CONVOLUTIONAL NEURAL NETWORKS AS FEATURE GENERATORS FOR NEAR-DUPLICATE VIDEO DETECTION

L. Nathan Perkins

Boston University
Department of Electrical and Computer Engineering
8 Saint Mary's Street
Boston, MA 02215
www.bu.edu/ece

December 7, 2015

Technical Report No. ECE-2015-05

Summary

With ever increasing online repositories of video content, accurate and fast near-duplicate detection is important; it can help deter piracy and distribution of illegal content, while improving user experience when searching existing video collections by de-duplicating results and providing contextual data, such as links to similar videos or helping to extract meta data by clustering content. Existing methods are a trade-off between frame-level accuracy, using complex spatial annotations as differences of gaussians, and computational efficiency, using video-level annotation of color and motion. With the increasing power and feasibility of deep neural networks, they provide a potential middle-ground, by providing a per-frame signature in terms of the dominant classification while still being relatively performant.

In this implementation, pre-trained networks from the Caffe project, including AlexNet, GoogleNet and the R-CNN networks are used to construct a searchable database of video signatures, that can be queried by looking at the intersection of dominant features across frames. Using an evaluation dataset of approximately 250 videos, the R-CNN network is shown to produce a set of features that is resilient to common distortions, such as small rotations, cropping, re-encoding and changes to color and brightness. ROC curves show the network has a high degree of accuracy at matching known videos and distorted videos, while rejecting novel content.

These results suggest that neural network-based near-duplicate detection is both feasible and accurate. Yet the current implementation is constrained by inherent limitations in the pre-trained networks, which were trained on a limited number of labels and are sensitive to strong color or rotational changes. Future work will look at training a novel neural network, as well as adapting the feature database to do similar-video search and meta-data extraction.

Contents

List of Figures

List of Tables

1 Introduction

Both widespread adoption of modern smartphones, which are capable of video capture and editing, and increasingly affordable bandwidth and storage have contributed to an explosion in the quantity and variety of videos that are posted online. And companies like Google and Facebook are at the center of this video proliferation, acting as repositories for organizing, tagging and searching through billions of hours of video content. Every second, over five hours of new video are uploaded to YouTube. A fundamental challenge in managing such repositories is the ability to compare video similarity. Identifying similar videos helps reduce duplicate content, group related videos, identify copyright violations and propagate meta data across content. To be most useful, such similarity metrics must be robust to video encoding artifacts, editing effects (changes in saturation, lighting, cropping) and minor visual differences (such as network logos associated with different station affiliates on live television). Depending on the exact application, it may also be beneficial to have an algorithm capable of identifying similar video sequences captured from different vantage points, enabling stitching together video of breaking events. This area of work and algorithm development is known as near-duplicate video detection or retrieval.

2 Literature review

Algorithms for near-duplicate detection and retrieval generally followed two broad approaches. In those cases where speed is of primary importance, algorithms often rely on global features such as motion, intensity and color histograms. These global feature vectors can be calculated for segments or full video clips, and similarity can be quantified by looking at correlations either on a moving average basis or a global basis [13, 4]. These global calculations are extremely efficient and are robust to color distortion and coding differences, but are not well suited to identifying similarities in a short segment of a longer video (for example, a remix video taking scenes from many movies) and are unable to identify looser measures of similarity, such as footage from multiple angles of the same event.

Alternatively, in cases where precision and resilience to a broad variety of distortions is necessary, slower algorithms have been implemented by adapting techniques from image analysis [9, 6], where key points are identified in an image through a difference of Gaussians (DoG) and those that persist at multiple scales of analysis are used as a feature vector for describing the image. These key points can then be compared across images, and are robust to scaling, shearing and rotating. Many variations on this approach have been used, considering various feature vectors, such as SIFT [1], various methods of assessing feature correspondence, such as entropy measures in feature alignment [14, 16, 12], and various techniques for indexing, storing and searching over large datasets of feature vectors [1].

These two approaches make tradeoffs across the computational efficiency and the robustness of the algorithm to identify similarity across variations in the video, rang-

ing from straight forward changes in coding, coloration and saturation to much more complex changes such as rotation, distortion and manipulation (derivative work).

3 Problem statement

An active and promising area of research in machine learning and especially machine vision has been the application of deep neural networks. Over the last several years, deep neural networks have both achieved the most substantial gains and the best overall scores in the ImageNet Large Scale Visual Recognition Challenge [10]. Specifically, these high performing deep neural networks use a structure known as a convolutional neural network where feedforward connections are tiled to convey information about overlapping regions of the image, similar to the convolution operation, which provides a flexible and scale-invariant framework for classifying images [7, 2]. Although the training of these neural networks is immensely complex and often relies on GPU parallelization, the evaluation of the trained neural network is very efficient and often consists of a series of matrix multiplications and nonlinear scaling operations. The output of these convolutional neural networks can potentially provide a rich and complex description of a frame.

The goal of this project is to adapt image classification neural networks to the problem of near-duplicate detection, by annotating a sequence of frames according to dominant features. Such features act as a summary of frame content in a way that should be robust to cropping, scaling, coloration and noise, while not requiring the local annotation, processing and comparison inherent in key point methods. Existing research has used neural network features to a similar end, to create a descriptive binary hash that aids in fast retrieval of images [8]. The hope is that this approach will provide a powerful middle ground between existing algorithms, able to identify a greater range of video distortions without the cost inherent in existing algorithms that use local, spatial features.

To implement this algorithm, I will rely on already trained neural networks (given the computational and data demands of performing the training) [5]. For the project, I will build, to the extent possible, a broad corpus of video, including many artificial distortions (including changing saturation, rotating video, cropping frames, adding noise, recompressing and resizing) of original videos. Videos will be annotated according to output vectors from the neural networks. In addition to considering the final output (which tends to have softmax outputs to highlight a dominant category), I will evaluate using intermediate values of the neural network as potential feature vectors. Those frames that have substantial discontinuities in the feature vector when compared with the prior frame will be denoted as keyframes. In order to query the corpus, a query video will be similarly processed and the database will be searched for a similar progression of keyframe feature vectors.

This approach will be evaluated for performance (ability to use artificially distorted videos to find the original), which is consistent with techniques used by others to evaluate near-duplicate detection [15]. In addition, a few different trained neural

networks will be used as sources of feature vectors, to understand the impact the underlying network has on the performance of the near-duplicate detection. Finally, I will collect performance data both on the process of building the corpus (extracting feature vectors) and on the process of querying the corpus (through profiling specific operations in time on a single machine), although my focus will be more on the processing of the video rather than the efficiency of storing and querying feature vectors (which has its own host of computational challenges). These metrics will help evaluate whether deep neural networks provide meaningful feature vectors that can be used in the classification and evaluation of near-duplicate video content.

4 Implementation

4.1 Deep neural networks

Initial work focused on simply compiling and linking the Caffe framework [5], which has a complex set of dependencies including BLAS for linear algebra, OpenCV for image processing and NVIDIA CUDA for GPU calculations. Caffe provides a flexible framework, including Matlab and Python interfaces, to train and process deep neural networks. In addition, Caffe provides a "model zoo" with pre-trained models, meant to reduce barriers to usage and avoid energy and time intensive network training. Using the Matlab interface and pre-trained models, I was able to successfully use four networks to perform image classification using novel images. Initial work focused on four of the more commonly used networks that they distribute: CaffeNet, AlexNet [7], R-CNN [3] and GoogLeNet [11]. The networks were trained using a research data set that is part of the ImageNet Large Scale Visual Recognition Challenge [10], which includes over 10 million images annotated with over 10,000 labels. Because of similarities between CaffeNet and AlexNet in both architecture and training, CaffeNet was excluded from subsequent work.

Each network is a variation on the deep, convolutional architecture, with inputs representing a color image and outputs representing potential labels for the image content. Although each network varies in the specifics of its architecture, all tend to use 15 or more "layers," where each layer has a set of nodes that take weighted inputs from all or a subset of nodes in the previous layer, and all tend to follow a common structure as shown in Figure 1. The term convolutional refers to a repetitive, tiling organization to projections usually found in the initial layers that mimics the convolution operator and helps achieve the scale- and rotation-insensitivity of the networks. After the convolutional layers, the networks tend to have pooling layers which perform non-linear downsampling that provides a level of translational invariance and decreases the complexity of the learning by downsampling the feature space [7]. Following the pooling layers, the networks will have one or more fully interconnected layers, where all nodes are connected to all nodes in the previous layer; the fully interconnected layers associate down-sampled convolutional features with labels in the training data. Finally, the networks have a softmax layer at the end, which

normalizes network outputs in a way that generally constrains the output to a single, dominant label or classification. To give a sense of scale, AlexNet has 650,000 neurons and over 60 million parameters. Training of the network is a slow process (often many days) and relies heavily on GPUs to perform parallel calculations to allow for regularized gradient descent.

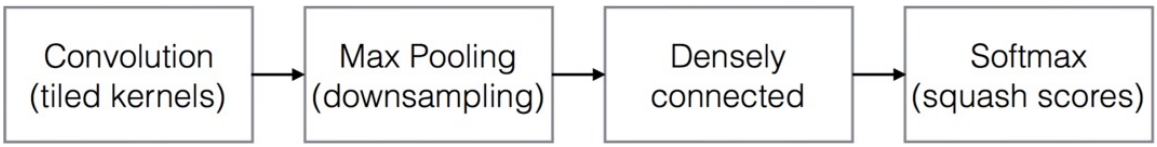


Figure 1: Although the specific strucutre and implementation details can vary widely, most convolutional neural networks for image recognition follow similar broad strokes. The initial layer acts as a convolution, tiling the image and learning kernels. Subsequent max pooling layers perform nonlinear downsampling. One or more densely or fully connected layers learn associations between features and labels. Finally, a softmax layer servers to squash the outputs and have one dominant label.

The networks accept square images of size 224×224 pixels in RGB colorspace after mean subtraction (where the mean is a square image representing the mean color across all images in the ILVRC training set; in actuality, this is close to a uniform gray image). In order to aid both the training process and the performance on the challenge, training images are augmented by cropping and/or flipping the original images ten times. Data augmentation is achieved by extracting windows at the top-left corner, the top-right corner, the bottom-left corner, the bottom-right corner and the center of the image, as well as horizontally flipped variants of each (see Figure 2). This augmentation enables a much larger corpus of training data by creating distinctive variations on the original image, and helps establish the robustness of the neural networks.

In order to validate that the networks are performing correctly, I tested a handful of images that were outside of the training set. Four example images are shown in Figure 3. Each image was fed into the AlexNet network, which produced scores corresponding to 1,000 labels from the training process. Table 1 shows the top five labels and scores as identified by the network. The network performs well at identifying familiar images, such as the elephant, guitar and lamp. But when the network is presented with an image that does not match any of the known labels, such as the crown, the results are less meaningful.

Performance-wise, initial tests showed a processing time of about 96ms per image when using the CPU.[1] Such performance is prohibitive for processing a large corpus of videos. In order to improve performance, the neural network processing was adapted to use the GPU where the per frame time was lowered to 50ms. Part of this latency reflects communication and memory transfer between the CPU and GPU. In order to

[1]Evaluations were performed on a 2013 MacBook Pro with a 2.6 GHz Intel Core i7 CPU and an NVIDIA GeForce GT 750M 2048 MB GPU.

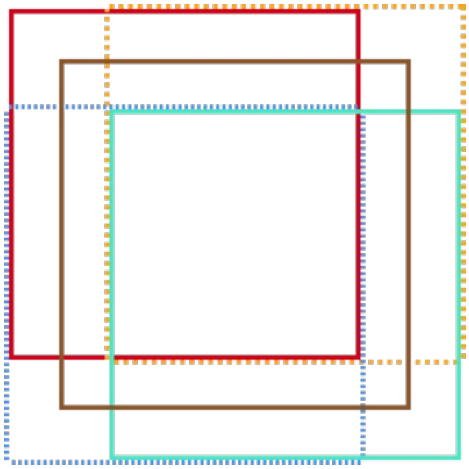


Figure 2: Training images are augmented by resizing to 256×256 pixels, then extracting five regions of 224×224 pixels, including the four corners and center. Horizontally flipped variants of each window are also used.

further increase performance, Caffe offers the ability to arbitrarily parallelize networks through a "resize" operation (essentially duplicating the underlying network) to allow simultaneous processing of images. Based on this, it is possible to process many frames in a batch, which reduces the per frame time to 9ms (using batches of 30 frames). Such performance gains enable building a nontrivial corpus of videos for testing and evaluation purposes.

4.2 Constructing video databases

Equipped with the three deep neural networks (AlexNet, GoogleNet and R-CNN) and an optimized, batch-oriented process for extracting features from images, the next step was to construct a database of videos and features. Approximately 250 short videos (each with a duration less than five minutes) were collected from YouTube to serve as both a training and testing dataset. In order to maximize the effectiveness of the selected neural networks, I opted to use videos that resemble the training set. To do this, videos were found by searching YouTube for one or two labels found in the training database (such as "elephant trumpet"), as such videos will likely contain imagery that resembles the training set. Identified videos are clipped to a maximum length of one minute. Each video was processed frame by frame. Frames were resized and mean-subtracted, then processed by the neural network to produce a corresponding feature vector for the frame. These initial datasets of features were 1-2GB total and provided a dense feature representation of the frame-by-frame progression of the videos.

In order to reduce the feature set to a more salient subset that can be quickly searched, frames with feature vectors that were substantially similar to the prior feature vector were discarded. Frames are considered similar if they have the same h

(a) Elephant

(b) Guitar

(c) Lamp

(d) Crown

Figure 3: Example images used to test the neural networks classification. The crown image was specifically selected as there is no corresponding label in the training set.

highest scoring labels in the same order. This created a sparser sampling of features at keyframes; the threshold for discarding features was selected to have a maximum of approximately two features per second ($h = 3$). This process was repeated for all three networks described above, as well as for modified versions of the networks after removing the final softmax layer that produces a dominant label. These six databases (one for each network, AlexNet, GoogleNet and R-CNN, and for modified versions of each network excluding the final softmax layer), each containing a few hundred videos, provided a corpus that allowed for evaluating relative scores of test videos and identifying the precision of matches of the near-duplicate detection algorithm.

4.3 Querying the database

Having constructed the corpus of videos, the next step is to provide functionality to input an unknown video and find, if it exists, the closest match in the database and a score representing the similarity between the videos. All input videos will undergo

Elephant		Guitar	
Label	Score	Label	Score
African elephant	0.707	acoustic guitar	0.983
tusker	0.192	electric guitar	0.011
Indian elephant	0.101	pick	0.003
water buffalo	0.0	cello	0.001
warthog	0.0	violin	0.001
Lamp		**Crown**	
Label	Score	Label	Score
lampshade	0.532	sax	0.729
table lamp	0.468	perfume	0.092
water bottle	0.0	candle	0.020
saltshaker	0.0	throne	0.019
pop-bottle	0.0	cornet	0.016

Table 1: Classification results from AlexNet using four example input images. The crown image was specifically selected as there is no corresponding label in the training set.

the same process of extracting features, by first passing each frame through the deep neural networks and then downsampling the feature vectors to a sparse representation based on keyframes. This sparse representation is then equivalent to that stored in the databases described above.

4.3.1 Euclidian-distance-based querying

Three distinct strategies were considered for querying the database. The first strategy used the Euclidian distance between keyframe features in the query video and all keyframe features stored in the database, where each feature is simply a vector representation of the output of the network – that is, the scores associated with each label. To perform the query, the system uses the feature vector corresponding with the first keyframe in the query video to finds the ten closest, in terms of Euclidian distance, keyframe features in the database. These ten closest feature vectors provide initial matches, as a close correspondence suggests that the frames have similar characteristics to the first frame of the query video. For each of these initial matches, the program compares the feature vectors associated with subsequent keyframes in both the query video and the potential matching videos. That is, for each subsequent keyframe in the query video, the system finds the closest keyframe in each of the potential matching videos, again by Euclidian distance between feature vectors. Because each keyframe is matched separately, they may not follow the same temporal progression; as a result, a penalty is applied if the matched keyframes are not temporally sequential.

Each potential matching video is scored according to the distance between features describing the query video and features describing the video in the database:

$$S_k = \frac{1}{r}\sum_{i=1}^{r} \min_{l} a(l,i) \sqrt{\sum_{j=1}^{n} (f_{k,j}(l) - f_{q,j}(i))^2},$$

where r is the number of frames in the query video, n is the number of features returned by the neural network, $f_{q,j}(i)$ is feature j for the query video at frame i, $f_{k,j}(l)$ is the feature j for the database video k at frame l, and $a(l,i)$ is a scaling factor that penalizes the score if the order of the keyframes matches are not temporally sequential. If matched keyframe l comes after the previously matched keyframe, then $a(l,i) = 1$, otherwise $a(l,i) = \alpha > 1$. This calculation will result in smaller scores for query and matching videos that have keyframes where the Euclidian distance between the feature vectors is low, and where the matching keyframes follow a similar sequential progression. In this case, the most likely match is that with the lowest score.

4.3.2 Order-based querying

Given some disadvantages to the above approach, which is computationally expensive (requiring many Euclidian distance calculations) and has some key limitations in its assumptions (specifically, it assumes that the full query video will appear within the matched video in the database and fails to do partial matches), a second strategy was developed for querying the database. Instead of using the full feature vector, this method relies on the order of the labels returned by the neural network. Much like the earlier tables showing the top labels for sample images (Table 1), the order-based query method turns the network output into a feature vector of dominant labels preserving the top three labels in order. Each keyframe has a feature vector of top labels, such as $[l_1, l_2, l_3]$ where l_1 is the 1^{st} most likely label based on the output of the neural network. This further reduces the complexity of the feature space, representing each video frame as a small selection of labels.

To use the order-based query method, the query video is similarly converted to a series of unique label orderings. The database is searched by looking for a frame with the same three labels, in the same order, as the first frame of the query video. If no match is found, the first frame of the query video is discarded and the process restarts, allowing for matching a segment of the query video. The query process will continue to match frames as long as the unique label order of subsequent frames continues to match, allowing some partial matches, where only one or two of the labels match. The number of partial matches is tunable to change the strictness of the results returned. A score is generated by counting the number of matching labels per frame and normalizing by the number of frames in the query video.

4.3.3 Intersection-based querying

Initial tests of the two methods above suggest strong performance for querying using the original videos, but performance dropped quickly with small distortions to the video. Small video distortions would create divergences in the ordering of labels or noise, especially in frames that were unlike the training set (text, animations, cross-fades, etc). A third method was developed, seeking to make the querying process more resilient to noise. Like the order-based query method, the query process looks for a frame with the same dominant labels in the same order as the first frame in the query video. For each match, the query looks at the overlap between subsequent labels in the query video and the matched video in the database. Matches are scored based on the relative overlap of the labels and based on how many of the labels appear in the same order. Scoring is a weighted mix of these two measures:

$$S_{\vec{m}} = \alpha \frac{|\vec{q} \cap \vec{m}|}{|\vec{q} \cup \vec{m}|} + (1 - \alpha) \sum_{i=1}^{min(|\vec{q}|,|\vec{m}|)} \delta(\vec{q}[i] - \vec{m}[i]),$$

where $\vec{q}$ is the vector of dominant labels for the query video (i.e., 1st label of 1st keyframe, 2nd label of 1st keyframe, 3rd label of 1st keyframe, 1st label of 2nd keyframe, ...), $\vec{m}$ is the vector of dominant labels for the matching video, α is a tuning parameter that prioritizes either the similarity or the order of labels that occur in each video and $|\cdot|$ is the number of elements in a set. As a result, $|\vec{q} \cap \vec{m}|$ represents the number of labels common to both the query and the matching video, while $|\vec{q} \cup \vec{m}|$ represents the total number of labels in both videos.

This intersection-based query method proved to be both extremely computationally efficient and resilient to noise and distortions in the video, and hence was used for all subsequent analysis.

4.4 Evaluation of algorithm

In order to evaluate the viability of both using deep neural network features as descriptors and the querying techniques, distorted versions of videos were generated using a range of distortion schemes, including resizing, recompression, cropping, rotating, brightness/saturation modifications and changes to the coloration. Examples of each distortion are shown in the appendix in Figure 6. Two evaluation schemes will be used. For each type of distortion, the resilience of the algorithm will be measured to see both whether it is able to match the distorted video and to see how the score decreases in response to further distortion. This will assess the robustness of the algorithm to intentional or unintentional video distortions and help assess specific strengths and weakness of the algorithm.

As a more comprehensive evaluation, the algorithm will be applied to a range of minimally distorted videos (akin to distortions seen in piracy, such as cropping, changes to color and brightness and recompression), including distorted versions from

the original corpus as well as distorted versions of novel videos. The results from this process provide an indication about how the threshold used to determine a match affects the number of false positives (incorrectly matched videos) and false negatives (videos that fail to match). This can be used to construct a general ROC curve that will describe the overall recall and precision of the neural network-based near-duplicate detection algorithm.

5 Experimental results

5.1 Resiliency to distortions

To better understand the robustness of using neural network features to perform near-duplicate detection, distorted versions of the videos used to construct the database were used as query inputs. The near-duplicate detection algorithm was assessed based on the ability to find the correct match and the score returned for the correct match. Figure 4 shows the results under various distortions.

As an initial verification of the premise, the near-duplicate detection was assessed after re-encoding videos. The algorithm was always able to match the videos and did so with a relatively high score, suggesting that the method is functioning as expected.

The results also show resilience of the algorithm to resizing, even when resizing a video to $\frac{1}{4}$ of its original size. Similarly, encouraging results appear when cropping videos horizontally (only horizontal cropping was used given the aspect ratio of the videos in the training set). After cropping up to 45% of the pixels from the video, the algorithm is able to match videos and returns a relatively high score.

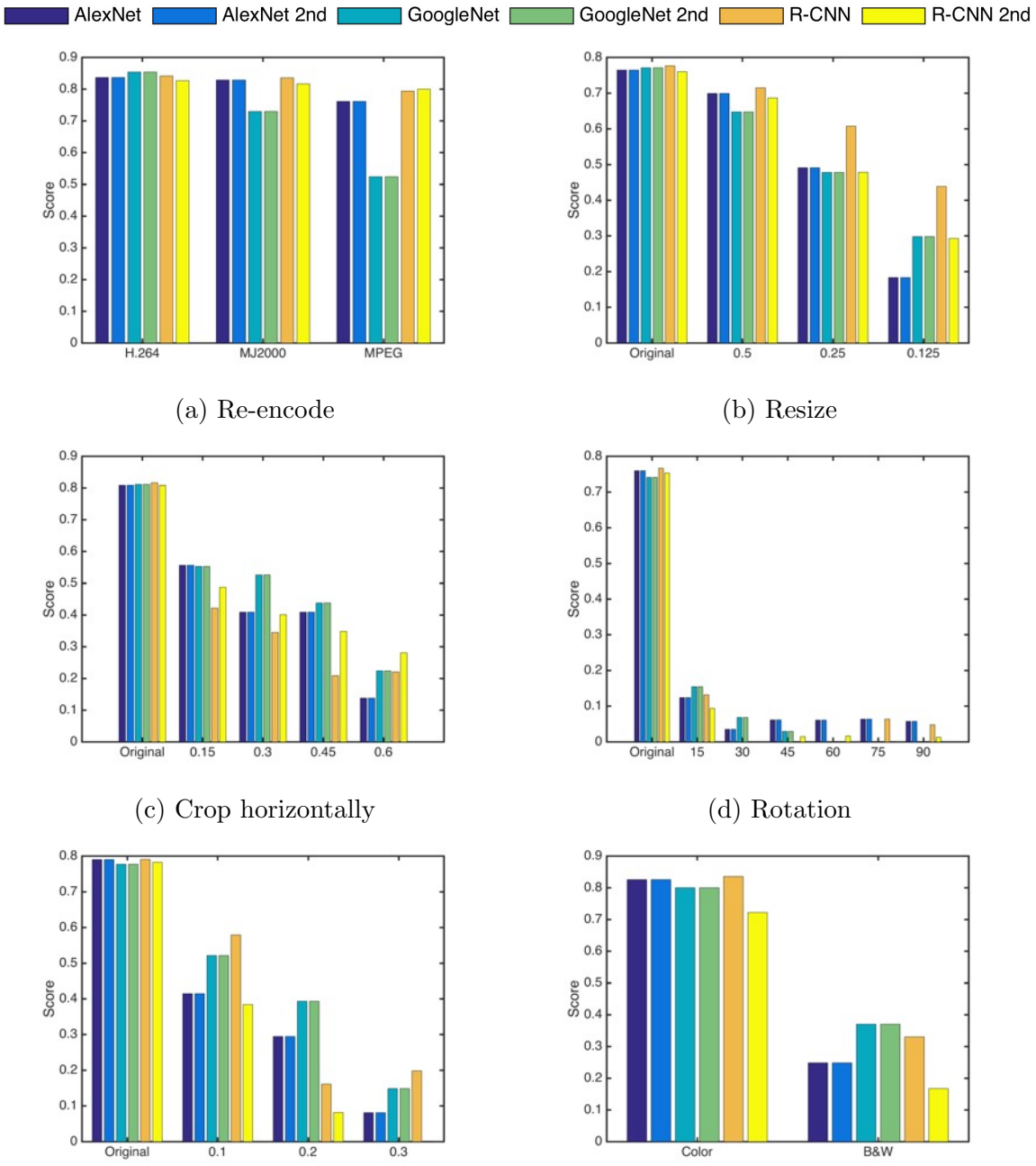
The algorithm is less performant after rotating videos. Small rotations, up to 15°, still tend to match the original video but have a low score. Larger rotations cause the algorithm to struggle to match the video.

Finally, it is interesting to observe that the algorithms are sensitive to the color and contrast seen in the videos. Color distortions (achieved by stretching the histogram for each color channel) reduced the accuracy of the matches by the algorithm, while converting videos to black and white substantially reduced the accuracy and score of the matching process. This suggests that the neural networks rely on color channel information to perform classification.

Broadly, these results show that the neural network-based near-duplicate detection algorithm performs well under a number of distortions, such as resizing, re-encoding, cropping and even small color or rotation changes. More extreme changes, such as substantial rotation or color filters, impair the method's ability to recognize video clips.

5.2 Accuracy

The charts and figures shown in the previous section show the resiliency of the network-based near-duplicate detection in terms of the scores returned, but do not



(a) Re-encode

(b) Resize

(c) Crop horizontally

(d) Rotation

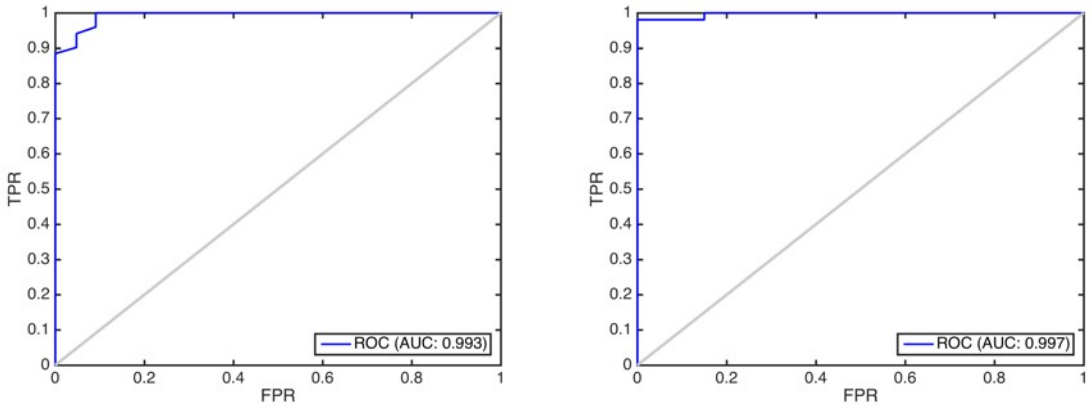
(e) Increase saturation & contrast.

(f) Conversion to B&W.

Figure 4: Results show the scores returned by different neural networks using the intersection-based query method when querying using distorted versions of training videos.

provide a comprehensive measure of the accuracy of the method. In order to get a sense of accuracy, Figure 5 depicts the receiver operating characteristics (ROC) curve reflecting the accuracy of results when querying the database. The curve is generated

by using a query set of known videos, known but mildly distorted videos (re-encoded, minimally cropped, minimally recolored) and novel videos. The algorithm is considered accurate if it matches the known videos and the known but mildly distorted videos, while not incorrectly matching the novel videos. The curve is generated by varying the threshold used for determining a match.



(a) Last layer (b) Second-to-last layer

Figure 5: ROC curves showing accuracy of matching using either the second to last or last layer of the R-CNN network and the intersectio-based query method.

The R-CNN network proved to be most effective for near-duplicate detection, likely due to its somewhat more flexible and resilient design. The network is designed to process extracted regions from larger images and is hence more resilient to stretching and other distortions [3]. ROC curves for the GoogleNet and AlexNet networks are shown in Figure 7 in the appendix.

The ROC curves for the R-CNN network indicate a high-accuracy rate; it is able to match both known and distorted videos without incorrectly matching novel videos. The R-CNN network has a slightly different architecture, forgoing the normal softmax final layer. The second to last layer instead encodes features based on a larger set of classifiers (over 4,000). This different architecture and larger number of layers likely accounts for the improved accuracy, especially when using the second-to-last layer as the source of our feature sets.

5.3 Performance

Although this project did not heavily focus on optimizing the code, and there are surely substantial performance gains possible in implementing a better optimized or hash-based query method, the computational performance of the method is an important factor to assess. The hope is that the network-based near-duplicate detection will provide a tradeoff between the computational complexity of frame-level complex feature annotation and the efficiency of video-level summaries such as histograms.

Despite the complexity in training the neural networks, applying the networks to novel frames, especially when leveraging the GPU and the batching operations described in the implementation section, is fast, with a per-frame time of less than 9ms. Even with this low per-frame time, this is still the slowest part of querying the database, as the query video must be similarly processed to extract features. Given a one minute query video, this means over 16 seconds of processing to identify features.

Once the feature vector has been extracted, querying the database is a fast process that benefits from standard computational tools for indexing and querying data, as the query process involves looking for a specific set of labels. Given the small data set and a naive implementation of this search process, querying took 7-43ms, depending on the number of potential matches.

Overall, these performance characteristics seem promising for a real-world implementation. Videos need only be processed for features once, and further optimizations to the query process, such as using an indexed data structure, would allow scaling to much larger datasets.

6 Conclusions

The results from the proposed neural network-based near-duplicate detection algorithm suggest that it is both a feasible and efficient mechanism for performing near-duplicate detection. After the somewhat computationally expensive process of feature extraction, videos can be described and stored as a progression of dominant labels. This progression of labels is readily queried, appears robust to distortions, including re-encoding, resizing, cropping, small rotations and minor changes to coloration, and is accurate over the limited data set tested for this project, both in terms of identifying known and distorted videos as well as rejecting novel videos.

The best results were achieved using the second-to-last layer of the R-CNN neural network, which produces a feature vector of 4,096 features per frame. This feature-space can be downsized by discarding those frames that are not substantially different from the prior frame, producing a sparse representation of the content of the videos.

Despite the initial success shown, more work is needed. The analysis here focused on videos that were tangentially related to the training sets used for building the networks. A more robust solution would look to build a new deep neural network specifically suited to the problem-space using a broad sampling of video frames for training. Such a network would be better able to encode video-specific visual features, such as text, effects (such as cross-fades) and specific visual styles (such as animation).

Although this project focused on matching videos and distorted versions of videos, the data set built for this project can be adapted to a variety of other problems, such as genre detection or similar-video clustering. Future work will involve consolidating features across a video into a single set of dominant features that describe the video as a whole. Such features can be used to find similar content or automatically annotate meta-data, such as genre.

Broadly, this project suggests that the success of deep neural networks in the

space of image search can be readily adapted to video labeling and search. This work provides proof of principle by building an accurate and robust-to-distortion near-duplicate detection algorithm using features extracted from videos using pre-trained neural networks designed for image classification.

References

[1] O. Chum, J. Philbin, and A. Zisserman, "Near Duplicate Image Detection: min-Hash and tf-idf Weighting," in *British Machine Vision Conference 2008*, pp. 50.1–50.10, British Machine Vision Association, July 2008.

[2] D. Erhan, C. Szegedy, and A. Toshev, "Scalable object detection using deep neural networks," *Computer Vision and ...*, 2014.

[3] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Computer Vision and Pattern Recognition*, 2014.

[4] A. Hampapur, K. Hyun, and R. M. Bolle, "Comparison of sequence matching techniques for video copy detection," *Electronic Imaging 2002*, vol. 4676, pp. 194–201, Dec. 2001.

[5] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional Architecture for Fast Feature Embedding," *arXiv preprint arXiv:1408.5093*, 2014.

[6] Y. Ke, R. Sukthankar, and L. Huston, "Efficient near-duplicate detection and sub-image retrieval," *ACM Multimedia*, 2004.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," pp. 1097–1105, 2012.

[8] K. Lin, H. F. Yang, J. H. Hsiao, and C. S. Chen, "Deep Learning of Binary Hash Codes for Fast Image Retrieval," in *Proceedings of the IEEE ...*, 2015.

[9] D. G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[10] O. Russakovsky, "ImageNet Large Scale Visual Recognition Challenge," *International journal of computer vision*, pp. 1–42, Apr. 2015.

[11] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, and S. Reed, "Going deeper with convolutions," *arXiv.org*, 2014.

[12] H.-K. Tan, C.-W. Ngo, R. Hong, and T.-S. Chua, *Scalable detection of partial near-duplicate videos by visual-temporal consistency*. New York, New York, USA: ACM, Oct. 2009.

[13] J. Yuan, L.-Y. Duan, Q. Tian, S. Ranganath, and C. Xu, "Fast and Robust Short Video Clip Search for Copy Detection," in *Advances in Multimedia Information Processing - PCM 2004*, pp. 479–488, Berlin, Heidelberg: Springer Berlin Heidelberg, Nov. 2004.

[14] W.-L. Zhao and C.-W. Ngo, "Scale-Rotation Invariant Pattern Entropy for Keypoint-Based Near-Duplicate Detection," *IEEE Transactions on Image Processing*, vol. 18, pp. 412–423, Dec. 2008.

[15] X. Zhou, X. Zhou, L. Chen, A. Bouguettaya, N. Xiao, and J. A. Taylor, "An Efficient Near-Duplicate Video Shot Detection Method Using Shot-Based Interest Points," *IEEE Transactions on Multimedia*, vol. 11, pp. 879–891, July 2009.

[16] J. Zhu, S. C. H. Hoi, M. R. Lyu, and S. Yan, "Near-duplicate keyframe retrieval by nonrigid image matching," in *Proceeding of the 16th ACM international conference*, (New York, New York, USA), pp. 41–50, ACM, Oct. 2008.

A Example distortions

(a) Original

(b) Color contrast adjustment

(c) Original

(d) Horizontally cropped

(e) Original

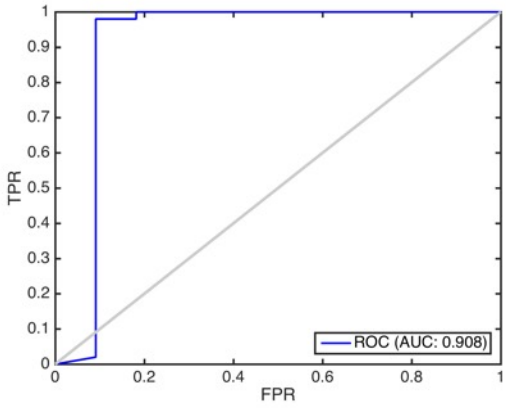
(f) Resized image

(g) Original

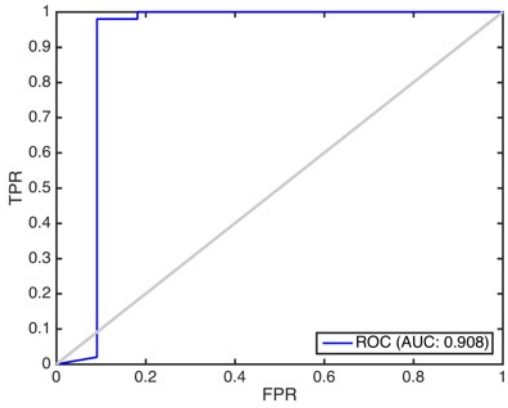
(h) Rotation

Figure 6: Representative distortions shown for frames. The original frame is on the left, the distorted frame is on the right. Two additional distortions were used: conversion to black and white and re-encoding using MPEG or Motion JPEG 2000.

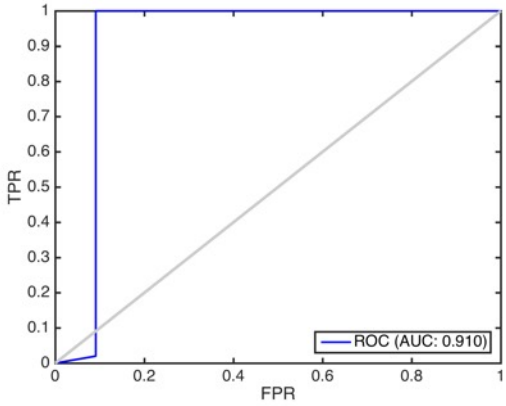
B ROC curves for other networks



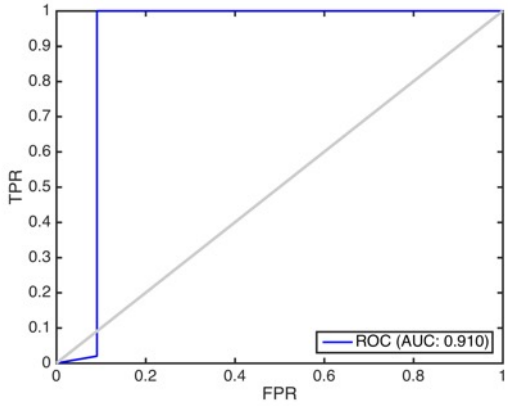
(a) Last layer of AlexNet



(b) Second-to-last layer of AlexNet



(c) Last layer of GoogleNet



(d) Second-to-last layer of GoogleNet

Figure 7: ROC curves showing accuracy of matching using either the second to last or last layer of the AlexNet and GoogleNet networks and the intersect-based query method.