Discovering Physics and Design Trends from Visual Temporal Structures
by
Donglai Wei
B.S., Mathematics, Brown University, 2011
S.M., Electrical Engineering and Computer Science, M.I.T., 2013

Submitted to the Department of Electrical Engineering and Computer Science in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in Electrical Engineering and Computer Science at the Massachusetts Institute of Technology
September 2017

© 2017 Massachusetts Institute of Technology
All Rights Reserved.

Signature of Author: ________ Signature redacted ________
Department of Electrical Engineering and Computer Science
August 31, 2017

Certified by: ________ Signature redacted ________
William T. Freeman
Thomas and Gerd Perkins Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by: ________ Signature redacted ________
Leslie A. Kolodziejski
Professor of Electrical Engineering and Computer Science
Chair, Department Committee on Graduate Students
Discovering Physics and Design Trends from Visual Temporal Structures

by Donglai Wei

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

Abstract

Living in a constantly changing world, we cannot help but notice the temporal regularities of visual changes around us. These changes can be irreversible governed by physical laws, such as glass bottles broken into pieces, or influenced by design trends, such as web pages adopting templates with larger background images. In this dissertation, we build computational models to discover and apply the knowledge of the physics for arrow of time, and the design trends for web pages from image sequences.

In the first part of the thesis, I train models to learn the visual cues that are indicative of the arrow of time from large real world video datasets. In the second part of the thesis, I investigate the evolution of visual cues and layout in web page design through screenshots over time. The knowledge of these visual temporal structures are not only of scientific interest by themselves, but also of practical uses demonstrated in this thesis.

Thesis Supervisor: William T. Freeman
Title: Thomas and Gerd Perkins Professor of Electrical Engineering and Computer Science, M.I.T.
Acknowledgments

I'd first like to thank my thesis advisor, Prof. William T. Freeman. From problem formulation to presentation skills, Bill has been a great mentor to guide me throughout my graduate school experience with patience and wisdom. Especially, I deeply appreciate Bill's superb taste in research problems, which pushes me to think more critically and creatively. Beyond teaching me the disciplines of doing research, Bill shows me through his example how to be a researcher with humility, integrity and generosity.

My deep gratitude also goes to my other thesis committee members, Prof. Andrew Zisserman and Prof. Justin Solomon. Andrew has always been a source of great ideas and encouragement during the two projects we collaborated on. Despite his exceptional achievement in the computer vision community, Andrew's passion and detail-oriented attitude towards research teaches me to enjoy the process of pursuing truth. It has always been a fun experience to discuss research questions with Justin, who is quick to decipher the mathematical structure behind the problem and draw connections with other seemingly unrelated topics. Also, I'm thankful to Justin for giving me many insightful and thorough comments on my thesis.

I want to thank other professors and researchers with whom I worked together during my graduate school: Dr. John W. Fisher III, Prof. Frédéric Durand, Dr. Ce Liu, Prof. Antonio Torralba, Prof. Bernhard Schölkopf, Prof. Changshui Zhang, Prof. Joseph Lim, Prof. Dahua Lin, Dr. Lindsay Pickup, Dr. Zheng Pan, Dr. Yichang Shih, Dr. Jason Chang, Dr. Ali Jahanian, Dr. Phillip Isola, Dr. Neal Wadhwa, Dr. Tali Dekel, Dr. Mikhail Bessmeltsev, Bolei Zhou, and Aric Bartle.

I'm thankful to share an office with Hyun Sung Chang, Roger Grosse, Daniel Zoran, Dilip Krishnan, Phillip Isola, Joseph Lim, Neal Wadhwa, Andrew Owens, Tianfan Xue, Jiajun Wu, Zhoutong Zhang and Xiuming Zhang. I will never forget the happy
memories and the intensive discussions.

Thank you to my academic advisor Prof. Charles Leiserson, who persuaded me to keep taking courses every year to explore new directions and shared with me his wisdom on academic growth and development. I am also thankful to my other labmates and friends in vision and graphics at CSAIL: Biliana Kaneva, Miki Rubinstein, Katie Bouman, Abe Davis, Ronnachai Jaroensri, Michael Gharbi, Guha Balakrishnan, Sebastian Claici and Yu Wang.

Lastly, this thesis is impossible without my parents love and support. Thank you, for everything.
Contents

Abstract 3
Acknowledgments 4
List of Figures 9
List of Tables 17

1 Introduction 19
  1.1 Overview ........................................... 20
  1.2 Contributions ....................................... 21
  1.3 My Other Work ...................................... 22

2 Physics from Temporal Structure: Visual Arrow of Time 23
  2.1 Seeing the Arrow of Time .......................... 25
     2.1.1 Related Work .................................. 26
     2.1.2 Dataset ....................................... 26
     2.1.3 Model ......................................... 29
     2.1.4 Experimental results .......................... 36
     2.1.5 Discussion .................................... 41
  2.2 Learning the Arrow of Time ......................... 41
     2.2.1 Related Work .................................. 43
     2.2.2 ConvNet Architecture ......................... 43
     2.2.3 Learning from Simulation Videos ............... 45
     2.2.4 Avoiding Artificial Cues from Real World Videos 47
### Contents

2.2.5 Learning from Real World Videos ........................................ 53
2.2.6 Applications ........................................................................ 56
2.2.7 Discussion ........................................................................... 64
2.3 Conclusion ................................................................................ 64

3 Trends from Temporal Structure: Visual Evolution of Web Design 65

3.1 Related Work ........................................................................... 66
3.2 WebTrend21 Database ................................................................ 67
  3.2.1 Database Construction ....................................................... 67
  3.2.2 Database Pruning ............................................................... 68
  3.2.3 Database Statistics ............................................................. 73
  3.2.4 Benchmark Construction .................................................... 74
3.3 Method Overview ...................................................................... 75
3.4 Trend Visualization for Design Inspiration .................................. 76
  3.4.1 Data and Model ................................................................. 76
  3.4.2 Analysis ........................................................................... 78
  3.4.3 Application ..................................................................... 81
3.5 Trend Colorization for Design Prototyping ................................. 85
  3.5.1 Data and Model ................................................................. 86
  3.5.2 Analysis ........................................................................... 86
  3.5.3 Application ..................................................................... 88
3.6 Trend Classification for Design Evaluation ................................. 89
  3.6.1 Data and Model ................................................................. 89
  3.6.2 Analysis ........................................................................... 90
  3.6.3 Application ..................................................................... 93
3.7 Conclusions ............................................................................. 94

4 Conclusion .................................................................................. 97

Bibliography .................................................................................. 99
List of Figures

1.1 Discoveries from visual changes. This thesis aim to answer the following two questions: (a) Physics: what visual cues indicate its arrow of time? (Chapter 2) (b) Trends: what makes a 2005 web page look like 2005? (Chapter 3) ................................................................. 21

2.1 Arrow of Time Challenge: given five frames from each video, can you tell if the frames are ordered in the forward direction of time or the reversed? (answer key below) ........................................... 24

2.2 Three examples of videos from our TA180 YouTube dataset, including one reverse-time example (right-most frames). The sampled frames are shown following the video frame order from top to bottom. ............ 27

2.3 Overview of the motion consistency (MC) method. We show three frames from a sequence in the TennisBall13 dataset, where a ball is rolled into a stack of static balls. Bottom row: regions of motion, identified using only the frames at $t$ and $t-1$. Notice that the two rolling balls are identified as separate regions of motion and colored separately in the bottom right-most plot. The fact that one rolling ball (first frame) causes two balls to end up rolling (last frame) is what the motion-consistency method aims to detect and use. .................. 30
2.4 Overview of the auto-regressive (AR) method. Top: tracked points from a sequence, and an example track. Bottom: Forward-time (left) and backward-time (right) vertical trajectory components, and the corresponding model residuals. Trajectories should be independent from model residuals (noise) in the forward-time direction only. For the example track shown, $p$-values for the forward and backward directions are 0.5237 and 0.0159 respectively, indicating that forward time is more likely.

2.5 Overview of the bag-of-words (BoW) method. Top: pair of frames at times $t-1$ and $t+1$, warped into the coordinate frame of the intervening image. Left: vertical component of optical flow between this pair of frames; lower copy shows the same with the small SIFT-like descriptor grids overlaid. Right: expanded view of the SIFT-like descriptors shown left. Not shown: horizontal components of optical flow which are also required in constructing the descriptors.

2.6 Left: construction of the divergence operator by summing contributions along straight-edge segments of a square superimposed on the flow word grid. Center & right: positive and negative elements of the div operator, respectively.

2.7 Left: Accuracy of AR model classifications over the 2D parameter space. Right: number of videos accepted. The AR model is particularly effective for small numbers of videos; as we force it to classify more, the performance falls to near chance level.

2.8 Video from the test set, correctly classified as being backwards in time. The most informative point in the video is the few frames around 34 (shown), where there is an inverse splash of water. Red dots here mark the locations of all strongly-backwards flow words. This type of motion (shown in the four flow word grids) was learned to be strongly indicative of a backwards time direction.

2.9 Video from the test set, correctly classified as being forwards in time. The most informative forwards-time words refer to the motion of the billowing steam, and occur periodically as the steam train puffs. Red dots marked on frame 224 show the locations of the most-forwards 10 flow words in this frame.
2.10 Illustration of the temporal class activation map network (T-CAM) for arrow of time (AoT) classification. Compared to the traditional VGG architecture for image recognition, (b) we first concatenate the conv5 features from the shared convolutional layers, (c) and then replace the fully-connected layer with convolution layers and global average pooling layer (GAP) [37, 57, 58, 73] for better localization. .................................................. 44

2.11 The 3-cushion billiard dataset. (a) Original frame from a 3-cushion video; (b) the frame warped (with a homography transformation) to an overhead view of the billiard table; and, (c) a simulated frame to match the real one in terms of size and number of balls. .................................................. 46

2.12 Visualization of test accuracy for sampled billiard ball videos. (a) For the system with only collision damping as the arrow of time signal, the model learns to detect the time of sudden drop of energy and becomes unconfident otherwise; (b) For the friction signal, however, the model is barely confident most of the time due to the weak signal. ..................... 47

2.13 Illustration of artificial signals from videos in UCF101 dataset. (a) The black framing of the clip has small non-zero intensity value, and a vertical slice over time displays asymmetric temporal pattern. (b) After training on the UCF101 test data, there are clusters corresponding to camera zoom-in. ................................................................. 48

2.14 The MJPEG Arrow of Time dataset. It consists of around 16.9k videos without inter-frame video coding from Vimeo. The dataset mostly contains human actions, and the rest are of various contents (e.g. water, animals, vehicle). ................................................................. 50
2.15 Heatmaps of correct predictions from the Flickr-AoT dataset. We use the “jet colormap” to indicate the probability of being forward (red, close to 1) or backward (blue, close to 0) and we only overlay the region where T-CAM is confident. We compute the flow for two consecutive frames in the middle of the clip. The black arrows indicate the direction and the magnitude (normalized by the largest value) of the flow on the sparse grid, and they are only shown for the colored regions. Our network learns to focus on the regions that inform about the AoT in both (a) general and (b) human scenes. (a) The network utilizes basic physics (e.g., water falling due to gravity) and video biases (e.g., people film trains coming towards them more often than trains leaving from them) to correctly decide the AoT. (b) When there is a human in the scene, the network decides the AoT by observing human actions.

2.16 Browser interface for our Amazon Mechanical Turk study to obtain human performance on the arrow of time prediction.

2.17 Heatmap results on the Hollywood-AoT dataset. (a) In these successful cases, our model utilizes basic physics (i.e., gravity) and human expressions to correctly decide the AoT, whereas (b) in some cases, the model fails to discriminate that the video is played backwards. Although the model fails overall in the train sequence, it extracts the correct AoT from the train steam.

2.19 Common cues for reverse film detection (cont. next page). For each reversely-played clip, we show its first, middle and last frame and its heatmap (blue for backward and red for forward) with normalized flow vector on the middle frame. Ideally, the heatmap should be blue, indicating the clip is played in the backward direction.

2.19 The T-CAM model consistently focuses on regions with (a) head motion, (b) motion against gravity, and (c) human body motion. The blue regions indicate that (a) global head motion and eye movement and (b) upward motion patterns of water and snow can be correctly recognized by the T-CAM model for backward AoT. For (c), the T-CAM model can be fooled by professional backward-acting (second row) and subtle motion (third row) where red regions are around performers.
2.20 Test accuracy on the lip reading dataset. (a) input motion of mouth region and (b) distribution of test accuracy over different words. 63

3.1 Tag cloud visualization of top-100 web domains in our WebTrend21 database. The size of the web domain name is in proportion to the number of web page screenshots. 68

3.2 Examples of bad web pages to prune. We remove web pages with rendering errors, either happened at the server-side during (a) capture and (b) retrieval, or at the browser-side during (c) loading. To remove web pages with disqualified content, we filter out (d) web pages that are no longer functional. 69

3.3 Histogram of the mean intensity of our web page dataset. We find that (a) the histogram has several sporadic peaks, (b) especially in the intensity range between 200 and 255. In the main text and Figure 3.4, we identify the cause of each of the four groups of peaks. After pruning, (c) the histogram varies smoothly with the mean intensity value. 69

3.4 Examples of web pages identified through mean intensity histogram method. (a) white pages, (b) google pages, (c) browser-side error, (d) server-side error. 70

3.5 Examples of web pages identified through geometry-based method. (a) pages that are under construction tend to have big page margins, and (b) pages that are rendered without proper CSS files tend to have big page height. 71

3.6 Examples of web pages identified from their html source code. (a) If the html source code does not have any body text, errors like “page not found” or “unfinished loading” can happen. Two common errors can be found through matching the keyword of (b) “Index of” and (c) “sorry.” 72

3.7 Distribution of data over years and distribution of colors over years. Magnitudes are in a logarithmic scale. The color space is a discretized version of sRGB with 4 bins per channel, and sorted based on hue. 73
Benchmark preprocessing. In height, we crop out the top 729 pixels for the first impression, avoiding over-compressing (a) long scrolled-down pages. In width, we crop out (b) one-sided margin due to the html layout not adapting to the browser window width and keep (c) two-sided margin by design.

Variational autoencoder model. Given input images, the model learns to compress them into more compact representation with the encoder module, which can well recover the original input with the decoder module. We only use the encoded mean vector $\mu$ for our latent code.

Comparison of VAE trained with different dimensions of latent variable. (a) we plot the reconstruction error over epoch number (the error does not go to zero partly due to the noise injected in the sampling module to avoid overfitting); (b) we show qualitative comparisons for three sampled images; (c) covariance matrix of 20 axes of latent representations with largest variance. With $\#z > 100$, the learned representation becomes distributed, not concentrating on certain dimensions.

Variational autoencoder model. Given input images, the model learns to compress them into more compact representation with the encoder module, which can well recover the original input with the decoder module.

Visualization for the first ten eigenvectors. For the first two year groups, blue and yellow are main colors.

Visualization for the first ten eigenvectors. For the first two year groups, blue and yellow are main colors.

2D representation from t-SNE. We first bin the t-SNE code into $40 \times 40$ grid cells and average over images whose codes belong to each cell. This 2D map is purely based on the appearance of web pages without any year labels, where each region correspond to one trend of design. Later, with year labels, we visualize how the design trends change over time.

For each year, we show the heatmap of the ratio of images at each rounded grid position.
3.16 Visualization of the nonlinear manifold learned by VAE. (a) We use t-SNE to project the latent code onto the 2D plane and round up the position to regular grid point; (b) for each group year, we show the heatmap of the ratio of images at each rounded grid position. The heatmaps change continuously over time, showing the design trend shift. For example, late 1990s web pages often have white margins or colorful background color, while 2010s web pages mostly have wide images at the top of the page.

3.17 Top 20 most frequent colors in each year. Magnitudes are in logarithmic scale. The color space is a discretized version of sRGB with 4 bins per channel, sorted based on hue. Through this analysis, we observe that the Internet is getting duller (less saturated) and a wider, more evenly spread, range of colors is being used as time progresses (distribution over colors is becoming higher entropy).

3.18 Conditional Generative Adversarial Network model. Given input gray scale images, the generator model outputs a color image, and the discriminator model classify if the input-output pair is fake or real.

3.19 Qualitative results for web page re-colorization by models trained on different groups of years. (a-d) input web pages are from 1996-2000, 2001-2005, 2006-2010, 2011-2016 respectively.

3.20 Illustration of the bricolage colorization application. Given (b) manually stitched web pages from (a) input templates, pix2pix models trained on web pages from different periods generate (c) diversified and self-consistent colorization from the gray-scale image of (b).

3.21 Illustration of our convolutional neural network, VGG-CAM [73]. Given an input web page, the model predicts its year label. The first five groups of convolutional layers (blue) are the same as VGG-16. Then there are three $1 \times 1$ convolutional layers (yellow) to replace the fully-connected layers in VGG-16. In the end, after a global-average-pooling layer (orange) and a softmax layer, we obtain the probability of the year label for the input web page.
3.22 Test results for VGG-CAM model. (a) confusion matrix for the exact year classification, (b) confusion matrix for the prediction to be within one year range of the ground truth, (c) example web pages from “brown.edu”, where the design trend can last for several years which makes it hard to predict the exact year of a web page.

3.23 Here we show, for each predicted year, the top four web pages our model is most confident belong to that year, revealing designs that are highly characteristic for each year, such as simple textual web pages in the 1990s, and more image-heavy designs of the 2010s.

3.24 Visualization of VGG-CAM neurons at different layers. We show all 64 filters for conv1 units and four sample filters for conv3 and pool5. Noticeably, the VGG-CAM learns mostly horizontal and vertical edges (conv1), simple shapes (conv3), and more complex visual patterns like logos and faces (pool5).

3.25 Heatmap of predicted years for web pages. We show (a) four input web pages ranging from 1996 to 2016 and (b) the predicted 14x14 heatmap of predicted years, scaled to the image size. Our model learns to find the logo region in the example from 1996 and the text region in the example from 2016 out of date.

3.26 These two web pages might seem generic, but each reflects the design trend of its own era. Can you tell what makes the 2016 web page look like 2016? Over each example web page above, we show regions our algorithm thinks are diagnostic of that web page’s year. On each row, one patch from the web page is shown, followed by four patches, from other pages, that the algorithm considers to have a similar visual pattern. Noticeably, the old design (1996) mainly uses simple text and textual links for communicating the content, and a modern design (2016) applies more diverse graphical elements such as special fonts for text, graphic logos, and natural images.
List of Tables

2.1 Arrow of time classification results on the TennisBall13 and TA180 dataset.
(a) the accuracy on TennisBall13 is high due to its small number of videos (b) we compare all four methods with 3-fold cross-validation on TA180: spatial-temporal orientated energy (SoE), motion consistency (MC), auto-regressive (AR), bag-of-words (BoW) ........................................ 37

2.2 Test accuracy on the simulated 3-billiard datasets and the real video. We compare T-CAM model with either one (T=1) or two (T=2) temporal segments for three different experiments. (1) For no signal case (None), the test performance is at the chance level; (2) Both models perform well when there is only collision damping in the system (Col.), but temporal fusion on conv5 helps to learn weak signals like friction (Fric.). (3) the simulation Col.+Fric. is made with a similar system configuration to the real videos (3-cush.), where the collision damping signal dominates. Thus, both models have similar performance. ........................................ 48

2.3 An examination of the artificial cues for arrow of time prediction. We show results of controlled experiments on the effect of black framing and camera motion on UCF101. ................................................................. 51

2.4 Effect of temporal codec on MJPEG-AoT dataset. We train and test on three versions of the MJPEG-AoT dataset and our model has similar performance. Thus, the added codec signal doesn't introduce significant artificial signals for our model to capture for the arrow of time. ....... 52
2.5 Test accuracy of the T-CAM model trained on the Flickr-AoT train dataset. (a) we first compare different parameters of the T-CAM model on the Flickr-AoT test dataset. (b) Other datasets: the MJPEG-AoT and the TA180 of [43]. T-CAM (raw) is trained on a larger set of videos from Flickr without pre-processing. ................. 54

2.6 Feature learning results for action recognition on UCF-101 split-1. We list out important factors besides the network initialization methods: input, architecture and layers to fine-tune. For the flow input, our initialization from arrow of time prediction is consistently better than the previous state-of-the-art ImageNet initialization [65] when only the fc layers are fine-tuned. ......................... 58

3.1 Colorization errors on different groups of years. We test four models trained on different years of web pages and a model trained on all years. We compute the L2 distance between the ground truth ab color channel and that from output. The model trained on each year group performs the best when tested on the same year group. ......................... 88

3.2 Test accuracy for year label prediction on WebTrend21. Colorization errors on different groups of years. To show that the ......................... 91
Chapter 1

Introduction

"No man ever steps in the same river twice, for it's not the same river and he's not the same man."

We live in a constantly changing world on all different levels. For atoms in isolated physical systems, the Second Law of Thermodynamics dictates their unidirectional change to reach higher levels of randomness. Not to mention design trends which are born and died equally fast.

Growing up in this dynamic environment, we unconsciously learn about statistical regularities from these diverse visual changes. For example, we will be more surprised if we see water drops come together than to splatter, as it is against our statistical knowledge of how things should change. Given a web page, we can often tell if its design looks out of fashion by finding statistical commonalities with web pages we visited before. Can we build computers to have the similar intelligence to discover these temporal structures?

Beyond scientific understanding, the knowledge of temporal statistical structures can have a wide range of applications. For example, in video forensics, i.e. identifying if a video is manipulated, such temporal knowledge can tell if some frames have been deleted or added to fake a person's presence in a surveillance video. In web page design, understanding the trend of each era can help designers come up with layout and color schemes that are either in fashion or out of date and give feedback on which part of the web page looks outdated.

However, compared to the extensive work on statistical structures for single images [7, 16, 29, 45, 52], there has been far less investigation for temporal structures for image sequences [17]. Recently, the temporal ordering of image sequences, a specific statistical property of temporal structures, is learned to reorder a collection of images.
from different cameras chronologically [5, 10] and to learn useful visual features for other vision tasks [14, 15, 41, 46], e.g. action recognition. In this promising direction, we not only need deeper scientific understanding of the temporal structures, e.g. what visual cues do the models capture, but also new statistical properties of temporal structures with broader applications.

In this thesis, we investigate two different statistical properties of temporal structures: direction and trend (Figure 1.1). We work on image sequences from different research communities: real world videos in computer vision, and web designs in the human computation interfaces. By exploiting the statistical structure of visual changes, we develop novel applications which can be used for video forensics and give interactive feedback during different stages of web page design process. Below, we give a high-level overview of my thesis, along with contributions and other works done during my doctoral program.

1.1 Overview

First, we address the question: “what visual cues are indicative of the arrow of time, i.e. the one-way direction of changes?” We train convolutional neural networks (CNN) to classify whether input videos are playing in the forward or backward direction. A model that can perform this task well can be used for video forensics and abnormal event detection, and its learned features are useful for other tasks, such as action recognition. However, CNNs can “cheat” and learn artificial signals from video production instead of the real signal. We design controlled experiments to systematically identify superfluous signals. After controlling these confounding factors, we analyze the visual cues learned on the large-scale Flickr video dataset, revealing both semantic and non-semantic cues for arrow of time and the photographer bias during video capture.

Next, we answer the question: “what makes 2016 web pages look like designed in 2016?” We first collect a large-scale dataset containing screenshots for top 11,000 popular web domains over 21 years (1996-2016). Trained on this database, our models investigate the design trend through visualization, colorization and evaluation. For visualization, we train variational autoencoder models to learn interpretable representations for design exploration on a 2D plane intuitively. For colorization, we use conditional generative adversarial networks to learn to re-colorize input web pages with different
Sec. 1.2. Contributions

Figure 1.1: Discoveries from visual changes. This thesis aim to answer the following two questions: (a) Physics: what visual cues indicate its arrow of time? (Chapter 2) (b) Trends: what makes a 2005 web page look like 2005? (Chapter 3)

color palettes based on their content and year labels. Finally, we train a CNN to predict the years in which input web pages were created and identify visual elements that are ahead of or lagging behind the trend.

1.2 Contributions

In this thesis, we expand the frontier of statistical understanding of temporal structures in the following three ways:

1. **Building large-scale databases.** To extract reliable and generalizable statistics for temporal structures, we need a massive amount of image sequence data. We collect two real world video databases for the arrow of time work and a database for web design screenshots over 21 years, all publicly available for the community.

2. **Analyzing learned representations.** Besides the competitive performance of the learned representations, we use methods to interpret the learned by the algorithm.

3. **Exploring new applications.** We propose novel applications to exploit the obtained knowledge of temporal statistical structures.
1.3 My Other Work

In this thesis, motion is used as input to shed light on videos' statistical temporal structures, revealing high-level knowledge of physics, trend and style. During my doctoral program, I also studied motion perception from four different perspectives: estimation, representation, segmentation and visualization.

Under the supervision of Dr. Ce Liu and Prof. William T. Freeman, I applied the data-driven matching framework, commonly used for high-level vision problems like semantic parsing, to improve the traditionally physics-based motion estimation in stereo and flow [69]. With Prof. Dahua Lin and Dr. John W. Fisher III, I studied the manifold of object deformations with Lie algebra representation and parallel transport technique [68]. With Dr. Jason Chang and Dr. John W. Fisher III, I co-developed a video motion segmentation method based on Dirichlet process models with Gaussian random field models on top [8]. I also participated in deviation magnification [63] with Dr. Neal Wadhwa, Dr. Tali Dekel and Prof. William T. Freeman. Interested readers can refer to these publications for details.
Chapter 2

Physics from Temporal Structure: Visual Arrow of Time

“Let us draw an arrow arbitrarily. If as we follow the arrow we find more and more of the random element in the state of the world, then the arrow is pointing towards the future; if the random element decreases the arrow points towards the past. That is the only distinction known to physics.”
— Arthur Eddington, The Nature of the Physical World

How much of what we see on a daily basis could be time-reversed without us noticing that something is amiss (Figure 2.1)? At a small scale, the physics of the world is reversible. Most fundamental equations look the same in both forward and backward directions of time. The collision of two particles looks the same if time is reversed. Yet at a macroscopic scale, time is not reversible and the world behaves differently whether time is playing forwards or backwards. For example, for a cow eating grass or a dog shaking water from its coat, we do not automatically accept that a complicated system (chewed food or the spatter pattern of water drops) will re-compose into a simpler one.

This seemingly innocent question is deeply-connected with fundamental properties of time. In physics, such asymmetric nature of temporal structures in our world is called the “arrow of time” (AoT), which has been studied in the context of thermodynamics and cosmology [44, 47]. While physics addresses whether the world is the same forwards and backwards, our goal is to assess whether and how the direction of time manifests itself visually. Can we train a machine to make that same judgment? What is it that lets us see that time is going the wrong way in reversed videos?

Apart from the fundamental science, the arrow of time plays important roles in
Figure 2.1: Arrow of Time Challenge: given five frames from each video, can you tell if the frames are ordered in the forward direction of time or the reversed? (answer key below)

Figure 2.1: Arrow of Time Challenge: given five frames from each video, can you tell if the frames are ordered in the forward direction of time or the reversed? (answer key below)

different areas of artificial intelligence. First, the arrow of time is a natural supervision signal for videos, i.e. we can trivially obtain supervision labels (i.e. the direction in

\footnote{Forwards: (a), (c), (e); backwards: (b), (d). Though in (c) the motion is too small to be visible to the eye.}
which the video is played) for massive amount of data to train large-scale classifiers. Besides the prediction task, these arrow of time labels can be used to learn visual representations for other video classification tasks, e.g. action recognition [15, 41]. Second, and more generally, temporal asymmetry priors have implications for methods for video processing, e.g. video decompression, optical flow estimation, and photo-sequencing [5]. Also, causal inference, of which the arrow of time is a special case, has been connected to machine learning topics, such as transfer learning and covariate shift adaptation [51].

In this chapter, we address the following questions for the direction of visual changes:

1. Can computers learn from human knowledge of physics to see the arrow of time in videos?

2. Can computers instead learn by themselves simply through watching real world videos? Further, what visual cues will they learn to use?

## 2.1 Seeing the Arrow of Time

Through our daily experience, we have unconsciously learned to “see” the arrow of time in videos, i.e. we can perceive if videos are playing in the forward or backward direction. In this section, we address the question: How can we teach computers?

Specifically, we formulate our approach within the framework of video classification, where we train classifiers to predict the direction in which the input video is played, i.e. forward or backward. For each video, we want to know not only how strong is its signal indicating the arrow of time, but also in which regions can we predict it reliably. We seek to use low-level visual information - closer to the underlying physics - to see the arrow of time, not object-level visual cues. We are not interested in learning that cars tend to drive forward, but rather in studying the common temporal structure of videos. We expect that such regularities will make the problem amenable to a learning-based approach. Some videos will be difficult or impossible; others may be straightforward.

In the sequel, we first curate two video datasets as common test beds for the arrow of time classification in Section 2.1.2. Then in Section 2.1.3, we propose four different ways to incorporate our knowledge of physics into machine learning algorithms to classify the direction of videos. We compare and analyze classification results of our models above in Section 2.1.4.
### 2.1.1 Related Work

Learning and representing statistical structures of visual input are crucial for a variety of computer vision tasks. These structures can capture both the spatial properties of images and temporal properties of image sequences.

For the spatial structure, many algorithms [16] are developed to exploit different aspects of the spatial regularity. For example, the now widely-used multi-resolution spatial pyramid architectures and convolutional neural networks [7, 34] are evolved from the statistical question, “Are images invariant over scale?” Many more statistical questions are asked to learn properties of human visual processing: “Can we recognize faces upside down?” [52], or about tonescale processing by asking, “Can we recognize faces if the tonescale is inverted?” [29]. Especially, to our interest, both symmetry and asymmetry properties of spatial structures are useful for visual inference. The assumption that light comes from above is asymmetric and helps us disambiguate convex and concave shapes [45]. Spatial translation invariance, on the other hand, allows us to train object detection or recognition systems without requiring training data at all possible spatial locations.

For the temporal structure [17], there has been far less investigation. Several recent papers have used temporal ordering. Basha et al. [5, 10] consider the task of photo-sequencing – determining the temporal order of a collection of images from different cameras. Others have used the temporal ordering of frames as a supervisory signal: for learning an embedding in [46]; for self-supervision training of a ConvNet [15, 41]; and to construct a representation for action recognition [14]. We here focus on a simple property of the temporal structure, i.e. temporal direction, asking: “Are video temporal statistics symmetric in time?”

### 2.1.2 Dataset

We collect two video datasets for the arrow of time classification task. The first TennisBall13 dataset, consisted of videos shot by ourselves with simple content, serves as a controlled environment to illustrate our new algorithms. The second TA180 dataset, containing general videos downloaded from Youtube, represents the complex visual world.
Sec. 2.1. Seeing the Arrow of Time

Figure 2.2: Three examples of videos from our TA180 YouTube dataset, including one reverse-time example (right-most frames). The sampled frames are shown following the video frame order from top to bottom.

**TennisBall13 dataset** We film a small number of video clips using a camera with the Motion-JPEG codec [49], which compresses each frame independently without artifacts holding any time-direction information. This dataset comprises 13 HD videos of tennis balls being rolled along a floor and colliding with other rolling or static balls. An example of one of these sequences is used later in Figure 2.3.

**Evaluation Procedure.** Tests on the TennisBall13 dataset are run in a leave-one-out manner, because of the small size of the dataset. Specifically, classification models will be trained on 12 videos (and their various flips), and then tested on the withheld video, and all 13 possible arrangements are run.

**YouTube dataset (TA180)** This dataset consists of 180 video clips from YouTube, which are obtained manually using more than 50 keywords. Our goal is to retrieve a diverse set of videos from which we might learn low-level motion-type cues that indicate the direction of time. Keywords include "dance," "steam train," and "demolition," among other terms. The dataset is available at [http://www.robots.ox.ac.uk/data/arrow/](http://www.robots.ox.ac.uk/data/arrow/).

**Selection Criteria.** Video clips are selected to be 6-10 seconds long, giving at least 100 frames on which to run computations. Thus, we discard clips from many professionally-produced videos where each shot's duration is too short, since each clip is required to be a single shot. We restrict the selections to HD videos, allowing us to subsample extensively to avoid block artifacts and minimize interactions with any given
compression method.

Videos with poor focus or lighting are discarded, as are videos with excessive camera motion or motion blur due to hand shake. Videos with special effects and computer-generated graphics are avoided, since the underlying physics describing these may not match exactly with the real-world physics underpinning our exploration of the arrow of time. Similarly, split-screen views, cartoons and computer games are all discarded. Also, videos are required to be in landscape format and minimal interlacing artifacts. In a few cases, the dataset still contains frames in which small amounts of text or television channel badges have been overlaid, though we try to minimize this effect as well.

Intentionally, the dataset contains clips from “backwards” videos on YouTube; these are videos where the authors or uploaders have performed the time-flip before submitting the video to YouTube, so any time-related artifacts that might have been introduced by YouTube’s subsequent compression will also be reversed in these cases relative to the forward portions of our data. In these cases, because relatively few clips meeting all of our criteria exist, multiple shots of different scenes are taken from the same source video in a few cases. In all other cases, only one short clip from each YouTube video is used.

In total, there are 180 videos in the dataset and 25 of them are reversed intentionally by youtube uploaders to achieve special effects. Two frames from each of three example videos are shown from top to bottom in Figure 2.2. The first two are forwards-time examples (baseball game, water spray), and the right-most pair of frames is from a backwards-time example where pillows are “un-thrown.” During learning, we feed the algorithms both the original video and their reversed versions, to avoid the unbalanced label distribution of the downloaded videos.

Evaluation Procedure. For evaluating the methods described below, the dataset is divided into 60 testing videos and 120 training videos, in three different ways such that each video appeared as a testing video exactly once and training exactly twice. For parameter selection, the 120 videos of the training set are further sub-divided into 70 training and 50 validation videos. The three train/test splits are labeled A, B and C, and are the same for each of the methods we report. The backwards-to-forwards video ratios for the three test sets are 9:51, 8:52 and 8:52 respectively.

The evaluation measure for each method is the proportion of the testing videos on
which it could correctly predict the time direction.

\section*{2.1.3 Model}

There are many underlying physical reasons why the forward and reverse time directions may look different on average. Below we probe how the arrow of time can be determined from video sequences in four distinct ways, drawing intuitions from energy, entropy, causality and statistical patterns.

**Energy-based: Spatio-temporal oriented energy (SoE)** In classical mechanics, the kinematic energy of a system changes in certain patterns. The energy may decrease in time due to its dissipation, and may fluctuate while in equilibrium. We use the “Spatio-temporal oriented energy” (SOE) \cite{11} as an off-the-shelf method to describe how the kinematic energy changes from the video motion.

SOE is a filter-based feature for video classification, which is sensitive to different space-time textures. Our implementation is faithful to that laid out in \cite{11}, and comprises third-derivative-of-Gaussian filters in eight uniformly-distributed spatial directions, with three different temporal scales. As in the prior work, we split each video into $2 \times 2$ spatial sub-regions, and concatenate the SOE responses for each sub-region for form a final feature.

**Entropy-based: motion consistency (MC)** Stated by the second law of thermodynamics, the total entropy of an isolate system can only increase. Thus, it is far more common for one motion to evolve into multiple motions than for multiple motions to collapse into one consistent motion. For instance, the cue ball during a snooker break-off hits the pack of red balls and scatters them, but in a reverse-time direction, it is statistically highly unlikely that the initial conditions of the red balls (position, velocity, spin etc.) can be set up exactly right, so that they come together, stop perfectly, and shoot the cue ball back towards the cue.

To exploit this observation of motion consistency, we first find pixels in the video where frame intensity differences are above a threshold. Then, we grow these pixels into consistent regions and compare them with those in the previous frame. These regions are illustrated in Figure 2.3. Broadly speaking, we expect more occurrences of one region splitting in two than of two regions joining to become one, in the forwards-time direction.
Figure 2.3: Overview of the motion consistency (MC) method. We show three frames from a sequence in the TennisBall13 dataset, where a ball is rolled into a stack of static balls. Bottom row: regions of motion, identified using only the frames at $t$ and $t - 1$. Notice that the two rolling balls are identified as separate regions of motion and colored separately in the bottom right-most plot. The fact that one rolling ball (first frame) causes two balls to end up rolling (last frame) is what the motion-consistency method aims to detect and use.

**IMPLEMENTATION DETAILS.** Given the image at current time $t$ ($I_t$), we warp the next frame ($I_{t+1}$) to it with estimated homography, which yields a warped image $W_{t+1}$. The difference between the current frames and the warped frame, $|I_t - W_{t+1}|$, now highlights moving areas. A smooth-edged binary mask of image motion is made by summing this difference over color channels, resizing down to 400 pixels, convolving with a disk function and then thresholding. The different regions in the motion mask are enumerated. Where one or more regions at time $t$ intersect more than one region each at time $t - 1$, a violation is counted for that frame pair, since this implies motion merging. We count the violations for each time direction of a sequence separately, with a variety of parameter settings. We use three different threshold-radius pairs: radii of 5, 6.6 and 8.7 pixels, and corresponding thresholds of 0.1, 0.01 and 0.056, where image intensities lie between 0 and 1. Finally, we train a standard linear SVM using the violation counts as 6d features (two time-directions; three parameterizations).
Causality-based: auto-regressive (AR) Researchers have recently studied the question of measuring the direction of time as a special case of the problem of inferring causal direction in cause-effect models. Peters et al. [42] shows that, for non-Gaussian additive noise and dynamics obeying a linear ARMA (auto-regressive moving average) model, the noise added at some point of time is independent of the past values of the time series, but not of the future values. This allows us to determine the direction of time by independence testing, and Peters et al. [42] uses it to successfully analyze the direction of time for EEG data. Intuitively, this insight formalizes our intuition that changes (noise) added to a process at some point of time influences the future, but not the past.

Here we consider the AR model, a special case of ARMA without the moving average part. However, we deal with vector-valued time series, which strictly speaking goes beyond the validity of the theoretical analysis of Peters et al. [42]. We find that a second order AR model works well in our setting. In a nutshell, we model the time series’ next value as a linear function of the past two values plus additive independent noise.

The assumption that some image motions will be modeled as AR models with additive non-Gaussian noise leads to a simple algorithm for measuring the direction of time in a video: track the velocities of moving points, fit those velocities with an AR model and perform an independence test between the velocities and model residuals (errors). This process is illustrated in Figure 2.4.

The independence testing follows the work of [42], and is based on an estimate of the Hilbert-Schmidt norm of the cross-covariance operator between two reproducing kernel Hilbert spaces associated with the two variables whose independence we are testing. The norm provides us with a test statistic, and this in turn allows us to estimate a \( p \)-value for the null hypothesis of independence. If the \( p \)-value is small, the observed value of the norm is very unlikely under the null hypothesis and the latter should be rejected. Ideally, we would hope that the \( p \)-value should be small (i.e. close to 0) in the backward direction, and large (i.e. significantly bigger than 0) in the forward direction.

Implementation details. We analyze the motion of a set of feature points extracted by KLT trackers [39,60], running tracking in both forward and backward directions. For each tracked point, velocities are extracted, and a 2D AR model is fitted. We then test the independence between the noise and velocity to determine the arrow of time.
Figure 2.4: Overview of the auto-regressive (AR) method. Top: tracked points from a sequence, and an example track. Bottom: Forward-time (left) and backward-time (right) vertical trajectory components, and the corresponding model residuals. Trajectories should be independent from model residuals (noise) in the forward-time direction only. For the example track shown, p-values for the forward and backward directions are 0.5237 and 0.0159 respectively, indicating that forward time is more likely.

Inferring causal direction of AR process is only possible when the noise is non-Gaussian, and when noise in only one temporal direction is independent. We define a valid trajectory to be one which spans at least 50 frames, for which noise in at least one direction is non-Gaussian as determined by a normality test, and for which the p-value test in one time-direction gives $p < 0.05$ whereas in the other it gives $p > 0.05 + \delta$ for some minimal gap $\delta$ (i.e. exactly one direction fails the null hypothesis test).

All valid trajectories are classified as forward or backward according to their $p$-value scores. Ideally, all valid trajectories for one video should imply the same direction of time, but in practice, tracking can be noisy in a way that violates the time-series model assumption. For this, we reject the videos with fewer than $N$ valid trajectories, where $N \geq 1$. We classify the accepted videos by a majority vote among the valid trajectories. We thus get a binary classifier (with the possibility of rejection) at video level. While
hypothesis testing is used to classify single trajectories, the overall procedure is not a hypothesis test, and thus issues of multiple testing do not arise. The hypothesis testing based trajectory classifiers can be seen as weak classifiers for video and the voting makes a strong classifier.

**Statistics-based: Bag-of-Word (BoW)** In the above three methods, we manually design the visual features from videos to capture known physical quantities, i.e. energy, entropy and causality. Besides physics, statistics-based perceptual priors are also indicative of the arrow of time. For example, it is physically possible for a man to walk backward while facing front, but we may think it more natural for the reverse which has higher probability to happen in our daily experience. Here we design visual features to capture such statistical regularity of the input. Inspired by the “bag-of-words” method for object recognition [36], we apply it to our optical flow input to discover motion clusters, referred to as flow words, exhibiting temporal asymmetries.

Each flow word is based on a SIFT-like descriptor of motion occurring in small patches of a video. We first register the frames of a video in order to compensate for hand-shake and intentional camera motion (panning and zooming etc.), and then we assume that any residual motion is due to objects moving in the scene. Rather than computing a SIFT based on image edge gradients, motion gradients from an optical flow computation are substituted, giving a descriptor which represents local histograms of image motion. An example of the descriptors used to build these flow words is shown in Figure 2.5.

These object-motion descriptors are then quantized to form a discrete set of flow words, and a bag-of-words descriptor representing the entire video sequence is thus computed. With sufficient forward and reverse-time examples, a classifier can be trained to discriminate between the two classes.

**Implementation Details.** In a similar manner to [64], computation starts from dense trajectories. However, instead of only representing the histogram of optical flow over a region, as in the HOF features of [33], we also represent its spatial layout in a local patch in a manner similar to a local SIFT descriptor.

In detail, frames are downsized to 983 pixels wide in a bid to remove block-level artifacts and to provide a consistent image size for the rest of the pipeline. Images at time $t - 1$ and $t + 1$ are registered to the image at time $t$, and an optical flow
CHAPTER 2. PHYSICS FROM TEMPORAL STRUCTURE: VISUAL ARROW OF TIME

Figure 2.5: Overview of the bag-of-words (BoW) method. Top: pair of frames at times $t-1$ and $t+1$, warped into the coordinate frame of the intervening image. Left: vertical component of optical flow between this pair of frames; lower copy shows the same with the small SIFT-like descriptor grids overlaid. Right: expanded view of the SIFT-like descriptors shown left. Not shown: horizontal components of optical flow which are also required in constructing the descriptors.

computation is carried out [6]. This is repeated over the video sequence in a temporal sliding window, i.e. giving $T - 2$ optical flow outputs for a sequence of length $T$. These outputs take the form of motion images in the horizontal and vertical directions. A normal VLFeat dense SIFT descriptor [61] uses intensity gradient images in the $x$- and $y$-directions internally in its computations, so to describe the flow maps instead of
the image structure, we simply operate on the vertical and horizontal motion images instead of the intensity gradients. This yields motion-patch descriptors whose elements represented the magnitudes of motion in various directions passing through the $t^{th}$ frame, on a small $4 \times 4$ grid (bin size 6 pixels) sampled once every 3 pixels in the horizontal and vertical directions.

We suppress static trajectories, because we are only interested in dynamic behaviour for this work. This is achieved by setting the motion-gradient images to zero where the magnitude is below a threshold, and not including motion-patch descriptors from such areas in further processing. This excises large areas, as can be seen in the constant background color in Figure 2.5.

**DIVERGENCE OPERATOR FOR FLOW WORDS:** In analyzing the results, it will be useful to quantify the divergence or other motion-field properties of a given flow word descriptor. Using the divergence theorem, we make an approximation to the integral of the divergence over a descriptor by considering the net outward and inward motion fluxes across a square that joins the centers of the four corner cells on the descriptor grid, assuming that motion at every point within a grid cell is drawn independently from the histogram which is represented at the centre of that cell. The resulting divergence operator for flow words is shown in the central and right parts of figure 2.6, with the negative and positive flux contributions separated for visualization.

**LEARNING.** A dictionary of 4000 words is learnt from a random subset of the training data ($O(10^7)$ examples) using K-means clustering, and each descriptor is assigned to its closest word. A video sequence is then described by a normalized histogram of visual flow words across the entire time-span of the video. Empirically, performance is improved if the square roots of the descriptor values are taken prior to clustering, rather than the raw values themselves.

For each video in the training/validation set, we extracted four descriptor histograms: (A): the native direction of the video; (B): this video mirrored in the left-right direction; (C): the original video time-flipped; and (D): the time-flipped left-right-mirrored version. An SVM is trained using four histograms A–D extracted from each video of a training set, 280 $(4 \times 70)$ videos in total. Similarly, the 50 videos of the validation set generate 200 histograms A–D in total, and each is classified with
Figure 2.6: Left: construction of the divergence operator by summing contributions along straight-edge segments of a square superimposed on the flow word grid. Center & right: positive and negative elements of the div operator, respectively.

the trained SVM. For a valid classification, $A$ and $B$ must have one sign, and $C$ and $D$ another. We combine the SVM scores as $A + B - C - D$, and this should give a positive score for forwards clips, a negative score for backwards clips, and some chance to self-correct if not all of the component scores are correct.

The $C$ parameter for the SVM is chosen to be the value that maximizes the classification accuracy on the validation set over all three train/test splits. Once this $C$ parameter is fixed, the whole set of 120 training videos, including the previously-withheld validation set, is used to learn a final SVM for each train/test split.

Testing proceeds in a similar manner to that used for the validation set: the SVM scores for the four bag-of-words representations for each testing video are combined as $A + B - C - D$, and the sign of the score gives the video's overall classification.

## 2.1.4 Experimental results

We first show how well our models can predict the arrow of time on two video datasets. Then, for the bag-of-words model, we visualize its confident prediction regions and the motion clusters that the model learns to use.

**Classification accuracy** On the TennisBall13 dataset, we only train and test motion consistency method and bag-of-words method. Due to the strong arrow of time signal in the dataset, both methods predict correctly 12 out of 13 times. Note that the
### Table 2.1: Arrow of time classification results on the TennisBall13 and TA180 dataset.

(a) Test on TennisBall13

<table>
<thead>
<tr>
<th>Model</th>
<th>MC</th>
<th>FW</th>
</tr>
</thead>
<tbody>
<tr>
<td>leave-one-out</td>
<td>92.3%</td>
<td>92.3%</td>
</tr>
</tbody>
</table>

(b) Test on TA180

<table>
<thead>
<tr>
<th>Split/Model</th>
<th>SoE</th>
<th>MC</th>
<th>AR</th>
<th>FW</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>57%</td>
<td>70%</td>
<td></td>
<td>90%</td>
</tr>
<tr>
<td>B</td>
<td>48%</td>
<td>73%</td>
<td>58%</td>
<td>77%</td>
</tr>
<tr>
<td>C</td>
<td>60%</td>
<td>73%</td>
<td></td>
<td>75%</td>
</tr>
</tbody>
</table>

(a) the accuracy on TennisBall13 is high due to its small number of videos (b) we compare all four methods with 3-fold cross-validation on TA180: spatial-temporal orientated energy (SoE), motion consistency (MC), auto-regressive (AR), bag-of-words (BoW)

misclassified videos by both methods are different.

On the TA180 dataset, we compare all four methods shown in Table 2.1b.

**Energy-based (SoE).** The relatively poor performance of the Spatial-temporal orientated energy method can be attributed to the difficulty in generalizing motion over different sub-regions, and the fact that these features were designed to characterize regular motion textures, rather than particular one-off dynamic events.

**Entropy-based (MC).** While the accuracy of the motion-consistency approach is weaker, it is at the same time computationally much simpler and quicker to compute than the bag-of-words method. There is also the chance that it may complement the capabilities of the bag-of-words method well, because the motion-consistency considers spatial location of motion, and comparison of motion between neighboring frames, whereas the bag-of-words method at present considers only snapshots of motion for each frame separately. Of the 50 videos that this system misclassified, only 10 overlapped with the misclassified set from the bag-of-words method.

**Causality-based (AR).** For the Auto-regressive method, there is a trade-off between the desirability of using a large gap $\delta$ in the p-value test, and requiring many valid trajectories per video must and the need not to reject excessively many videos.

For large values $(\delta, N) = (0.24, 13)$, only 5 videos are accepted, but 4 are classified

---

1 No training is needed and we directly test on all video clips.
correctly (80%). For lower parameter values, \((\delta, N) = (0.15, 4)\), 101 videos are accepted, of which 58 are correctly classified (58%). Figure 2.7 shows more of this trade-off of between accuracy and number of videos classified.

**Statistics-based (BoW).** Of the videos that were misclassified by the bag-of-words method, a few have lots of individuals moving separately in the shot: groups of people or animals, or a large number of small balls, which might not be well-suited for the scale of feature we are using. A further three were stop-motion films of seeds growing, which tend to have jerkier motions than most normal videos.

**Analysis** We can use the trained bag-of-words model to look at which flow words were particularly characteristic of forwards or backwards motion. This was achieved by taking the SVM weights fixed using the training set and applying them to the descriptors for the test videos in their native playing direction. We then record the 10 words (0.25% of the total vocabulary size) contributing most positively to the scores for forwards-direction clips, or most negatively for the backwards-direction clips.

Figures 2.8 and 2.9 show the bag-of-words results on two of the test cases. Each of these figures is split into four parts: a synopsis of the video motion, a plot showing where in the video clip the 10 most informative flow words for classification originated (out of 4000 possible flow words from this vocabulary), the most-informative frame marked with all the instances of any of these 10 highly-weighted words, and then finally the top four most informative flow words for the given video clip.
Figure 2.8: Video from the test set, correctly classified as being backwards in time. The most informative point in the video is the few frames around 34 (shown), where there is an inverse splash of water. Red dots here mark the locations of all strongly-backwards flow words. This type of motion (shown in the four flow word grids) was learned to be strongly indicative of a backwards time direction.

In Figure 2.8, the example comes from one of the videos which was uploaded to YouTube after being time-reversed, and the synopsis images show a boy emerging from a pool with an inverse-splash. The motion of the water in this splash is the strongest indicator in this clip that time is running in the backwards direction. In Figure 2.9,
Figure 2.9: Video from the test set, correctly classified as being forwards in time. The most informative forwards-time words refer to the motion of the billowing steam, and occur periodically as the steam train puffs. Red dots marked on frame 224 show the locations of the most-forwards 10 flow words in this frame.

time runs in the forwards direction as usual, and the synopsis shows a steam train gradually approaching the camera. The gentle upwards divergence, arising from the plume of steam from the train, is a good indicator of Time's Arrow, and interestingly, the most useful words for capturing this appear periodically in the video clip as the
rhythm of the train motion causes the fumes to billow in a regular way.

Overall, there is a very slight tendency for a flow word describing forwards-time videos to have a higher divergence score than those describing backwards videos. Correlation scores between model weight and flow word divergence are small but consistent, with models learned on the three splits having correlation coefficients of 0.0329, 0.033 and 0.038 respectively.

2.1.5 Discussion

We have addressed a new and very fundamental problem of telling which direction time is flowing in a video, and presented four complementary methods for determining this direction. Our results indicate that the statistics of natural videos are not symmetric under time reversal: for if they were, then for each video, there would be an equally likely video where time is reversed, so there would be no way to solve the classification problem better than chance (assuming our training sample is unbiased / representative of the prior).

Of course, each method can be improved. For example, the statistics-based bag-of-words method performed well above chance on this video set, but its performance can doubtlessly be improved by extensions to use multi-resolution in both space and time. State of the art features developed for other tasks with a temporal dimension, such as actions and activity recognition, could also be employed [19, 64]. Also, the causality-based auto-regressive method can be extended to longer temporal intervals – for example to capture events that are causal but delayed, e.g. a tree branch oscillating some time after a person has brushed past it.

More generally, we have deliberately concentrated on low-level statistics in this work, but priors and classifiers can also be developed for high-level semantic information (like the fact that animals move head-forwards) to complement the low-level analysis.

2.2 Learning the Arrow of Time

In the previous section, we made an initial attempt to correctly classify and identify visual cues for the arrow of time in videos. To deepen the understanding, we ask the following three questions: 1) How to improve the model to handle large video dataset with automatically learned visual features? 2) What does the model learn in order to
solve the arrow of time classification, the real arrow of time or other artificial signals (e.g. camera motion)? 3) How to apply the learned arrow of time classification model for other computer computer vision tasks?

To address the first question, we train a ConvNet model with arrow-of-time competency by exploiting the large number of recorded videos available online, and let the data determine which visual features to use (section 2.2.2). This model has been designed taking into account the recent developments in ConvNets for human action recognition, and also in visualization – since we wish to visualize what has been learnt to enable time's arrow to be determined. Compared to the work in the previous section using hand crafted features trained on hours of video, we are now able to benefit from deep learning networks trained on thousands of hours of video.

For the second question about the visual cues learned by the classification model, we carefully design three step-by-step experiments. We begin by testing the model on a controlled simulation dataset (section 2.2.3), where we know the real arrow of time cues and we can test if the model learns them. However, to reach the point in being able to train a ConvNet from natural videos requires care in avoiding artificial cues to the direction of time, for example, based on time-asymmetric video compression schemes or on prototypical camera motions selected by directors. We investigate which video pre-procession are necessary to avoid such artifacts (section 2.2.4). Lastly, we move onto real data for training, using a 150k subset of the Flickr dataset [59], evaluating test performance (section 2.2.5).

In terms of applications in the third question, we show how the arrow of time signals can be used in other computer vision tasks like video feature learning, video forensics and action understanding (section 2.2.6). For video feature learning, we follow the standard procedure to start from our trained model for arrow of time classification and fine-tune it on the action recognition task. For video forensics application we detect clips of films that were recorded, then temporally reversed for display. This may be done for special visual effects, or to make otherwise dangerous scenes safe to film. We show competence at this task in a newly collected dataset of films containing time-reversed components, and also visualize the cues in the videos that the network uses to make the classification. More generally, this application illustrates that the trained network could provide a tool to detect videos that have been tampered with in this way. For action understanding, we focus on “talking heads” to see if any spoken words
are visually symmetric in time.

### 2.2.1 Related Work

Several recent papers have used temporal ordering, in some manner, rather than temporal direction. Basha et al. [5, 10] consider the task of photo-sequencing – determining the temporal order of a collection of images from different cameras. Others have used the temporal ordering of frames as a supervisory signal: for learning an embedding in [46]; for self-supervision training of a ConvNet [15, 41]; and to construct a representation for action recognition [14]. With the exception of Pickup et al. [43], none of these previous works address the task of detecting the direction of time. [43] explores three representations for determining time's arrow in videos: asymmetry in temporal behaviour (using hand-crafted SIFT-like features); evidence for causality; and an auto-regressive model to determine if a cause influences future events.

In terms of ConvNet architectures, we borrow from recent work that has designed ConvNets for action recognition in videos [28, 54, 65, 66], particularly those that have used optical flow to explicitly capture motion information [54]. We also employ the CAM visualization of [73].

### 2.2.2 ConvNet Architecture

In this section, we describe the model to use for the arrow of time binary classification. In order to concentrate on the temporal aspect of the video, we only use optical flow as input, but not its RGB appearance. Below, we first introduce the intuition of the architecture, and then describe implementation details.

**Model design** Our aim is to design a ConvNet that has an extended temporal footprint and that also enables what has been learnt to be visualized. We also want the model to have sufficient capacity to be able to detect subtle temporal signals. To this end, we base the model on three prior ConvNets: the VGG-16 network [55] as the backbone for the initial convolutional layers, this gives sufficient capacity; the temporal chunking in the model of [13] to give an extended temporal footprint; and the class activation map (CAM) model of [73] that provides the visualization.

Our “T-CAM” model is a direct temporal extension of [73]: we start from the CAM model which is effective to localize the discriminative signal by replacing fully-connected
Figure 2.10: Illustration of the temporal class activation map network (T-CAM) for arrow of time (AoT) classification. Compared to the traditional VGG architecture for image recognition, (b) we first concatenate the conv5 features from the shared convolutional layers, (c) and then replace the fully-connected layer with convolution layers and global average pooling layer (GAP) [37, 57, 58, 73] for better localization.

Implementation details In terms of the model, we start from a “conv1-modified” VGG-16 model for motion classification [65], where convolutional layers before conv5 are initialized from the VGG-16 model pre-trained on ImageNet. Then we add three more convolution layers with size 3×3×1024, stride 1×1 and pad 1×1 before the GAP layer, and add BatchNormalization layers [22] after each convolution layer.

We adopt many good practices for video classification described in [65]. We use TV-L1 [70] for optical flow extraction, and we do fixed five-corner cropping and horizontal flipping for data augmentation during both training and test. The final classification score for one video is given by averaging the classification scores of 25 evenly sampled group of frames with the augmentation above.

Additionally, specific to our arrow of time classification task, many videos may not
be indicative due to the small magnitude of flow value or their symmetry in time, which can add noise to the training. We use simple statistics (e.g. the flow magnitude) to remove such videos from the training set.

2.2.3 Learning from Simulation Videos

As an initial evaluation of the T-CAM model, we first avoid the compounding factors in real world videos (e.g. temporal codec or sample bias) and turn to graphic simulations where we have full control of the physics. We choose to simulate a simple world, the three-cushion billiards game (Figure 2.11a), where the principal signals for the arrow of time are: rolling friction and energy loss at collisions/bounces. We then apply the T-CAM model, trained on the simulations, to sequences of real billiards games.

Three-Cushion Dataset

We creat the following two versions of the dataset.

SIMULATION VIDEOS. We extend the physics engine in [18] to handle multiple balls with friction and collision damping. The parameters (e.g. size, mass) of the simulation are chosen to match those of the real videos (Figure 2.11b-c). For a given billiard system, we simulate 5k videos (100 frames each) and do 70-30 split for training and testing.

REAL WORLD VIDEOS. We download the video recordings of the first three qualifying games from the 2016 USBA National Championships from Youtube [2]. 167 individual shots (around 200 frames each) are extracted. As the cameras are placed at different angles, the perspective projection may cause the ball to appear to move faster as it comes towards the camera. To avoid this artifact, we warp the pool table images into a canonical overhead position, the same as that of the simulator. This dataset is only used for testing.

Experiments

Below, we test the T-CAM model for three tasks: the first two on the simulation data; and then on the real videos. We compare the T-CAM model with two different configurations, T=1 and T=2, with 10 consecutive optical flow maps as input. The flow for simulation is calculated from the ground truth position of the ball, which can be erroneous during collision. The flow for the real video is estimated by a KLT tracker [53], so that we can avoid the undesirable motion signals from the surroundings (e.g. human motion, lighting change).
Figure 2.11: The 3-cushion billiard dataset. (a) Original frame from a 3-cushion video; (b) the frame warped (with a homography transformation) to an overhead view of the billiard table; and, (c) a simulated frame to match the real one in terms of size and number of balls.

EXP 1: NO SIGNAL. To make sure there is no artificial signal for time's arrow in the simulation pipeline, we test the models on a world without an arrow of time: no friction, no collision damping, and uniform distribution of initial positions and velocities. The results are given in Table 2.2. As expected the models perform at chance level.

EXP 2: CONTROLLED AoT SIGNAL. Now, we add the known arrow of time signal, collision damp and friction, one by one and see how well T-CAM model learns for each case. Shown in Table 2.2, we find both T-CAM models perform well for the collision damp case, where the loss of energy is evident as an instant event. However, for friction, T-CAM model with T=1 has trouble to learn such weak signals, whose effects are accumulated over a period of time. To qualitatively understand how the T-CAM model (T=2) learns these two signals differently, we uniformly sample across each video and plot the test accuracy over time (Figure 2.12). For each case of collision damping and friction, we pick three videos with different initialization. We plot their kinetic energies over time and color it with the test accuracy.

EXP 3: GENERALIZABLE REPRESENTATION. The goal is to train a model from simulations that works on the real world “3-Cushion dataset”. We first estimate the physical parameters of the real videos through the trajectories of the balls. This gives a rolling friction coefficient of $\mu = 0.5$, and a collision damping factor $\eta = 0.5$. With these
Figure 2.12: Visualization of test accuracy for sampled billiard ball videos. (a) For the system with only collision damping as the arrow of time signal, the model learns to detect the time of sudden drop of energy and becomes unconfident otherwise; (b) For the friction signal, however, the model is barely confident most of the time due to the weak signal.

parameters, we generate simulation videos to train the model, and then test on the real sequences directly (there is no domain adaptation). Table 2.2 shows that temporal fusion (i.e. $T=2$) helps the most when the signal is weak (i.e. friction) and the collision damping is a stronger signal. Note, there are frequent collisions, and so collision damping makes a significant contribution to the time asymmetry. For the collision damping, the T-CAM model effectively capture the collision event and the accuracy is close to chance for other places. In contrast, the T-CAM model is barely confident about the friction signal. The irregular drop of the energy results from collisions when we cannot recover the ground truth velocity based on the change of position.

### 2.2.4 Avoiding Artificial Cues from Real World Videos

Many real world videos available online have artificial signals introduced during production or compression (e.g. motion encoding), which can provide cues to indicate the video's temporal direction. A learning-based algorithm may cheat and solve the arrow-of-time task using such artificial cues.

Below, we investigate three artificial signals from different stages of video production that are common in videos: camera motion, black framing, and temporal codecs. For the
Table 2.2: Test accuracy on the simulated 3-billiard datasets and the real video. We compare T-CAM model with either one (T=1) or two (T=2) temporal segments for three different experiments. (1) For no signal case (None), the test performance is at the chance level; (2) Both models perform well when there is only collision damping in the system (Col.), but temporal fusion on conv5 helps to learn weak signals like friction (Fric.). (3) The simulation Col.+Fric. is made with a similar system configuration to the real videos (3-cush.), where the collision damping signal dominates. Thus, both models have similar performance.

<table>
<thead>
<tr>
<th></th>
<th>Exp 1</th>
<th>Exp 2</th>
<th>Exp 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>T-CAM</td>
<td>None</td>
<td>Col.</td>
<td>Col.+Fric.</td>
</tr>
<tr>
<td>T=1</td>
<td>50%</td>
<td>90%</td>
<td>94%</td>
</tr>
<tr>
<td>T=2</td>
<td>50%</td>
<td>98%</td>
<td>85%</td>
</tr>
</tbody>
</table>

Figure 2.13: Illustration of artificial signals from videos in UCF101 dataset. (a) The black framing of the clip has small non-zero intensity value, and a vertical slice over time displays asymmetric temporal pattern. (b) After training on the UCF101 test data, there are clusters corresponding to camera zoom-in.

black framing and camera motion signal, we design experiments on the popular video dataset UCF-101 [56], where 46% videos have black framing, and 73% of videos have camera motion. To study the temporal codec signal, we have to collect a video dataset without temporal codec compression.

**MJPEG Arrow of Time Dataset (MJPEG-AoT)** To avoid codec artifacts we introduce a new dataset (16.9k shots from 3.5k videos) for which there is no motion coding. We ensure this by only using the MJPEG and ProRes codes, for which each frame is
compressed independently. We can then evaluate performance with and without codecs by using the original frames or the frames extracted after the video has been compressed using a codec.

Unlike Youtube, where videos are mostly H.264 compressed, Vimeo hosts many professional videos in a variety of original formats, which are known. To download videos without temporal codec compression, we search on Vimeo with keywords such as “mjpeg”, “prores”, and “cannon+raw” etc. We verify the codec of the downloaded video with “ffmpeg” and obtain around 7,000 videos.

**VIDEO PRE-PROCESSING.** For pre-processing the videos, we use the following settings in sequential order:

- **Black frame removal.** For each video, we select five uniformly spaced frames and compute the mean rgb value for each row and column. To remove the black frame, we find the first or last consecutive rows or columns whose mean value are less than our manually set threshold.

- **Shot detection.** We compute the frame difference for each video and save the mean rgb difference for each frame. Then, we select 41-frame-long clips whose mean frame differences are not 0 (e.g. static frames) and are not large (e.g. shot transition or fast camera motion).

- **Clip stabilization.** For each 41-frame-long clip, we first compute the homography between each frame and the central frame independently, and then smooth the estimated homography with outlier rejection. We stabilize each clip with these estimated homographies.

- **Human selection.** As a final check on quality, Amazon Mechanical Turk (AMT) is used to identify clips with either black stripes, multiple-shots or an unstabilized camera. Then, we manually refine the selection for the final round.

**Experiments** Below, we examine artificial signals that can affect the arrow of time classification. We train the T-CAM model on two versions of the same dataset, one original (A) and one with identified artificial signals removed (A*). If the model trained on A has significantly better test result on A than A*, then it is likely that this model learns to rely on artificial signals for predictions. If both models have similar test
BLACK FRAMING. Black bars present at the boundary of many videos may not be completely black after video compression (Figure 2.13a). The resulting small image intensities can cause different flow patterns for forward and backward temporal motion, providing an artifactual cue for the arrow of time.

Given a video with black framing, we can remove the artificial signals in one of two ways: zero out the flow values in the black bar region or crop out the black bar region. Given the three versions of the UCF-101 dataset, we train and test our T-CAM model for all nine possible combinations of train and test sets (Table 2.3a).

Consistently, the model trained on “original” data has significantly higher test accuracy on “original” test data, implying that the black framing is significantly contributing to the video direction classification. In contrast, all three models have similar test ac-
Table 2.3: An examination of the artificial cues for arrow of time prediction. We show results of controlled experiments on the effect of black framing and camera motion on UCF101.

Accuracies on either version of the modified data, suggesting both removal procedures are effective to avoid black frame signals. Thus, to avoid learning the artificial signals from black frame, we can either zero-out or crop-out the corresponding flow maps regions.

The test accuracy of the AoT classification drops from 98% to around 90% if we redo the experiment after either cropping out black bar regions or zeroing the flow values in the black bar regions. These steps can remove this artificial AoT cue.

Camera motion. Around 73% of the UCF101 videos have camera motion. We first train our model the UCF101 dataset with all information removed from the black bar regions. Then we cluster the GAP feature for the most confidently classified test clips. Consistently, we find the clustered clips share similar camera motions (e.g. zoom-in in Figure 2.13b or tilt-down), suggesting that our model learns to exploit cinematic camera motion conventions as an artificial cue to classify the AoT signal.

Out of the 13.3k videos from UCF101, we select 9.6k videos which can be well-stabilized within a chunk of 41 frames. To avoid the effect of black framing, we crop out the black bar region of the videos. We train and test on original videos and the stabilized ones (Table 2.3b).
Table 2.4: Effect of temporal codec on MJPEG-AoT dataset. We train and test on three versions of the MJPEG-AoT dataset and our model has similar performance. Thus, the added codec signal doesn’t introduce significant artificial signals for our model to capture for the arrow of time.

The T-CAM model trained on “original” videos has significantly worse performance on the stabilized videos, where camera motion cues (e.g. zoom-in) no longer exist. On the other hand, both models have similar test accuracy on the stabilized videos, suggesting the stabilization method is effective to cancel out the camera motion bias. Thus, to avoid learning the artificial camera motion bias, we can stabilize the videos.

**INTER-FRAME CODECS.** For efficient storage, most online videos are compressed with temporally-asymmetric video codecs, i.e. inter-frame codecs. Forward and backward “frame prediction” are often employed, which may offer an artificial signal for the direction of time.

For controlled experiments, we want to train and test our model on videos before and after inter-frame codec compression. Thus, we have to resort to inter-frame codec free videos from our collected MJPEG-AoT dataset, as it is almost impossible to reverse the inter-frame codec compression for videos from other large scale video datasets. In addition, we cropped out the black frames and stabilized the camera motion to single out the inter-frame codec signal. Starting from the original videos from the MJPEG-AoT dataset, we encoded them with the popular H.264 codec in either the forward or backward direction, to simulate the corruption from the inter-frame codec. Hence, we created another two versions, referred as “H.264-F” and “H.264-B” for forward and backward direction compression respectively.

In Table 2.4, we show results of our T-CAM model trained and tested on one of the three versions of the dataset, nine different combinations in total. After removing black
Sec. 2.2. Learning the Arrow of Time

bar and camera motion, the inter-frame codec H.264 does not introduce a significant signal that can be learned by our model. This finding offers a procedure for building a very large scale video dataset starting from videos that have been H.264 encoded, without being concerned about artificial signals.

2.2.5 Learning from Real World Videos

After verifying the model on the simulation videos and removing the known artificial signals from the real world videos, now we are ready to examine visual cues that the model learned to determine the arrow of time.

Flickr Arrow of Time Dataset (Flickr-AoT) To use a large amount of unlabeled video to train our model, we find a subset of videos satisfying our criteria from the Flickr dataset collected in [59, 62]. Starting from 1.7M videos, we first remove the black framing and find clips that can be well-stabilized. In the end, we harvest around 150k clips, which is an order of magnitude bigger than MJPEG-AoT. The dataset pruning procedure is the same as the MJPEG-AoT dataset.

Experiments We first report classification results on the Flickr-AoT dataset with different configurations of the T-CAM model. Then, to interpret the cues learned by the T-CAM model, we visualize the localization results of the arrow of time. To demonstrate that the T-CAM model learns about the generic arrow of time, we also test on two other arrow of time datasets. For comparison, we not only implement the state-of-the-art method [43], but also conduct human experiments using Amazon Mechanical Turk.

Test on Flickr-AoT. In Table 2.5a, we compare the T-CAM model with different number of temporal segments, number of input flow maps and the overlapping ratio between adjacent temporal segments. The best configuration is two temporal segments with 10 frames each without overlapping, which has test accuracy 79%.

Localization on Flickr-AoT. For spatial localization, we follow the procedure for the CAM model [73]. In Figure 2.15, we use the "jet colormap" to indicate the probability, $p$, of being forward (red, close to 1) or backward (blue, close to 0) and we only overlay the region where T-CAM is confident (i.e. $|p - 0.5| > 0.2$). We compute the flow for two consecutive frames in the middle of the clip. The black arrows indicate
the direction and the magnitude (normalized by the largest value) of the flow on the sparse grid, and they are only shown for the colored regions. All the clips are correctly predicted by T-CAM and the clip forward/backward label is indicated by the dominant color of the heatmap.

**Test on other AoT datasets.** We compare our model performance on two other AoT datasets: the MJPEG-AoT and TA180 [43].

<table>
<thead>
<tr>
<th>#temporal segment</th>
<th>T=1</th>
<th>T=2</th>
<th>T=4</th>
</tr>
</thead>
<tbody>
<tr>
<td>overlap ratio/#frame</td>
<td>10</td>
<td>10</td>
<td>20</td>
</tr>
<tr>
<td>0%</td>
<td>65%</td>
<td>69%</td>
<td>71%</td>
</tr>
<tr>
<td>50%</td>
<td>N/A</td>
<td>68%</td>
<td>73%</td>
</tr>
</tbody>
</table>

(a) Test on Flickr-AoT

<table>
<thead>
<tr>
<th>Method</th>
<th>MJPEG-AoT</th>
<th>TA180</th>
</tr>
</thead>
<tbody>
<tr>
<td>[43]</td>
<td>N/A</td>
<td>81%</td>
</tr>
<tr>
<td>Human (AMT)</td>
<td>61.7%</td>
<td>79.4%</td>
</tr>
<tr>
<td>T-CAM (raw)</td>
<td>52.4%</td>
<td>56.1%</td>
</tr>
<tr>
<td>T-CAM (raw)</td>
<td>62.4%</td>
<td>82.5%</td>
</tr>
</tbody>
</table>

(b) Test on other AoT data

Table 2.5: Test accuracy of the T-CAM model trained on the Flickr-AoT train dataset. (a) we first compare different parameters of the T-CAM model on the Flickr-AoT test dataset. (b) Other datasets: the MJPEG-AoT and the TA180 of [43]. T-CAM (raw) is trained on a larger set of videos from Flickr without pre-processing.

MJPEG-AoT serves as the gold standard as it is a clean dataset without temporal codec pollution. Table 2.5b shows the classification results. For training on the Flickr-AoT dataset, there is a significant improvement with the artificial signals removed, achieving similar performance to the model trained on MJPEG-AoT.

On the TA180 dataset [43], despite the fact that our model is trained on a different dataset (Flickr-AoT), without any fine-tuning we still achieve about 76.2%, which is

---

2Unlike T-CAM which is trained on Flickr-AoT and test on TA180, [43] performs 3-fold cross-validation on TA180.
Figure 2.15: Heatmaps of correct predictions from the Flickr-AoT dataset. We use the "jet colormap" to indicate the probability of being forward (red, close to 1) or backward (blue, close to 0) and we only overlay the region where T-CAM is confident. We compute the flow for two consecutive frames in the middle of the clip. The black arrows indicate the direction and the magnitude (normalized by the largest value) of the flow on the sparse grid, and they are only shown for the colored regions. Our network learns to focus on the regions that inform about the AoT in both (a) general and (b) human scenes. (a) The network utilizes basic physics (e.g., water falling due to gravity) and video biases (e.g., people film trains coming towards them more often than trains leaving from them) to correctly decide the AoT. (b) When there is a human in the scene, the network decides the AoT by observing human actions.

comparable to the state-of-the-art performance of 81%. To estimate the difficulty of the task, we also conduct an Amazon Mechanical Turk study. As shown in Table 2.5b, the model achieves similar performance to humans.
2.2.6 Applications

In this section, we show three applications of the arrow of time signal: video feature learning, video forensics, and action understanding.

**Video Feature Learning: Action Recognition**  To evaluate the learned video feature from various vision tasks [15, 38, 41], UCF-101 dataset is used as a popular test bed, where the original model is fine-tuned for action recognition. In the same manner, we directly take the T-CAM model trained on Flickr-AoT dataset from scratch. On the UCF-101 training data, we fine-tune our model with action labels for three different sets of layers separately: the last layer, all layers after temporal fusion and all layers.
Sec. 2.2. Learning the Arrow of Time

In previous comparisons [15,38,41], each vision task is only listed with its test accuracy, not controlling the input data or architecture of the model or number of layers to fine-tune. In Table 2.6, we provide a systematic comparison by listing out the control parameters and have the following four observations.

1. For the input data, video features learned from flow are superior to those learned from rgb images or dynamic RGB (D-RGB). When fine-tuned for all layers, even a random network with flow input [54] out-performs all previous feature learning methods with rgb input.

2. For the same input data, the choice of architecture can greatly improve the result. For the rgb input data, recent feature learning methods [38,41,67] with AlexNet or FCN architecture hardly beats random network with AlexNetM architecture.

3. Initialization from image classification model on ImageNet performs much better for rgb input than flow input. As discovered in network understanding works [72], ConvNet models learn object detectors, e.g. human head and car wheels, in the convolutional layers, it's not surprising that these features can be directly used for action recognition [54]. However, the flow input has different statistics from that of rgb images and direct initialization from it may not achieve good result, except fine-tuning all layers [65].

4. With the flow input, our arrow of time task yields better video feature than that from ImageNet models if only retrain classification layers. When fine-tuning all layers, the test accuracy doesn't differ significantly among different initialization methods.

**Video Forensics: Reverse Film Detection**  
Reverse action is a type of special effect in cinematography where the action that is filmed ends up being shown backwards on screen. Such techniques can not only create artistic scenes that are almost impossible to make in real life (e.g. broken pieces coming back together), but also make certain effects easier to realize in the reverse direction (e.g. targeting a shot precisely). With careful examination, humans can often detect such techniques, as the motion in the

---

4Either the same architecture or with slight modification
4compared to our implementations, [65] uses more extensive data augmentation methods.
### Table 2.6: Feature learning results for action recognition on UCF-101 split-1.

We list out important factors besides the network initialization methods: input, architecture and layers to fine-tune. For the flow input, our initialization from arrow of time prediction is consistently better than the previous state-of-the-art ImageNet initialization [65] when only the fc layers are fine-tuned.

---

**DATASET.** We collect the Hollywood-AoT dataset, containing clips from Hollywood films, which are displayed in reverse deliberately. Thanks to the “trivia” section on the IMDB website, shots that use reverse action techniques are often pointed out by the fans as Easter eggs. With keyword matching (e.g. “reverse motion”) and manual refinement on the trivia database, we collected around 76 clips from 25 popular movies, including ‘Mary Poppins’, ‘Brave Heart’ and ‘Pulp Fiction’. Notably, ‘Top Secret!’ has a 1.5-minute long single shot of reverse action!

**CLASSIFICATION RESULTS.** The overall test accuracy is 59.2%, where human perfor-
Sec. 2.2. Learning the Arrow of Time

Figure 2.17: Heatmap results on the Hollywood-AoT dataset. (a) In these successful cases, our model utilizes basic physics (i.e., gravity) and human expressions to correctly decide the AoT, whereas (b) in some cases, the model fails to discriminate that the video is played backwards. Although the model fails overall in the train sequence, it extracts the correct AoT from the train steam.

Performance (using Amazon Mechanical Turk) is 72.3%. With the T-CAM model, we visualize both successful and failure cases with the heatmap score of being backward in time (Figure 2.17). The successful cases are consistent with our earlier finding that the model learns to capture the cue of gravity. For the failure cases, some are due to rarely-seen motion patterns in real life (e.g. bullet motion) and some are due to the failure to capture more complex concepts (e.g. a precise shot). The clips are from (left to right): (a) ‘Mary Poppins (1964)’, ‘Anaconda (1997)’ and ‘Memento (2000)’; (b) ‘Brave Heart (1995)’, ‘Mary Poppins (1964)’ and ‘The Railway Children (2000)’.
Figure 2.19: Common cues for reverse film detection (cont. next page). For each reversely-played clip, we show its first, middle and last frame and its heatmap (blue for backward and red for forward) with normalized flow vector on the middle frame. Ideally, the heatmap should be blue, indicating the clip is played in the backward direction.
Figure 2.19: The T-CAM model consistently focuses on regions with (a) head motion, (b) motion against gravity, and (c) human body motion. The blue regions indicate that (a) global head motion and eye movement and (b) upward motion patterns of water and snow can be correctly recognized by the T-CAM model for backward AoT. For (c), the T-CAM model can be fooled by professional backward-acting (second row) and subtle motion (third row) where red regions are around performers.
ANALYSIS OF VISUAL CUES. To understand the common cues used by the T-CAM model to detect the reverse-playing movie clips, we find clips with high prediction confidence, cluster their $fc7$ feature and manually associate these clusters with interpretable cues. In Figure 2.19, we show three clips (all played in the backward direction) from each of three identified motion cues: human head, gravity and human body. For each 10-frame clip, we show its first, middle and last frame and its prediction heatmap (red for "forward" and blue for "backward") overlayed on the middle frame with normalized flow vector in the confident regions.

- **Human head cluster.** Shown in Figure 2.19a, the T-CAM model achieves high accuracy and the heatmaps are dominated by blue color in the head region. The first and the third row have global head motion while the second row shows eye movement.

- **Gravity cluster.** Shown in Figure 2.19b, the T-CAM model is correctly confident in the region where either water or snow is moving upward in certain patterns against gravity.

- **Human body cluster.** Shown in Figure 2.19c, the T-CAM model achieves around chance accuracy, where the heatmap color varies for different human motion patterns. Some motion is indicative to human for backward arrow of time (e.g. stepping backward motion of the man behind in the first row), while some motion can be subtle (third row). Notably, the second row features Charlie Chaplin’s performance which intentionally tries to make his backward motion seem natural and successfully fools our T-CAM model.

**Action Understanding: Talking Heads** It is interesting to consider whether human actions are visually reversible. For example, a person sitting down is aided by gravity, but standing up must work against gravity. This asymmetry is reflected in the muscle patterns and the temporal sequence of the body posture. As a case study, we take a very subtle action: talking heads, and investigate if it is possible to determine if a face is speaking backwards or forwards using visual information alone. That the direction can be determined will depend, of course, on what is spoken – some words or phrases will be the visual equivalent of a palindrome, and it will not be possible to tell the direction for these. Conversely, if we start with a character palindrome (like ‘racecar’), it does
not follow that this will be a visual palindrome, just as it does not follow that it will be an audio palindrome. Besides statistical understanding, detecting the asymmetry in talking head videos have clinical applications [50], where the detection rate is strongly correlated with the cochlear implant success rate.

DATASET. We use the recently released large-scale ‘Lip Reading in the Wild’ (LRW) dataset [9]. This has 1-second long video clips for 500 different words, with around 1000 examples for each word ‘spoken’ by hundreds of different speakers. We use the dataset’s training and test partitions, so that all tests are on unseen samples of 50 clips per word for the 500 word test set. The dataset provides a stabilized lip region (which we train and test on) as well as full faces. An example clip is shown in figure 2.20a.

MODELS AND RESULTS. We train a new T-CAM model on the LRW dataset, using the training procedure described in section 2.2.2. The performance is significantly better than that on generic videos (i.e. on Flickr-AoT), with the T-CAM model able to capture the arrow-of-time signal very well in this specific domain. Training on the entire training set gives a time’s arrow classification test performance of 97.6%. However, the model is actually able to learn from far fewer words than this – for example it can reach a performance of 83.2% when trained on as few as 10 words (meaning that it has not seen examples of the other 490 words at all).

We show the histogram of the test accuracy over different words (Figure 2.20b), where most have around 95% test accuracy. For the extremes, we find that the top five

2.2.7 Discussion

After studying and carefully removing artifactual cues to the arrow of time in online videos, we have curated a large-scale dataset of videos to use to train a powerful T-CAM model to determine the direction of time.

We obtain good discrimination if we first remove videos for which AoT discrimination is impossible for both humans and machines: videos with either no motion, or motion that is symmetric in time. After removing such videos from the training set, we can train an AoT classifier that performs at 79% on a large unlabeled video dataset.

The trained model lets us examine the visual cues learned to distinguish forward from backward time. Some of the AoT cues involve physical effects, such as waterfalls descending. Many exploit regularities of human action.

Of special interest is the Reverse Action dataset, of special effect time-reversed film clips. Humans can identify the direction of time correctly 72%, and our algorithm gives a test accuracy of 59% for that challenging dataset.

All the curated datasets will be publicly released for future research.

2.3 Conclusion

In this chapter, we propose a novel problem for the computer vision community, predicting and learning the arrow of time in videos, as an initial step towards understanding the temporal statistical structure of the visual world. We first show that we can encode our prior knowledge of physics into visual features for computers to use for arrow of time detection. Then with a large scale dataset and a ConvNet model, computers can learn the arrow of time from raw input, reveal artificial signals in videos for prediction and apply the learned feature and arrow of time to other computer vision tasks.
Chapter 3

Trends from Temporal Structure: Visual Evolution of Web Design

"Web users ultimately want to get at data quickly and easily. They don’t care as much about attractive sites and pretty design.”
— Tim Berners-Lee

We now investigate the visual temporal structure in the Internet world. Unlike real world objects governed by arrow-of-time related physics, web pages are heavily influenced by design trends of their time in terms of layout, font, and color scheme. Over the years, the web design trend has changed dramatically, from the animated gifs and marquees of the mid-1990s, to the embossed corners and drop shadows of the mid-2000s, to the flat minimalist pages of the 2010s. Understanding the design trends is helpful for designers to learn from the past, e.g. how information is organized and what color schemes have emotional impact.

In this chapter, we answer the question: “What makes a 2016 web page look like 2016?” Given web pages with the year they were created, i.e. temporal structures on the Internet, we use machine learning algorithms to visualize, colorize and classify web design trends.

Inspired by recent success in image understanding, e.g. object [31] and scene [71] recognition, we contribute to the web design community by introducing a large-scale temporal database of web pages and three deep learning models to harness the data complexity. In Section 3.2, we describe the curated “WebTrend21” database, consisting
of 21 years of web pages scraped from the Internet Archive\textsuperscript{1} to study the visual evolution of web design. Then in Section 3.3, we overview three aspects of trend understanding for web pages through mapping them into interpretable representations, re-colorizing them consistently with time and predicting their for year labels in Section 3.4–3.6.

\section*{3.1 Related Work}

To study design, recent data-driven approaches have contributed to design creation and evaluation for web pages, mobile apps and magazine covers. Kumar et al. [32] suggest a design mining platform with the goal of querying web design assets (e.g., banners) based on the html structure and graphical elements from more than 100,000 web pages. Reineke et al. [48] model aesthetics at first impression from a dataset of web pages, using two measures of visual complexity and colorfulness. Miniukovich and De Angeli [40] further model the aesthetics of graphical user interfaces based on two datasets of screenshots from web pages and mobile apps. Jahanian et al. [27] model association of colors and linguistic concepts (color semantics) by design mining from a dataset of 2,654 magazine covers from 1998 to 2013, for 71 titles and 12 genres using topic modeling. However, understanding the temporal aspect of data has not been the goal of these studies.

To study the trend of design over time, Ivory and Rodrick [25] evaluate the design evolution of more than 1,500 sites and 22,000 pages over the years of 2000, 2002, and 2003. They first define more than 150 quantitative measures over basic html elements (e.g., number of text lines, links, images), page formatting (e.g., use of interactive elements), performance (e.g., download speed), and site architecture (e.g., consistency of the pages) [26]. Then, they use these measures to classify the pages based on quality ratings (e.g., visual design and ease of navigation) and find several consistent design patterns for three years, confirming good design guidelines [24]. More recently, Alharbi and Yeh [3] suggest a design trend mining framework for analyzing design patterns in Android apps with a dataset tracking 24,436 apps over 18 months. Then, they define a list of features including app details (e.g., title and description of the app), visual appearance, and behavioral features, and use these features to classify their data. Through this analysis, they delineate design patterns for eight aspects of user experience.

\textsuperscript{1}https://archive.org/index.php
interface design (e.g., home screen widgets and navigation strategies). Another relevant work is Ginosar et al. [20] that studies the visual evolution of American high school yearbook photos. Through descriptive statistics and a classification task, they discover the defining style elements for a given decade, trends in fashion and social norms over time.

Compared to previous works, we here build a larger-scale database of web pages with wider range of years and analyze trend with the learned representation instead of hand-crafted measures. Also, beyond the extracted good design principles from trend analysis, we build novel applications useful for web designers for different stages of design process: inspiration, prototyping and evaluation.

3.2 WebTrend21 Database

In this section, we introduce our curated WebTrend21 database, containing screenshots of web pages in the past 21 years (1996-2016), containing 164,641 screenshots from the top 10,859 web domains (without adult-content domains) ranked by traffic. We first explain the construction and pruning of the database, then plot descriptive statistics of color and year labels, and create two benchmark datasets for later tasks.

3.2.1 Database Construction

To create a temporal dataset for the web, we retrieve web pages from the Internet Archive [4]. Previous scraped datasets from this archive are either headless, i.e. the Document Object Model (DOM) without the rendered page, or nonpublic screenshot datasets scraped for different goals than ours (for details, refer to the datasets and services available on the Internet Archive offered by individuals and private parties). We developed a scraper using the Selenium WebDriver\(^2\) and Java with the Firefox driver. The scraper takes a screenshot of the entire page (scrolling to load full pages) and downloads the page DOM. The scraper also captures some details about how graphical elements are laid out, e.g., locations of graphical elements (i.e. `<img>` tags) on the page while it is rendered.

To generate a list of web domains to scrape, we use the list of high-traffic urls

\(\text{http://www.seleniumhq.org/projects/webdriver}\)
according to the Alexa\(^3\) service. We then filter out the adult-content domains using Web of Trust\(^4\). For each domain, we use the Internet Archive API to obtain one screenshot per month (whenever a snapshot for that month is available) from 1996 to September 2016. Due to the non-existence of web pages on the Internet Archive server, our around 2 million requests \((10859 \times 12 \times 21)\) only retrieve around 500 thousand valid web page screenshots. In Figure 3.1, we show the tag cloud visualization of top-100 web domains in the WebTrend21 database.

### 3.2.2 Database Pruning

To study the design trend of web pages in their intended condition, we need to discard screenshots of web pages with either rendering errors or disqualified content from the crawled database. Rendering errors can happen at either server or browser side during the download process. For the server-side error, web pages can be unavailable when the Internet Archive attempted to snapshot them, e.g. 404 page not found (Figure 3.2a),

\(^3\) http://www.alexa.com
\(^4\) https://www.mywot.com/
Figure 3.2: Examples of bad web pages to prune. We remove web pages with rendering errors, either happened at the server-side during (a) capture and (b) retrieval, or at the browser-side during (c) loading. To remove web pages with disqualified content, we filter out (d) web pages that are no longer functional.

Figure 3.3: Histogram of the mean intensity of our web page dataset. We find that (a) the histogram has several sporadic peaks, (b) especially in the intensity range between 200 and 255. In the main text and Figure 3.4, we identify the cause of each of the four groups of peaks. After pruning, (c) the histogram varies smoothly with the mean intensity value.

or the snapshots are no longer stored on the Internet Archive (Figure 3.2b). For the browser-side error using our scraper, the screenshot image can be taken before the page is fully loaded where Internet Archive logos are displayed for replacement (Figure 3.2c). In terms of web page content, we discard web pages that are not functional, e.g. for sale (Figure 3.2d) or under construction. Also, to reduce the number of web pages to prune, we find for each domain the set of unique web page screenshots which are different from
Although it’s easy to specify the error source, it is laborious to thoroughly identify all error cases due to the large amount of data and the diversity of the errors. Below, we describe four approaches, complementary to each other, to semi-automatically prune the database.

**Http-based method** We discard web pages with images that are unable to be loaded from the Internet Archive server. To check the existence of missing images for a web page, we send separate requests to the Internet Archive server. With this simple approach, we reduce the number of screenshots from the initial 500k to 290k.

**Appearance-based method** When an error occurred during the crawling, certain visual templates will be recorded in the screenshot, e.g. logo of Internet Archive and the text stating “Error.” Thus, we can apply standard object detection techniques to find erroneous pages based on appearance. Specifically, we use the normalized-cross-correlation method for template matching [35].

But how to come up with these visual templates in the first hand? Besides eyballing
Figure 3.5: Examples of web pages identified through geometry-based method. (a) pages that are under construction tend to have big page margins, and (b) pages that are rendered without proper CSS files tend to have big page height.

randomly sampled web pages, we discover visual templates for erroneous pages through observations from the mean color histogram. We assume that the histogram of the web pages’ mean intensity (ranging from 0-255) should vary smoothly instead of jumping suddenly. However, as shown in Figure 3.3a, there are several sporadic peaks in the histogram indicating abnormally consistent visual patterns, mostly located in the high intensity region. We find the following visual patterns in the descending order of the mean intensity to explain the four labeled groups of peaks in Figure 3.3b.

**WHITE PAGES.** There are thousands of web pages with mean intensity rounded up to 255, which can be of minimal style of design or erroneous pages with error messages(Figure 3.4a). Empirically, we remove web pages with mean intensity over the threshold 254.5, indicating the lack of enough content on the web page.

**GOOGLE PAGES.** For the top 10k web domains where we crawl web pages, many are from Google with different top-level web domains from different countries, e.g. google.dk from Denmark. These pages may share similar visual designs and we run another round of duplication removal for all Google-related web pages (Figure 3.4b).

**BROWSER ERROR PAGES.** Due to network problems, our crawler may occasionally lose connection to the server, returning the browser-side error pages with similar visual template e.g. “server not found” and “connection reset” (Figure 3.4c).

**SERVER ERROR PAGES.** Sometimes, the web page snapshot is no longer available on the
Figure 3.6: Examples of web pages identified from their html source code. (a) If the html source code does not have any body text, errors like “page not found” or “unfinished loading” can happen. Two common errors can be found through matching the keyword of (b) “Index of” and (c) “sorry.”

Internet Archive server. Thus we obtain the default server-side error page. Instead of matching the whole page, we choose to create the template from the “Wayback” logo, which also shows up in various situations (Figure 3.4d).

After the removal of the erroneous pages above, we get a smoothed version of the mean intensity histogram (Figure 3.3c).

Geometry-based method The size of the web page screenshots and their margins are also useful to find error web pages to remove. Apart from the consistent erroneous page templates above, many visual templates only appear for a few times, making it less efficient to build one detector for each one. For example, for pages under construction, different web domains tend to have their own versions. Thus, instead, we make use of the page size and the content size of the screenshot to find potential unqualified web pages for later manual labeling. Specifically, we have the following two observations: pages under construction tend to have big margins (Figure 3.5a); pages without proper CSS layout tend to be long in height (Figure 3.5b). In terms of implementation, we compute the margin size by thresholding the standard deviation of the pixel value of each row or column.

Text-based method Besides the visual information from the screenshots, the html body text extracted from the web page source files can also indicate if errors occurred. With the scraped html body text, we match the following three types of keywords to find web
Figure 3.7: Distribution of data over years and distribution of colors over years. Magnitudes are in a logarithmic scale. The color space is a discretized version of sRGB with 4 bins per channel, and sorted based on hue.

pages which are hard to spot by the above two methods to prune manually. Specifically, we first find web pages whose body text is empty, indicating that error might occur when it was archived (Figure 3.6a). Then we search keywords for “Index of,” where the web page URL leads to the server folder instead of rendering the html file (Figure 3.6b). Finally, we search for keywords, e.g “error”, “sorry” and “construction”, for web pages whose design is not ready yet (Figure 3.6c).

### 3.2.3 Database Statistics

To understand the data distribution of the database, we plot two histograms over year labels and dominant color hues.

Figure 3.7a illustrates the number of screenshots by year in our database. Although we try to balance the number of web pages for each year by uniformly sampling each web domain in time, web pages in early years are not many to start from and many do not meet our pruning criteria, yielding the skewing distribution.

Figure 3.7b illustrates the histogram shows the log counts of colors over all years. Given a web page screenshot, we count 64 different colors from the discretized sRGB color space with 4 bins for each of the 3 channels. We can see that black and white are the mostly used colors, followed by blue yellow and red, while green and purple are less used.
Figure 3.8: Benchmark preprocessing. In height, we crop out the top 729 pixels for the first impression, avoiding over-compressing (a) long scrolled-down pages. In width, we crop out (b) one-sided margin due to the html layout not adapting to the browser window width and keep (c) two-sided margin by design.

### 3.2.4 Benchmark Construction

Here we describe two subsets of WebTrend21 database used as benchmarks with different design purposes, WebTrend21-130k and WebTrend21-50k.

**Preprocessing** Although web pages are scraped with the same settings from the browser window with the size of 729×1184 (close to 10:16 computer screen size ratio), some have bigger height due to scroll-down pages (Figure 3.8a) and some have wide one-sided margins due to its inability to adapt to the browser window size (Figure 3.8b). To standardize the web page sizes reflecting their intended designs and enabling comparison with one another, we use the following preprocessing methods.

For height, we crop out the top 729 rows of each screenshot, corresponding to the first impression of web pages, i.e. initial rendering from the web browser without scrolling down. For width, we compute the standard deviation of each column of pixels to find columns that are almost constant in color as page margin region. To distinguish between the improper margin display (Figure 3.8b) and the intentional margin (Figure 3.8c), we compute the difference between the left and right margin size and crop it to have the same margin if the difference is beyond a threshold (i.e. 15 pixel). Furthermore, most deep learning models assume the input having the size 256×256 due to model architecture and GPU memory size. To make full usage of the input data, we resize the
cropped first impression screenshots into 256×256 without keeping the aspect ratio.

**WebTrend21-160k**  For a trend analysis unbiased by designs from Google, which has 200 web domains but similar designs, we intentionally combine all sub-domains of Google (e.g. google.co.uk) as one domain and only keep at most one screenshot for each month. Thus, we have 162,933 screenshots left from 9,535 web domains as the WebTrend21-130k dataset, which is used for the year classification task in Section 3.6.

**WebTrend21-50k**  To balance the weights of different web domains, we create a balanced version of the WebTrend21-160k dataset, by uniformly sampling in time at most 10 web pages from each web domain. This new benchmark dataset, WebTrend21-50k, contains 53k web pages from 9,430 web domains.

### 3.3 Method Overview

To analyze the design trend for a given year, e.g. 2016, we train deep learning models to visualize, synthesize and discriminate the trend. Furthermore, these trained models can be applied during three different stages of a design process: inspiration, prototyping and evaluation.

1. **Trend visualization for design inspiration:** we learn a nonlinear embedding of web page screenshots using a variational auto-encoder (VAE), so that we can navigate through the space of available designs intuitively. In this learned interpretable embedding, we discover design shifts happened in 2016 and build a “design walkthrough” for web domains to provide design inspiration.

2. **Trend colorization for design prototyping:** we learn to re-colorize web pages to be consistent with the given period of time with a Conditional Generative Adversarial Network (cGAN). The trained model reveals how different layout and visual elements are colored differently in 2016 compared to previous years and can be used to generate colorization suggestions for design prototyping.

3. **Trend classification for design evaluation:** we train a Convolutional Neural Network (CNN) to predict year labels from web page screenshots. We discover visual elements that are more probable to be used in 2016 web pages, and use it to evaluate which part of input designs look before or behind a given year.
3.4 Trend Visualization for Design Inspiration

The key of this section is to learn to represent web page designs with lower-dimensional vectors, where linear combinations of these vectors still yield valid web pages and changing along each dimension is interpretable in terms of design change. We first describe the data processing and model settings to learn such representation. Then, with this new representation, we analyze how trends change over time and build an application to interpret the design trend of given web domains for design inspiration.

3.4.1 Data and Model

Data Processing  We here focus on the low-resolution impression of the web design, e.g. layout and color palette, neglecting the fine-resolution details, e.g. body text font. Thus, we downsample the screenshots from the WebTrend21-50k dataset, balancing the contribution of different web domains, into 64×64 images.

Model  To learn the latent representation for web page screenshots, we use the popular variational autoencoder model (VAE) [30] which is known for its interpretable results [21]. Given an input image, VAE encodes it into a Gaussian mean vector $\mu$ and a standard deviation vector $\sigma$, which are used to generate random samples to be decoded into the original input (Figure 3.9). For our latent representation, we only use the mean vector $\mu$ as the latent code $z$.

Following the common practice, we set the trade-off parameter between the reconstruction loss and the loss from $z$'s prior distribution, $\beta$, as 1, and we need to select
Figure 3.10: Comparison of VAE trained with different dimensions of latent variable. (a) we plot the reconstruction error over epoch number (the error does not go to zero partly due to the noise injected in the sampling module to avoid overfitting); (b) we show qualitative comparisons for three sampled images; (c) covariance matrix of 20 axes of latent representations with largest variance. With $\#z > 100$, the learned representation becomes distributed, not concentrating on certain dimensions.

the number of dimensions for the latent representation $z$. Intuitively, we want $z$ to contain enough information to reconstruct input images, and to be compact enough to use as few dimensions as possible. Below, we show results for training VAE with dimension $\#z \in \{50, 100, 300, 500\}$. For reconstruction, we observe that $z \geq 100$ leads to reasonable results to faithfully recover layout pattern and color palette both quanti-
Figure 3.11: Variational autoencoder model. Given input images, the model learns to compress them into more compact representation with the encoder module, which can well recover the original input with the decoder module.

To measure the compactness of $z$, we compute the covariance matrix from all web pages in the dataset and count the minimum number of dimensions needed to account for 80% of the total variance. In Figure 3.10b, we found that the number of dimension needed to capture 80% variance grows linearly with the its size of $z$. To balance between the two factors, we choose $z = 100$ for the analysis below.

### 3.4.2 Analysis

Given the learned interpretable representation for web page screenshots, we first apply Principal Component Analysis (PCA) to extract design changes and t-Distributed Stochastic Neighbor Embedding (t-SNE) to display all design changes onto a 2 dimensional space to visualize the shift of design trends. Such analysis is only possible with the learned $z$ representation, where any latent code in this space corresponds to a valid sample. Otherwise, with raw pixel representation for example, it is hard to synthesize web-page-like images through combination of input data (Figure 3.11a).

**PCA Analysis** We first compute PCA bases, so that changes along each PCA bases are independent from each other. Below, we analyze the change of mean and basis coefficients to discover design trends among four groups of year: 96-00, 01-05, 06-10, 11-16.
MEAN. Given the latent code $z$ of web page screenshots encoded by our trained VAE model, we compute the mean for all the images and the mean for each year group respectively. In Figure 3.11b, we show the decoded images from these mean latent codes. For the total mean, the decoded image has a typical layout (e.g. banner, side bar and white margins) with the prevalent blue color. For the mean of each year group, we notice the design trend shift in terms of growing size of image regions and the usage of more diversified color. In contrast, we can barely notice such trend from the original pixel space (Figure 3.11a).

BASIS COEFFICIENT. We first interpret each PCA basis as a design change through visualization and reveal how the design trend changes over the years.

To visualize the top 10 PCA bases $B_i$, based on the variance of their coefficients $\sigma_i^2$, we use two different methods. First, for the numerical method, we create new latent codes through adding the total mean with variation along a basis, $\{±0.1, ±0.5\}\sigma_i B_i$. Then we decode each of these new codes to see how it changes the mean web page design. In the top part of Figure 3.12a, we see these PCA bases capture design changes in terms of brightness (basis 1), layout (basis 2-4) and hue (5-10). Interestingly, shown in Figure 3.13, we sort the numerical visualization of the PCA bases by average hue value, and the top bases are complementary in the HSV hue wheel. It suggests that the VAE model learn to recover how web designers choose color palette. To verify the numerical visualization results, we use a data-driven approach to bin and average image pixel values by the coefficient magnitude for each basis. However naive averaging will lead to almost gray image as in Figure 3.11a where the signal is overwhelmed by the brightness of images (i.e. captured by the first PCA basis). To fix it, we only do averaging over a subset of input images whose coefficients for previous PCA basis are within a certain distance (i.e. $0.5\sigma_i$) from the mean latent code, alleviating the data variance from those bases. In the bottom part of Figure 3.12a, we see that the data-driven averaging method agree with the numerical method, confirming the interpretation of the learned PCA bases. Also, in Figure 3.13 we show PCA bases that represent color change of the design roughly move along the diameters on the HSV hue wheel.

To visualize the trend change, we plot the mean and standard deviation of the PCA coefficients for the top 10 bases in Figure 3.12c-d. In terms of the mean changes, the PCA 2nd and 6th bases have drastic change, moving towards positive values over time.
Trend change along these two bases reveal that more web designs are having layouts with white background with images on top and more side bars to fill up the margin. For the standard deviation in general, there's a continuous increase of over the years due to the appearance of diversified design patterns. However, for the 8th basis, year group 1996-2000 has bigger variance, which is explained by the prevalent usage of the
Sec. 3.4. Trend Visualization for Design Inspiration

Figure 3.13: Visualization for the first ten eigenvectors. For the first two year groups, blue and yellow are main colors.

either blue or yellow background color and blue text links in the early years of Internet.

**t-SNE Visualization** After understanding each design change with PCA, we now perform a holistic analysis by projecting the latent representation onto a 2-dimensional “design space” with t-SNE.

In Figure 3.14, we show the learned “design space,” where all images from WebTrend21-50k dataset are averaged according to their t-SNE coordinates. The change of design is continuous in this space and regions reflect different design patterns. For example, designs with colorful background are clustered in the top region; with colorful banner and side bars in the bottom right region; with white margins in the bottom left regions; with different size of images at the page top in the upper-left and middle-left regions.

Then, we overlay the heatmap of web design density over the years to reveal the trend shift. In Figure 3.15, we observe that the heatmaps change almost continuously over the years, with bottom-left regions (designs with wide white margin) disappear while upper-left regions (designs with big images on top) appear.

### 3.4.3 Application

**Web Domain Walk Through** We display the design change of a given web domain in the learned “design space.” As t-SNE does not handle adding new data points onto the learned manifold. We train a 3-layer convolutional neural network to regress the t-SNE position from the raw input. In Figure 3.16, we show results for two example web domain **craigslist.org** and **bbc.com.** For **craigslist.org,** we see a loop in the design
Figure 3.14: 2D representation from t-SNE. We first bin the t-SNE code into $40 \times 40$ grid cells and average over images whose codes belong to each cell. This 2D map is purely based on the appearance of web pages without any year labels, where each region correspond to one trend of design. Later, with year labels, we visualize how the design trends change over time.

space, indicating the recurring design pattern, i.e. design in year 2016 is similar to that in year 2004. For \textit{bbc.com}, we can see three clusters of design groups, indicating that
Figure 3.15: For each year, we show the heatmap of the ratio of images at each rounded grid position.

The design uses light color background from 2001-2008, then red as theme color and black and white.
Figure 3.16: Visualization of the nonlinear manifold learned by VAE. (a) We use t-SNE to project the latent code onto the 2D plane and round up the position to regular grid point; (b) for each group year, we show the heatmap of the ratio of images at each rounded grid position. The heatmaps change continuously over time, showing the design trend shift. For example, late 1990s web pages often have white margins or colorful background color, while 2010s web pages mostly have wide images at the top of the page.
Figure 3.17: Top 20 most frequent colors in each year. Magnitudes are in logarithmic scale. The color space is a discretized version of sRGB with 4 bins per channel, sorted based on hue. Through this analysis, we observe that the Internet is getting duller (less saturated) and a wider, more evenly spread, range of colors is being used as time progresses (distribution over colors is becoming higher entropy).

Figure 3.18: Conditional Generative Adversarial Network model. Given input gray scale images, the generator model outputs a color image, and the discriminator model classify if the input-output pair is fake or real.

### 3.5 Trend Colorization for Design Prototyping

We here focus on the trend of color usage in web design. Traditional methods capture the color statistics $P(\text{color}|\text{year})$, i.e. what color palettes are trending for a given year (Figure 3.17). However, there are two problems: (1) even in the same year, the color palette of web pages varies greatly depending on layout and content; (2) given the color palette, it is non-trivial to automatically assign colors to different regions of web pages.

In this section, we examine the trend of color change through the lens of image colorization task. Given an input gray-scale web page, we directly colorize the web page in a consistent manner based on its content and the selected year.
3.5.1 Data and Model

Data Processing We use the 256×256 screenshot images converted into the CIE Lab color space, and use the L channel (gray-scale) as input and a,b channels as output. We divide the WebTrend21-50k dataset into four subsets by years: 1996-2000, 2001-2005, 2006-2010 and 2011-2016. We use 80-20 split of web domains into train and test set, and divide each subset of screenshots accordingly. By having the clear separation of web domains, we avoid models that correctly colorize web pages through memorization of similar training data from the same domain.

Model As a conditional generation problem, a direct approach is to train a regression mapping an input gray-scale image to its color channels with L2 loss. However, as discovered in [23] (pix2pix), L2 loss tends to produce blurry results, due to the non-unique solution to the colorization problem. Thus, we adopt the same architecture in [23] and replace L2 loss with the combination of L1 loss and a discriminator loss, following the generative adversarial framework. Specifically, we use a skip-layer architecture for generating color images and a patch-based discriminator to classify if the pair of the input gray-scale image and the output color image is real or fake (Figure 3.18). We use the default settings of the architecture and optimization algorithm.

3.5.2 Analysis

We first show qualitative results for web page re-colorization by models trained on different groups of years. (a-d) input web pages are from 1996-2000, 2001-2005, 2006-2010, 2011-2016 respectively (Figure 3.19). As expected, the trained model prefers different color palette for screenshots with different layout and content. Also, given a web page, models trained on different years' web pages provide diverse but consistent re-colorization result.

To show that the trained models capture the color trend, we test these models along with a model trained on all training images on web pages from different year groups. We compute the L2 distance between the ground truth ab color channel and that from output. In Table 3.1, we find that the model trained on each year group performs the best when tested on the same year group.
Figure 3.19: Qualitative results for web page re-colorization by models trained on different groups of years. (a-d) input web pages are from 1996-2000, 2001-2005, 2006-2010, 2011-2016 respectively.
Table 3.1: Colorization errors on different groups of years. We test four models trained on different years of web pages and a model trained on all years. We compute the L2 distance between the ground truth ab color channel and that from output. The model trained on each year group performs the best when tested on the same year group.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1996-2000</td>
<td>9.12</td>
<td>7.96</td>
<td>8.88</td>
<td>8.87</td>
<td>10.95</td>
</tr>
<tr>
<td>2001-2005</td>
<td>8.06</td>
<td>7.88</td>
<td>7.18</td>
<td>7.81</td>
<td>8.34</td>
</tr>
<tr>
<td>2006-2010</td>
<td>7.73</td>
<td>7.78</td>
<td>7.98</td>
<td>6.91</td>
<td>7.62</td>
</tr>
<tr>
<td>2011-2016</td>
<td>7.89</td>
<td>7.93</td>
<td>8.14</td>
<td>7.73</td>
<td>7.02</td>
</tr>
</tbody>
</table>

Figure 3.20: Illustration of the bricolage colorization application. Given (b) manually stitched web pages from (a) input templates, pix2pix models trained on web pages from different periods generate (c) diversified and self-consistent colorization from the gray-scale image of (b).

### 3.5.3 Application

**Bricolage Colorization** During prototyping, designers often come up with a draft design by bricolaging different parts of other web pages (Figure 3.4b). However, it is hard to
make the color consistent. With the trained models above, we can not only re-colorize the bricolage prototypes consistently, but with different years as style. In Figure 3.20c, we show the diversified colorization results. For more qualitative results, we select web pages from different years and re-colorize them with models trained on different period. In Figure 3.20, we see that the pix2pix model not only learns to choose different color palette based on the web page content, but also assigns colors in a reasonably manner.

3.6 Trend Classification for Design Evaluation

We aim to find discriminative design elements that are indicative of time, which can be low-level features (e.g., usage of blue buttons) or high-level patterns (e.g., usage of face photos). However, the design elements are often outnumbered by other trend-irrelevant parts of the web pages. One way to find needles in the haystack is to use handcrafted features and an improved clustering technique [12]. On the other hand, recent works [72] show that convolutional neural network (CNN) models can automatically learn hierarchical features to use for pattern discovery through label prediction tasks.

Below, we adopt the latter approach and train a VGG-CAM model, Class Activation Mapping model (CAM) [73] based on VGG-16 network [55] on the task of year prediction. With this network, we first find salient web pages that reveal temporal trends. Then, to further dissect the trends into visual patterns, we visualize the learned neurons in the model. For application, we visualize heatmaps for the predicted average year and analyze the contributions of these visual patterns with data-driven approach.

3.6.1 Data and Model

Data Processing To find discriminative design elements, we use the WebTrend21-160k dataset, where the input screenshots have the size 256×256. During training, we use the standard practice and randomly generate one of the five fixed 224×224 crops (four corners and center) with random left-right flip of images. We use the 80-20 split of the data into train and test, without any intersecting web domains.

Model In Figure 3.21, we sketch the VGG-CAM model [73] with its feature maps output from each layer. Compared to the standard VGG-16 network, the VGG-CAM model replaces the fully-connected layers with 1×1 convolution layers (colored yellow)
Given an input web page, the model predicts its year label. The first five groups of convolutional layers (blue) are the same as VGG-16. Then there are three 1x1 convolutional layers (yellow) to replace the fully-connected layers in VGG-16. In the end, after a global-average-pooling layer (orange) and a softmax layer, we obtain the probability of the year label for the input web page.

and adds a global-average-pooling layer (colored orange) before the softmax layer. Such modification of the network allows the last layer feature map to be used as the prediction heatmap with 14x14 spatial resolution.

We train the model from scratch, classifying input screenshots into one of 21 years from 1996 to 2016.

### 3.6.2 Analysis

**Classification Accuracy** This network achieves 19.6% top-1 classification accuracy after re-balancing the test data to have equal weight (chance = 1/21 or 4.8%). In Figure 3.22a, we show the confusion matrix for the year label classification, where the diagonal elements are not dominant. One reason for the low classification accuracy is that design trends usually last for a few years. Thus web pages with the same design trend may have multiple plausible year labels, which makes it hard to predict a single year label correctly. As shown in Figure 3.22c, sample web pages from “brown.edu” for each year (1997-2016) can be categorized into different design trends groups, some of which even last for 4 years.

Instead, a more reasonable metric is to evaluate the percentage of model predictions
Figure 3.22: Test results for VGG-CAM model. (a) confusion matrix for the exact year classification, (b) confusion matrix for the prediction to be within one year range of the ground truth, (c) example web pages from “brown.edu”, where the design trend can last for several years which makes it hard to predict the exact year of a web page.

<table>
<thead>
<tr>
<th>Model/Evaluation</th>
<th>top-1</th>
<th>±1 year</th>
<th>±2 years</th>
</tr>
</thead>
<tbody>
<tr>
<td>VGG-CAM</td>
<td>19.6%</td>
<td>47.6%</td>
<td>64.3%</td>
</tr>
<tr>
<td>random</td>
<td>4.8%</td>
<td>14.3%</td>
<td>23.8%</td>
</tr>
</tbody>
</table>

Table 3.2: Test accuracy for year label prediction on WebTrend21 colorization errors on different groups of years. To show that the ...
Figure 3.23: Here we show, for each predicted year, the top four web pages our model is most confident belong to that year, revealing designs that are highly characteristic for each year, such as simple textual web pages in the 1990s, and more image-heavy designs of the 2010s.

confuses them with older web pages as some web domains did not update their design over those years. However, for web pages after 2005, our model mostly confuses them with newer web pages.

With VGG-CAM, we now can reveal the design trends over the years through web pages with top prediction scores. In Figure 3.23, we show the four test images that the CNN is most confident belong to each year. These are web pages that the model thinks are highly characteristic of each year. By examining these representative images, we can qualitatively identify design trends that are summarized by experienced web designers [1]. For web pages from 90s (e.g. 1996), there were mostly icons, plain text and hyperlinks; for early 2000s (e.g. 2000, 2004), sidebars and menus appeared and
design layouts became more structured; afterwards (e.g. 2008, 2012, 2016), bigger-size images were used and designs of bigger sizes prediction scores.

**Visualization of Internal Units** Further, the VGG-CAM model learns a hierarchical visual representation of design elements that are indicative of temporal trends. Specifically, we visualize the discriminative patterns stored inside neurons in the first, middle, and last layers of the network (conv1, conv3 and pool5) (Figure 3.24). We also compare with the VGG-CAM model trained on ImageNet database. For the conv1 layer, the neurons can be directly visualized as RGB images, where they capture horizontal and vertical patterns, perhaps picking up on the axis-aligned nature of web designs. For the conv3 and pool5 layers, we adopt the data-driven methods of [72] to find patches from the test data with the top activation for each neuron (we visualize 4 selected neurons for each of these layers in Figure 3.24). With increasing layer depth, neurons begin to capture more semantically interpretable patterns: e.g., boxes and text characters in the conv3 layer; logos and human faces in the pool5 layer.

### 3.6.3 Application

By design, VGG-CAM model produces 14×14 heatmap for the probability of year labels. Thus, we can apply the model to identify which specific regions are causing input web pages to look out of date. Such a tool could also guide a user to design a vintage look, by telling the user when their design takes on the appearance of, say, the year 2000.

**Heatmap Prediction** Given the 14×14 response before the global average pooling layer, we apply softmax at each spatial location to get the probability map for each year. Then we compute the average year label for each location, resize it to the size of image and overlay it on top of the input image. In Figure 3.25, we apply this method to two seemingly generic web pages, one from 1996 and the other 2016. Next to each detected patch, we show other patches in the test data that also highly activate these neurons, revealing the visual patterns each neuron is sensitive to (as described in the paragraph above).

**Template Explorer** We not only show the visual elements that are indicative of years, but also visualize the template similar WebTrend21 database. In Figure 3.26, we apply this method to two seemingly generic web pages, one from 1996 and the other 2016. Next to each detected patch, we show other patches in the test data that also highly
activate these neurons, revealing the visual patterns each neuron is sensitive to (as described in the paragraph above). This analysis suggests that the two web pages have distinct design elements, evolving from text and simple graphic elements to natural images and icons.

### 3.7 Conclusions

In this section, we take the deep learning approach to quantify design trend analysis.

We first present a large-scale temporal dataset for web pages from 1996 to 2016, which will be publicly available to push forward the web design trend study. Then we train three different models to analyze and apply three different aspects of trends:
Figure 3.25: Heatmap of predicted years for web pages. We show (a) four input web pages ranging from 1996 to 2016 and (b) the predicted 14×14 heatmap of predicted years, scaled to the image size. Our model learns to find the logo region in the example from 1996 and the text region in the example from 2016 out of date.

(1) being able to navigate through web design space intuitively can help designers to better understand the past and current trend. (2) being able to colorize a web page with consistent color based on its content and year label enables quick prototyping to realize the draft design of layout. (3) being able to predict the apparent date of a web page opens up a number of design applications.
Figure 3.26: These two web pages might seem generic, but each reflects the design trend of its own era. Can you tell what makes the 2016 web page look like 2016? Over each example web page above, we show regions our algorithm thinks are diagnostic of that web page’s year. On each row, one patch from the web page is shown, followed by four patches, from other pages, that the algorithm considers to have a similar visual pattern. Noticeably, the old design (1996) mainly uses simple text and textual links for communicating the content, and a modern design (2016) applies more diverse graphical elements such as special fonts for text, graphic logos, and natural images.
Chapter 4

Conclusion

TIME is the great sculptor of the world and there is rich information hidden in the visual temporal structure. On one hand, human beings are capable of acquiring common sense knowledge of physics and design trends through daily visual experience. On the other hand, it is unclear how computers can learn these dynamic properties of the world, as it is hard to represent and label such abstract knowledge. Acquiring common sense knowledge from image sequences is a less-explored area in the computer vision community, as traditional algorithms focus on estimating static properties of the visual world, e.g. geometry and semantic labels, which often have well-defined ground truth.

In this thesis, we propose acquiring common sense knowledge of these dynamic properties by using the temporal labels of image sequences. To get around the difficulties of directly labeling the knowledge, we train models to predict the temporal labels as the proxy and analyze the learned representation to extract the common sense knowledge. Specifically, we study two different properties: the arrow of time from frame orders in real world videos and web design trends from year labels of web pages. In both endeavors, we demonstrate that models trained on the temporal labels open up new analysis and applications.

In Chapter 2, Visual Arrow of Time, we train a classifier to tell whether input videos are playing in the forward or backward direction. In the first part, we manage to manually encode knowledge of physics into four different models, while in the second part, we train models to learn by themselves. For the latter part, we first design systematic control experiments to identify superfluous signals, avoiding our model learning these artificial signals from video production instead of the real signal. Then, we train the model on the large-scale Flickr video dataset, and analyze the learned visual cues.
showing both semantic and non-semantic cues for arrow of time and the photographer bias during video capture. Finally, we apply the trained arrow of time classifier to video forensics tasks, and use its learned features for other tasks, such as action recognition.

In Chapter 3, Visual Evolution of Web Design, we quantify the trends in web design with three applications. We first collect a large-scale database, consisting of screenshots over the past 21 years (1996-2016) for the top 11k web domain. Then we visualize the trends over time in a learned representation through a reconstruction task. We also train a conditional generation model which re-colorizes input web pages with different color palettes based on their content and year labels. Finally, we train a classifier to predict the years in which input web pages were created and identify visual elements that are ahead of or lagging behind the trends.

For the future work, we plan to explore visual temporal structures in different sources of image sequences to learn other common sense knowledge, e.g. cartoon videos for stylish object deformation and instructional videos for temporal logic of problem solving.


