around zero index (typically 3, 5 or 7 in each dimension) therefore it can be quickly computed using GPUs. Aside from the shift invariance and locality convolutional layers have far fewer parameters than fully-connected layers which makes them less prone to overfitting to the training data.

2. Related Work

This chapter reviews recent approaches towards textual annotation. First, we discuss available datasets, then multiple approaches in image classification. Lastly, we examine object localization in an image. Unfortunately, we do not include video annotation in our work since video processing and training still creates unfeasible demands on computational power.

2.1 Datasets

Recent breakthroughs in machine learning could not happen without large-scale datasets since they are necessary to train and evaluate algorithms. Image datasets can be divided into three groups depending on what they are addressing: image classification, object localization and semantic segmentation.

The objective of image classification tasks is to decide whether given object is or is not present on an image. There are multiple low-resolution datasets such as CIFAR-100 [18] containing 60,000 images in 100 categories with resolution 32-by-32 pixels. Since 2009 there is ImageNet [1], a large-scale, high-resolution dataset with an aim to populate the majority of the 80,000 WordNet synsets with 500 to 1000 images. In research, a subset of the ImageNet database called ILSVRC2012 containing 1000 categories is usually used.

The task of object localization embodies stating what is in an image and wherein the image it is. The location is usually given as a bounding box. As of 2018, ImageNet database holds bounding boxes for over 3000 synsets with an average of 150 images per synset. Similarly to ImageNet, Open Images Dataset [19] contains over one and a half million annotated images with more than twice as many bounding boxes belonging into 600 categories. Semantic segmentation goes a step further. Its goal is to not only distinguish and localize an object in an image but also to assign each pixel to an object it belongs to. Probably the most recognized dataset The Microsoft Common Objects in COntext (MS COCO) [20] contains 91 common object categories with 82 of them having more than 5,000 labeled instances. There are also many more datasets focused on a specific task such as Places [21], designed for scene recognition, or YFCC100M [22] also used for unsupervised learning [23].

2.2 Image Classification

Prior to 2012 state-of-the-art approaches towards image classification involved using SVM classifiers trained on handcrafted features. Even though those methods can be tweaked, they rise and fall with a quality of the features. Since 2010 ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [1] has been used to benchmark computer vision systems. Their dataset contains 1000 categories, each with around 1000 high-resolution images. The top-5 error rate for the SVM classifiers hovered at over 25% until in 2012 an entry from A. Krizhevsky, et al. [24] disturbed machine learning community with by far the best result of 15.3% using deep convolution neural network (DCNN) and pushed the whole

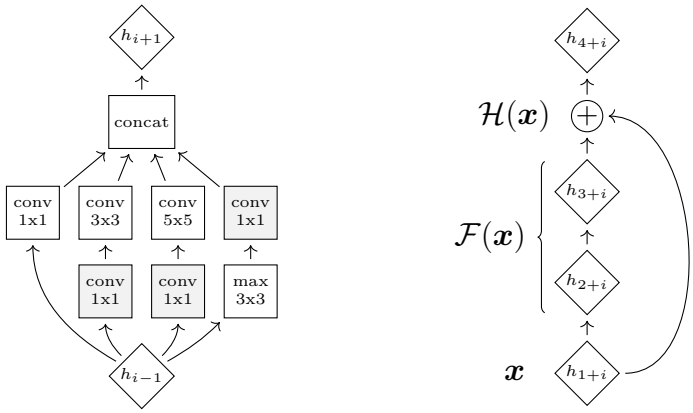

Figure 2.1: Inception block of GoogLeNet [28] (2014, left) and a skip connection of a residual network [30] (2015, right).

industry towards neural networks, which until then were used only in a limited number of cases such as handwritten character recognition. The main advantage of neural networks is that they, in contrast to SVM classifiers, can learn the best features themselves and thus they seem ideal for such tasks. But we were unable to train larger models due to multiple issues that were overcome in the last few years.

Fast graphics cards, novel stochastic optimization algorithms [25], clever weight initialization [26], non-saturating activation functions, batch normalization [27] and many other techniques allowed for deeper and bigger networks. One of the key ideas, a so-called network in network, takes elementary building blocks such as convolutional and max-pooling operations with different filter sizes and stacks them into larger, more complex layers, sometimes called blocks or cells. This idea has been used in efficient deep neural network architecture codenamed Inception [28] which uses 1x1, 3x3 and 5x5 convolution layers together with max pooling side-by-side (Figure 2.1). Additional pointwise (1x1) convolutions (in gray) are used to project the output of a previous layer onto a new filter space, greatly reducing the number of trainable parameters in 3x3 and 5x5 convolutions. This enabled GoogLeNet [28] (later called Inception V1), comprised of these blocks, to win with 6.7% top-5 error rate ILSVRC competition in 2014, defeating 20 times larger DCNN by Simonyan and Zisserman [29] only made of classical convolutional layers.

By looking at those described networks it seems that adding more layers is beneficial. But increasing the depth of suitably deep models can lead to accuracy degradation [30]. Yet this is not caused by overfitting since we can also observe higher training error. It could be partly due to vanishing or exploding gradients even though there are methods to moderate the problem such as clever weight initialization or gradient normalization. To tackle the problem a deep residual learning framework [30] was proposed. Instead of training a set of layers to fit a desired underlying mapping $\mathcal{H}(\boldsymbol{x})$ directly, it lets layers to fit a residual mapping $\mathcal{F}(\boldsymbol{x}) - \boldsymbol{x}$ such that $\mathcal{H}(\boldsymbol{x}) = \mathcal{F}(\boldsymbol{x}) + \boldsymbol{x}$. In neural networks, it can be realized by "skip connections" (Figure 2.1). ResNet-152, a DCNN utilizing this idea with whooping 152 layers, achieved 4.49% top-5 error rate on ILSVRC in 2015.

The most recent attempts to automate a network's architecture engineering

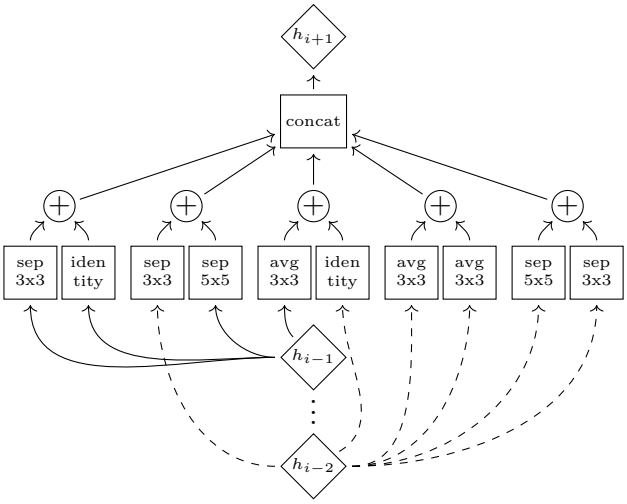

Figure 2.2: A state-of-the-art block of NASNet-A [32] (2017), created by a recurrent neural network, outperformed all blocks ever designed by humans.

process were made using reinforcement learning (RL) [31, 32]. The methods utilize a controller recurrent neural network (RNN) that predicts a network architecture. This architecture is then trained and its achieved accuracy is used to scale the gradients to update the RNN controller. Despite being extremely computationally demanding needing thousands of GPU-hours even on small datasets, [32] shows that architectures learned on CIFAR-10 can translate easily to ImageNet's ILSVRC achieving best results ever with 3.8% top-5 error rate. Even smaller networks using the same structure outperform equivalently-sized human-designed models. A cell of the best performing architecture can be seen in Figure 2.2.

2.3 Other Approaches to Annotation

It is not uncommon an image given for annotation contains dozens of objects, then the classical approach to image classification starts to fall apart since a standard softmax classification layer prefers to assign only one class. Ordinarily, multiple cutouts of an image are taken and annotated separately, the final classification is a mean of distributions for each cutout. It may be sufficient for the task but it does not give us locations of the objects. To obtain a bounding box of an object in an image, a region-based convolutional network (R-CNN) has been proposed [33]. It utilizes two deep convolutional networks, a region proposal network (RPN) suggesting regions of interest and a standard classification DCNN annotating suggested regions. RPN uses first convolutional layers of a standard DCNN such as [29] to generate a feature map. Then a small network $\mathcal{N}$ takes an $n \times n$ spatial window of the feature map and predicts k bounding boxes, for each box also stating a probability it contains an object (Figure 2.3). Depending on a size of the feature map and number k, which is usually set to 9, the network generates thousands of regions of interest. The most probable ones are then taken.

As shown in [34], first layers of CNN learn low features such as lines and corners with later layers learning more complex representations of images. This can be used for semi-supervised learning. Deep convolutional representations

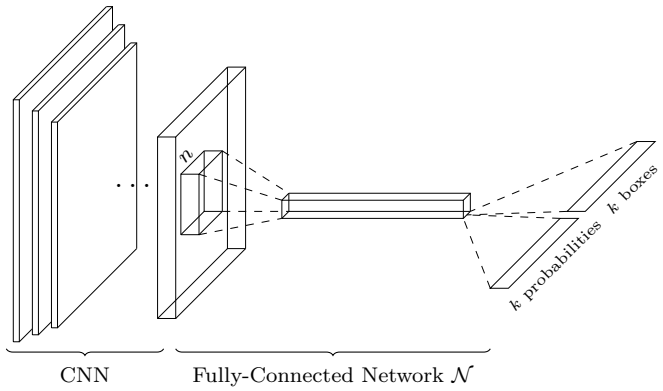

Figure 2.3: A region-based convolutional network [33] (2016), generating k bounding boxes for each sliding window $n \times n$.

learned on similar annotated dataset are applied to new tasks where supervised learning is expensive to do [35]. K-nearest neighbors algorithm can be then used to annotate unknown images by projecting them onto a feature space of the network's later layers. Such an approach can be scaled to much bigger datasets than ImageNet, working reasonably well even on YFCC100M [23].

3. Retrieval Model

Modern KIS tools suitable for large or very large collections depend on automatic retrieval models. We select the appropriate model for our task and propose modifications needed for the model to be useful for KIS. We also describe steps how we created training data for our model.

3.1 Model Selection

As mentioned, there are multiple approaches towards textual annotation. For our task to create a model with at least a thousand classes we chose image classification. This is because object localization datasets are much smaller and a resulting model would be more demanding on pre- and post-processing, negatively affecting already complicated processing pipeline.

Today's state-of-the-art approaches to the problem use exclusively deep neural networks, discussed in Related Work, thus we limit ourself to DCNNs. Every year there is a new architecture achieving better results, yet improved accuracy usually comes at cost of bigger models. A notable exception is NASNet network [32] that beats all other models in every size category; however, the model was released recently and hence it is not in our consideration. Many top performing models over the last few years were already presented thus we only show all considered models in Table 3.1.

Model	Number of parameters	ILSVRC2012 Top-1 Acc.*	ILSVRC2012 Top-5 Acc.*
AlexNet [24]	60 M	59.3†	81.8†
GoogLeNet [28]	**6.6 M**	69.8‡	89.9
VGG-16 [29]	138 M	74.4	91.9
Inception V3 [36]	23.8 M‡	78.8	94.4
Inception-ResNet-v2 [37]	55.8 M‡	**80.1**	**95.1**

* Without averaging over multiple cropped images and ensemble of models.

† Achieved by averaging four corner patches and a center patch of image.

‡ Value taken from [32].

Table 3.1: Considered neural networks and their performance.

With limited computational resources, emphasis in our decision was given on model size. We selected GoogLeNet since its size allowed us to train it on GPU with 2 GB of memory with batches of size 32 reasonably fast. Our decision is also supported by [38] where the network ranked among the most efficient ones utilizing well its parameter space. The network consists of initial convolution layers with max pooling reducing spatial dimensions of input to $28 \times 28 \times 192$ followed by 9 Inception blocks (shown in Figure 2.1) interleaved by occasional max pooling and with output dimensions $7 \times 7 \times 1024$. Then there is an average pooling further reducing output to $1 \times 1 \times 1024$. The last layers are depicted in Figure 3.1 also with modifications discussed in the next sections. For the ones more interested in the architecture, we point to the original paper [28] for the

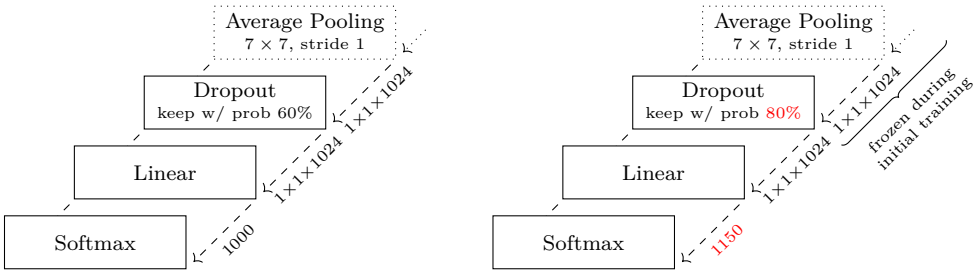

Figure 3.1: The last layers of GoogLeNet with their output sizes as presented in [28] (left) and modified version used in our experiments with changes in red (right).

detailed description of the network and [27] for modifications by adding batch normalization to layer outputs.

3.2 Dataset Selection

Due to the unmatched number of classes to chose from and extensive use in the scientific community, we have decided to use the ImageNet database [1] as a starting point. The database is based on WordNet [39], giving us an advantage when designing search model later in Chapter 4. The most used dataset ILSVRC2012 has a few flaws, however. It was designed for a competition thus it does not contain many common objects, yet on the other hand, its 120 dog classes could pose a challenge when searching because even a human could then make a mistake classifying a searched dog incorrectly. Bigger ImageNet10K [40], despite solving the lack of common objects, amplifies possible human error and greatly reduces the performance of any neural network. We thus decided to create our own dataset. All classes with at least 1000 example images from Winter 2011 release of ImageNet were taken into consideration, yielding to 6642 classes in total.

We tried selecting classes for our network automatically from WordNet structure, yet results were unsatisfactory therefore human judge was used to select viable classes. To simplify selection and orientation in the classes, we propose the following steps:

1. Mapping $\mathcal{M}$ of each image class to n-dimensional vector space is computed.
2. Image classes are divided into k clusters (we used $k = 50$).
3. For each cluster, WordNet directed acyclic graph (DAG) with edges representing hypernym–hyponym relation is created enabling closer inspection of similar classes.

We defined mapping $\mathcal{M}$ of image class c as

$$\mathcal{M}(c) = \mathbb{E}_{\boldsymbol{x}\sim\hat{p}_c} f(\boldsymbol{x};\boldsymbol{\theta}) \approx \frac{1}{|S_c|}\sum_{\boldsymbol{x}\in S_c} f(\boldsymbol{x};\boldsymbol{\theta}) \tag{3.1}$$

where $\boldsymbol{x}$ are images drawn from distribution $\hat{p}_c$ representing class c, $f(\cdot;\boldsymbol{\theta})$ is a function parametrized by $\boldsymbol{\theta}$ mapping image to n-dimensional vector and S_c is a subset of images of class c taken from ImageNet database. Since DCNNs proved

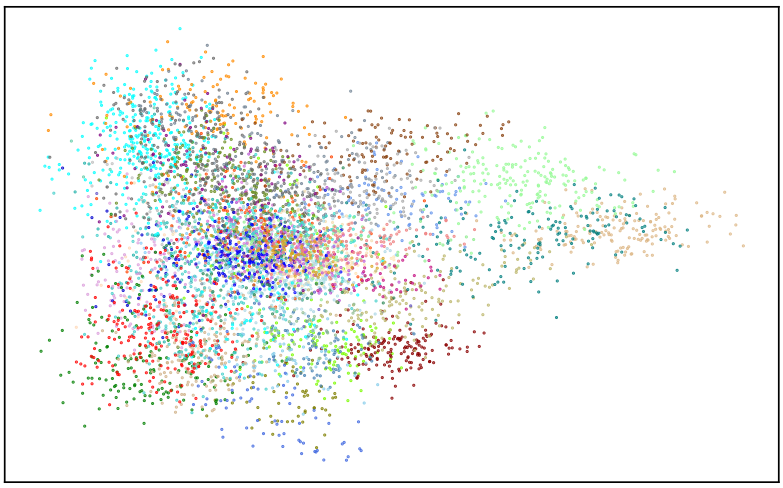

Figure 3.2: 6642 ImageNet classes in 50 clusters projected into a 2D plane. Higher dimensional vectors $\mathcal{M}(c)$ are clustered using k-means++ [44] algorithm. Each vector is then projected into 2D using principal component analysis [45]. For example, the rightmost clusters in the image contain only flowers and trees.

to be efficient in mapping images to lower dimensional space [35], standard GoogLeNet is used as a good f. The network is initialized to publicly available weights [41] trained on ILSVRC2012 with the last linear layer with softmax classifier removed. In our experiments $|S_c| = 10$ proved to be sufficient approximation of the mean value.

WordNet structure, no matter how precise in words' meaning, does not reflect human visual perceptions. For example, a man can quite easily associate a picture to word *barbecue*, yet it can have two meanings – *a cooker* and *a meal* that are far apart in WordNet. Unfortunately, the images associated with those categories are similar, therefore confusing a neural network and adding ambiguity to a user's search. In order for a human judge to identify such conflicts, grouping visually similar images is necessary, since one cannot orientate in thousands of classes. We use an open-source implementation [42] of k-means clustering algorithm to group image classes into clusters. Even though clustering into two clusters is already NP-hard [43], using a version of the algorithm known as k-means++ [44], where cluster centers are chosen with probability proportional to the square of their distance to the closest already chosen center, resulted into having multiple clusters with only one kind of an object, such as *a person*, *a tree*, *a bird* or *a horse*. For example, the *horse* cluster contained synsets from multiple branches of WordNet structure connected to *racing*, *hunting*, *transportation*, *people*, *buildings* and even *clothing* hence demonstrating the usefulness of the clustering. Further, we constructed WordNet DAG for each cluster and selected useful synsets. Such an approach also revealed multiple synsets with wrongly assigned images based on ambiguity in their names. For example, two synsets *a smoker* and *a nonsmoker* with descriptions *'A passenger [railway] car for passengers who wish to (or not to) smoke'* contain photos of people smoking and photos of automobiles respectively.

3.2.1 Image Preprocessing

Using the outlined method, 1150 image categories were chosen out of the 6642 considered (see Attachment A.3 for the complete list) and corresponding images were downloaded from [46]. Only 1000 images per each class were selected to reduce the size of the dataset and to equalize the number of images in each class. Every image was then resized in a way that its shorter size had 400 pixels further reducing the dataset size but still enabling cropping as a dataset augmentation method during training. Furthermore, train and validation datasets were created. The train dataset containing nine-tenths of images of each class and the validation dataset containing the rest. Creation of a test set was unnecessary since we have not been designing completely new network architecture.

During training, we employ several data augmentation methods such as [47] to reduce overfitting. Firstly, each image is randomly cropped. We ensure the new image has at least 80% of the area of the original image. Then the brightness and saturation are randomly adjusted and each image is with probability 1/2 vertically flipped. All the methods are not computationally intensive and therefore they can be performed online during training without the need to store distorted images on a disk.

3.3 Model Training

Our goal is to train a model parametrized by $\boldsymbol{\theta}$ to map a set of images $\boldsymbol{X}$ to a set of labels Y. Assuming $(\boldsymbol{x}, y) \in (\boldsymbol{X}, Y)$ are drawn from true but unknown distribution p_{world} independently, the objective can be written as maximum likelihood estimation in following way (see [16] for more details):

$$\boldsymbol{\theta}_{ML} = \arg\max_{\boldsymbol{\theta}} \prod_{i=1}^{n} p_{model}\left(y^{(i)} \mid \boldsymbol{x}^{(i)}; \boldsymbol{\theta}\right) \tag{3.2}$$

$$= \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{n} \log p_{model}\left(y^{(i)} \mid \boldsymbol{x}^{(i)}; \boldsymbol{\theta}\right) \tag{3.3}$$

The second equality holds because logarithm is always increasing function and $\log(ab) = \log(a) + \log(b)$. Further dividing by n, using empirical distribution $\hat{p}_{data}$ and minimizing negative of the original function we can write:

$$\arg\min_{\boldsymbol{\theta}} -\mathbb{E}_{(\boldsymbol{x},y)\sim\hat{p}_{data}} \left[\log p_{model}\left(y \mid \boldsymbol{x}; \boldsymbol{\theta}\right)\right] \tag{3.4}$$

Such function can be optimized using stochastic gradient descent (SGD) as described in Chapter 1. However, for training the network, we use a more advanced algorithm based on SGD called Adam [25] which computes individual adaptive learning rates for different parameters from estimates of first and second moments of the gradients. Default values of $\beta_1 = 0.9$ and $\beta_2 = 0.999$ were used.

Training as big models as GoogLeNet [28] requires a lot of computing power that could be potentially wasteful since GoogLeNet has been already trained on images with similar low-level features. Therefore we use a transfer learning technique firstly described by [35]. This approach cuts training time needed by initializing model's weights to values of a model trained on different but related

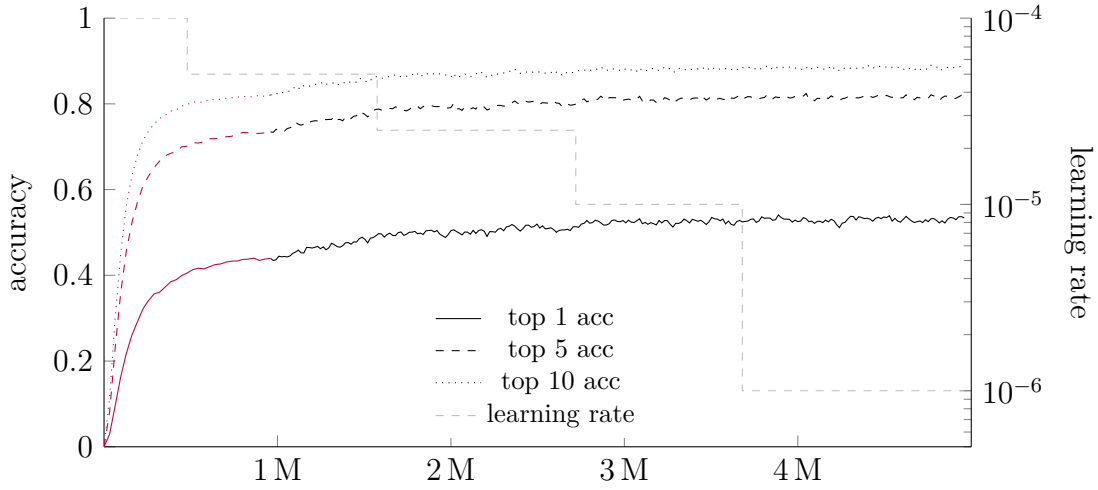

Figure 3.3: The accuracy of GoogLeNet [28] during training on a random subset of the validation dataset. The horizontal axis shows the number of images the model has seen to a given point in training. Initial training of the last layer only is shown in red. Curves are smoothed by exponential moving average, i.e. $S_i = \frac{1}{2}V_i + \frac{1}{2}S_{i-1}$ where V_i is model's accuracy and S_i is the shown smoothed value.

task (normally, prior to a training, model weights would be initialized to random values such as [26]). In our work, we use TensorFlow [48], an open-source machine learning framework with an extensive collection of pretrained models [41] also containing GoogLeNet with weights trained on the ILSVRC2012 dataset. We use those initial weights for the whole network except for the last layer which is replaced by a new one of different dimension (see Figure 3.1) with its weights initialized from a uniform distribution $U\left[-\sqrt{6/(n_j+n_{j+1})}, \sqrt{6/(n_j+n_{j+1})}\right]$ where $n_j + n_{j+1}$ is number of neurons in two adjacent layers (see [26] for more details).

During the initial phase of training (Figure 3.3 in red) only the last layer was trained with the other weights fixed because in the beginning random gradients can destroy learned low-level features worth keeping. The first phase consisted of 15,000 batches of size 64, firstly with learning rate 10^{-4} and after 7,500 batches with $5 \cdot 10^{-5}$. Further in the training, the whole network was fine-tuned with the batch size of 32 examples and learning rate between $5 \cdot 10^{-5}$ and 10^{-6} as shown in Figure 3.3. The selected batch size is due to limited GPU memory and it is possible that better results could be obtained if a bigger batch size was used, even though the paper discussing batch normalization of GoogLeNet [27] also uses the batch size of 32.

Evaluation Method	Top-1 Acc.	Top-5 Acc.	Top-10 Acc.
baseline (whole image)	56.0	83.8	90.1
center cutout	56.1	83.8	90.2
10 patches averaged	**57.1**	**84.7**	**90.9**

Table 3.2: Trained model performance on the validation set.

At the test time, three evaluation methods were tried. As a baseline, images were resized using bilinear interpolation to the network's input size (224×224). The second approach was to take a center patch of an image with width and height of $87,5\%$ of the original image again resized by bilinear interpolation to the network's input size. The third approach used a similar technique as described in [24]. Additionally to the center patch, it takes four 224×224 corner patches from a rescaled image of size 320×320 and averages the predictions made by the network's softmax layer on them as well as on their horizontal reflections. Results of all methods are summarized in Table 3.2.

4. Textual Search

The model described in the previous chapter assigns each image $\boldsymbol{x}$ a vector $\hat{\boldsymbol{y}}$. In theory, it represents how likely each class is contained in the image. However, this representation is far from useful for a user. Therefore multiple techniques are used to ensure the model is a convenient yet powerful search tool even for novices.

4.1 Supported Labels

Our tool supports only a finite set of labels L that are prompted by the interface when users form the query described in greater detail in Section 4.3. It could be possible to construct a query given higher level description transformed by a task-specific set of neural language processing rules [49], but this approach may mislead user if a searched high-level concept is not in an internal model. Thus a set L_m is created containing labels corresponding to the image classes with their names, descriptions and hyponyms taken from WordNet [39]. We further utilize WordNet structure creating a new set L_h of hypernyms of all labels in L_m. The final set of labels is then $L = L_m \cup L_h$. With hypernym–hyponym relation as a directed edge, L can be viewed as a directed acyclic graph (DAG) where all vertices (i.e. labels) with no outgoing edges are in L_m (however, there can be vertices in L_m with outgoing edges).

4.2 Query Formulation and Ranking

The users are allowed to specify sets of supported labels $N_i \subseteq L$, wherein each set the labels are connected by logical OR, while the sets N_i are connected by logical AND. For k such sets, the user query Q_u is then written as

$$Q_u = \bigwedge_{i=1}^{k} \left(\bigvee_{\forall label_j \in N_i} label_j \right) \tag{4.1}$$

and further in the text, for convenience, we use data representation of Q_u defined as $\{N_i\}_{i=1}^{k}$ with the same meaning as in Equation 4.1. If the query contains any hypernyms (labels with some outgoing edges in L), further preprocessing needs to be done. By default, every hypernym h is substituted by a set of all labels in L_m that are reachable from h in DAG L. For hypernyms in L_m (i.e. they correspond to a class recognized by the model), the substitution can be disabled. Finally, the preprocessed query Q_p contains only labels from L_m directly recognized by our model.

The ranking $r(\cdot)$ for each image $\boldsymbol{x}$, given preprocessed query Q_p and model parametrized by $\boldsymbol{\theta}$, is calculated as

$$r\left(\boldsymbol{x}; Q_p, \boldsymbol{\theta}\right) = \prod_{\forall N_i \in Q_p} \left(\sum_{\forall label_j \in N_i} \hat{\boldsymbol{y}}_{label_j} \cdot idf\left(label_j\right) \right) \tag{4.2}$$

where $\hat{\boldsymbol{y}} = f(\boldsymbol{x}; \boldsymbol{\theta})$ is model prediction on the image $\boldsymbol{x}$ and $\hat{\boldsymbol{y}}_{label_j}$ represents its relevance score of containing $label_j$ and $idf(\cdot)$ represents an inverse document frequency (IDF) [50]. Let us explain the underlying reasoning why to introduce IDF on an example: When a user searches *a person in front of an excavator*, person keyword completely dominates the query since in the dataset there are thousand times as many persons as excavators. Standard IDF, however, uses the number of documents over the number of documents containing a given term which is unusable because softmax layer gives small nonzero values to almost all classes. We thus define inverse document frequency as

$$idf(label) = \log\left(\frac{\max_{i \in labels} \sum_{\boldsymbol{x}} \hat{\boldsymbol{y}}_i}{\sum_{\boldsymbol{x}} \hat{\boldsymbol{y}}_{label}} + 1\right) \tag{4.3}$$

where $\hat{\boldsymbol{y}}$ is the same as in Equation 4.2, therefore dependent on image $\boldsymbol{x}$. If we think of $\hat{\boldsymbol{y}}_i$ as a probability image $\boldsymbol{x}$ contains label i, the fraction in Equation 4.3 can be viewed as how likely any given image contains the most common label in a collection over how likely it contains the searched label.

4.3 Application Prototype

This section discusses our user-friendly implementation of described approaches. Due to the project we are taking part in, the user interface is programmed in C# language using the .NET framework and only focuses on an implementation of keyword search since the rest is developed jointly. Also note that model training, simulations, data annotation and other offline tasks presented in this work are programmed in Python and are available together with documentation on the attached CD (Attachment A.2). The user application can be logically divided into three parts described in the following sections: *Main Window*, *Keyword Model* and *Suggestion Text Box*. Overview of the most important classes, their methods and interactions can be seen in Figure 4.1.

4.3.1 Main Window

The Main Window class is used as an entry point for the application defining its user interface and connecting the text box to the retrieval model. It is also responsible for correctly initializing all the components with appropriate external data files defined in a configuration file. Further, since the application is only a demo, the class loads and displays images to a user based on a model's ranking.

4.3.2 Keyword Model

Keyword model implements the ranking algorithm as described in the section above. An inverted index is utilized for fast query retrieval. As the creation of the index is the most demanding operation due to the image annotation by DCNN, it is created once in the preprocessing phase. The neural network assigns each image $\boldsymbol{x}^{(i)}$ a set of labels with its score $\{(label_j, \hat{\boldsymbol{y}}_{label_j}) \mid \hat{\boldsymbol{y}} = f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}),\ label_j \in L_m\}$. Further, labels with the score lower than 0.001 are discarded without loss of model's quality as discussed in Chapter 5. The inverted index then contains for

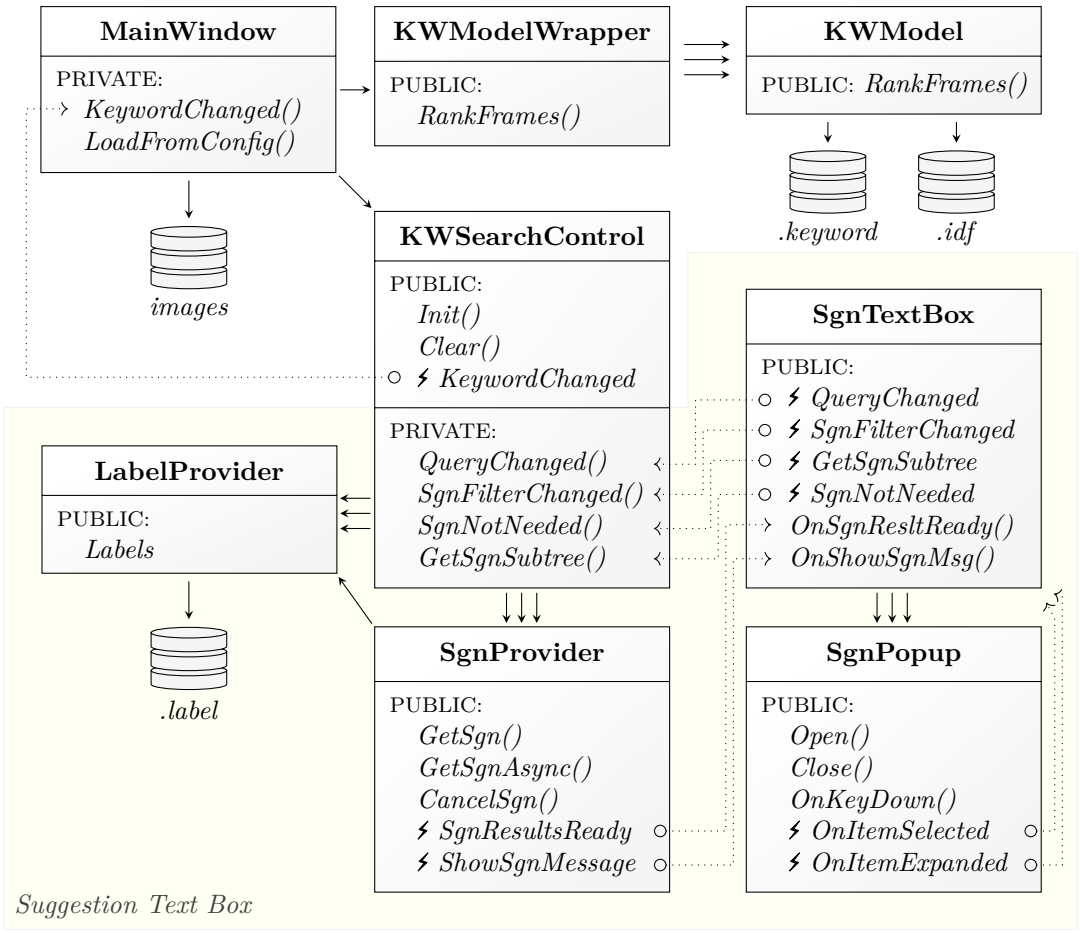

Figure 4.1: Application prototype structure. A solid arrow represents a reference to a class instance. Multiple arrows denote, that reference to more instances is possible. A dotted arrow represents a reference to a function. Interaction with external data files is denoted by a database icon.

each label $j \in L_m$ a list of tuples $(i, \hat{\boldsymbol{y}}_{label_j})$ – the image id and the assigned score. The file also incorporates label-to-offset map that enables fast seek instructions to individual label lists.

During application startup, only the label-to-offset map is loaded into RAM. Lists corresponding to given labels are copied into the memory only when the model is queried by a user. For small datasets, the overhead is negligible since there are more I/O intensive operations such as image rendering, etc. For large collections, the slowdown can be noticed if a compound query is created yet this approach saves space in RAM that can be used by other more memory demanding models such as similarity model. Also note, that the most labels are not used each time the application is run since only a handful of keywords are used per each target image. To mitigate the possible slowdown the model uses two caches. The bigger one stores already loaded labels avoiding virtually any repetitive label read during regular usage. The smaller one stores clauses (i.e. $\bigvee_i label_i$) enabling faster retrieval of repeated or more complex queries. The application also supports multiple keyword sources, therefore, Keyword Model Wrapper is used to redirect model calls to appropriate models.

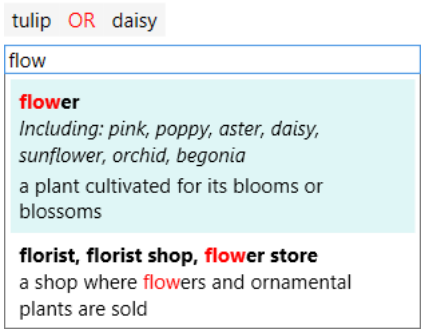

Figure 4.2: Keyword search interface. When selecting the *flower* label, hyponyms *pink*, *poppy*, etc. will also be searched. To select only *flower*, a user can hold the control key while pressing the enter. If the label is expanded by pressing the right key, the user can also select only individual flowers.

4.3.3 Suggestion Text Box

The suggestion-providing text box is comprised of multiple classes encapsulated by Keyword Search Control class enabling effortless integration to any existing application. If text box's content changes, a handler in Keyword Search Control selects Suggestion Provider belonging to the currently selected keyword source. The provider asynchronously searches all labels $l \in L$ and hands over the results to the text box when completed. Suggestion Popup is then populated with the results, displaying them to a user. If any suggestion is selected to update the query, Keyword Search Control passes a list of selected labels to a registered handler responsible for a query ranking.

The text box accepts only the labels L, however, full-text search over their WordNet names and descriptions is performed using Aho–Corasick string search algorithm. This way, a user gets suggestions even if the desired label is not present and thus can select any similar label based on its description or quickly see that the query needs to be formulated differently. As user types, the suggestions are presented with the searched phrase highlighted (see Figure 4.2). The labels are distinguished by color. Red labels are the added WordNet hypernyms L_h, green and gray labels represent basic 1150 labels L_m. The green labels represent hypernyms for gray labels. The suggestions are browsed by up and down keys, while the selection of keyword is done only by enter. We find mouse control unnecessary since interaction via keyboard is faster, but we may include more intuitional and user-friendly mouse control for novices as well. When a user browses to a hypernym, the right key can be used to expand the hypernym, following the WordNet structure. When a label is selected, it appears on the top panel, connected by logical OR with previously selected labels. Logical relation between the labels can be changed to AND and back by left mouse click, clicking on the labels themselves removes them from the query.

4.4 Other Retrieval Models

Our work is part of a bigger project focused on video retrieval with a goal of creating a tool that provides a user with a powerful means to interactively search his video collection. In this section, we therefore summarize our tool's other

retrieval models and model fusion used for Lifelog Search Challenge (LSC) workshop at ICMR2018 [15] and for Video Browser Showdown (VBS) competition at MMM2018 [14].

Mainly for visual KIS tasks, our tool features easy-to-use canvas for color sketches. Every sketch is represented as a set $Q_c = \{(q_i, r_i, t_i)\}_{i=1}^{k}$ of k color points where $q_i \in \mathbb{R}^3$ is color in CIE Lab color space, r_i represents region where the given color shall be located and $t_i \in \{\text{ALL}, \text{ANY}\}$ specifies whether ALL or just ANY of the pixels in the region should be considered. The ranking $c(\cdot)$ of an image $\boldsymbol{x}$ is then computed as

$$c\left(\boldsymbol{x}; Q_c\right) = -\left(\sum_{\substack{\forall (q_i, r_i, t_i) \in Q_c \\ t_i = \text{ANY}}} \min_{p \in \boldsymbol{x} \text{ in } r_i} L_2\left(q_i, p\right) + \sum_{\substack{\forall (q_i, r_i, t_i) \in Q_c \\ t_i = \text{ALL}}} \operatorname{avg}_{p \in \boldsymbol{x} \text{ in } r_i} L_2\left(q_i, p\right) \right) \tag{4.4}$$

where $p \in \boldsymbol{x}$ in r_i represents all pixels of image $\boldsymbol{x}$ in specified region r_i. Due to performance reasons, each image in the database is represented only as 20×15 color points or 'pixels'. Our user interface also currently supports regions in the shape of ellipses only. If a keyword or color-sketch search is not successful or a user wants to simply browse similar images, our tool utilizes deep features from GoogLeNet [28] after the last average pooling layer for retrieval of semantically similar images. Rank of an image given a query example is the cosine similarity between their deep feature vectors.

All the retrieval models provide a relevance score function inducing a similarity based ordering of all database objects with respect to a given query. Even though our tool supports multiple query modalities at once by normalizing the ranking of each modality to the interval $[0, 1]$ and then summing them up, combining multiple models can be tricky. This is because each model has a different distribution of rank values, e.g. similarity ranking assigns to most of the database rank between 0.6 and 0.85 whereas in keyword ranking almost all images are assigned rank smaller than 10^{-2}. However, our tool enables us to use other models as filters where their thresholds can be dynamically set for each model interdependently. Such an approach proved to be much more useful than sorting joined ranking from multiple models.

5. Model Evaluation on KIS Task

In Chapter 3 we evaluated our model on classification task which differs from KIS task. In this chapter, we, therefore, discuss the differences between classification and known-item search, propose possible evaluation methods and present their results. We also show and discuss this year's Video Browser Showdown competition we successfully participated in.

In KIS task the goal of a retrieval model parametrized by $\boldsymbol{\kappa}$ is to minimize a position (or rank) r of the searched image $\boldsymbol{x}$ given a user query Q. If the position is treated as a random variable the objective can be written as

$$\arg\max_{\boldsymbol{\kappa}} \mathbb{E}_{(\boldsymbol{x},Q)\sim\hat{p}_{data}} \left[\sum_{t=1}^{|C|} p_{model}\left(r \leq t \mid \boldsymbol{x}, Q; \boldsymbol{\kappa}\right) \right] \tag{5.1}$$

where $p_{model}\left(r \leq t \mid \cdot\right)$ is a probability image's position is less than or equal to t and $|C|$ is size of the collection. Intuitively, the ideal user constructing the most specific queries Q for images $\boldsymbol{x}$ together paired with an ideal model capable of distinguishing all images from each other would always have $p_{model}\left(r \leq 1\right) = 1$, therefore yielding the highest possible value. Unfortunately, unlike the function 3.4 of the classification task, function 5.1 is not differentiable thus we can only use it as a quality measure of our model. Yet another issue arises with $\hat{p}_{data}$ distribution since it is dependent on a set of supported labels and on a dataset, the images $\boldsymbol{x}$ are taken from. In order to evaluate our model, a thorough user study would be needed yet a slight change to the model could nullify its results. In the next section, we construct simple simulations providing us with an insight into model's performance without the need for a broad study.

5.1 User Simulations

To simplify the simulations only queries containing OR will be considered. We can, therefore, write $Q = \{N_1\}$ and for a sake of simplicity, we will use $Q = N_1 \subseteq L_m$. We take inspiration from [51] on approaches toward generating user queries and define the following users:

Network User. Assumes a user is coherent with the model $\boldsymbol{\theta}$. Query Q_n of length $|Q_n|$ for a random image $\boldsymbol{x}$ is generated by drawing $|Q_n|$-times from set $\{1, \ldots, L_m\}$ with probability $p(label) = \hat{\boldsymbol{y}}_{label}$ where $\hat{\boldsymbol{y}} = f\left(\boldsymbol{x}; \boldsymbol{\theta}\right)$ is the distribution of classes predicted by our neural network.

Real User. A human judge is given a random image $\boldsymbol{x}$ from a dataset and describes it in a set of labels $Q_r \subseteq L_m$ generating one $(\boldsymbol{x}, Q_r)$ pair.

Distribution User. Given a set of image-query pairs generated by humans we can infer distribution $\mathcal{C}$ how likely user selects top-k^{th} label the neural network assigned to an image for all $k \in \{1, \ldots, |L_m|\}$. Query Q_d of length $|Q_d|$ for a random image $\boldsymbol{x}$ is generated by drawing $|Q_d|$-times c from distribution $\mathcal{C}$ and taking top-c^{th} label from $\hat{\boldsymbol{y}}$.

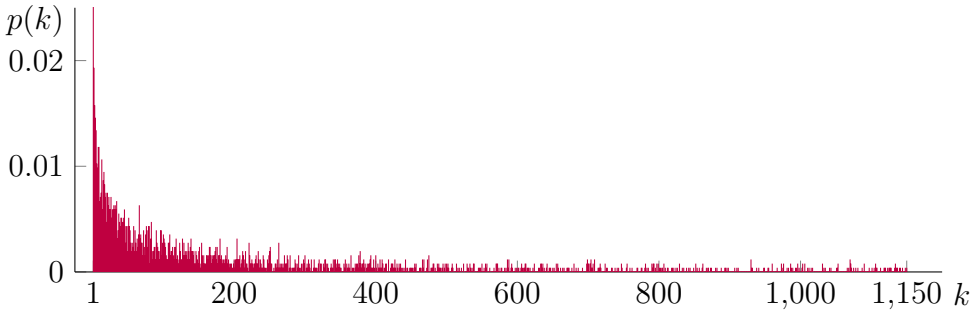

Figure 5.1: Distribution $\mathcal{C}$ how likely user selects top-k^{th} label the neural network assigned to an image. Ideally if user was fully coherent with the neural network $p(1) = 1$ and $p(k) = 0$ for all other $k \in \{2, \ldots, 1150\}$.

VBS competition uses TRECVid Internet Archive Creative Commons (IACC.3) dataset [52] containing 600 hours of recordings in 4593 videos making it a good candidate dataset for the simulations. Since our model is based on images, we only considered 335,944 master shot reference (MSR) keyframes [53][1].

To create the *distribution* user, the author annotated 250 random images from MSR collection by one to five labels from the set L (transformed to a subset of L_m by rules described in Chapter 4.2). During the annotation, images containing random noise and ornaments as well as some low-quality shots were skipped. Further, the distribution $\mathcal{C}$ how likely user selects top-k^{th} label was created (Figure 5.1). We tried to smooth the distribution but it usually deteriorated the results, therefore, we kept using the unmodified one. Comparison of the *network*, the *real* and the *distribution* user is shown in Figure 5.2. We can see, not surprisingly, that the *network* user is an optimistic estimate of the *real* user whereas the *distribution* user is a pessimistic estimate. The difference between the *real* and the *distribution* user is profound especially for a rank in the first dozens of thousands. We suspect this is because if the *real* user selects the top-1$^{\text{st}}$ label, such a label is quite unique and distinguishable in the collection. On the other hand, the *distribution* user can select top-1$^{\text{st}}$ label on a random noise image where its rank can be unpredictable. To address this issue we introduce a new user:

> **Compound User.** Each query Q_c for an image $\boldsymbol{x}$ is generated by the *network* user with probability 1/2 and by the *distribution* user with probability 1/2.

As can be seen in Figure 5.2, the *compound* user much more accurately simulates the *real* user, however, test data would be needed to support our hypothesis.

Simulations show that keyword retrieval model can alone, even in the most optimistic case, find only 50% of queries considering only ranks up to small thousands since browsing in even 10 thousand images cannot be done in reasonable time. But on the other hand, the graphs show that approximately 10% of searched images can be found in the first few hundreds indicating that some browsing after keyword query can be beneficial. We can also see that extending the query by new labels increases the chance of success even in the case of the conservative distribution user. To reduce memory requirements of our retrieval model we also introduce a threshold μ discarding all labels with probability smaller than μ.

[1] During the actual competition, our tool worked with a bigger set of keyframes to better deal with competition's challenges.

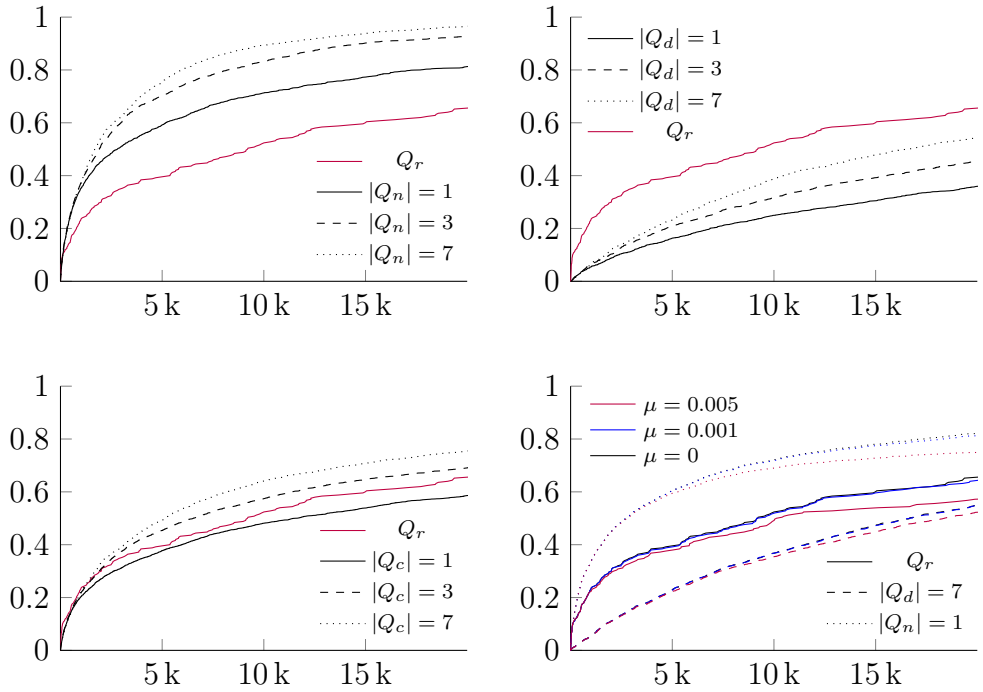

Figure 5.2: Comparison between the *real* user (Q_r) and the *network* (Q_n, top left), the *distribution* (Q_d, top right) and the *compound* user (Q_c, bottom left). The horizontal axis shows position t and the vertical axis shows probability $p(r \leq t)$ how likely random image will be in the first t images given a query constructed by a given user. The bottom right graph shows how discarding all labels with probability less than μ affects the quality of ranking. Based on 2000 queries of the *network* and the *distribution* user, $2000 + 2000$ queries of the *compound* user and 250 queries of the *real* user.

Using simulations, we investigate how different values of μ affect the quality of the ranking. As can be seen in Figure 5.2 (bottom right), using $\mu = 0.005$ deteriorates the quality of ranking quite significantly whereas the threshold of 0.001 has negligible effect. Therefore in production $\mu = 0.001$ is used yielding to approximately 80 labels per image on average, reducing the inverted index size to only 7 percent of the original size.

5.1.1 Similarity Re-Ranking

Many interactive video retrieval tools enable a user to query a database in many modalities. The most relevant shots (images) D are then displayed to the user. If searched image $\boldsymbol{x}$ is not present in the set D, usually the user can sequentially browse the results. However, as datasets get bigger this approach quickly becomes unfeasible. Retrieval systems, therefore, provide an option of similarity browsing based on handcrafted features or more recently deep neural network features [14, 54] speeding up the search if visually and/or semantically similar image is present in the set D. Therefore we evaluate our model together with similarity browsing based on our networks deep features to better estimate the model's performance in the known-item search task. We take the definition of similarity user from [51]:

Ideal **Similarity User.** Given a target image $\boldsymbol{x}$ and a set of the most relevant

images D, the most similar image to $\boldsymbol{x}$ from D is selected.

We define similarity of two images as the cosine of the angle between their activation values in the last average pooling layer of GoogLeNet denoted as $f'(\cdot;\boldsymbol{\theta})$:

$$similarity(\boldsymbol{a},\boldsymbol{b}) = \frac{\langle f'(\boldsymbol{a};\boldsymbol{\theta}), f'(\boldsymbol{b};\boldsymbol{\theta})\rangle}{\|f'(\boldsymbol{a};\boldsymbol{\theta})\|\|f'(\boldsymbol{b};\boldsymbol{\theta})\|} \tag{5.2}$$

where $\boldsymbol{a}$ and $\boldsymbol{b}$ are two images, $\langle\cdot,\cdot\rangle$ is dot product and $\|\cdot\|$ is L_2 norm.

In the simulations the keyword users are used to assign each image in a collection initial rank, and $|D|$ top ranking images are passed to the similarity user. Even though, as reported by [51], humans usually do not select the most similar image based on our *similarity* measure we can use this optimistic approximation to show theoretical limits of our model and to compare how well keyword initialization helps with the *page zero problem* of similarity browsing. Results in Figure 5.3 shows that one re-ranking significantly improves the chance of success whereas the added value of every other re-ranking lowers probably because we converge to some cluster in high dimensional feature space. We can also see that even given the *network* keyword user and the ideal *similarity* user 20% of

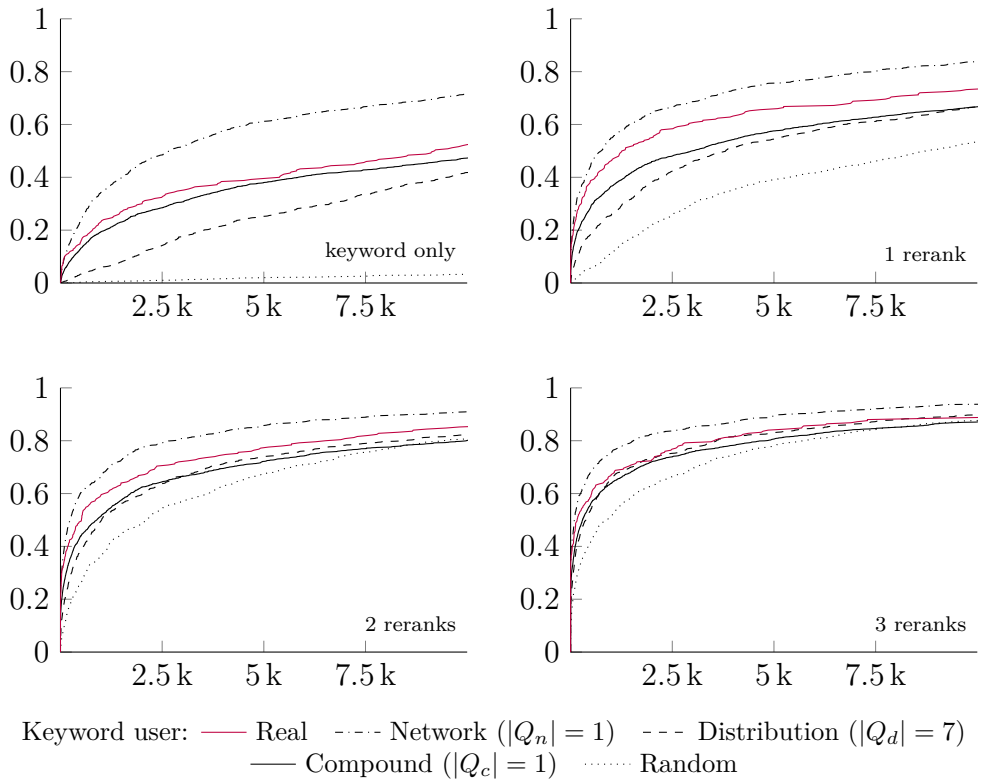

Figure 5.3: Influence of similarity browsing on a ranking. The horizontal axis shows position t and the vertical axis shows probability $p(r \leq t)$ how likely random image will be in the first t images given a query constructed by a given user and a number of re-rankings by the *similarity* user with D being the 50 highest ranked images from the previous iteration or keyword initialization. Based on 1000 queries of the *network*, the *distribution* and the *random* user, $1000 + 1000$ queries of the *compound* user and 250 queries of the *real* user.

images we not found, therefore, proving interactive search using other approaches such as color or edge sketches as a necessity. With respect to the results of simulations with random query initialization, we argue that such small gap between the *random* and the *distribution* user after three re-rankings is due to target dataset's uniformity where keyword initialization cannot help much. This also supports conclusion of [51] that some images are unreachable by similarity re-ranking. Searching for dataset's distinct objects using only similarity browsing is on the other hand, based on our experience with the dataset, almost impossible.

5.2 VBS Competition

Since 2012, The Video Browser Showdown competition [8, 13], organized at the *International Conference on MultiMedia Modeling*, serves as a stage where researches evaluate their interactive video retrieval tools. SIRET group of Charles University debuted at the competition in 2014 [55], winning it in years 2014, 2015 and newly with our keyword-based retrieval model in 2018. Note that an older set of 1390 labels was used there. The set missed multiple key concepts and also contained many duplicate or very similar concepts thus we introduced the label clustering described in Chapter 3.2. However, the rest stayed unchanged hence we can in this section introduce the competition and discuss our tool's results.

VBS competition focuses on known-item search (KIS) and ad-hoc video search (AVS) tasks in a highly interactive way with a human in the loop rather than using only automated retrieval systems. Such a task can highly differ from just automated systems and maybe surprisingly some tools [56, 57] with little to no automated retrieval models can hold up to tools designed on deep learning models. In 2018, VBS comprised of three types of tasks on TRECVid's IACC.3 600 hour dataset [52]: *(1)* Visual known-item search (KIS) task with users searching short selected clip played in loop but with clip getting continually more and more blurred to prevent teams to focus on one image of the clip only. Also recording the played clip is forbidden since such a task is addressed in other computer vision benchmarks. *(2)* Textual known-item search (KIS) task with users searching for short clip only described by text. Due to its inherent difficulty, more information appears over time. *(3)* Ad-hoc search (AVS) task where users are given short text description and goals is to find as many scenes as possible corresponding to the description.

5.2.1 Scoring

For the visual and the textual KIS tasks the following formula was used at VBS 2018:

$$\max\left(0,\ 50 + 50 \cdot \frac{t_L - t}{t_L} - 10 \cdot |WS|\right) \tag{5.3}$$

where t_L is the duration of the task (5 minutes for visual and 7 minutes for textual KIS), t is a time of submission and $|WS|$ is a number of wrong submissions before the correct one. If there is not any correct submission sent in time, no points are awarded. For AVS task time-independent function was used:

$$\frac{|C|}{|C| + |I|/2} \cdot \frac{|q(C)|}{|q(P)|} \tag{5.4}$$

Team	Visual KIS	Textual KIS	AVS	Visual KIS Novice	AVS Novice	**Overall**
SIRET (our) [14]	95 (2)	**100** (1)	71 (4)	67 (4)	83 (2)	**83**
ITEC1 [58]	64 (6)	37 (5)	60 (7)	**100** (1)	64 (4)	65
ITEC2 [6]	29 (8)	47 (3)	81 (2)	85 (3)	75 (3)	63
HTW [54]	91 (3)	41 (4)	61 (6)	50 (5)	61 (5)	61
NECTEC [59]	68 (5)	0	**100** (1)	0	**100** (1)	54
VIREO [60]	**100** (1)	0	50 (8)	86 (2)	30 (8)	53
VITRIVR [61]	79 (4)	55 (2)	64 (5)	0	35 (7)	47
VERGE [62]	62 (7)	0	72 (3)	0	46 (6)	36
VNU [63]	21 (9)	0	41 (9)	48 (6)	14 (9)	25

(·) Standing in individual round.

Table 5.1: VBS 2018 results.

where $|C|$ is a number of correct and $|I|$ a number of incorrect team's submissions and P is set of all correct submissions by all teams representing an artificial recall of 100%. Function $q(\cdot)$ clusters shots of one scene into one frame encouraging teams to find different videos (or segments) rather than to send all frames of one scene (see [13] for a detailed explanation).

During the competition, there were five runs each with four tasks – visual KIS, textual KIS and AVS rounds with tools being operated by experts (usually the creators of the tools) and another visual KIS and AVS rounds with tools operated by novice users selected from conference visitors. The overall score was determined by averaging all five rounds with a winner of each round getting 100 points and others proportionally to the winner.

5.2.2 Results

The results of VBS 2018 are presented in Table 5.1 where it can be seen we won by significant margin scoring 27 percent more points than the second team. However, in Table 5.4, we can see there are still areas for improvement especially in the textual KIS tasks where two of the four tasks were not solved by any team. But we can proudly report that SIRET was the only team to successfully solve both solved textual KIS tasks and all expert visual KIS task even though we finished the round second after a team with only three solved tasks because of the wrong submission penalty in the first task. Interestingly even though strong in the expert visual run, in the novice run we were not able to solve two out of four tasks. This trend of lower performance in novice run can be however observed

$\frac{V}{E}$	$\frac{V}{E}$	$\frac{V}{E}$	$\frac{V}{E}$	$\frac{T}{E}$	$\frac{T}{E}$	$\frac{T}{E}$	$\frac{T}{E}$	$\frac{A}{E}$	$\frac{A}{E}$	$\frac{A}{E}$	$\frac{A}{E}$	$\frac{V}{N}$	$\frac{V}{N}$	$\frac{V}{N}$	$\frac{V}{N}$	$\frac{A}{N}$	$\frac{A}{N}$	$\frac{A}{N}$	$\frac{A}{N}$
2	4	8	7	0	3	0	3	9	9	9	9	1	3	4	3	9	9	9	8

Table 5.2: A number of successful teams per video KIS (V), textual KIS (T) and AVS (A) task with the tool being operated by experts (E) or novices (N). Red tasks were not solved by our tool.

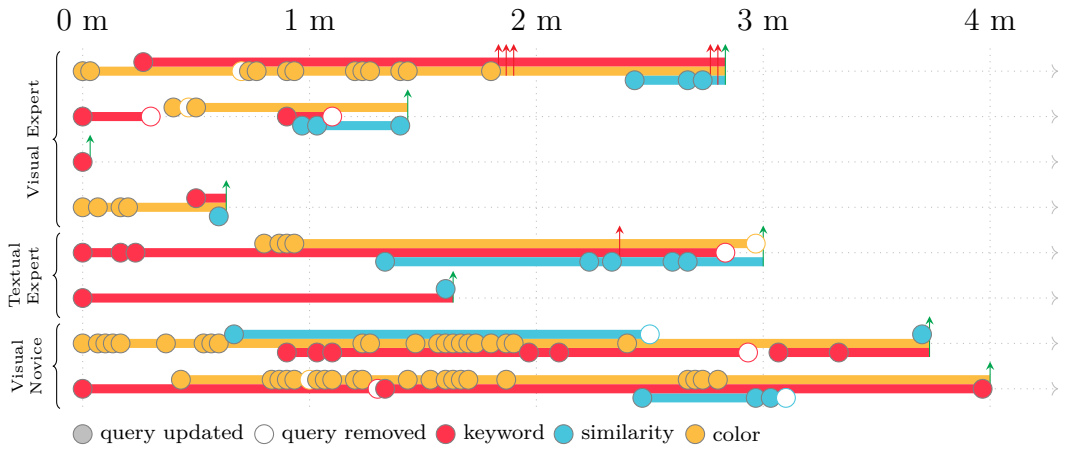

Figure 5.4: Use of our tool's retrieval models in all solved visual and textual KIS task at VBS 2018. The horizontal axis represents time since the task's start, green and red arrows represent correct and incorrect submissions to the competition's server.

with all teams except ITEC1 and ITEC2 with the former being the only team solving three out of four tasks.

Since VBS 2018 mandatory logging of user actions was established such that with each frame sent to the server basic set of actions that lead to the result was submitted. Even though a complete analysis will be published in a separate journal, we can at least show and analyze logs from our tool presented in Figure 5.4. Looking at the visual KIS expert tasks, we can see that both keyword and color sketch can be the equally good initialization of a display from which we can semantically browse the collection. On the other hand in the textual KIS tasks, keyword plays the utmost role in success with all teams starting with keyword queries except ITEC2 which utilized their advanced color sketch interface but in the end falling back on keyword search anyway. When comparing expert and novice users, we can see the expert's more systematic approach utilizing fewer models at once. At least in the second and fourth expert visual task, we can with high probability say a user has been systematically using the color sketch as a filter wheres novices with their rapid model changes could hardly do some systematic filtering. The AVS tasks show the dominance of keyword search with the most of the tasks being solved without using any other model. Visual representation of AVS logs can be seen in the Attachment A.1.

Conclusion

With the help of deep neural network features and clustering, we created an image dataset of commonly occurring objects. We then used the dataset to retrain a neural network and build powerful, easy-to-use query interface. We created artificial users to accurately approximate the capabilities of our model, fine-tune its parameters and to select the best retrieval strategies. We also discussed our successful participation at Video Browse Showdown 2018 which we won by a significant margin.

All in all, we showed that a retrained neural network on our set of carefully selected labels and powerful query formulation interface together with color sketches and semantic browsing based on features from neural networks could beat other state-of-the-art retrieval tools.

In future, we would like to focus on extending the text labeling beyond a fixed set of labels using multimodal approaches of deep learning.

Bibliography

[1] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[2] George Awad, Asad Butt, Jonathan Fiscus, David Joy, Andrew Delgado, Martial Michel, Alan F. Smeaton, Yvette Graham, Wessel Kraaij, Georges Quénot, Maria Eskevich, Roeland Ordelman, Gareth J. F. Jones, and Benoit Huet. Trecvid 2017: Evaluating ad-hoc and instance video search, events detection, video captioning and hyperlinking. In *Proceedings of TRECVID 2017*. NIST, USA, 2017.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[4] Bolei Zhou, Aditya Khosla, Àgata Lapedriza, Aude Oliva, and Antonio Torralba. Object detectors emerge in deep scene cnns. *CoRR*, abs/1412.6856, 2014.

[5] Rui Hu and John Collomosse. Motion-sketch based video retrieval using a trellis levenshtein distance. In *Pattern Recognition (ICPR), 2010 20th International Conference on*, pages 121–124. IEEE, 2010.

[6] Andreas Leibetseder, Sabrina Kletz, and Klaus Schoeffmann. Sketch-based similarity search for collaborative feature maps. In *MultiMedia Modeling*, pages 425–430, Cham, 2018. Springer International Publishing.

[7] Adam Blažek. Efficient video retrieval using complex sketches and exploration based on semantic descriptors. 2016.

[8] Claudiu Cobârzan, Klaus Schoeffmann, Werner Bailer, Wolfgang Hürst, Adam Blažek, Jakub Lokoč, Stefanos Vrochidis, Kai Uwe Barthel, and Luca Rossetto. Interactive video search tools: a detailed analysis of the video browser showdown 2015. *Multimedia Tools and Applications*, 76(4):5539–5571, 2017.

[9] Nicolae Suditu and Francois Fleuret. Heat: Iterative relevance feedback with one million images. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2118–2125. IEEE, 2011.

[10] Klaus Schoeffmann, David Ahlström, and Marco A Hudelist. 3-d interfaces to improve the performance of visual known-item search. *IEEE Transactions on Multimedia*, 16(7):1942–1951, 2014.

[11] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: Lessons learned from the 2015 mscoco image captioning challenge.

IEEE transactions on pattern analysis and machine intelligence, 39(4):652–663, 2017.

[12] Luca Bertinetto, Jack Valmadre, Joao F Henriques, Andrea Vedaldi, and Philip HS Torr. Fully-convolutional siamese networks for object tracking. In *European conference on computer vision*, pages 850–865. Springer, 2016.

[13] J. Lokoc, W. Bailer, K. Schoeffmann, B. Muenzer, and G. Awad. On influential trends in interactive video retrieval: Video browser showdown 2015-2017. *IEEE Transactions on Multimedia*, pages 1–1, 2018.

[14] Jakub Lokoč, Gregor Kovalčík, and Tomáš Souček. Revisiting siret video retrieval tool. In *MultiMedia Modeling*, pages 419–424, Cham, 2018. Springer International Publishing.

[15] Jakub Lokoč, Tomáš Souček, and Gregor Kovalčik. Using an interactive video retrieval tool for lifelog data. In *Proceedings of the 2018 ACM Workshop on The Lifelog Search Challenge*, LSC '18, pages 15–19, New York, NY, USA, 2018. ACM.

[16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[17] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251 – 257, 1991.

[18] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

[19] Ivan Krasin, Tom Duerig, Neil Alldrin, Vittorio Ferrari, Sami Abu-El-Haija, Alina Kuznetsova, Hassan Rom, Jasper Uijlings, Stefan Popov, Andreas Veit, Serge Belongie, Victor Gomes, Abhinav Gupta, Chen Sun, Gal Chechik, David Cai, Zheyun Feng, Dhyanesh Narayanan, and Kevin Murphy. Openimages: A public dataset for large-scale multi-label and multi-class image classification. *Dataset available from https://github.com/openimages*, 2017.

[20] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[21] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[22] Bart Thomee, David A. Shamma, Gerald Friedland, Benjamin Elizalde, Karl Ni, Douglas Poland, Damian Borth, and Li-Jia Li. The new data and new challenges in multimedia research. *CoRR*, abs/1503.01817, 2015.

[23] Giuseppe Amato, Fabrizio Falchi, Claudio Gennaro, and Fausto Rabitti. Searching and annotating 100m images with yfcc100m-hnfc6 and mi-file. In *Proceedings of the 15th International Workshop on Content-Based Multimedia Indexing*, page 26. ACM, 2017.

[24] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[25] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[26] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

[27] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[28] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014.

[29] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[30] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[31] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.

[32] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. *arXiv preprint arXiv:1707.07012*, 2017.

[33] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[34] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

[35] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *International conference on machine learning*, pages 647–655, 2014.

[36] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826, 2016.

[37] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, volume 4, page 12, 2017.

[38] Alfredo Canziani, Adam Paszke, and Eugenio Culurciello. An analysis of deep neural network models for practical applications. *CoRR*, abs/1605.07678, 2016.

[39] George A. Miller. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41, November 1995.

[40] Jia Deng, Alexander C Berg, Kai Li, and Li Fei-Fei. What does classifying more than 10,000 image categories tell us? In *European conference on computer vision*, pages 71–84. Springer, 2010.

[41] N. Silberman and S. Guadarrama. TensorFlow-Slim image classification model library, 2016. `https://github.com/tensorflow/models/tree/master/research/slim`. Last accessed on 2018-04-02.

[42] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[43] Sanjoy Dasgupta. *The hardness of k-means clustering*. Department of Computer Science and Engineering, University of California, San Diego, 2008.

[44] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.

[45] Hervé Abdi and Lynne J Williams. Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, 2(4):433–459, 2010.

[46] Stanford University, Princeton University. ImageNet download original images, 2016. `http://www.image-net.org/download-images`. Last accessed on 2018-04-02.

[47] Patrice Y Simard, David Steinkraus, John C Platt, et al. Best practices for convolutional neural networks applied to visual document analysis. In *ICDAR*, volume 3, pages 958–962, 2003.

[48] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan,

Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[49] Anastasia Moumtzidou, Theodoros Mironidis, Fotini Markatopoulou, Stelios Andreadis, Ilias Gialampoukidis, Damianos Galanopoulos, Anastasia Ioannidou, Stefanos Vrochidis, Vasileios Mezaris, Ioannis Kompatsiaris, et al. Verge in vbs 2017. In *International Conference on Multimedia Modeling*, pages 486–492. Springer, 2017.

[50] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern information retrieval*. Addison-Wesley, 2011.

[51] Gregor Kovalčík. Comparison of signature-based and semantic similarity models. 2017.

[52] George Awad, Jonathan Fiscus, David Joy, Martial Michel, Alan F Smeaton, Wessel Kraaij, Maria Eskevich, Robin Aly, Roeland Ordelman, Gareth J. F Jones, Benoit Huet, and Martha Larson. TRECVID 2016. Evaluating video search, video event detection, localization and hyperlinking. In *TRECVID 2016, 20th International Workshop on Video Retrieval Evaluation, 14-16 November 2016, Gaithersburg, Ma, USA*, 11 2016.

[53] National Institute of Standards and Technology. IACC.3 master shot reference, 2016. `https://www-nlpir.nist.gov/projects/tv2016/pastdata/iacc.3.master.shot.reference`. Last accessed on 2018-07-01.

[54] Kai Uwe Barthel, Nico Hezel, and Klaus Jung. Fusing keyword search and visual exploration for untagged videos. In *MultiMedia Modeling*, pages 413–418, Cham, 2018. Springer International Publishing.

[55] Jakub Lokoč, Adam Blažek, and Tomáš Skopal. Signature-based video browser. In *MultiMedia Modeling*, pages 415–418, Cham, 2014. Springer International Publishing.

[56] Wolfgang Hürst, Rob van de Werken, and Miklas Hoet. A storyboard-based interface for mobile video browsing. In *MultiMedia Modeling*, pages 261–265, Cham, 2015. Springer International Publishing.

[57] Aaron Duane, Cathal Gurrin, and Wolfgang Huerst. Virtual reality lifelog explorer: Lifelog search challenge at acm icmr 2018. In *Proceedings of the 2018 ACM Workshop on The Lifelog Search Challenge*, LSC '18, pages 20–23, New York, NY, USA, 2018. ACM.

[58] Manfred Jürgen Primus, Bernd Münzer, Andreas Leibetseder, and Klaus Schoeffmann. The itec collaborative video search system at the video browser showdown 2018. In *MultiMedia Modeling*, pages 438–443, Cham, 2018. Springer International Publishing.

[59] Sitapa Rujikietgumjorn, Nattachai Watcharapinchai, and Sanparith Marukatat. Sloth search system. In *MultiMedia Modeling*, pages 431–437, Cham, 2018. Springer International Publishing.

[60] Phuong Anh Nguyen, Yi-Jie Lu, Hao Zhang, and Chong-Wah Ngo. Enhanced vireo kis at vbs 2018. In *MultiMedia Modeling*, pages 407–412, Cham, 2018. Springer International Publishing.

[61] Luca Rossetto, Ivan Giangreco, Ralph Gasser, and Heiko Schuldt. Competitive video retrieval with vitrivr. In *MultiMedia Modeling*, pages 403–406, Cham, 2018. Springer International Publishing.

[62] Anastasia Moumtzidou, Stelios Andreadis, Foteini Markatopoulou, Damianos Galanopoulos, Ilias Gialampoukidis, Stefanos Vrochidis, Vasileios Mezaris, Ioannis Kompatsiaris, and Ioannis Patras. Verge in vbs 2018. In *MultiMedia Modeling*, pages 444–450, Cham, 2018. Springer International Publishing.

[63] Thanh-Dat Truong, Vinh-Tiep Nguyen, Minh-Triet Tran, Trang-Vinh Trieu, Tien Do, Thanh Duc Ngo, and Dinh-Duy Le. Video search based on semantic extraction and locally regional object proposal. In *MultiMedia Modeling*, pages 451–456, Cham, 2018. Springer International Publishing.

List of Figures

List of Tables

A. Attachments

A.1 AVS Tasks at VBS 2018

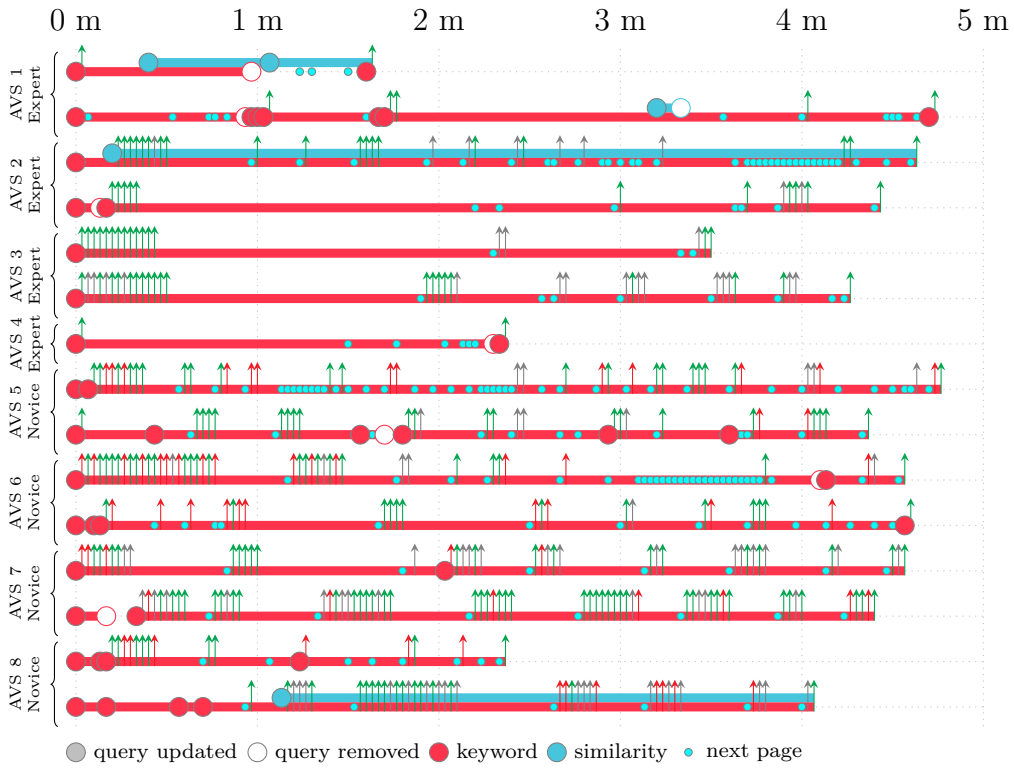

Figure A.1: Use of our tool's retrieval models in all AVS task at VBS 2018. The most of the tasks have two timelines because submissions were made from two computers. The horizontal axis represents time since the task's start, green and red arrows represent correct and incorrect submissions to the competition's server. Gray arrow represents a repeated submission of the same frame.

A.2 CD Content

The attached CD contains the following content:

- Python scripts used for model training and offline data processing with a user documentation.
- WPF (Windows Presentation Foundation) demo application source codes written in C#.
- Compiled application targeted for 32bit Windows operating system with .NET 4.6.1. We include a small dataset for a demonstration.
- List of selected ImageNet classes.

A.3 Selected ImageNet Classes

n00007846	n01741943	n02125311	n02537085	n02839910	n03002341	n03236735	n03496892	n03721590
n00326094	n01743086	n02128385	n02584449	n02840245	n03010915	n03239726	n03497657	n03722288
n00433802	n01747885	n02128925	n02607201	n02843553	n03028079	n03240683	n03505504	n03725035
n00440747	n01754876	n02129165	n02626762	n02843684	n03031422	n03249569	n03506370	n03728437
n00440941	n01770393	n02129604	n02667379	n02846511	n03035252	n03251533	n03508101	n03736064
n00441073	n01772222	n02130308	n02686379	n02849154	n03037709	n03251766	n03512147	n03743279
n00442115	n01776313	n02131653	n02686568	n02850732	n03038685	n03256166	n03513137	n03743902
n00444846	n01791625	n02132136	n02687172	n02853016	n03046257	n03261776	n03517760	n03744276
n00444937	n01792042	n02132580	n02687423	n02858304	n03047052	n03271574	n03518943	n03745146
n00445055	n01792158	n02134084	n02687821	n02867715	n03057021	n03281145	n03521544	n03753077
n00445226	n01794158	n02139199	n02690373	n02870526	n03063338	n03287733	n03531281	n03759954
n00445351	n01795088	n02164464	n02691156	n02870880	n03064758	n03291741	n03532672	n03760671
n00445802	n01806143	n02165456	n02692232	n02871439	n03072201	n03294833	n03541091	n03761084
n00446980	n01811909	n02190166	n02692877	n02876657	n03073296	n03303217	n03541696	n03763968
n00447073	n01812337	n02200198	n02694662	n02877962	n03074380	n03307792	n03541923	n03764276
n00448466	n01816887	n02206856	n02699494	n02879087	n03074855	n03309808	n03544143	n03773035
n00450335	n01823013	n02208280	n02699629	n02879718	n03075634	n03316406	n03544360	n03773504
n00451370	n01843383	n02212062	n02701002	n02880940	n03075768	n03320046	n03545150	n03775199
n00452293	n01844917	n02213107	n02704949	n02882190	n03079230	n03320421	n03545470	n03788365
n00453935	n01846331	n02219486	n02710201	n02883344	n03080497	n03322704	n03546340	n03790512
n00463543	n01855672	n02226429	n02715229	n02892767	n03082807	n03327234	n03547229	n03793489
n00464894	n01858441	n02233338	n02726305	n02897820	n03082979	n03329302	n03551395	n03797390
n00468480	n01877134	n02274259	n02732072	n02898711	n03084420	n03329663	n03555426	n03804744
n00469651	n01882714	n02283201	n02733524	n02900705	n03085013	n03335030	n03558176	n03805725
n00471613	n01887474	n02309337	n02738535	n02908217	n03085219	n03343560	n03560430	n03814906
n00478262	n01887623	n02311060	n02740533	n02909870	n03085602	n03345487	n03577090	n03815615
n00479440	n01887787	n02317335	n02749479	n02913152	n03087069	n03345837	n03581125	n03819994
n00479887	n01910747	n02324045	n02753044	n02916179	n03089879	n03346455	n03584254	n03820318
n00480211	n01915811	n02326432	n02760429	n02924116	n03094503	n03349469	n03584829	n03822656
n00480508	n01922303	n02330245	n02761557	n02927161	n03099274	n03351979	n03589791	n03837869
n00480993	n01944390	n02331046	n02764044	n02930766	n03101156	n03352628	n03593526	n03839671
n00482298	n01956481	n02342885	n02766320	n02932019	n03101517	n03354903	n03594734	n03841666
n00825773	n01970164	n02355227	n02766534	n02932400	n03103396	n03358172	n03594945	n03842012
n01055165	n01976957	n02360282	n02769290	n02933112	n03106898	n03359137	n03595860	n03854065
n01322604	n01982650	n02363005	n02769748	n02934168	n03108853	n03367410	n03596285	n03857828
n01439514	n01986806	n02364673	n02773037	n02934451	n03113152	n03372029	n03604156	n03862676
n01443537	n02002075	n02374451	n02774630	n02936714	n03113657	n03379204	n03610418	n03868863
n01459791	n02007558	n02382437	n02776205	n02938886	n03115180	n03386011	n03612814	n03873416
n01482330	n02021795	n02382948	n02776631	n02942699	n03120491	n03388043	n03619890	n03874599
n01495701	n02022684	n02387254	n02777734	n02944459	n03120778	n03390983	n03620967	n03877845
n01503061	n02041246	n02391049	n02778456	n02946348	n03121298	n03397266	n03621377	n03880531
n01504344	n02051845	n02391994	n02778669	n02946921	n03126707	n03397947	n03623556	n03885028
n01514668	n02055803	n02395406	n02782093	n02948072	n03128248	n03400231	n03624134	n03888257
n01514859	n02062744	n02396014	n02785648	n02950826	n03128519	n03404149	n03627232	n03904909
n01518878	n02064816	n02398521	n02789487	n02951358	n03135532	n03408054	n03636248	n03906997
n01519873	n02068974	n02402425	n02790669	n02952585	n03136369	n03416489	n03636649	n03908204
n01529672	n02071294	n02403325	n02791270	n02956699	n03139464	n03417042	n03642806	n03908714
n01533339	n02076196	n02403454	n02793495	n02958343	n03141327	n03417345	n03643253	n03916031
n01579260	n02077923	n02410509	n02797295	n02959942	n03145522	n03427296	n03649909	n03918737
n01605630	n02084071	n02411705	n02800497	n02963503	n03147509	n03432129	n03656484	n03928116
n01610955	n02087551	n02416519	n02802426	n02964843	n03150232	n03438257	n03660909	n03930630
n01613294	n02092468	n02419796	n02806379	n02965300	n03151077	n03439814	n03661043	n03931765
n01616318	n02099029	n02423022	n02807616	n02968473	n03154073	n03441112	n03662601	n03933933
n01621127	n02099601	n02430045	n02807731	n02970685	n03160309	n03443371	n03662887	n03938244
n01629276	n02100399	n02437136	n02808440	n02971167	n03179701	n03444034	n03665366	n03948459
n01639765	n02104523	n02439033	n02810471	n02974697	n03180011	n03445617	n03665924	n03955489
n01662784	n02106662	n02444819	n02812201	n02977058	n03180504	n03446070	n03666591	n03963645
n01663401	n02106854	n02445715	n02813544	n02977936	n03187268	n03452267	n03673450	n03965456
n01670092	n02108672	n02456962	n02814860	n02978881	n03188531	n03457902	n03676483	n03967942
n01674464	n02109811	n02480495	n02818832	n02980441	n03200701	n03461288	n03679712	n03973628
n01674990	n02110185	n02480855	n02821627	n02981911	n03201776	n03461385	n03682487	n03977966
n01693783	n02110341	n02481823	n02823428	n02984203	n03206908	n03467517	n03683079	n03984381
n01695060	n02110958	n02484322	n02824448	n02988304	n03207941	n03467984	n03684823	n03985232
n01696633	n02114100	n02496913	n02827606	n02991302	n03211117	n03478907	n03688943	n03991062
n01697178	n02115335	n02503517	n02828884	n02991847	n03218198	n03481172	n03690938	n03995372
n01713764	n02117135	n02508021	n02834778	n02992529	n03220513	n03485794	n03694639	n03995856
n01726692	n02118333	n02510455	n02837789	n02999936	n03221720	n03488188	n03706653	n03996145
n01727646	n02121808	n02512053	n02839110	n03000684	n03233905	n03492250	n03709206	n03996416
n01729977	n02122948	n02534734	n02839351	n03001627	n03234164	n03492542	n03710193	n04004475

n04004767	n04149813	n04314914	n04508949	n07592481	n07747607	n07868200	n09406793	n10618342
n04009552	n04152593	n04315948	n04511002	n07597365	n07747951	n07868340	n09409752	n10622053
n04009801	n04153751	n04317175	n04517823	n07598256	n07749582	n07869775	n09416076	n10630188
n04012084	n04154565	n04317325	n04519153	n07599998	n07749731	n07870167	n09421799	n10655594
n04019101	n04161981	n04323819	n04520170	n07605040	n07750872	n07871810	n09433442	n10665698
n04026417	n04190052	n04330267	n04523831	n07607605	n07751004	n07873807	n09436444	n10679174
n04028315	n04194289	n04330340	n04524313	n07609840	n07751148	n07874780	n09436708	n10694258
n04028581	n04197110	n04334599	n04525305	n07612367	n07751451	n07875436	n09437454	n10707233
n04039381	n04197391	n04335209	n04526964	n07612996	n07753113	n07880458	n09468604	n10718131
n04041069	n04199027	n04335435	n04532670	n07614500	n07753275	n07880751	n09472597	n10721321
n04041544	n04202417	n04335886	n04535524	n07619409	n07753592	n07880968	n09618957	n10755080
n04044716	n04204081	n04347754	n04536866	n07631926	n07753743	n07881800	n09619168	n10787470
n04057047	n04204238	n04350905	n04543158	n07642933	n07755411	n07886849	n09624168	n11536673
n04061793	n04204347	n04355338	n04545858	n07648913	n07757132	n07891726	n09632518	n11608250
n04063373	n04205318	n04356056	n04547592	n07679356	n07758680	n07892512	n09636339	n11624531
n04065789	n04206225	n04358117	n04549629	n07680932	n07762244	n07892813	n09767197	n11669921
n04068441	n04208210	n04361260	n04550184	n07686873	n07762913	n07893642	n09785659	n11807979
n04070727	n04208936	n04366367	n04552348	n07687381	n07763629	n07906111	n09792969	n11900569
n04074963	n04210120	n04370048	n04554684	n07690273	n07763987	n07907943	n09818022	n11931918
n04078574	n04215402	n04371430	n04555897	n07695742	n07764155	n07908411	n09820263	n11939491
n04081281	n04217882	n04371563	n04557648	n07695965	n07764847	n07911677	n09828216	n11944196
n04090263	n04220250	n04373894	n04560292	n07697100	n07765073	n07923748	n09841188	n11978233
n04091097	n04225729	n04376876	n04568557	n07697537	n07767344	n07924033	n09842047	n12041446
n04096066	n04225987	n04379243	n04579667	n07700003	n07767847	n07927512	n09859152	n12102133
n04097866	n04228054	n04380346	n04586932	n07710616	n07802026	n07927931	n09874725	n12142085
n04099175	n04228581	n04382880	n04587559	n07711080	n07803093	n07929172	n09894445	n12205694
n04102406	n04230808	n04389033	n04587648	n07712856	n07803545	n07929351	n09913455	n12268246
n04105068	n04231693	n04392113	n04588739	n07713895	n07804323	n07929519	n09918248	n12300840
n04108268	n04233124	n04397768	n04590553	n07714990	n07806221	n07932841	n09918554	n12301445
n04112252	n04235860	n04398044	n04592099	n07715103	n07812184	n07933274	n09930876	n12360108
n04113406	n04238128	n04404997	n04592741	n07715221	n07818277	n07935737	n09990415	n12387633
n04114844	n04243546	n04409515	n04594218	n07715561	n07822197	n07936263	n10019406	n12387839
n04115456	n04243941	n04411264	n04594489	n07718472	n07822518	n07954211	n10043643	n12405714
n04116294	n04250224	n04417809	n04594828	n07719213	n07822845	n08182379	n10049363	n12421683
n04118021	n04251701	n04421872	n04605726	n07720442	n07823460	n08249459	n10084295	n12454159
n04120489	n04252077	n04438304	n04606251	n07720615	n07831267	n08256735	n10091651	n12582231
n04122685	n04252225	n04442312	n04606574	n07721325	n07840804	n08494231	n10112129	n12633638
n04122825	n04252653	n04446276	n04610013	n07722217	n07841495	n08505018	n10120330	n12651821
n04123740	n04254777	n04449966	n06267145	n07723330	n07841639	n08517676	n10142391	n12707781
n04124098	n04256520	n04453156	n06278475	n07723559	n07842308	n08518171	n10142747	n12724942
n04125021	n04257790	n04453390	n06359193	n07724943	n07842753	n08640531	n10148035	n12752205
n04127904	n04257986	n04456115	n06470073	n07725376	n07843636	n08640739	n10151760	n12822115
n04128413	n04259630	n04459362	n06595351	n07730207	n07844042	n09217230	n10153594	n12987056
n04128499	n04264914	n04463679	n06596364	n07730406	n07848196	n09238926	n10182190	n12992868
n04133789	n04265904	n04465501	n06793231	n07731952	n07849336	n09246464	n10226413	n12997919
n04134008	n04266375	n04465666	n06794110	n07732636	n07852833	n09247410	n10249459	n12998815
n04134523	n04266486	n04467307	n06874185	n07734017	n07858978	n09256479	n10260800	n13084184
n04137444	n04269944	n04467665	n06892775	n07734744	n07859583	n09270735	n10334009	n13104059
n04138977	n04270147	n04468005	n06998748	n07735510	n07860988	n09282208	n10340312	n13136316
n04139395	n04273569	n04469514	n07572957	n07735687	n07862348	n09288635	n10366966	n13136556
n04139859	n04284002	n04469813	n07574602	n07736692	n07863374	n09289331	n10405694	n13137409
n04140064	n04285008	n04476259	n07575076	n07739125	n07863547	n09303008	n10421470	n14696793
n04141076	n04285803	n04477219	n07575726	n07742704	n07864934	n09308572	n10467179	n14698884
n04141975	n04294879	n04487394	n07576182	n07743544	n07865484	n09308743	n10469874	n14844693
n04143140	n04295881	n04490091	n07580592	n07743902	n07866015	n09348460	n10470779	n14974264
n04143897	n04296562	n04495843	n07582609	n07744811	n07866723	n09359803	n10493685	n15019030
n04146050	n04309049	n04505036	n07585557	n07745466	n07867021	n09376198	n10513823	n15075141
n04146614	n04309348	n04507155	n07588947	n07745940	n07867324	n09393605	n10582746	
n04148054	n04313503	n04508489	n07590320	n07747055	n07867616	n09403734	n10599806	