Learning to Compress Images and Videos

Li Cheng Li.Cheng@nicta.com.au
S.V. N. Vishwanathan svn.vishwanathan@nicta.com.au
Statistical Machine Learning, National ICT Australia
Research School of Information Sciences & Engineering, Australian National University, Canberra ACT 0200

Abstract

We present an intuitive scheme for lossy color-image compression: Use the color information from a few representative pixels to learn a model which predicts color on the rest of the pixels. Now, storing the representative pixels and the image in grayscale suffice to recover the original image. A similar scheme is also applicable for compressing videos, where a single model can be used to predict color on many consecutive frames, leading to better compression. Existing algorithms for colorization – the process of adding color to a grayscale image or video sequence – are tedious, and require intensive human-intervention. We bypass these limitations by using a graph-based inductive semi-supervised learning module for colorization, and a simple active learning strategy to choose the representative pixels. Experiments on a wide variety of images and video sequences demonstrate the efficacy of our algorithm.

1. Introduction

The explosive growth of the Internet, as witnessed by the popularity of sites like YouTube and image google, has exponentially increased the amount of images and movies available for download. As more and more visual data is being exchanged, there is an ever-increasing demand for better compression techniques which will reduce network traffic. Typical compression algorithms for images work in the frequency domain, and use sophisticated techniques like wavelets. In the case of video clips, these algorithms not only compress each frame, but also use compression across frames in order to reduce storage requirements. For instance, frames within a scene are likely to be very similar, and hence it is sufficient to encode the differences between consecutive frames. Motion prediction, optical flow, and other tools are also used to further improve performance.

In this paper, we take a slightly different approach. Instead of performing a frequency transformation we store a grayscale version of the image and color labels of a few representative pixels. Using the stored information we learn a model which predicts the color for the rest of the pixels. Turning to video, essentially the same idea works, but now we only need to store color information sampled from a single frame and use the same model to predict on all closely related frames. Two key questions are: a) How does one learn the model? b) How to choose the representative pixels automatically? In what follows, we will answer both these questions systematically.

1.1. Paper Outline

In section 2 we will briefly discuss semi-supervised learning algorithms with particular emphasis on graph based methods. In section 3 we will describe the colorization algorithm of Levin et al. (2004), and formally show it is a transductive semi-supervised learning method. We then extend their formulation to an inductive setting by adapting a graph-Laplacian based manifold regularization algorithm due to Belkin et al. (2006). In section 4 we show that the task of choosing representative pixels can be automated by using a simple active learning approach. In section 6 we present experiments on image and video data to demonstrate the effectiveness of our algorithm. The paper concludes with a outlook and discussion.

2. Semi-Supervised Learning

Semi-supervised learning refers to the problem of learning from labeled and unlabeled data. It has

---

Appearing in *Proceedings of the 24^{th} International Conference on Machine Learning*, Corvallis, OR, 2007.

attracted considerable attention in recent years (see Zhu (2005) for a comprehensive survey). Of particular interest to us are graph based methods, examples of which include Smola & Kondor (2003), Belkin & Niyogi (2003), and Belkin et al. (2006). In this section we will briefly survey graph-based semi-supervised learning algorithms.

2.1. Notation

A graph G consists of an ordered and finite set of n vertices V denoted by $\{v_1, v_2, \ldots, v_n\}$, and a finite set of edges $E \subset V \times V$. A vertex v_i is said to be a neighbor of another vertex v_j if they are connected by an edge. G is said to be undirected if $(v_i, v_j) \in E \iff (v_j, v_i) \in E$ for all edges. The adjacency matrix of G is an $n \times n$ real matrix W with $W_{ij} = 1$ if $(v_i, v_j) \in E$, and 0 otherwise. If G is weighted then W can contain non-negative entries other than zeros and ones, *i.e.,* $W_{ij} \in (0, \infty)$ if $(v_i, v_j) \in E$ and zero otherwise. Let D, the degree matrix, be an $n \times n$ diagonal matrix with entries $D_{ii} = \sum_j W_{ij}$. The graph Laplacian is the matrix $L = D - W$ while the normalized graph Laplacian $\Delta := D^{-1/2} L D^{-1/2}$. In what follows, by default, we can either use the graph Laplacian or the normalized graph Laplacian interchangeably.

2.2. Goal of Semi-Supervised Learning

Let $\mathcal{X}$ be the space of observations, and $\mathcal{Y}$ the space of labels. We assume that $\mathcal{Y}$ is a finite subset of $\mathbb{R}$ and use $|\mathcal{Y}|$ to denote its size. The semi-supervised learning problem can be formally formulated as follows: Given a sequence $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^m$ of labeled examples drawn from $\mathcal{X} \times \mathcal{Y}$, $\{\boldsymbol{x}_i\}_{i=m+1}^n$ of unlabeled examples drawn from $\mathcal{X}$, and a loss function $l : \mathcal{X} \times \mathcal{Y} \times \mathcal{H} \to \mathbb{R}$, learn a function $f \in \mathcal{H}$ which minimizes the loss on the labeled examples and also generalizes well to unseen examples.

Our compression problem fits perfectly in this framework. In the case of images, given a set of color pixels (labeled examples) and a set of grayscale pixels (unlabeled examples) we want to learn a function which will predict color (labels) on the grayscale pixels. In the case of video, our function also needs to generalize well to predict on unseen (but closely related) frames.

Clearly, semi-supervised learning is meaningful only in situations where the true underlying distribution of examples, which the unlabeled data will help elucidate, is relevant for the classification problem. Therefore, certain smoothness assumptions, *e.g.* if two observations are close then their corresponding labels should be similar, are often made. In our application, these assumptions are natural: If two pixels have similar intensity values and are spatially close to each other then it is very likely that they have similar color values.

2.3. Graph Based Methods

Graph-based semi-supervised methods construct a *problem graph*, $\mathcal{G}$, whose nodes are the examples (both labeled and unlabelled), and edges encode nearest neighbor relationships. Often times, the edges are weighted by a kernel function to reflect the similarity between neighboring examples. Now the semi-supervised learning problem can be posed as that of estimating a smooth function that respects neighborhood relations on the graph. Following Smola & Kondor (2003), Belkin & Niyogi (2003), Belkin et al. (2006), and others, we minimize the following regularized risk:

$$J(f) = c\,||f||_{\mathcal{H}}^2 + \frac{\lambda}{n^2}\,||f||_{\mathcal{G}}^2 + \frac{1}{m}\sum_{i=1}^{m} l(\boldsymbol{x}_i, y_i, f). \quad (1)$$

Here $\mathcal{H}$ is a Reproducing Kernel Hilbert Space (RKHS) of functions $f : \mathcal{X} \to \mathbb{R}$. Its defining kernel is denoted by $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, which satisfies $\langle f, k(\boldsymbol{x}, \cdot)\rangle_{\mathcal{H}} = f(\boldsymbol{x})$ for all $f \in \mathcal{H}$. c and λ are trade-off parameters for the regularizers. The regularizer $||f||_{\mathcal{G}}^2$ is defined as

$$\|f\|_{\mathcal{G}}^2 = \boldsymbol{f}^\top \nabla_{\mathcal{G}} \boldsymbol{f}, \quad (2)$$

where $\boldsymbol{f}$ denotes the vector $[f(\boldsymbol{x}_i), \ldots, f(\boldsymbol{x}_m), \ldots f(\boldsymbol{x}_n)]$, and $\nabla_{\mathcal{G}} \in \mathbb{R}^{n \times n}$ is a function of $\mathcal{G}$ which determines the specific form of regularization imposed. Two choices are particularly relevant to us:

$$\|f\|_{\mathcal{G}}^2 = \boldsymbol{f}^\top \Delta \boldsymbol{f}, \quad (3)$$

and

$$\|f\|_{\mathcal{G}}^2 = \boldsymbol{f}^\top L^2 \boldsymbol{f} = ||L\boldsymbol{f}||^2. \quad (4)$$

Finally, we make the assumption that l only depends on f via its evaluations at $f(\boldsymbol{x}_i)$ and that l is piecewise differentiable. Specifically, when we use the square loss,

$$l(\boldsymbol{x}_i, y_i, f) = (f(\boldsymbol{x}_i) - y_i)^2, \quad (5)$$

we obtain the so-called Laplacian Regularized Least Square (LapRLS) algorithm, which generalizes the Regularized Least Squares algorithm (Belkin et al., 2006). As a consequence of the representer theorem (Schölkopf & Smola, 2002) there exist coefficients α_i such that f can be expressed as

$$f(\cdot) = \sum_{i=1}^{n} \alpha_i k(\boldsymbol{x}_i, \cdot). \quad (6)$$

Furthermore, there is a closed form solution

$$\boldsymbol{\alpha} = (I_m K + cmI + \frac{\lambda m}{n^2} \nabla_{\mathcal{G}} K)^{-1} \boldsymbol{y}, \qquad (7)$$

where $\boldsymbol{\alpha}$ denotes the vector $[\alpha_i, \ldots, \alpha_m, \ldots \alpha_n]$, $I_m \in \mathbb{R}^{n \times n}$ contains the identity matrix of size $m \times m$ on the top left hand corner and zeros elsewhere, I is the identity matrix, K is the Gram matrix $K_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$, and $\boldsymbol{y}$ denotes the vector $[y_i, \ldots, y_m, 0, \ldots, 0]$. We will use this formulation in all our experiments.

2.4. Transductive vs Inductive

In transductive learning one is given a (labeled) training set and an (unlabeled) test set. The idea of transduction is to perform predictions only for the given test points (Chapelle et al., 2006). This is in contrast to inductive learning, where the goal is to output a prediction function which is defined on the entire space (Chapelle et al., 2006). In our context, this is a key difference. While an inductive algorithm can easily be used to predict labels on closely related images, transductive algorithms are unsuitable. This limits the applicability of transductive algorithms to video compression.

All the algorithms we discussed above are inductive, but, it is easy to turn them into transductive algorithms. Let X denote the set of labeled and unlabelled points, then, work with functions $f : X \to \mathbb{R}$, drop the regularization term $||f||^2_{\mathcal{H}}$ from the objective function, (1), and minimize (Zhu, 2005; Belkin et al., 2006):

$$J(f) = \frac{\lambda}{n^2} \, ||f||^2_{\mathcal{G}} + \frac{1}{m} \sum_{i=1}^{m} l(\boldsymbol{x}_i, y_i, f). \qquad (8)$$

We note in the passing that if we drop the regularization term $||f||^2_{\mathcal{H}}$ from the objective function but continue to work with $f \in \mathcal{H}$, *i.e.,* $f : \mathcal{X} \to \mathbb{R}$, we get an inductive algorithm whose prediction function is required to be smooth only on the observed examples (both labeled and unlabelled). In applications (like ours) where f is only used to predict on examples that are very similar to the observed examples, this suffices.

3. Colorization by Semi-Supervised Learning

Colorization – the process of adding color to a grayscale image or video sequence – has attracted considerable research interest recently. Unfortunately, existing algorithms are tedious, and labor-intensive. For our purposes, a particularly relevant algorithm is due to Levin et al. (2004), which we now present using notation that makes it easy to see connections to semi-supervised learning.

Given a grayscale image with a few color patches, we enforce the constraint that two neighboring pixels should have similar colors if their intensities are similar. Formally, let $X = \{\boldsymbol{x}_i\}_{i=1}^n$ denote the set of all pixels in an image, and $\{\boldsymbol{x}_i, y_i\}_{i=1}^m$ denote the set of pixels for which color information, y_i, is available. We minimize the following objective function:

$$\sum_{i=1}^{n} \left(f(\boldsymbol{x}_i) - \sum_{i \sim j} \omega_{ij} f(\boldsymbol{x}_j) \right)^2 + \sum_{i=1}^{m} \delta(f(\boldsymbol{x}_i), y_i). \quad (9)$$

Here, $f : X \to \mathbb{R}$ is a function that assigns color values to pixels, $i \sim j$ implies that pixel $\boldsymbol{x}_j$ is a neighbor of pixel $\boldsymbol{x}_i$, and for each i the weights ω_{ij} are nonnegative and sum to one. The predictor f is forced to take on user-specified values on all pixels where color information is available, by the loss function $\delta(f(\boldsymbol{x}_i), y_i)$ which is 0 if $f(\boldsymbol{x}_i) = y_i$ and ∞ otherwise. The weights ω are computed using a normalized radial basis function or a second-order polynomial, and takes into account the similarities in intensity values.

To show that the above algorithm is a graph-based transductive semi-supervised learning algorithm, we begin by constructing a weighted adjacency matrix W such that $W_{ij} = \omega_{ij}$. Since $\sum_i \omega_{ij} = 1$, the degree matrix $D = I$, and the graph Laplacian can be written as $L = I - W$. It is now easy to verify that

$$\boldsymbol{f}^\top L^2 \boldsymbol{f} = \sum_{i=1}^{n} \left(f(\boldsymbol{x}_i) - \sum_{i \sim j} \omega_{ij} f(\boldsymbol{x}_j) \right)^2 ,$$

and hence, modulo some scaling factors, the objective function of Levin et al. (2004) is identical to (8).

It is worthwhile mentioning here that Levin et al. (2004) also use their algorithm to perform colorization on a video sequence. Now, the notion of a neighbor also takes into account temporal information, that is, two pixels are deemed neighbors if either they are close to each other on a single frame or if they appear at the same position on two consecutive frames. For our application, this approach suffers from several drawbacks. First, the size of the optimization problem grows with the number of related frames thus making it unsuitable for real-time compression. Second, the algorithm propagates color information from frame to frame. A better approach is to learn how to predict color on a single frame and reuse this model to predict on all closely related frames. Third, when streaming data on the Internet, one might need to compress on demand since all the frames might not be available apriori. Our algorithm, which we describe next, addresses all these issues.

3.1. Our Algorithm

We extend the formulation of Levin et al. (2004) in many ways. First, we work with an graph-based inductive algorithm LapRLS (see section 2.3). This has the advantage that we can learn a model for one frame of video and reuse the same model to predict on related frames, thereby addressing the shortcomings discussed above. Second, we use the square loss (5) instead of the δ loss. Besides being analytically tractable, our loss favors smoother functions whose color predictions might differ slightly from user-supplied values. Third, we use the normalized graph Laplacian, Δ, for regularization instead of the L^2 regularization favored by Levin et al. (2004). Recent studies (e.g. Zhang & Ando, 2005) have shown that using the normalized graph Laplacian is both theoretically and practically more appealing. Finally, we extract features out of each pixel and construct our nearest neighbor graph in feature space. Instead of just respecting spatial proximity, our features also respect local texture. We discuss implementation details in section 5.

4. Active Learning for Compression

Active learning is a framework that allows the learner to *ask* for informative examples. The goal is as usual to construct an accurate classifier, but the labels of the data points are initially hidden and there is a charge for each label you want revealed. The hope is that by intelligent adaptive querying, one can get away with significantly fewer labels than one would need in a regular supervised learning framework.

Recall that we want to colorize an image or a video by building a model which takes as input color information of few *representative* pixels. The key question is: How to choose these representative pixels? We solve this by casting our question as an transductive active-learning problem. Each pixel we query for a label (color information) increases our cost (the amount of storage needed to reconstruct the image). Therefore, it is advantageous to query for as few pixels as possible.

While sophisticated algorithms with guaranteed theoretical bounds exist for active learning, we use a simple strategy. The learner starts off with a few randomly chosen labeled pixels, and learns a model. The prediction of the model is then evaluated on the image and high error areas are identified and clustered. The algorithm then chooses a representative from each cluster and queries it for label information, adds it to its label set, and the process repeats.

5. Implementation Details

In this section we describe various “tricks of the trade” we employed to apply our algorithm to images and videos.

Following Levin et al. (2004), we work in YUV space where Y is the intensity (luminance) channel, U and V are the chrominance channels that encode the color. We also predict U, V values independently. As noted in section 3.1, our algorithm maps each pixel using a feature map. Our feature maps encode both the spatial location of the pixel in the image grid as well as the local texture information obtained by sampling a 5×5 local grid around each pixel. Similar to Levin et al. (2004), we also construct a 4-nearest neighbor graph in feature space. While only spatially adjacent pixels are connected in their graph, our graph also takes local texture into account when connecting pixels. For the kernel we choose the stock standard Gaussian kernel (Schölkopf & Smola, 2002) with the variance σ tuned separately for each problem.

A key issue to be addressed is computation of α's which involves inverting a dense $n \times n$ matrix, where n is the number of pixels in the image (see (7)). Advances in parallel computing and matrix algebra notwithstanding, it is still computationally challenging to invert a large dense matrix with tens of thousands of rows and columns. To reduce our computational burden we first observe that there is a lot of redundancy amongst spatially nearby pixels, since they tend to be spectrally homogeneous. Therefore, we can preprocess input images or frames to obtain an over-segmented representation, also called as a *super-pixel* representation by Ren & Malik (2003), and pick pixels randomly from these segments. Typically, after quantization, the number of segments ranges between 1000 - 5000 depending on the complexity of the input image.

We address the issue of measuring the quality of our solution. We employ Peak Signal to Noise Ratio (PSNR) score, a standard scheme for measuring the quality of image and video compression. It measures fidelity in the logarithmic decibel scale as

$$\mathrm{PSNR} = 20 \log_{10} \frac{255}{\sqrt{\mathrm{MSE}}}, \tag{10}$$

where the empirical Mean Square Error (MSE) between two images I and $I^{'}$ of size $n \times n$ is

$$\mathrm{MSE} = \frac{1}{n^2} \sum_{i,j=1}^{n} (I_{ij} - I^{'}_{ij})^2. \tag{11}$$

If the image contains more than one channel (*e.g.* R, G, B or Y, U, V) then, the MSE is the average of the MSE measured on each channel.

Now we turn our attention to the stopping criterion employed by the active learning algorithm. We need to balance between two conflicting requirements. On one hand, we want to reduce the number of labeled examples. On the other, we want to have a high PSNR. In our experiments, we stop the algorithm by default when either a PSNR of 38 is achieved or a max of 5000 pixels have been queried for color information.

6. Experiments

Our algorithm can work in two different modes: A human-assisted mode, where the active learning module is switched off, and a completely automatic mode which requires an oracle supplying ground truth. The human-assisted mode is useful in situations where ground truth is either not available or it is expensive to label pixels. On the other hand, the automated mode can be used for compression. We experiment with both images and video and report our results below. [1]

Human-Assisted Image Colorization Given a gray-scale image, when its color image are not known apriori, we could label a few pixels with color and hand to our algorithm which will learns the predictor f. The color image are then revealed by applying f to the whole image. Figure 1 presents an experiment on a image (panel (a)), where with the aid of the labeled pixels (panel (a), in color), our semisupervized algorithm is able to produce a visually appealing color image (panel (b)).

Image Compression To test the efficacy of our image compression scheme we perform two experiments. The aim of the first experiment is to show that our active learning approach outperforms humans in choosing pixels for labeling. Here we work with the color image of a colony of bees on hive frames. The image is of size 640×853 and is depicted in panel (a) of Figure 2. The corresponding grayscale image is depicted in panel (b). On this image we asked a human volunteer to label certain pixels with color. The pixels chosen by the human are depicted in panel (c), and the colorized image is depicted in panel (d). Notice that the predicted image is not visually pleasing. For instance, there are certain patches which are marked with a bluish tinge because sufficient labels were not available to learn those textures. This shows that for images with rich texture human-assisted colorization might be tedious and time-consuming. In contrast, panel (e) depicts the pixels chosen by our active learning approach, and the corresponding colorized image is depicted in panel (f). Observe that this image is visually indistinguishable from the ground truth. Not only does the active learning approach produce more visually pleasing results (PSNR of 27.00 for the human labeling vs a PSNR of 31.49 for the active learning approach), but also requires far fewer number of labeled pixels (8558 vs 2534 pixels).

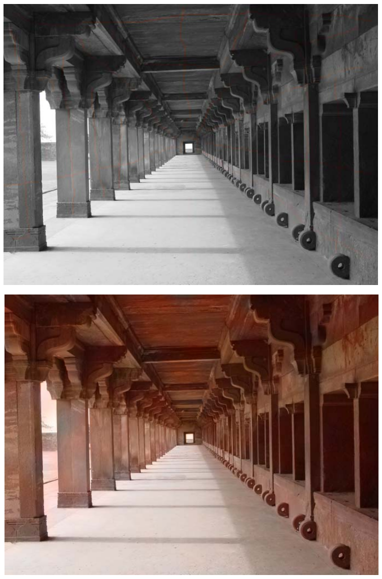


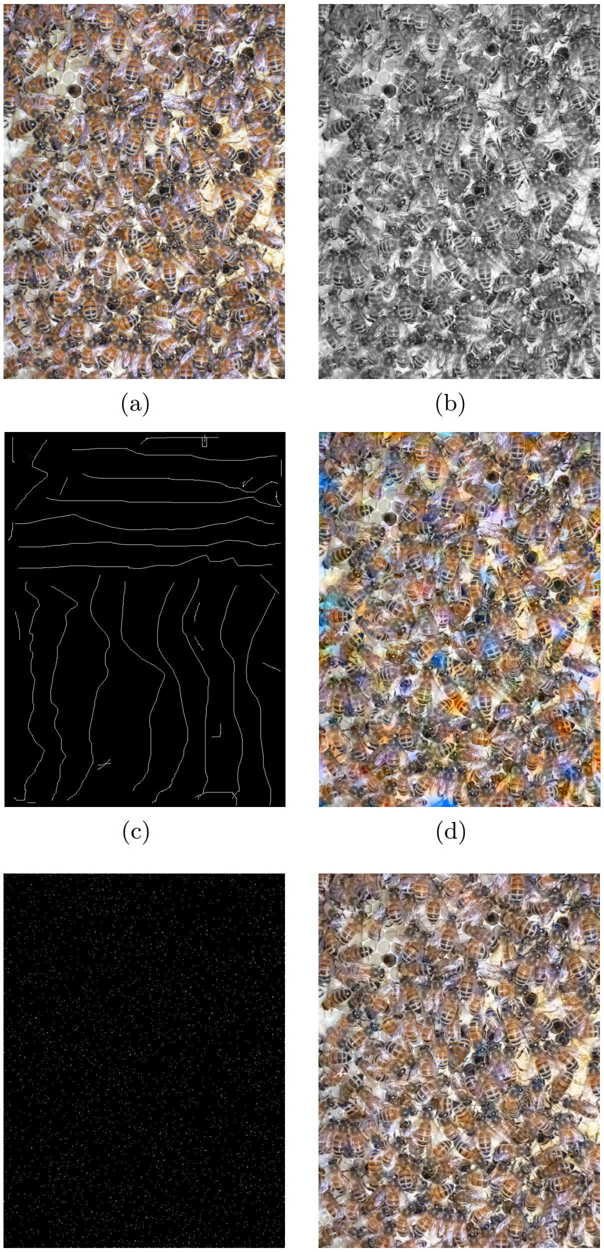
Figure 1. An image colorization example. When presented with this gray-scale image of size 683×512, as well as the labeled pixels (panel (a)), our algorithm is able obtain a visually appealing colorized image (panel (b)).

Recall that our algorithm works in iterations and chooses pixels to query for labels. We plot the evolution of the PSNR score as the iterations proceed in panel (a) of Figure 3. It can be seen that the PSNR curve plateaus after 4 - 6 iterations.

In our second experiment (see Figure 4) we work with the image of a girl. The aim of this experiment is to show that the active learning approach outperforms labeling randomly chosen pixels. The original image of size is 512×683 is depicted in panel (a) of Figure 4. The corresponding grayscale image is depicted in panel

---

[1] The results can be found from a dedicated webpage sml.nicta.com.au/∼licheng/LearnCompressImgVid/ LearnCompressImgVid.html.

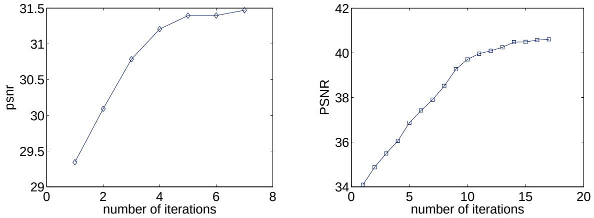


(a) (b)

(c) (d)

(e) (f)

Figure 2. Image compression example. See text for details.



(a) (b)

Figure 3. The PSNR scores for (a) bees and (b) girl vs number of iterations of the active learning algorithm.

(b). The random pixels chosen for labeling are depicted in panel (c), and the colorized image is depicted in panel (d). Notice again that the predicted image exhibits some artifacts (*e.g.* whitish color around the forehead area). In contrast, panel (e) depicts the pixels chosen by our active learning approach, and the corresponding colorized image is depicted in panel (f). Our predicted image is visually indistinguishable from the ground truth. Now the PSNR values are 38.41 and 40.95, and the number of pixels chosen are 2976 and 2766, for the random and active learning approaches respectively. The evolution of the PSNR score with the number of iterations is shown in panel (b) of Figure 3.

Finally, we test the compression ratios achieved by our approach on both the images. In the case of the bees, the original JPEG image occupies 595845 bytes on disk, while the grayscale version occupies 439303 bytes. In addition, we have to store 2534 color pixels. Each pixel for which we store color information requires 4 bytes of additional storage: 2 bytes for the color information (the luminance channel is already present in the grayscale image), and 2 bytes to encode its location. This adds a modest 10136 bytes (approx 10Kb) of extra storage, leading to a compression ratio of 0.754.

For the girl image, the figures are: 220641 bytes for the color JPEG, 161136 bytes for the grayscale, and 2766 colored pixels, leading to a compression ratio of 0.781.

Human-Assisted Video Colorization The aim of this experiment is to show that the color predictor learnt from a single frame can be successfully deployed to predict color on many successive frames, without any visible distortions.

We work with a grayscale image sequence (146 frames of size 240 × 130) of a baby holding a milk bottle, and manually scribble 1542 color labels on the first frame. We then learn a predictor using the labeled and unlabelled pixels of the first frame and use it to predict it on successive frames. Figure 5 shows our results: Panel (a) depicts the first frame of the sequence, panel (b) the grayscale version with the manually annotated labels, panel (c) the prediction results of our algorithm on the first frame, panel (d) the 32^{nd} frame of the video sequence, and panel (e) the prediction of our algorithm.

Video Compression The aim of this experiment is to explore the utility of our method for compressing color videos. We experiment with a video stream that contains 302 frames, each of size 240 × 130, of a call

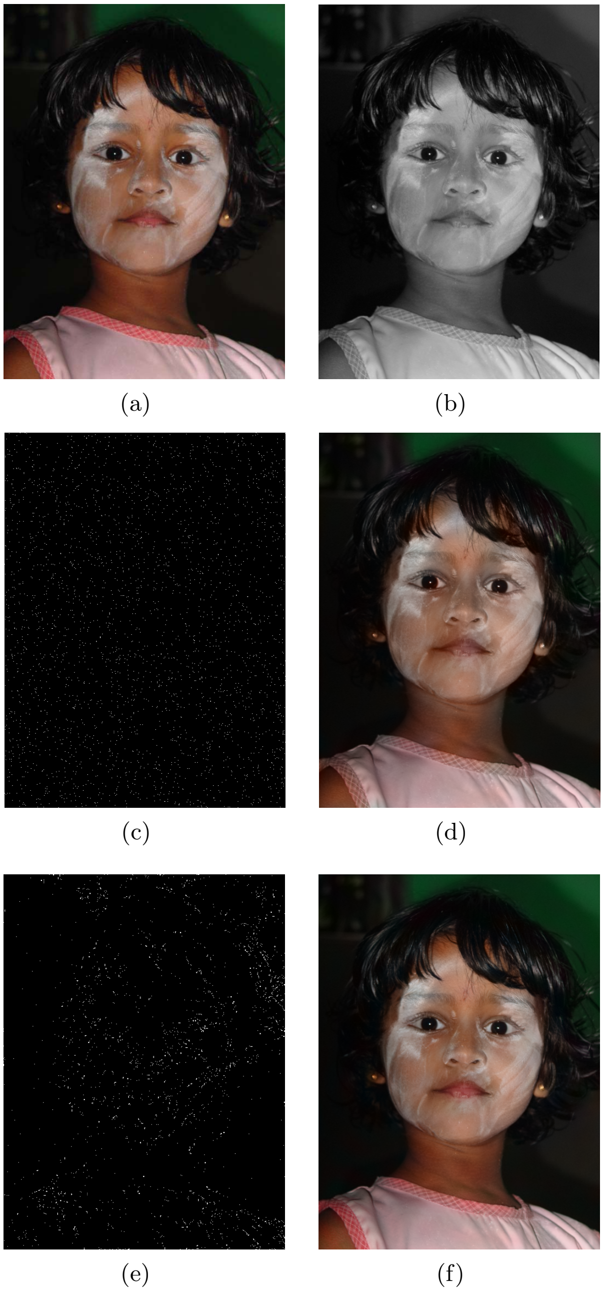

(a) (b) (c) (d) (e) (f)

Figure 4. Image compression example. See text for details.

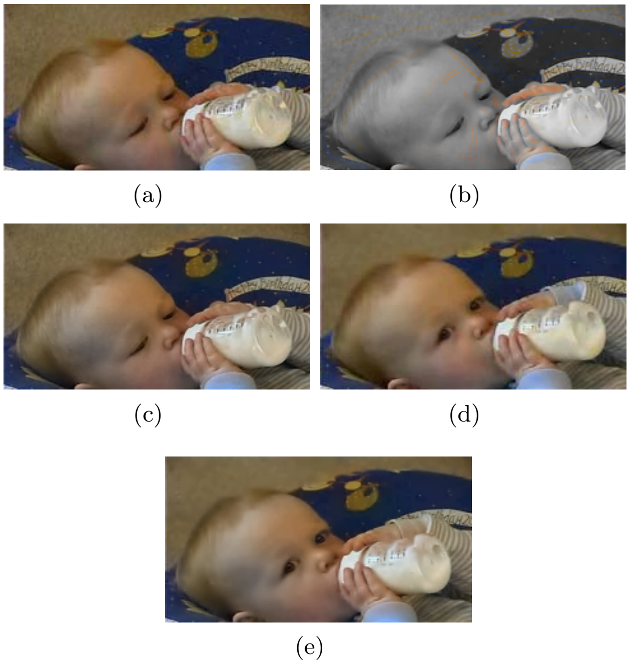

(a) (b) (c) (d) (e)

Figure 5. Human assistant video compression example. See text for details.

center employee. Figure 6 shows our results: Panel (a) depicts the first frame of the sequence, panel (b) the grayscale version, panel (c) the color pixels chosen by our active learning approach, and panel (d) the prediction results of our algorithm. We use our learnt model to predict frames 1 to 49. In the same figure, panel (e) depicts ground truth for frame number 50 and panel (f) the prediction of our algorithm. Notice the raised hand of frame 50 generates heavy color distortion when using a predictor learnt using information only from frame 1. Since our previously learnt model fails to predict well, we now update model with additional color pixels from the current frame. Panel (g) depicts the color pixels chosen and panel (h) depicts the prediction of the new model.

Our active learning approach selectively queries labels for pixels along the boundaries where color changes occur. For instance, in panel (g) notice that it queries for color information around the fingers, since this is a difficult to learn region. During processing the whole sequence, the final model learnt requires 7005 labeled pixels.

We compressed the original color video into a H.264 format movie (using QuickTime Professional in default settings), and the resultant filesize was 816419 bytes. On the other hand, the grayscale movie compressed with the same codec occupied 698865 bytes. Our method also needs to store color information for

7005 pixels. Each pixel for which we store color information requires 4 bytes of additional storage: 2 bytes for the color information (the luminance channel is already present in the grayscale image) and 2 bytes to encode its location. This adds a modest 35025 bytes (approx 34Kb) of extra storage, leading to a compression ratio of 0.899.

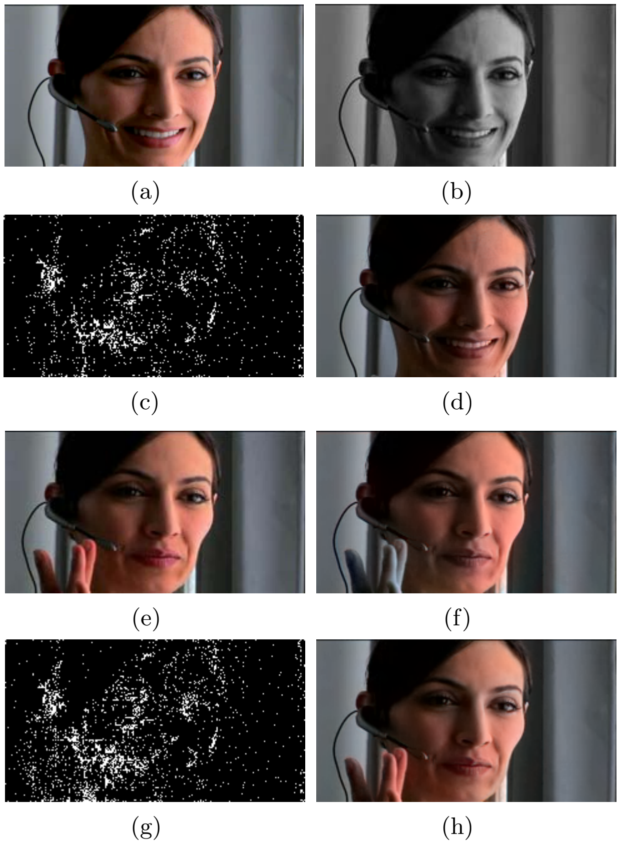


Figure 6. Video compression example. See text for details.

7. Outlook and Discussion

JPEG and H.264 are widely considered state-of-the-art compression techniques for images and video respectively. In this paper, we presented a machine learning approach which is able to compress images and video better than these algorithms, often times achieving competitive compression ratios.

The observation that the colorization algorithm of Levin et al. (2004) is a transductive graph-based semi-supervised learning algorithm led to the research presented in this paper. We enhance and extend the original algorithm in many different ways, which are well motivated from a machine learning viewpoint, and applied it to a novel application.

In the standard active learning paradigm, there is a cost associated with querying for a label. But, there is no reward for forgetting labels. Our algorithm iteratively queries for labels, but never forgets previously queried labels. But, it is possible that we might be able to achieve the same PSNR values with far fewer number of pixels. Extending it to forget labels is part of our future research. Proving performance bounds for our algorithm and addressing non-stationary video sequences are also fertile areas of future research.

Acknowledgements

NICTA is funded by the Australian Government's Department of Communications, Information Technology and the Arts and the Australian Research Council through Backing Australia's Ability and the ICT Center of Excellence program. This work is supported by the IST Program of the European Community, under the Pascal Network of Excellence, IST-2002-506778.

References

Belkin, M., & Niyogi, P. (2003). Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, *15*(6), 1373–1396.

Belkin, M., Niyogi, P., & Sindhwani, V. (2006). Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *J. Mach. Learn. Res.*, *7*, 2399–2434.

Chapelle, O., Schölkopf, B., & Zien, A., eds. (2006). *Semi-Supervised Learning.* Cambridge, MA: MIT Press.

Levin, A., Lischinski, D., & Weiss, Y. (2004). Colorization using optimization. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, 689–694. New York, NY, USA: ACM Press.

Ren, X., & Malik, J. (2003). Learning a classification model for segmentation. In *Proc. 9th Int'l. Conf. Computer Vision*, vol. 1, 10–17.

Schölkopf, B., & Smola, A. (2002). *Learning with Kernels.* Cambridge, MA: MIT Press.

Smola, A. J., & Kondor, I. R. (2003). Kernels and regularization on graphs. In B. Schölkopf, & M. K. Warmuth, eds., *Proc. Annual Conf. Computational Learning Theory*, Lecture Notes in Comput. Sci., 144–158. Heidelberg, Germany: Springer-Verlag.

Zhang, T., & Ando, R. K. (2005). Graph based semi-supervised learning and spectral kernel design. Tech. Rep. RC23713, IBM T.J. Watson Research Center.

Zhu, X. (2005). Semi-supervised learning literature survey. Tech. Rep. 1530, Computer Sciences, University of Wisconsin-Madison. Http://www.cs.wisc.edu/~jerryzhu/pub/ssl_survey.pdf.