Kobe University, NICT and University of Siegen at TRECVID 2017 AVS Task

Zhenying He*, Takashi Shinozaki†‡, Kimiaki Shirahama‡, Marcin Grzegorzek‡, and Kuniaki Uehara*
* Graduate School of System Informatics, Kobe University
† Center for Information and Neural Networks, National Institute of Information and Communications Technology (NICT)
‡ Pattern Recognition Group, University of Siegen

Abstract—This paper presents the result of the TRECVID 2017 AVS task by kobe_nict_siegen team. Consisting of three research institutes Kobe University, NICT and University of Siegen. We submitted the following three runs.

1) kobe_nict_siegen_D_M_1: This run uses feature vectors extracted by a pre-trained convolutional neural network (CNN) as input for a small-scale multi-layer neural network called micro neural network (microNN). The microNN is a lightweight detector fine-tuned to a target concept using a balanced set of positive and negative data from ImageNet and IACC video datasets. Finally, the results of several microNNs are combined to generate the ranked result.

2) kobe_nict_siegen_D_M_2: This run is basically identical to kobe_nict_siegen_D_M_1, but applies additional motion features. The motion features are extracted by a motion CNN which involves biologically-inspired motion thresholding and competitive learning.

3) kobe_nict_siegen_D_M_3: This run is a degraded version of kobe_nict_siegen_D_M_1, and apply fully averaged pooling to feature vectors extracted with CNN, the dimension of the vector is reduced from $3 \times 3 \times 2048$ to 2048.

We also use a word2vec model to find synonyms of concepts in order to improve the performance.

I. INTRODUCTION

By taking part in the ad-hoc video search task at TRECVID 2016, we were able to ascertain the efficiency of training microNNs for various concepts as well as their reasonable performance in concept retrieval tasks. This year [2], we employed a word2vec model to compute vector representations of words in the text description of a query, and find suitable concepts. Features extracted from a CNN have shown a great effectiveness in various recognition problems. Using the features obtained from pre-trained CNN as input, we can train several microNNs which are individually lightweight concept detectors for concepts. Thanks to having only one hidden layer, each microNN can be trained much faster than a lot traditional methods such as SVM. The final classification results are obtained by combining the outputs of multiple microNNs.

II. THE PROPOSED METHOD

Given an ad-hoc query sentence, we manually pick up its key concepts. We manually pick up its key concepts. This process is enhanced by computing similarities between vector representations of words in the query sentence and the ones of concept names based on a word2vec model [8]. Key concepts are selected from a set of concepts with high similarities. From each shot, we take one frame every 30 frames as a key frame. Feature vectors are extracted from these key frames using a CNN, and used to train a microNN. Supplementary feature vectors extracted from images in ImageNet dataset are used.

Fig. 1 shows the list of concepts’ name in ImageNet and TRECVID video dataset selected for each query in this study. An arrow indicates that a model in the left-hand side is transferred to the one in the right-hand side. For example, in query 531, the concept “people” in ImageNet can be transferred to the concept “Two_People” in TRECVID video dataset. Fig. 2 shows an overview of the three methods used to construct microNNs for the concepts in Fig. 1. The outputs of a hidden layer in a CNN are used as features to train microNNs in kobe_nict_siegen_D_M_1, kobe_nict_siegen_D_M_2, and kobe_nict_siegen_D_M_3. Transfer learning is performed to train a microNN on video features by initializing its weight parameters as the ones learned by a microNN on image features. In addition, we combine the features obtained from ResNet [3] and motion CNN as input to microNNs in the kobe_nict_siegen_D_M_2.

ResNet is a deep residual network with shortcut connection added to each pair of filters. It won the first place on tasks such as ImageNet detection, ImageNet localization, COCO detection, and COCO segmentation [4] in ILSVRC & COCO 2015 competitions. ResNet has a lower computational complexity despite its very deep architecture. For the best tradeoff between computational cost and accuracy, we use the 101 layers version of ResNet constructed by Chainer [5], which is a flexible framework for deep learning similar to Caffe [6], TensorFlow [7] and so on. It supports various network architectures and CUDA acceleration.

A. Concept Selection Using Word2Vec

Our first task is to find concepts related to a given topic. To this end, we employ a word2vec model that produces vector representations of words based on a neural network with a single hidden layer [8]. This model is used to compute vector
representations of words in the topic and the ones from the concept names. Then, for each word in the topic, we select concepts whose vector representations are highly similar to that of the word. The word2vec model can accurately measure this kind of linguistic similarities because of its simple network architecture. In other words, compared to complex neural networks (with non-linear hidden layers), the word2vec model can be trained on a much larger dataset, so that its final performance is better than those of the complex ones.

Our word2vec model is based on the “Continuous Bag-Of-Words” (CBOW) architecture [8]. Its training criterion is to correctly classify a word $w_t$ based on $N$ surrounding words $w_{t+k}$ where $k \in \{-N/2, \cdots, -1, 1, \cdots, N/2\}$ (i.e., $N/2$ previous and $N/2$ future words). Here, $w_t$ (or $w_{t+k}$) is represented as a $V$-dimensional one-hot vector where $V$ is the vocabulary size, and only the dimension corresponding to $w_t$ (or $w_{t+k}$) takes one and all the others zero. The input layer consists of one-hot vectors for $w_{t+k}$s. The hidden layer projects these vectors into $D$-dimensional vectors using a common $V \times D$ projection matrix $W$, and takes their average vector as the overall intermediate representation. It should be noted that, assuming $w_{t+k}$ is the $i$th word in the vocabulary (i.e., the $i$th dimension of its one-hot vector is one), the projection of $w_{t+k}$ picks up the $i$th row in $W$. Hence, once $W$ is optimised, each row will be the vector representation of a word. After the hidden layer, the overall intermediate representation for $w_{t+k}$ is projected back into a $V$-dimensional vector using a $D \times V$ matrix $W'$ with softmax normalization. This vector is a probabilistic estimation of $w_t$, and training our word2vec model is equivalent to optimising $W$ and $W'$ that yield the most accurate estimation for words in the vocabulary.

In our implementation, we build a word2vec model using “Gensim”, a Python library for text analysis on large-scale corpora [9]. The model is trained using the English Wikipedia corpus with a vocabulary of size $V = 1,449,029$, including single words and word pairs (bi-grams) that appear more than 10 times in the corpus (word pairs are treated in the same way as single words). Our preliminary experiment showed that this setting leads to more meaningful vector representations than only considering single words. Our model is trained on $D = 200$-dimensional vector representations of words based
on $N = 10$ surrounding words. For each word in a topic, our word2vec model is used to select 50 most related concepts, among which a few really meaningful concepts are chosen manually.

### B. CNN Feature extraction

In general, it is not reasonable to learn a deep neural network from scratch for concept detection in videos, because of the huge computational cost to train it using a large amount of data. Pre-trained networks, such as AlexNet [10], ResNet [3] and GoogLeNet [11], are usually transferred to a classifier suitable for a target problem. In this study, we use ResNet implemented in Chainer [5] framework for the feature extraction. We focus on a phenomenon where lower layers in a deep neural network characterize visual features that can be used universally for various images or videos. Based on this, we use a layers of ResNet101 as feature extractor.

We take the output of the second last layer in ResNet101 as a feature vector. We use different pooling approaches to obtain feature vectors with varied dimensionals, and aim to compare their effect on detection accuracies. In kobe_nict_siegen_D_M_1 and kobe_nict_siegen_D_M_2, we change the fully averaged pooling to partly averaged pooling and obtain the feature of $3 \times 3 \times 2048$ dimensions. As for kobe_nict_siegen_D_M_3, we retain the original fully averaged pooling layer, and take the $2048$ dimensional feature. In kobe_nict_siegen_D_M_2, we combine a feature extracted by a motion CNN with the one by ResNet, in order to find out whether their combination affects the recognition accuracy or not. According to the following experiments, we found that kobe_nict_siegen_D_M_1 has achieved a better performance in this task. This feature extraction is applied to images in the ImageNet dataset and videos in the TRECVID dataset.

### C. Motion Feature Extraction

In order to extract motion features in videos, we develop a novel method which applies traditional competitive learning to a conventional CNN. Previous DNN studies on motion analyse largely used two-stream convolutional networks [13] which employ dense optical flow [14] for motion feature extraction. Optical flow is obviously a hand engineered feature, and does not use the most important characteristic of DNN, representation learning, which automatically obtains optimal features through the learning process. Another choice for motion analysis is C3D [15]. It enables end-to-end motion analysis by DNN, and performs representation learning of motion features. However, it basically uses $3 \times 3$ filters, and the resolution, especially the spatial resolution, is not enough for some complex movies. A filter of conventional CNN only has spatial dimensions, but a motion filter has an additional temporal dimension. The increase in the dimension drastically expands the degree of freedom of the filter and makes it difficult to learn parameters. To address this problem, we introduce biologically-inspired motion thresholding and competitive learning into a CNN.

Competitive learning is a traditional learning method for NNs [16]. It exhibits quite strong representation learning, and is applied for Neocognitron which is an archetype of CNN [17], and also self organizing maps (SOMs) [18]. The learning process is based on winner takes all (WTA) dynamics: only the winner neuron which has the maximum output value is updated by Hebb rule. In this study, we unified competitive learning with back propagation (BP) learning of CNN and tried to extract motion features of videos. Firstly, competitive learning is used as pre-training to obtain primitive motion features. Then, subsequent BP learning performs fine tuning with a discrimination task.

Fig. 3 shows the structure of the motion CNN which processes motion information. The structure is based on AlexNet [19], and performs additional competitive learning in the first layer.

The input frame is firstly converted to grayscale, and the edges extracted by Laplacian filter. Then, three temporally consecutive frames are used as input for the motion CNN.

The information flow of static and motion images are segregated in the biological visual processing system [20]. With this idea, our network also separates the motion information from the static one by thresholding the temporal difference of the images. For each receptive field (RF), we calculate the spatial average of the difference between two temporally consecutive frames. The RF outputs a value only when the difference exceeds the threshold. Otherwise, the output is zero. We determined the threshold $\theta$ as 1.2 empirically.

The parameter optimization is performed with conventional stochastic gradient descent (SGD). In competitive learning, the gradient is calculated by feedforward information, and weights are updated with SGD. We use IACC.1.A-C as the training dataset. The mini-batch size is 100, and 5,000 iterations are applied. Fig. 4 represents motion filters obtained by competitive learning. If we don’t use any threshold ($\theta = 0.0$), almost all filters end up processing static information (Fig. 4(a)). If we set appropriate threshold ($\theta = 1.2$), the network can effectively obtain motion filters (Fig. 4(b)). In the subsequent BP learning for the fine-tuning, the gradient is calculated from the backward information, and weights are updated with SGD. We use UCF-101 [21] as the training dataset. Fig. 5 shows error rates and losses during fine-tuning. When the iteration reaches around 20,000, the loss decrease tends to be stabilize.
After the fine-tuning, we use the output of fc6 layer which is a vector with 4,096 dimensions as the motion feature. We combine it with the output of ResNet for the static image, resulting a vector with 22,528 dimensions as the extracted motion features.

![Motion filters](image)

Fig. 4. Motion filters in the first layer of motion CNN obtained by the competitive learning. Three colors represents temporally consecutive three frames: the first, second and third frames correspond to red, blue and green, respectively. (a) No thresholding ($\theta = 0.0$) results in almost all filters being static. (b) Appropriate threshold ($\theta = 1.2$) produce only motion filters.

D. Micro Neural Networks structure

In our study, microNN is a binary classifier that outputs two values depending on the presence or absence of the concept. Each microNN consists of the input, hidden and output layers, which are fully-connected and contain $3 \times 3 \times 2048$, 32 and 2 nodes in kobe_nict_siegen_D_M_1, 2048, 32 and 2 nodes in kobe_nict_siegen_D_M_3, 22528, 32, and 2 nodes in kobe_nict_siegen_D_M_2 respectively. A microNN is constructed for each concept based on features extracted by ResNet.

This small-scale structure allows the microNN to be efficiently trained. During learning, we apply Dropout [22] to improve learning by ignoring randomly selected nodes. Dropout can avoid overfitting by reducing the degrees of freedom, thus raising the generalization performance.

E. Learning MicroNNs

Under the settings described above, we gradually perform transfer learning for each concept using the following two steps:

(i) We train a microNN using images from the ImageNet dataset.

(ii) We refine the microNN using videos from the TRECVID dataset from the weight parameters learned in the first step as initial values.

If the annotation of the concept is available only in the image dataset (ImageNet) or the video dataset (TRECVID), the microNN is trained using only that dataset. In general, CNN learning is strongly affected by the initial values. Especially in the case of insufficient training data, it is important to obtain suitable initial values to prevent overfitting. Therefore, compared to training a microNN from scratch, we often obtain better results using parameters that have been optimized on images as initial values. In addition, only a few minutes are required to learn a microNN for a concept.

F. Shot Retrieval based on Selected Concepts

We assume that the concepts related to a given query are selected based on Fig. 1. In order to balance the output values produced by microNNs for different concepts, we normalize the output values for each concept so that the maximum and minimum are 1 and -1 respectively. Through the last year’s experiments, we found that combining the output values of microNNs for different concepts by summation has better performance than multiplication. That’s because summation is
more tolerant to errors in concept detection. So, for each shot, we calculate the sum of the output values of the microNNs for the selected concepts to use as the overall score representing the appropriateness of the shot for the query.

### III. Evaluation Experiment Results

Fig. 6 shows retrieval results for kobe_nict_siegen_D_M_1, kobe_nict_siegen_D_M_2 and kobe_nict_siegen_D_M_3. Each row shows the results for query 532 by displaying key frames of shots ranked in first, second, third, 50th, 100th, 500th and 1000th positions. Each key frame is selected as the middle frame in a shot.

![Fig. 6. An illustration of retrieved results of kobe_nict_siegen_D_M_1, kobe_nict_siegen_D_M_2 and kobe_nict_siegen_D_M_3. Each row shows the result for query 532 by displaying a keyframe of shots ranked at the first, second, third, 50th, 100th, 500th and 1000th positions. Here, each keyframe is selected as the middle frame in a shot.](image)

We demonstrate the efficiency of microNNs for the AVS task using IACC.3. Fig. 7 shows a performance comparison between kobe_nict_siegen_D_M_1, kobe_nict_siegen_D_M_2 and kobe_nict_siegen_D_M_3 on each of the 30 queries. The line in blue, orange and gray represent Average Precisions (APs) of kobe_nict_siegen_D_M_1, kobe_nict_siegen_D_M_2 and kobe_nict_siegen_D_M_3 respectively. This figure indicates that, for most of the queries, using features extracted by ResNet with partly averaged pooling layer leads to a higher APs than those with fully averaged pooling layer.

![Fig. 7. Performance comparison among our methods.](image)

Fig. 8 presents a comparison between our methods and other methods developed for the manually-assisted category in AVS task. The orange bars indicate our three submitted results.

![Fig. 8. Performance comparison between our and other methods developed for the manually-assisted category in AVS task. The orange bars indicate our three submitted results.](image)

Fig. 9 shows a comparison between our method and all the other methods developed in AVS task. The orange bars indicate the MAPs of our three submitted results.

![Fig. 9. Performance comparison between our method and all the other methods developed in AVS task. The orange bars indicate the MAPs of our three submitted results.](image)

### IV. Conclusion and Future Work

In this paper, we have introduced our AVS methods that use microNNs to detect concepts related to a query. A microNN has a simple small-scale structure compared to ResNet, so it can be efficiently trained for a large number of concepts required for the AVS task. Based on transfer learning, a microNN can be trained efficiently. In addition, it can be incrementally refined using different training data. For example, the microNNs in this paper are first trained using the ImageNet dataset, and are then refined using the TRECVID video dataset. Although we manually extract concepts from each query, automatic selection is expected to be not difficult because the rule used in manual selection is very simple. Our current concept detection method works as an object recognizer to classify an object located in the center of an input image. We aim to extend this to a scene recognizer by considering correspondences of object and scene across the entire image. This will allow us to acquire more accurate understanding of an image by combining the object and scene recognizers. In addition, concepts selected from queries are mostly nouns, there is plenty of room to improve the treatment of verbs. We hope to find an appropriate database and a more effective method to process verbs to further improve the accuracy.

### References


