UNIVERSITÄT KOBLENZ · LANDAU
Faculty 4: Computer Science

WeST People and Knowledge Networks
Institute for Web Science and Technologies

Shot Detection in Screencasts of Web Browsing with Convolutional Neural Networks

Bachelor's Thesis

in partial fulfillment of the requirements for
the degree of Bachelor of Science (B.Sc.)
in Informatik

submitted by
Daniel Vossen

First supervisor:	Prof. Dr. Steffen Staab Institute for Web Science and Technologies
Second supervisor:	Raphael Menges, M.Sc. Institute for Web Science and Technologies

Koblenz, September 2020

Statement

I hereby certify that this thesis has been composed by me and is based on my own work, that I did not use any further resources than specified – in particular no references unmentioned in the reference section – and that I did not submit this thesis to another examination before. The paper submission is identical to the submitted electronic version.

	Yes	No
I agree to have this thesis published in the library.	☒	☐
I agree to have this thesis published on the Web.	☒	☐
The thesis text is available under a Creative Commons License (CC BY-SA 4.0).	☒	☐
The source code is available under a GNU General Public License (GPLv3).	☒	☐
The collected data is available under a Creative Commons License (CC BY-SA 4.0).	☒	☐

.....

(Place, Date) (Signature)

Note

- If you would like us to contact you for the graduation ceremony, please provide your personal E-mail address: .....
- If you would like us to send you an invite to join the WeST Alumni and Members group on LinkedIn, please provide your LinkedIn ID : .....

Zusammenfassung

Analysen von Benutzerverhalten auf grafischen Oberflächen, welche sich durch Nutzerinteraktionen oder im Hintergrund laufende Scripte verändern können, werden zur Bewertung der Oberfläche und als Hilfe bei ihrem zukünftigen Gestaltungsprozess herangezogen. Die Herausforderung besteht darin, die Daten der Interaktionen mit einer Oberfläche in verständliche und nützliche Informationen aufzubereiten. In dieser Arbeit befassen wir uns eingehend mit Bildschirmaufnahmen von Interaktionen auf Webseiten und damit, wie visuelle Veränderungen der Webseitenoberflächen erkannt werden können. Die Erkennung visueller Veränderungen ist ein elementarer Bestandteil, um Interaktionsdaten mehrerer Benutzer auf einen gemeinsamen Raum für eine spätere Analyse zu aggregieren. Dazu betrachten wir wie visuelle Veränderungen in traditionellen Videos erkannt werden und ob die gleichen Methoden auch für Bildschirmaufnahmen von Webseiteninteraktionen verwendet werden können. Wir stellen fest, dass künstliche neuronale Netze, die zum Erkennen visueller Veränderungen genutzt werden, zwar für normale Videos mit großem Erfolg eingesetzt werden, dass es aber noch keine derartige Arbeit für Aufzeichnungen von Webseiteninteraktionen gibt. Daher verwenden wir in meiner Bachelorarbeit ein Convolutional Neural Network, um visuelle Veränderungen in Aufzeichnungen von Webseiteninteraktionen zu erkennen, und vergleichen unsere Ergebnisse mit anderen gängigen Verfahren. Wir finden heraus, dass unsere Herangehensweise in manchen Situationen ähnliche oder bessere Ergebnisse erzielen kann, in den meisten Fällen jedoch keine Alternative zu gängigen Verfahren darstellt.

Abstract

Behavior analysis of users on a graphical interface, which can change its appearance by interactions or background scripts, is used to evaluate the interface and to be a help in its future designing process. The challenge is to turn data of interactions with an interface into comprehensible and useful information. In this work, we take a closer look into screen recordings of interactions with websites and how visual changes in the interfaces can be detected. Detecting visual change is a key part of the aggregation of interaction data of several users onto a common space for later analysis. For this, we observe how visual change gets detected in traditional videos and whether the same methods can find a use for recordings of interaction with websites. We find that, while artificial neural networks tasked with the detection of visual change get used with great success for videos recorded with cameras, there exists no such work for recordings of interactions of websites yet. Therefore, in this bachelor thesis, we use a convolutional neural network to detect visual changes in recordings of interactions of websites and compare our results with other state-of-the-art procedures. We find out that in some cases we can reach comparable or even better results as state-of-the-art approaches but most of the time our approach does not offer a reasonable alternative.

Contents

1 Motivation **1**

2 Related Work **2**

3 Foundations **5**
3.1 Machine Learning 5
3.1.1 Overfitting And Underfitting 5
3.2 Artificial Neural Networks 6
3.2.1 Backpropagation 7
3.2.2 Activation Function 7
3.2.3 Loss Function 9
3.3 Convolutional Neural Networks 9
3.3.1 Convolutional Layer 10
3.3.2 Pooling Layer 11

4 Data Set **12**
4.1 Data 12
4.2 Structure 12
4.3 Task 14

5 Method **15**
5.1 Preparing The Data 15
5.2 Apparatus 16
5.3 Scenarios 17
5.4 Scenario 1 18
5.5 Scenario 2 21
5.6 Scenario 3 23
5.7 Scenario 4 25

6 Discussion **27**

7 Conclusion **29**

8 Appendix **33**

1 Motivation

Behavior analysis of users on graphical interfaces is often utilized to evaluate the design of interfaces. The results of behavior analysis aim to support the designers in providing a user experience as intended. To analyze the behavior it is important to understand how a user is interacting with an interface. While questioning users about an interface is a common practice, the answers may not be detailed. Users could subjectively rate usability and give an insufficient explanation about their experience. The handling of prearranged questions is error-prone as users can run into unforeseen situations not covered by the questionnaire or users may misunderstand the questions in the first place.

With today's technology, it is possible to record eye and mouse movements as well as mouse clicks and keyboard input. We call this kind of data *gaze and interaction data* in the following. Detailed knowledge about where users look at the interface may allow designers to perform informed decisions based on actual user experience. To draw conclusions about the actual user experience of an interface, it is necessary to have the recording of multiple users interacting with the interface. When the gaze and interaction data is mapped to a video screencast, a comprehensive recording of a user's behavior with an interface is created. The appearance changing nature of such interfaces makes it difficult to use video recordings as a common space to aggregate several users' gaze and interaction data. To create such a common space the video recordings would have to be synchronized while different paths of interaction between users had to be considered. A screenshot of the interface can provide a better common space instead.

We want to take a closer look at the case of webpages as those are probably the most used interfaces as of today. In the early days of the internet, webpages were static and simple. It would have been enough to group all users and their gaze and interaction data on the same webpage for analysis. A single screenshot of the entire webpage would suffice as common space to map the data into as every user would see the same while browsing the page. Changes in the look of a webpage could have been recognized easily by metadata like changes in the URL as every new look required to query for a new webpage document.

But modern webpages are often dynamic and can change their appearance rapidly. Everyday interaction with a webpage, like opening or closing popup menus, will change the displayed information while the URL of the page stays the same. Scripts running in the background may also change the look of a webpage without any interaction. This makes a webpage and its changes hard to understand from its structure alone and renders the behavior analysis of multiple users difficult.

To aggregate gaze and interaction data from various users on the same, yet dynamic, interface for analysis, the recordings should be cleverly grouped according to displayed contents. The starting point of grouping the recordings and also the core of my work is recognizing when the visual contents visible in a screencast of a webpage change. At first, we define what makes a visual change relevant. For ex-

ample, one could argue switching the displayed content of a carousel may imply a relevant visual change. On the other hand, one could also argue the visual change is not relevant if the carousel is the only visual change and the layout of the webpage remains the same. The definition should be suited to the task at hand.

After we have come up with a reasonable way to decide about visual change, we want to segment our screencasts accordingly with an automated decision process. For this, we compare each frame with its consecutive frame in our screencasts and label them accordingly if visual change exists. Because we are talking about examining images and recognizing visual change we make use of artificial neural networks (ANNs) that have already shown to be successful at classifying images. ANNs are mainly used for pattern recognition and possible predictions based on trained input data. In the traditional approach to detect visual change, features are chosen manually and get extracted out of the images. Feature extraction is done by running the images through various filters and by reducing the huge amount of input data in each image to a reasonable size while trying to get meaningful results. The extracted features are then used as input on an ANN for classification. For a more modern approach the feature extraction can be done by an ANN, too. The most commonly used ANNs for image analysis are convolutional neural networks (CNNs). CNNs are doing the part of feature extracting and image processing themselves. CNNs use backpropagation to learn their own filters that we believe have great potential to be well fitted to the task at hand. We choose to create and train our own CNN with labeled example data. In the end our trained CNN classifier will take a pair of frames as direct input and return the predicted label.

2 Related Work

In this work we want to split screencasts of interactions with websites into visual coherent sequences. Thus, we map this challenge to the field of automated shot segmentation, for which a broad set of work exist.

Video abstraction and analysis is a widely explored research topic to make it possible to search through or get a high-level perspective on videos without having to watch them completely. The hierarchical structure of a video is simple. Single frames make up the lowest structural level. Several successively recorded frames make up a shot. A sequence of shots related by location, actors, and time make up a scene. All scenes together form a video. The segmentation of a video's scenes and shots is a key element in the structural analysis of a video [1].

According to Del Fabro and Böszörmenyi [2] most approaches to scene segmentation follow a shot boundary detection. Shot boundaries are the temporal edges of a shot. Shot boundaries lie between the last frame of one shot and the first frame of the following one. This means one shot lies completely between two boundaries. Following the structural hierachy, a video is first broken down to its individual shots which are then grouped back together into several scenes. Accordingly scene segmentation could also be described as a segmentation of a video into groups of re-

lated shots.

Shot detection most often uses feature deviation between two consecutive images. The feature deviations are compared to a predefined or adaptive threshold of deviation to estimate shot boundaries. Features can consider anything from a global point of view to recognition of remarkable points in smaller parts of an image. Color histograms [3] for example represent the color distribution of an image. This may be useful for detecting global changes in a displayed webpage. Edge change ratio [4] features represent incoming and outgoing edges. These features may be useful for webpages with opening and closing menus and other elements with borders. Optical flow [5] is the distribution of motion or velocity of vertices in relation to the observer. Optical flow could be useful for webpages with moving elements like carousels and animated buttons. SIFT/SURF [1] detect and recognize local features and objects with the help of a reference database. Detecting local features may be useful to recognize reoccuring layout elements like menus or banner ads. Non-visual features, for example transcripts [6] of the audio track, may also be considered. Since we cannot make use of audio tracks and their transcriptions from screencasts for the simple fact that they usually do not exist it may be possible to extract text-based features with character recognition instead. A bag-of-words approach may be applicable on extracted text-based features.

From 2001 to 2007 the annual TRECVid benchmarking exercise included the shot boundary detection task. The participating groups were asked to detect shot boundaries in videos with possibly visible transitions between shots. P. Over et al. [7] describe four types of transitions. The first type of transitions is a cut. A cut has no visual transition between shots. The ending of one shot is followed directly by the beginning of the next shot. The second and third types are dissolves and fade-ins/-outs. In case of a dissolve the second shot fades in while the first shot fades out. If the second shot fades in only after the first shot fades out the transition type is a fade-in/-out. Transitions that are not properly described by the first three types make up the forth type. Smeaton et al. [8] made the observation that seven of the top ten performing groups used support vector machine classifiers (SVM). These SVM classifiers were trained mostly with image features of sample data to recognize transitions between shots and replace the otherwise used thresholds.

M. Gygli [9] used a CNN in end-to-end fashion to detect shot boundaries indicated by the aforementioned transitions. The CNN looks at ten consecutive frames and classifies if the fifth and sixth frame belong to the same shot. A fully convolutional architecture allows the labeling of more than one pair of frames for bigger input without the cost of redundant computation. They trained the CNN with an automatically generated and annotated dataset. For this dataset M. Gygli took video snippets from different videos from YouTube that had no or very few transitions themselves. Some of the video snippets were combined with different transitions. The CNN was then fed with snippets where all frames belong to one shot and snippets that include a transition.

Menges et al. [10] transfered the problem of shot detection from movies and videos

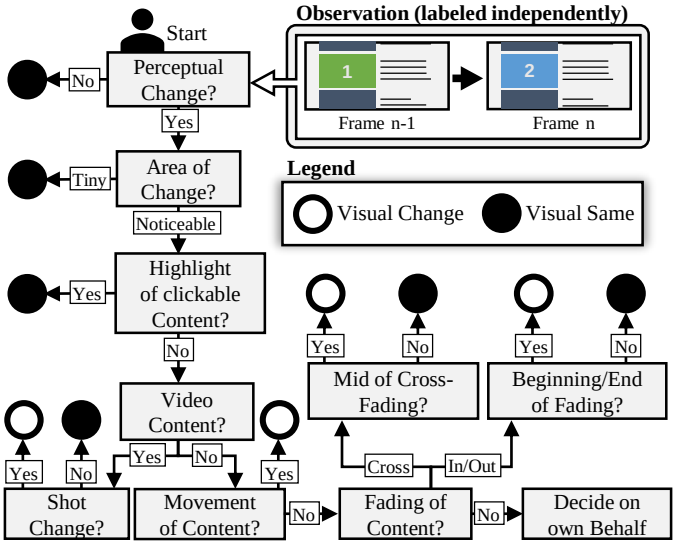


Figure 1: Labeling of visual change decision tree

to recordings of interactions with websites. They argue that traditional transitions do not exist in recordings of web browsing and thus there is little to no point to detect traditional transitions. A new decision process shown in Figure 1 was designed to define shot boundaries. Then they evaluated a support vector classifier and a random forest classifier to detect visual changes that may imply shot boundaries. These classifiers were trained with multiple deviation features of two consecutive frames and their shot boundary label. The random forest classifier performed slightly better. Menges et al. compared the used features against each other for importance. The edge change fraction was standing out as the biggest contributer to the decision process in shot boundary detection. Optical flow, histograms, and text recognition on the other hand were not deemed as important for the classifiers.

Protasov et al. [11] extracted features in videos using a CNN trained with a large set of real world scene types. They argued that understanding the background of an image will give a good description of the place where the scene is filmed and can be used to seperate shots and scenes. The CNN they used takes ten patches per image and returns ten fuzzy class vectors. The fuzzy class vectors consist of 205 classifications to different location types. The location types range from landscapes like mountains or airfields to more specific real-world places like mountain paths or runways. Fuzzy class vectors have the adventage to provide implicit information instead of discrete classes. Protasov et al. united the ten fuzzy class vectors into a single feature vector. Feature deviation between two consecutive frames was used to detect shot boundaries. To reduce the effect of impulse noise Protasov et al. used additional features of frames surrounding the two observed frames. The further apart in time from the observed pair of frames the less weight was given in the

decision process. It is questionable if this network trained with real-world scene types is useful for our use-case of visual change detection on websites.

While CNNs are used to great effect in finding shot boundaries in traditional videos they have not been used yet to classify visual changes on websites.

3 Foundations

At first, we take an introductionary look at machine learning. Then we look at artificial neural networks and how their components work. At last we try to understand convolutional neural networks and their special layers.

3.1 Machine Learning

Machine learning is a subfield of artificial intelligence. Hand-crafted AI methods for problem solving like search algorithms or heuristics are reaching their limits the larger and more complicated the data to process gets. Instead machine learning algorithms develop a mathematical model from already existing data to solve a problem.

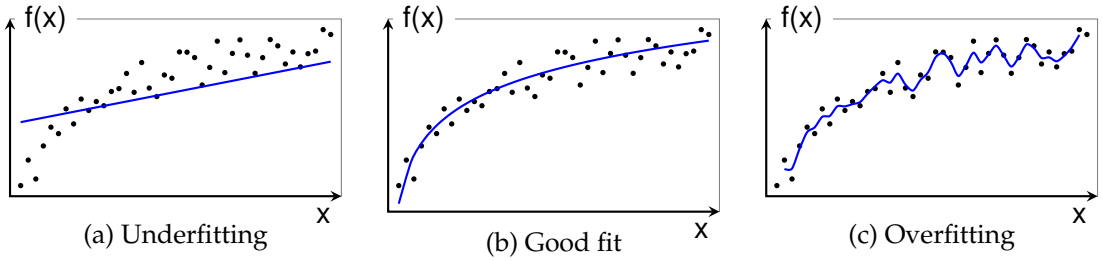
Machine learning approaches are being used for a wide varity of tasks on big data like classification, clustering and regression analysis. State-of-the-art machine learning technology already outperforms conventional AI at complex games like chess and is even successfully used for autonomous car driving.

Approaches in machine learning can be split roughly into supervised, unsupervised and reinforced learning. In supervised learning the input data includes the associated labels. The aim is to come up with a decision process for mapping inputs to desired output labels. Unsupervised learning is used for feature learning or pattern recognition in unlabeled input data. In reinforcement learning the program interacts with an environment and gets feedback after making decisions. The decision-making is adjusted on the fly to get the best possible feedback in the future.

Our classification problem to detect visual change belongs to the supervised learning category as we have labeled data from which our model hopefully learns general rules to label unknown data correctly. To see if such a classifier was trained correctly the labeled data is typically split in two parts, the training data and the testing data. A model learns from the training data set only and is then evaluated on the testing data. This ensures that we can test the model on data previously unknown to the model while we can compare its results to the labels we already know about. The training process is typically carried out multiple times over the whole training data set. One iteration of training data is called an *epoch*.

3.1.1 Overfitting And Underfitting

Training a mathematical model can lead to problems with a desired generalization. The more complex the architecture of a neural network gets the more options are



(a) Underfitting (b) Good fit (c) Overfitting

Figure 2: Problems with Over-/Underfitting

available to the model to use as parameters in its decision process. On the one hand if a trained model uses more parameters than necessary and specializes on the training data to reduce training loss it might be unsuited as generalization and is therefore overfitted 2c. An overfitted and therefore excessively complex model might try to fit even errors in noisy training data which in turn reduces its capability to make good predictions. It performs better on training data than on unknown data. On the other hand if a model misses relevant parameters or is too simple to describe a more complex data structure it might be underfitted 2a.

There are several ways to avoid the problem of over- or underfitting. One of the more obvious methods is to include more training data. A lack of training data can make it harder to find important features or make the model narrow on features that are not relevant on the broad scale. Similarly can the choice of training duration make an impact on generalization. Too many training iteration may cause the model to become too complex while an early end may stop the model from becoming more precise. The complexity of a model can be reduced by removing features. This method is called pruning. Features with very low impact on the final output are often irrelevant enough for a generalization to get rid off. Another way to keep complexity low is the use of regularization. The more features are used in the model the higher the cost gets. The target is to minimize the cost and thus keeping the complexity low. Cross validation can be used to detect overfitting. A subset of training data is left out of the training process and used for testing the resulting model. If the model performs much better on training data than the testing data from the same pool of data we can assume that the model is fitted too strongly to the training data.

3.2 Artificial Neural Networks

An artificial neural network (ANN) consists of connected nodes called artificial neurons which can send signals between each other. Such a network is structured like a directed weighted graph where the neurons are connected by edges. ANNs are loosely based on biological neural networks as the names of the used terms imply. Each neuron that receives signals from other neurons processes the signals in

some nonlinear way described in the following sections and can send its own signal to other neurons. The edges transmitting the signals have weights amplifying or weakening transmitted signals. When we train a network we are searching for the best weights to transmit signals to get from the input signal to the expected output of the network.

A typical ANN consists of three layers of neurons. The input layer receives the input data while the output layer returns the data processed by the network's layers. In between input and output layers lies one hidden layer. As a user only interacts with the input and output layers the hidden layer is not directly observable by the user. If all neurons in one layer are connected to every neuron in the previous layer it is called *fully connected*. The more hidden layers there are in a network, the deeper it gets. If more than one hidden layer exists the approach is called *deep learning*.

3.2.1 Backpropagation

A signal in a mathematical construct like an artificial neural network is nothing more than some value $\in \mathbb{R}$. Each neuron as described in figure 3 has one or more values as input and calculates a new value. The newly calculated value can then be used as input to other neurons or, if the neuron is part of the output layer, as the final output of the network.

Let us consider a neuron N with inputs I and activation function φ. Every input $i \in I$ has a value x_i and the weight w_i of its edge. The output of the neuron is calculated by applying the activation function on the sum of all values multiplied by their respective weights: $f_N(I) = \varphi(\sum_i x_i * w_i)$

In the case of supervised learning, the calculation of an neuron's output is adjusted by backpropagation. Input data is fed forward through the network and thus through all neurons. A loss function is applied to compare the network's output with the actual target. The lower the calculated error the closer the output is to the target. The error is propagated back through the network and the weights of each edge are adjusted proportionally to their impact on the loss. This process is repeated over the whole dataset while the error is hopefully getting lower. If there are no significant changes to the error anymore we can assume the trained model is as close as it gets to its reachable accuracy.

3.2.2 Activation Function

The activation function determines the neuron's output as defined in the previous section. We use two different activation functions in our work: the sigmoid function and the rectified linear unit (ReLU).

The ReLu is one of the most commonly used activation functions in deep networks. The rectifier function is the simple $f(x) = max(0, x)$ with the codomain $[0, \infty[$. That means a positive value gets passed on as it is while all negative values are getting set to zero (cf. Figure 4).

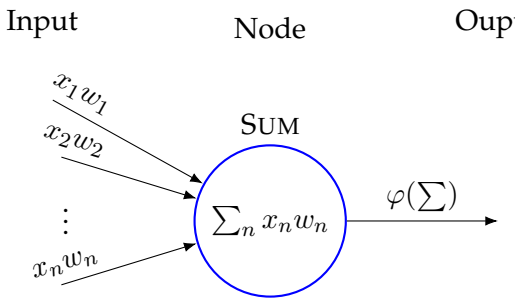


Figure 3: Neuron with incoming axons x_i, weights w_i and activation function φ

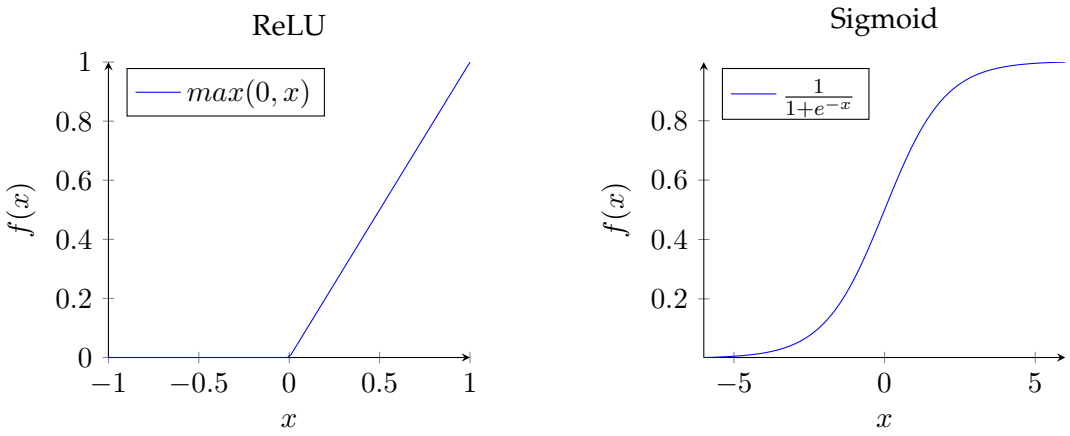


Figure 4: Activation functions

The ReLU is much faster than other non linear activation functions [12]. While there is a risk that neurons can die off completely, the sparse activation of neurons speeds up the processing of the network. Since the application of the ReLU on any value only needs a single operation to produce its output (the comparison to zero) is also very easy on the need of computational power. With this, the ReLU is perfectly suited for deep and complex networks where processing speed can often become crucial.

The other activation function we utilize is the sigmoid function. The function is defined as $f(x) = \frac{1}{1+e^{-x}}$ with the codomain $[0, 1]$. When we apply the sigmoid function on any value, we are squashing it to a range of 0 to 1 (cf. Figure 4). A value between 0 and 1 is perfectly suited to an artificial network's output layer in classification problems where the final outputs are the predicted labels for each class.

We use the ReLU throughout all layers of our network and apply the sigmoid only to our output layer.

3.2.3 Loss Function

The goal of training a neural network is the minimization of the error calculated by the loss function. The loss function is used to calculate the loss which is then again used for the aforementioned backpropagation. Thus the definition of the loss function is an essential part in the neural networks design. A bad choice of loss function can have a negative impact on the training process.

The binary cross-entropy (BCE) loss function used in this work is defined as:

$$BCE(Obs) = -\frac{1}{N}\sum_{i=1}^{N} l_i * log(p_i) + (1 - l_i) * log(1 - p_i)$$

Let us dissect this function. N is the number of observations where l is the actual label and p the model's output, therefore the predicted label. We understand visual change to be labeled as 1 and visual same labeled as 0. When we look at the term inside the sum we notice that depending on our label $l \in \{0, 1\}$ one part of the addition will result in zero. For $l = 1$ this means we only calculate $log(p_i)$, and for $l = 0$ we calculate $log(1 - p_i)$. These two logarithmic functions are mirrored by the perpendicular line $x = 0.5$. This means that a prediction which is off the label by some margin Δ results in the same loss, no matter if it missed label 1 by Δ or label 0 by Δ. Since we know that $0 \leq p \leq 1$ we will get the *negative* loss shown in figure 5. That is why we use the negative mean $-\frac{1}{N}\sum_{i=1}^{N}$ as the BCE loss.

Figure 5 also shows that the loss is getting bigger the further away the prediction is from the actual label. While this might also apply to any linear function the logarithmic curve punishes predictions much harder the further they are off the label. As a result the model makes predictions that are rather extreme close to either 1 or 0. By using the BCE loss function we try to ensure that our model produces confident predictions that do not end up somewhere in the middle between 1 and 0 what would imply uncertainy where even the smallest changes in the input data could push the prediction on the other side of the 0.5 rounding threshold and therefore switch class.

3.3 Convolutional Neural Networks

The convolutional neural network (CNN) belongs to deep learning and is the typical choice of a neural network to analyze images. [13] A CNN has the advantage of learning image filters itself without the need for human interference. Little to no image pre-processing is needed.

While the typical fully connected ANN has no problems with 1-dimensional input data, it is not suited for 3-dimensional data like images ($width \times height \times channels$) and their high count of data points.

The key idea behind CNNs is the use of a small receptive field on the input image to limit the number of inputs for each neuron as each neuron is responding only to the receptive field instead of the whole input image. The entirety of all

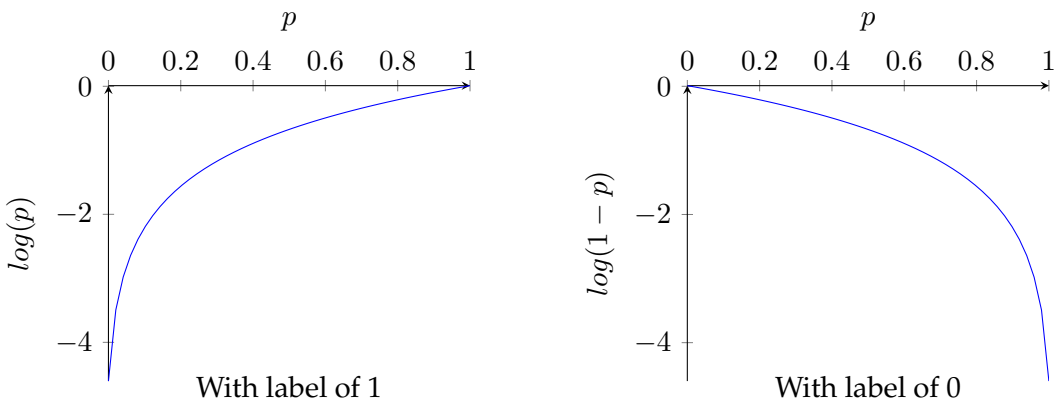


Figure 5: Negative loss calculated with the logarithmic function

(partially overlapping) neurons spreads over the image completely. These locally-responding neurons are either part of a convolutional layer or a pooling layer. The typical CNN has several convolutional layers most often followed by a pooling layer each. After the last layer, the 3-dimensional data is flattened and run through fully connected layers for classification. The fully connected layers that would be fed typically with features already extracted by hand-crafted methods get features that the CNN deemed relevant for its task instead.

3.3.1 Convolutional Layer

An digital image can be understood as tensor with shape $(height) \times (width) \times (depth)$. In our case a tensor is a multi-dimensional matrix where every element contained in said matrix is of the same data type, like float for example. For a color image in the RGB color space the depth would be three, one for each color channel. As a convolutional layer takes more than one image or feature map as input the input tensor's shape is $(number\ of\ images) \times (image\ height) \times (image\ width) \times (image\ depth)$. After the convolution the convolutional layer returns a feature map with the shape $(number\ of\ images) \times (feature\ map\ height) \times (feature\ map\ width) \times (feature\ map\ depth)$. The convolution is done by filters called kernels that are moved over the input. A kernel is typically a 3×3 or 5×5 matrix and uses the full depth of the input. If we think of the input as image again we can imagine the kernel as a small square that looks at all image layers on the small region it is placed upon. The kernel represents the receptive field of a neuron as a neurons inputs are only coming from the region the kernel is laid on. Instead of using the weights all neuron's incoming edges as adjustable parameters in the training period of our network only the kernel's parameters are learnable. A 3×3 sized kernel for example has only 9 learnable parameters. The number of parameters is therefore independent of the input image size in contrast to a fully connected layer where each single

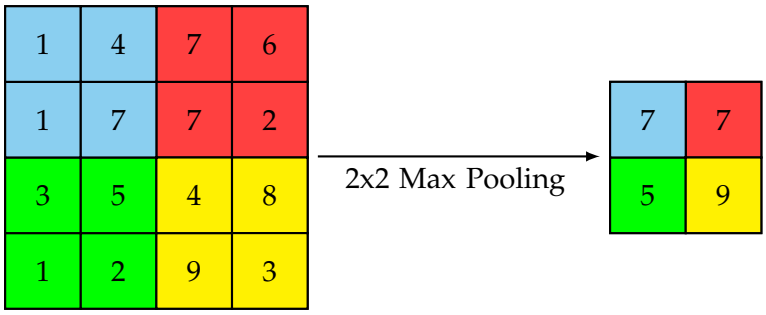


Figure 6: Max pooling with filter size 2×2 and stride 2

neuron would have $width \times height$ parameters. This is important as the number of parameters is already unfeasible for the smallest of images. One kernel returns a 2-dimensional feature map as output. The convolutional layer's actual output feature map depth is the number of kernels used in the convolutional layer. The kernel's outputs are laid on top of each other and create a new channel each. The parameters of each kernel are initialised randomly to ensure that every resulting channel in the layer's output is unique allowing different starting points in the gradient descent of the training process which seeks for local minima.

3.3.2 Pooling Layer

The job of the pooling layer is the reduction of the dimensions of an image or feature map. Image data is downsampled to reduce the amount of overall parameters to process by the network while hopefully losing as little relevant data as possible. This reduction of complexity also helps against overfitting. Clusters of values are compressed into single values by a regional filter. The filter is typically a 2×2 or 4×4 matrix and is moved over an image or feature map similiarly to a convolutional kernel. The distance that the filter is moved by each step is specified by the so called stride. In most cases the stride will have the same size as the cluster. A filter with size 2×2 and stride 2 for example will have no overlapping clusters and covers the input completely. A higher stride would result in gaps in its coverage and lost data while a lower stride would result in values being used multiple times and less downsampling which might not be wanted. The most commonly used pooling is max pooling shown in figure 6. The filter used in max pooling returns the highest value of its cluster as new value and discards the rest. Another example among others would be the average pooling where the new value is computed by taking the mean of all values inside the cluster. We will use exclusively max pooling in this work as most available literature on the subject of image analysis.

4 Data Set

Menges et al. [10] kindly provide their now publicy available data set[1] for us to use.

4.1 Data

The data set contains screencasts of web browsing from four different users on the same twelve websites each. Menges et al. declared that each website is belonging to one of four categories with three websites each as follows: Walmart, Amazon and Steam belong to the category Shopping; Reddit, CNN and the Guardian to the category News; NIH, WebMD and MayoClinic to the category Health; and General Motors, Nissan and Kia to the category Car. The screencasts were captured with 5 frames per second and contain a total of 45,310 video frames. Each pair of consecutive frames that differed by at least one pixel was manually labeled in accordance with the decision progress depicted in figure 1 resulting in 4,446 pairs labeled as visually different. Additionally to each screencast a datacast is provided including information extracted from the web browser and website's DOM tree, as well as the user's interaction data. The interaction data holds 893 clicks, 10,742 scrolls, 11,058 mouse movements and 837,716 gaze points.

4.2 Structure

The data set is seperated into several directories and files. On the highest level the data is split accordingly to different participants in the study. One such directory holds all session data of that specific user on all websites. All files belonging to a session on a specific website are named with a unique prefix to tell them apart. The data set structure for one user on one website is visualized in figure 7.

The screencast is provided as a .webm file and depicts the exact same view the user was presented with when interacting with the website. Each screencast has a width of 1024, a height of 768 and lies in the YUV color-space. All other session information is found inside of a .json file including DOM tree mutations and interaction data.

Menges et al. also provided their own generated session data as well as their results after processing said data in .csv format for use. The most important file in this work features_meta.csv includes information about all relevant observations. Observations are pairs of the same webpage layer in contiguous frames that are not pixel identical after taking scrolling offsets into account. A layer can be understood as the composition of different elements of the webpage and is defined in the DOM tree. The scrollable part of the webpage is the root layer and is part of every webpage. Other layers are fixed elements like navigation bars or menus. Each entry of the .csv file describes exactly one observation of the screencasts. Observation ID,

[1] https://zenodo.org/record/3908124

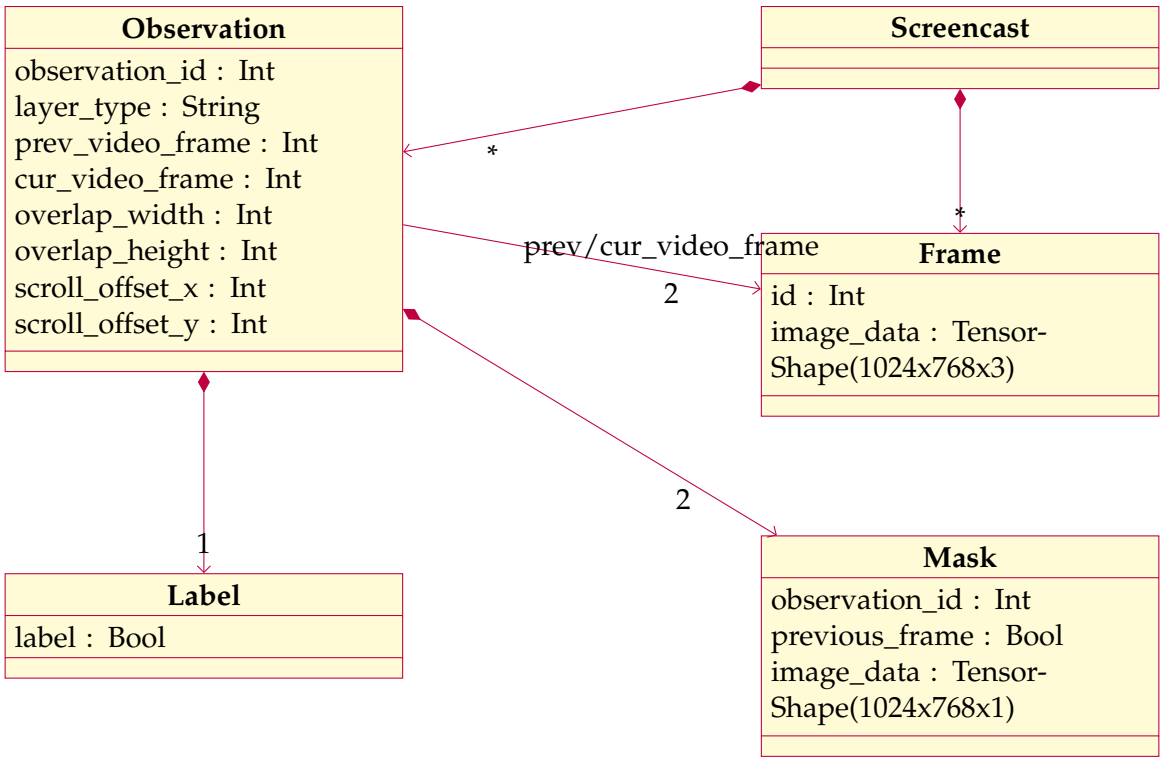


Figure 7: Visualization of the provided dataset's structure as UML diagram. Label describes visual change.

Table 1: We show the number of all observations per user per website. Read as follows: *obs_vs/t_obs (percentage)*, where *obs_vs* is the number of observations labeled as visual change, *t_obs* is the total number of observations and *percentage* is the calculated percentage share of observations with visual change to the total number of observations

	p1	*p2*	*p3*	*p4*	*total*
Walmart	65/280(23%)	195/596(32%)	142/454(31%)	266/980(27%)	668/2310(28%)
Amazon	28/141(19%)	168/697(24%)	94/503(18%)	288/1147(25%)	578/2488(23%)
Steam	55/210(26%)	211/760(27%)	164/601(27%)	241/942(25%)	671/2513(26%)
Reddit	72/324(22%)	36/256(14%)	73/574(12%)	76/551(13%)	257/1705(15%)
CNN	40/211(18%)	83/862(09%)	49/928(05%)	102/818(12%)	274/2819(09%)
Guardian	54/241(22%)	72/519(13%)	39/555(07%)	72/730(09%)	237/2045(11%)
NIH	50/155(32%)	72/286(25%)	116/400(29%)	128/608(21%)	366/1449(25%)
WebMD	81/277(29%)	147/489(30%)	42/350(12%)	152/863(17%)	422/1979(21%)
MayoClinic	28/129(21%)	26/141(18%)	37/222(16%)	76/489(15%)	167/981(17%)
General Motors	36/240(15%)	63/413(15%)	92/538(17%)	82/739(11%)	273/1930(14%)
Nissan	99/302(32%)	158/425(37%)	144/520(27%)	214/748(28%)	615/1995(30%)
Kia	77/434(17%)	76/406(18%)	82/655(12%)	104/799(13%)	339/2294(14%)
total	685/2944(23%)	1307/5850(22%)	1074/6300(17%)	1801/9414(19%)	4867/24508(19%)

layer type, frame IDs of the frame pair, overlap and scroll offsets which are relevant for us can be found here.

Each observation is labeled with a boolean flag for visual change. The label can be found in labels-l1.csv and can be mapped to features_meta by the observation id.

View masks complete the observation data. Each observation possesses two view masks in .png format, one for each frame. View masks can be laid over the frames to show only the layer that is observed and black out the rest in each frame.

The distribution of observations between users and websites is shown in table 1

4.3 Task

The labeling by hand through an usability expert is not a reasonable way to process big data like screencasts. A more rational way is the automization of such labeling by recreating the decision process used by the expert to decide about visual change. The automization method used by Menges et al. utilizes a classifier which uses a high number of computer-vision features deemed as important for classification. The features were hand-picked and are based on heuristic computer vision methods. As a consequence thereof there might be some better way to extract features. Therefore the use of a CNN appears as a valid alternative for classification.

5 Method

First, we introduce ways to prepare the data set. It has to be transformed such that it can be used as input for our CNN. Second, we describe the architecture of our CNN we use to create a classifier. Last but not least, we train classifiers for different scenarios and analyze the results.

5.1 Preparing The Data

The preperation of training and testing input data can be seen as a two step process. In the first step all existing observation are filtered for a variety of criteria. In the second step the frame pairs of the filtered observations are made ready to use as inputs for the CNN. Both steps of filtering observations and arranging the frames as inputs can be done in various different ways.

All observations can be filtered by users and websites. Training and testing data can be split for example by using the observations of one subset of users for training and the other users for testing. The same idea can be applied for websites. Another way to filter observations that can be used to split training and testing data is the simple use of keeping a percentage of observations by random for training and using the removed ones for testing. To reproduce the same randomized split of data that may be used in training the same random seed should be used while training and testing. Keeping only specific layers is also a way of filtering observations. For example we could keep the root layer only and throw away all observations belonging to different layers – like fixed headers. Several observations across layers of and the same pair of frames can also be combined into one observation instead. The newly generated observations would be labeled as visually different if at least one of the participating layers is also labeled as visually different and visually same otherwise. Observations with a low overlap can also be filtered. In accordance with Menges et al. all observations with an overlap of 32 or less are dropped when using this filter.

Each observation will be used once per epoch. As we have a very skewed ratio between observations labeled as visually same and visually different we can balance the observations by reducing the number of observations labeled as belonging to the majority class to that of the other label. To avoid undersampling of the cut short majority class, observations are not left out completely but rotated through with each epoch. We use exclusevely balanced training data throughout our entire work to hinder our classifiers from heavily favouring the majority class.

Observations are fed to the network in batches. The observation's frames are only prepared batch by batch. Using more than 10 prepared frame pairs at once was found to be impracticable with the memory limitations of the available hardware.

Both frames of one observation are extracted from their .webm file by ID. OpenCV decodes each frame to an RGB image with the dimensions of 1024×768. It is possible to restrict the input frames to their corresponding layer of the observation by utilizing the enclosed view masks. All pixel values that are not representing the

current layer are set to zero. The scrolling offset can also be corrected. For this all overhanging pixels are cut off. Then the position of the remaining pixels are adjusted to make the frames overlap completely with each other. Let us take the action of scrolling down as example. The first frame will be cut at the top and the second will be cut at the bottom as the cut off pixels have no matching partner in the other frame. The now smaller frames are copied onto an empty frame each at the same position. This will ensure them to have a perfect overlap while also retaining the original dimensions of 1024×768.

In the final step both frames are laid on top of each other resulting in a 6 channel tensor. The first three as well as the last three channels are the RGB colors of their respective frames. The result after factoring in the batch size is an input tensor with the dimensions of $(batch\_size) \times 6 \times 768 \times 1024$.

5.2 Apparatus

Typically CNNs work with really small input images of 32×32 pixels for example. As our input images have the dimensions of 1024×768 pixels we are working with a much larger scale. Our CNN structure takes inspiration from other established networks working with large-scale inputs such as AlexNet [14] and VGG [15].

Our CNN inputs for the training period are tensors with a fixed size of 1024×768 and 6 channels. The input tensor is processed by 8 convolutional layers and 6 max-pooling layers. Figure 8 shows the positions of layers within the CNNs architecture. Based on the AlexNet structure the first and second convolutions have a kernel size of 5×5 and the remaining convolutions use a smaller kernel with a size of 3×3. Based on the VGG structure early pooling layers are more frequent while the input resolution is still high. Our first four convolutional layers are each followed by a max-pooling layer. With smaller but deeper feature maps, convolutional layers are used in succession with max-pooling layers following the 6th and 8th convolution. All max-pooling layers are using a window size of 2×2 pixels and a stride of 2. Following the convolutional and pooling layers are three fully connected layers with 2048, 1024, and 1 channel. The output of each convolutional layer and each fully connected layer except the last one is run through a ReLU. The sigmoid activation function is applied to the last output for the purpose of binary classification. We use the binary cross-entropy (BCE) [16] as loss function. The Adam optimization algorithm [17] is used for backpropagation.

Our CNN was implemented using the Pytorch framework. Pytorch is currently the leading machine learning framework in research[2]. It has a good python integration and is easy to use. Additionally, we are using the OpenCV library for Python for all image manipulations.

The two scripts train_cnn.py and test_cnn.py of the publicly available code[3] can

[2]https://thegradient.pub/state-of-ml-frameworks-2019-pytorch-dominates-research-tensorflow-dominates-industry/

[3]https://gitlab.uni-koblenz.de/dvossen/visual_change_cnn.git

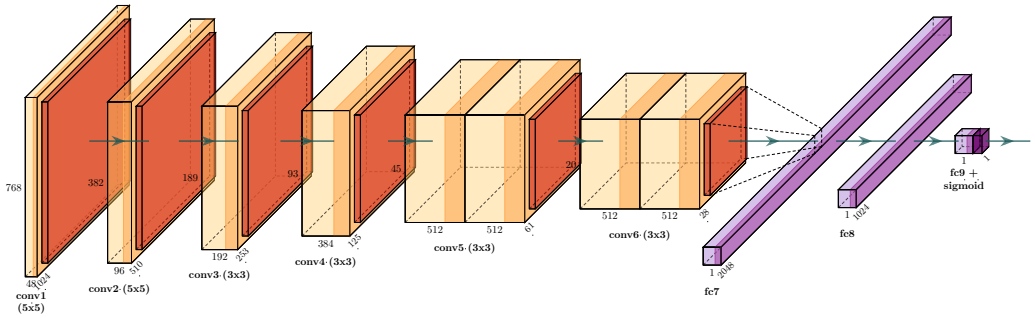


Figure 8: CNN structure

take any number of arguments to reproduce all methods described in the section 5.1 without changing one line of code.

5.3 Scenarios

As our available data set gives a wide range of options to prepare the training and testing data of our CNN we come up with several reasonable scenarios that might comparable to production environment use cases. The architecture of our CNN is the same throughout all scenarios, only the data used changes between scenarios. We showcase four different ways to split the data set into training and testing data. At first, we compare different data preparation techniques on a more general split. We take a look at how our CNN will work with raw image data that was not tampered with in any way. But our main interest is the performance of our CNN when it is trained and used on images that were prepared with the additional data available to us: scrolling offsets, masks, overlap. As one might consider the root layer as the most interesting and important layer we also consider using our CNN on the root layer only. After going over the different data preparation techniques, we look further into other possible choices of training data. We look at two scenarios where we split training and testing data by users and in the last scenario, we split our data by webpage categories.

The analysis of the resulting classifications is done in Python3 with scikit-learn [18]. As we train models for several examples we evaluate them by accuracy and F_1score. The F_1score is a binary classifier's measurement of performance on testing data. We report the F_1score per class (visual same and visual change) which is defined as the harmonic mean of precision and recall:

$$F_1 = 2 * \frac{precision * recall}{precision + recall}$$

For each label, the number of correctly labeled predictions divided by the number

of all predictions labeled as such is the precision. For example, if we predict 10 observations to be visually different but only 7 of these are labeled as such, we have a precision of 70%. The recall on the other hand is calculated by taking the number of all right predictions of one label and dividing it by the number of all predictions that should have been predicted with the same label. For example, if our testing data has 10 observations labeled as visually different but only 8 of these are predicted as visually different we have a recall of 80%. Depending on task and goal precision and recall might be of different importance. A look at the F_1score only might not suffice.

5.4 Scenario 1

In our first scenario, we create a general classifier over the whole data set. Our aim is a classifier that works on data of the same users and the same websites it was trained with. For this, we split all observations randomly in 80% training and 20% testing data. We train four different classifiers. Two setups use raw images as input and the other two setups get input images prepared in accordance with masks, scrolling offsets, and minimum overlap. In both cases, we have one classifier used on all layers, and one classifier used only on the root layer. Table 2 displays F_1scores and accuracy of all classifiers on all websites.

Let us take a look at the classifiers on raw image data first. The classifier *raw* that was trained and used on observations for all layers has an overall accuracy of 75% and the overall F_1scores of 81% and 63% for the classes visual same and visual change. The comparable low F_1score for visual change is composed of 93% recall and 48% precision. This means that nearly all observations with visual change are predicted correctly but over half of all positive predictions are false positives. As less than 25% of all observations belong to the class of visual change we have better accuracy if we only predict the majority class of visual same.

We try to understand why accuracy is rather underwhelming. As all layers are combined into one observation our classifier might have problems with different kinds of visual changes. Big visual changes that might appear on the root layer are labeled the same as small visual changes on small fixed elements. We have situations as well where visual change is found on more than one layer but only labeled as change overall. Our classifier misses possibly vital information in both cases and might be hindered by it in its training process.

Let us compare our findings with our other classifier *rawR* which was trained and used on raw image data as well but on observations with layer type root only. *RawR* achieved an overall accuracy of 82% and overall F_1scores of 88% and 64% for the classes of visual same and visual change. While we see only a small difference to *raw* in the overall statistics we can see some improvement on a few websites.

One might think that the use of the root layer only should show a bigger improvement to the combination of all layers. The diversity is lower so the classifier should be able to specify. The root layer is also the biggest layer after all that would imply

Table 2: Classifiers of visual change. We are randomly splitting the data into 80% training data and 20% testing data. We compare different data preperation: no image manipulation and observations on the same frame pair for differend layers combined into one observation (*raw*), no image manipulation and only observations of the root layer are used (*rawR*), images prepared in accordance with masks, scrolling offset and minimum overlap with all observations used (*prep*), images prepared in accordance with masks, scrolling offset and minimum overlap and only observations of the root layer are used (*prepR*). Best classifier result per website is printed in bold font.

Shopping Sites	Walmart				Amazon				Steam			
Classifier	*raw*	*rawR*	*prep*	*prepR*	*raw*	*rawR*	*prep*	*prepR*	*raw*	*rawR*	*prep*	*prepR*
Visual Same [F_1]	66%	91%	93%	**97%**	83%	91%	93%	**95%**	77%	83%	**87%**	86%
Visual Change [F_1]	68%	79%	88%	**90%**	70%	74%	81%	**84%**	66%	71%	**72%**	71%
Accuracy	67%	87%	91%	**95%**	79%	86%	90%	**92%**	73%	78%	**82%**	81%

News Sites	Reddit				CNN				Guardian			
Classifier	*raw*	*rawR*	*prep*	*prepR*	*raw*	*rawR*	*prep*	*prepR*	*raw*	*rawR*	*prep*	*prepR*
Visual Same [F_1]	78%	93%	96%	**99%**	86%	89%	93%	**95%**	83%	89%	97%	**98%**
Visual Change [F_1]	54%	57%	75%	**77%**	46%	28%	**59%**	54%	45%	42%	**74%**	71%
Accuracy	70%	88%	93%	**98%**	78%	80%	89%	**90%**	74%	82%	**95%**	**95%**

Health Sites	NIH				WebMD				MayoClinic			
Classifier	*raw*	*rawR*	*prep*	*prepR*	*raw*	*rawR*	*prep*	*prepR*	*raw*	*rawR*	*prep*	*prepR*
Visual Same [F_1]	88%	98%	95%	**99%**	79%	90%	**93%**	**93%**	79%	91%	**97%**	95%
Visual Change [F_1]	80%	86%	85%	**87%**	58%	65%	**80%**	74%	50%	74%	**88%**	81%
Accuracy	85%	96%	92%	**98%**	72%	85%	**90%**	89%	70%	87%	**95%**	93%

Car Sites	General Motors				Nissan				Kia			
Classifier	*raw*	*rawR*	*prep*	*prepR*	*raw*	*rawR*	*prep*	*prepR*	*raw*	*rawR*	*prep*	*prepR*
Visual Same [F_1]	75%	77%	**91%**	**91%**	76%	78%	**97%**	95%	95%	93%	97%	**98%**
Visual Change [F_1]	48%	41%	**66%**	61%	74%	59%	**89%**	83%	81%	61%	**84%**	79%
Accuracy	66%	67%	**86%**	85%	75%	71%	**95%**	93%	93%	88%	95%	**96%**

that the biggest visual changes can also be found there. But all these thoughts do not hold. It seems more likely that there is no difference in difficulty in detecting visual changes of fixed elements or the root layer. This is contradictory to our attempt at understanding our previous classifier.

We conclude that the use of observations of all layers and the use of root layer observations only make no real difference in our CNN classifier's performance. As both classifiers get whole untampered images as input we assume that *raw* and *rawR* work very similarly internally.

To see if the similarity of visual change classification on fixed elements and root layers holds if we do not give the full-sized images as input we train another two classifiers. The classifier *prep* has observations of all layers as input and the classifier *prepR* uses only root layer observations. Both classifiers take image data prepared in accordance with masks, scrolling offsets, and minimum overlap as input.

Overall *prep* achieves 91% accuracy and F_1scores of 94% for visual same and 78% for visual change. *PrepR* achieves 91% accuracy identically and very similar F_1scores of 95% and 76% for visual same and visual change. Table 2 shows that these numbers are not only close in the overall scores but on individual websites, too. We can see that image preparation has a drastic effect on the classification. *Prep* and *prepR* beat *raw* and *rawR* consistently over the board. The only exception seems to be on the website Steam where all four classifiers score very close to each other and image preparation is barely an improvement.

There might be several reasons why our preparation of input data shows great classification results like these but they pretty much come down to the fact that we take work out of the CNNs hands. Let us consider that our classifier looks for visual changes in its input data. The application of masks reduces the region to search through where visual change might have happened. Additionally the readjusting of the scrolling offsets reduces the region further to the overlap only and the classifier does not need to learn about scrolling altogether. This is also an explanation about the below-average results for Steam where the raw input classifiers are barely worse. A look at the data set tells us that all observations on the Steam website belong to the root layer and no fixed elements are involved. Accordingly, the masks are just blank and no part of the images is excluded from the input. The small edge *prep* and *prepR* have over *rawR* might be attributable to the scrolling offset only. Strangely enough, the similarity between the two classifiers of prepared input data works as a counter-argument as *prep* should show the better results as fixed elements are much smaller and thus smaller regions have to be searched for visual change. This is where the black box problem of CNNs shows. It is impossible to understand how the images are processed internally and what features are extracted and used for classification so we will not get a final answer. Our takeaway from these comparisons at least is the fact that different layer types do not matter in the grand scheme of things.

Let us take a step back and compare the classifier results between websites. We notice that each classifier achieves above-average and below-average scores on the same websites as the other classifiers. That fact is most noticeable when we com-

pare the F_1scores of visual change. Again we find a possible explanation by looking at our data set. The F_1scores correlate in proportion to the website's ratio of observations of the class visual change to observations belonging to the class visual same. The smaller the percentage of observations with visual change the worse the scores become. The observations of the website CNN for example have the lowest percentage of visual changes of all websites with around 9%. Accordingly, we find that all our classifiers have their worst scores on the CNN website. On the other side of the spectrum, the websites Nissan and Walmart have the highest percentage of visual changes with around 30% and 28%. Our classifiers show some of their best scores there. We offer two possible explanations. Explanation one is about the useable visual changes in training data. A low percentage and thus low number of visual changes in observations might not be enough training data for the classifier to differentiate between visual same and visual change on testing data of the same website. CNNs are always in need of a big number of training data for each class which is not the case for our data set. With this, we can also assume that visual changes on one website do not suffice as training data to detect a visual change on other websites. Explanation two questions the use of F_1scores to judge a classifier's success. Let us consider two test cases. In both cases the number of false positives as well as the recall of visual change is identical but in case one we have two times as many observations belonging to the class visual change as in case two. The precision of visual change in case two would be half of that of case one and tank the F_1score with it. The percentage of observations with visual change ranges among websites from 9% to 30% which can be seen as rather unbalanced thus making for the big range of F_1scores.

Let us recapitulate our first scenario. We have shown that the extra step of preparing the input data accordingly to masks, scrolling offsets, and minimum overlap is followed by an impressive improvement of a classifier's performance. We have also shown that layer types have only a marginal effect if any on a classifier's performance. We demonstrated that the unbalanced ratio of the classes visual same and visual change has a big impact on F_1scores and deducted that the low percentage of visual change is not suited to either our CNN classifiers or the F_1scoring system or both.

5.5 Scenario 2

In our second scenario, we train four classifiers, one classifier per user, each with all of the user's observations on all websites. We test each classifier on the session data of the three remaining users. The results we achieve with this split of training and testing data might give us insight if the session data of one user is enough to generate a reasonable classifier for session data of unknown users on the same websites. This should be important for training data limited to sessions of a few users where a classifier is desired for a large number of users. A usability expert for example would only need to label observations of one user and can automate the

Table 3: Classifiers of visual change. We use a four-fold cross validation. One user session acts as training data, and three user sessions act as test data. Menges et al. used one website as training data for the *R.F* (random forest classifier) and *B.* (baseline classifier) and tested on the same website only. *CNN* and *CNN3* use all websites as training data. *CNN3* was trained on the session data of three users and tested on one. Best classifier result per website is printed in bold font.

Shopping Sites	Walmart				Amazon				Steam			
Classifier	*R.F*	*B.*	*CNN*	*CNN3*	*R.F*	*B.*	*CNN*	*CNN3*	*R.F*	*B.*	*CNN*	*CNN3*
Visual Same [F_1]	93%	87%	78%	**94%**	**94%**	90%	85%	90%	**92%**	72%	70%	81%
Visual Change [F_1]	87%	78%	63%	**88%**	**81%**	73%	55%	74%	**80%**	55%	55%	68%

News Sites	Reddit				CNN				Guardian			
Classifier	*R.F*	*B.*	*CNN*	*CNN3*	*R.F*	*B.*	*CNN*	*CNN3*	*R.F*	*B.*	*CNN*	*CNN3*
Visual Same [F_1]	**96%**	84%	85%	93%	**93%**	77%	78%	88%	**97%**	81%	88%	92%
Visual Change [F_1]	**69%**	42%	36%	39%	**58%**	32%	28%	29%	**79%**	42%	46%	40%

Health Sites	NIH				WebMD				MayoClinic			
Classifier	*R.F*	*B.*	*CNN*	*CNN3*	*R.F*	*B.*	*CNN*	*CNN3*	*R.F*	*B.*	*CNN*	*CNN3*
Visual Same [F_1]	**97%**	87%	85%	95%	85%	86%	85%	**92%**	**98%**	95%	90%	91%
Visual Change [F_1]	**92%**	72%	70%	70%	55%	54%	56%	**67%**	**88%**	78%	51%	77%

Car Sites	General Motors				Nissan				Kia			
Classifier	*R.F*	*B.*	*CNN*	*CNN3*	*R.F*	*B.*	*CNN*	*CNN3*	*R.F*	*B.*	*CNN*	*CNN3*
Visual Same [F_1]	**97%**	83%	75%	84%	**97%**	92%	86%	93%	**97%**	93%	80%	96%
Visual Change [F_1]	**80%**	42%	38%	52%	**90%**	80%	61%	84%	**84%**	47%	42%	74%

process for the other users. As we got our best results so far with the preparation of image data we do not further consider the use of raw untampered images as input data. As we found no noteworthy differences between the use of all layers and the root layer only we do not see the need to consider both cases again and settle on the use of all layers. We report the mean F_1score of our classifiers in table 3 under the name *CNN* for each website. The overall accuracies of each classifier range from 60% to 81% with an average of 74%. The F_1scores of visual same range from 70% to 87% with an average of 82% and the F_1scores of visual change range from 32% to 65% with an average of 51%.

We observe three noticeable problems. First, we notice the big performance fluctuations between our four classifiers. This is not only the case for the overall scores but different classifiers on the same website fluctuate even worse. To just give one example of many, the F_1score of visual change on the MayoClinic website is around 74% for the classifiers trained with the session data of the users p2 and p4, 55% for the classifier trained with session data of the user p1 and only 1% for the classifier trained with session data of the user p3. Second, we notice similar fluctuations be-

tween the mean F_1scores on different websites. Third, the comparable classifier *prep* of the previous scenario showed much better results, as well as much more consistency between websites, even tho the preparation of input data was the same. The big fluctuations also complicate a direct comparison as we do not seem to have clear results at hand.

As the classifier *prep* is the most similar classifier to *CNN* as far as the preparation of data input is concerned, we take a look at the difference between these two to explain the problems we notice. As the split of training and testing data is the only difference, we can offer two possible explanations. The first is about the nature of the split. While *prep* is trained with session data of all users *CNN* has the session data of only one user to train with. The data of the other users is completely unknown to the classifier in testing. Does this mean that observations of one user are different enough to observations of other users that across-user classification can not be performed as well as the classification of observations of already known users? In the context of user behavior analysis, this question sounds plausible because every user interacts differently with websites. But we only use images from the screencasts as input and no user interaction data whatsoever. Thus the difference between users has to be in the images only. There is a problem with that line of thought as the visuals are mainly determined by the website and not the user. The decision process in labeling the dataset was identical for all users as well as the tasks are given to the users while the data set was generated. To create visual changes whose detection can not be learned from visual changes of other users on the same website a user would probably need to interact with a website in seemingly strange and counter-intuitive ways deliberately. A website as a common space among users also limits the variety of possible visual changes that can happen between users. Much more plausible than different kinds of visual changes between users is our second explanation. Using the session data of one user only as training data is much less than 80% of the whole data set used in scenario 1. Our classifier *CNN* can be seen as an example of why CNNs in general need a rather high amount of training data compared to simpler machine learning methods. This claim is supported by comparing our classifiers with the number of observations used for training. The classifier trained with session data of p1 consisting of around 3,000 observations, and therefore the least observations of all users, performs worst (61% overall accuracy) while the classifier trained with over 9,000 observations of p4 performs best (81% overall accuracy).

We conclude that for our data set the session data of one user is insufficient as training data for our CNN to train a classifier for session data of other users. We assume the size of the data set to be the main factor in our failure and not the difference of session data between users.

5.6 Scenario 3

Based on our experience from scenario 2 we again train four across-user classifiers. This time we use the session data of three users for training and test the classi-

Table 4: Comparision of results between the different users. The session data of three users was used as training data. The results of the testing done on the forth user is shown here. Image data was prepared in accordance to masks, scrolling offsets and minimum overlap. Best performance per user session used as testing data is printed in bold font.

Test User	*p1*	*p2*	*p3*	*p4*	*average*
Visual Same [F_1]	93%	**94%**	91%	91%	92%
Visual Same precision	95%	95%	**99%**	**99%**	97%
Visual Same recall	**92%**	**92%**	85%	85%	88%
Visual Change [F_1]	70%	**73%**	66%	68%	69%
Visual Change precision	64%	**68%**	51%	54%	58%
Visual Change recall	76%	79%	**93%**	**93%**	88%
Accuracy	89%	**90%**	86%	86%	87%

fier on the fourth. Input data preparation stays the same as before. We try to get more insight if our assumptions in scenario 2 hold true as we have more training data available for each classifier while the testing is again applied on session data of users unknown to the classifiers. Under the name *CNN3* we report the mean F_1scores in table 3 for each website. Compared to the classifier *CNN* of scenario 2 CNN3 is much more consistent while outperforming *CNN* on every website except the news site Guardian. Likewise, we compare the performance of *CNN3* to the performance of *prep*. The scores of *CNN3* are very close and only slightly worse than that of the classifier of scenario 1. This might not come as a surprise since the ratio of testing and training data is similar between the aforementioned classifiers. Let us get into more detail about *CNN3*. For this we report the overall accuracy and F_1scores separately for each classifier in table 4 next to the averages of all classifiers. Our classifiers are named after the specific users on whose session data the classifiers are tested with. We notice that all classifiers score very similar to each other. The fluctuations we saw in scenario 2 are gone completely. The overall consistency is only flawed at the F_1score of visual change where *p1* and *p2* have noticeable more precision but less recall than *p3* and *p4*. The classifiers *p1* and *p2* also show slightly better accuracy and score the closest to *prep*. A look at the data set gives us possible explanations. As the session data of the users p1 and p2 holds the smallest number of observations, the split of training and testing data of the classifiers *p1* and *p2* is the closest to the ratio of 80% to 20% of scenario 1. The user session data of p1 and p2 also have a higher percentage of observations with visual change. Again we observe a drop in the precision score at visual change as we observed for all other classifiers on websites.

We sum the findings of our scenario up in two points. First, we have shown that across-user classification is indeed possible with consistently reasonable results and that to classify a user's observation data the session data of the said user does not need to be included in the training data set. Second, we demonstrated that the size

and variety of the training data set are of utmost importance to produce classifiers with consistently reasonable results.

5.7 Scenario 4

In our last scenario, we try our CNN out at across-category classification. An across-category classification is interesting for training data limited to a small pool of websites. The results of our scenario could give us hints if a general classifier for a wide range of websites can be generated from a small set of websites different in layout, style, or content. In the previous scenarios, we deducted that across-user session classification provides no real problem as our classifier is trained on image data and the session data of each user includes all websites. There were no visual surprises so to say as the image data depends on the website and is independent of the users. Instead of across-category classification, our classifier predicts labels of observations on websites completely unknown to the classifier. The visual content can differ greatly from website to website because of things like color schemes and layout. That might hinder our classifier from performing reliable predictions. We train four different classifiers each with the session data of all users on three website categories and use the session data of all users on the remaining website category as testing data. We call our across-category classifiers *CNNc* from here on. We display the F_1scores for each category used in testing in table 5. Not shown is the accuracy for each category: 72% on shopping sites, 81% on news sites, 79% on health sites, and 86% on car sites. The mean overall accuracy of all classifiers is 80%.

Let us compare our across-category classifier with our across-user session classifier *CNN3* of scenario 3. The F_1scores of *CNNc* are consistently lower and overall accuracy is around 10% worse. We notice more fluctuations of the F_1scores between *CNNc* classifiers than between the four classifiers of scenario 3. This time we do not find any correlation between performance and the number of observations used as training data. The health sites category has the least amount of observations but the scores of the classifier tested on health sites are below average. The F_1scores of visual change correlate again with the percentage of observations with visual change. The classifier tested on websites of the news site category has the lowest score of all classifiers. Accordingly, the websites of the news site category are the websites with the lowest, second-lowest, and fifth-lowest percentage of observations with visual change.

As we do not see any connections between the performance of our *CNNc* classifiers and the size of the training data set we have to look elsewhere to explain the underperformance. The data split is the obvious answer. As already explained at the beginning of this section the classifier uses image data as input and unknown websites might have unknown features that could not be learned beforehand to detect visual change. A simple example might clarify this further. If our classifier uses features based on color to detect visual changes and an unknown website uses a completely different distribution of colors the learned features might become use-

Table 5: Comparision between the different classifiers of Menges et al. and our classifier. *SVC* is a support vector classifier, *R. F.* is a random forest classifier and *B.* is the baseline classifier. *CNNc* is our convolutional neural network classifier. User sessions of three categories were used as training data and tested on the forth category shown here. The best classification result is printed in bold.

Category	Shopping				News			
Classifier	*SVC*	*R. F.*	*B.*	*CNNc*	*SVC*	*R. F.*	*B.*	*CNNc*
Visual Same [F_1]	92%	**93%**	74%	79%	**95%**	94%	82%	89%
Visual Same precision	94%	95%	**99%**	90%	96%	**97%**	**97%**	96%
Visual Same recall	90%	**92%**	59%	70%	**95%**	91%	70%	83%
Visual Change [F_1]	80%	**83%**	63%	60%	**64%**	62%	38%	44%
Visual Change precision	76%	**80%**	47%	49%	**60%**	51%	25%	33%
Visual Change recall	84%	86%	**98%**	78%	68%	78%	**83%**	70%

Category	Health				Cars			
Classifier	*SVC*	*R. F.*	*B.*	*CNNc*	*SVC*	*R. F.*	*B.*	*CNNc*
Visual Same [F_1]	80%	82%	74%	**86%**	95%	**97%**	87%	91%
Visual Same precision	89%	**96%**	91%	88%	94%	**98%**	93%	94%
Visual Same recall	72%	72%	62%	**84%**	**97%**	96%	82%	89%
Visual Change [F_1]	51%	**61%**	50%	55%	73%	**85%**	53%	62%
Visual Change precision	40%	46%	36%	**51%**	79%	**82%**	43%	55%
Visual Change recall	68%	**88%**	78%	60%	67%	**87%**	70%	71%

less or even worse counterproductive.

Let us conclude our last scenario. We have seen that across-user session classification is much easier than across-category classification. For the aspect of behavior analysis, we suggest to generate and label data on as many versatile websites as possible if a CNN is used as a classifier. Data of different users on the same websites generates not enough variety and might be even redundant due to the lack of impact we witnessed.

6 Discussion

In the discussion, we want to compare our work with related state-of-the-art first and then comment about some aspects of our work that are not directly connected to our use-cases.

Let us start by comparing our work to the state-of-the-art methods described by Menges et al. Menges et al. describe two scenarios, one with across-user session classification and one with across-category classification. In both scenarios, the preparation of image data is almost identical to ours but no fixed image size is needed as input Menges et al. crop the images where we set pixel values to zero. Menges et al. calculate computer-vision features of the pair of cropped frames for each observation. These features (see table 6, Appendix) are used as input to the classifiers.

Menges et al. list three classifiers used: a random forest classifier, a support vector classifier, and a baseline classifier. The random forest classifier uses 100 decision trees and the mode result, the most common label of all results, is used as the prediction. The support vector classifier (SVC) "uses a radial basis function kernel and balanced class weighting using the synthetic minority over-sampling technique". Both classifiers are part of machine learning as is our CNN. The baseline classifier uses the feature of the overall pixel difference between both frames as a single input and compares it to a threshold for predictions. The threshold is calculated by searching for the best achievable F_1score on the training data set. We hope to demonstrate that a CNN can be a viable alternative to classify visual change in recordings of interactions with websites.

Let us start with the across-user session classification. The session data of one user on one website is used as training data. The classifiers are then tested on the same website with the session data of the remaining three users. This is done once per user and website. An exact recreation of this method is unreasonable for our CNN for two reasons. The first reason is the lack of data. A session of one user on one website does not produce enough observations to justify the use of a CNN or guarantee a satisfying classification. The training data size of a simple CNN should be at least in the four-figure range per class. Some user website combinations did not even produce 200 observations total. The visual similarities between pairs of observations as well as the high depth of the network architecture warrant an even bigger size of input data. The second reason is time. One training process of our

CNN can take up to half a day until it is completed. To train and test one model for four users on twelve websites each would require nearly one month of non-stop computing.

Instead, we can compare the classifiers of Menges et al. with our classifier of scenario 2 where we use the session data of one user on all websites. The basic idea stays the same as we analyze if the session data of one user is enough to create a classifier working on different users. But we focus our comparison to our other classifier *CNN3* where three user sessions were used for training instead as it was the only one showing good and consistent performance.

The F_1scores after a four-fold cross validation done by Menges et al. are found in table 3 next to our classifiers. *R.F* is the random forest classifier and *B.* is the baseline classifier.

We can keep our comparison to the classifier *CNN* of scenario 2 very short. The scores can hardly keep up with the baseline classifier. We have shown several problems with our classifier which get confirmed yet again. Simply the difference in pixel values is already enough for the same and better results. As it is we can not claim our classifier to be a reasonable alternative of any kind and therefore not consider it any further.

CNN3 proves to be much more competitive. The F_1scores are better than the baseline classifier on most websites and are hardly worse on others. Compared to the *R.F* our classifier is overall still lacking. The random forest classifier outperforms our *CNN3* on almost all websites and at around half websites with a margin of over 5% at F_1scores of visual same and a larger margin on F_1scores of visual change. The F_1score of visual change of the random forest classifier seems to be influenced by the percentage of observations with visual change as well as our classifiers but in a less extreme way. *CNN3* still managed to score higher on the websites Walmart and WebMD.

Based on the comparison we would like to conclude that our convolutional neural network is a viable alternative for across-user session classification but based on the difference in used training data we assume the use as an alternative to be a very situational one only. The use of our CNN as a classifier in this scenario shows great potential but is held back by the need for more training data. State-of-the-art performance might be reached more consistently with enough training data for each class.

The second scenario of Menges et al. is an across-category classification. Training and testing data are split exactly as in our across-category classification of scenario 4 which allows us a direct comparison. According to Menges et al. an across-category classification is "the more difficult task, as we attempt to find a general classifier that is trained by visuals on Web sites which are from a category different than the classified Web sites." So far we could already confirm this statement as our *CNNc* scored worse than our *CNN3*. We compare the F_1scores of our classifier to the scores of Menges et al. classifiers *SVC*, *R.F* and *B.* in table 5.

Looking at the F_1scores our *CNNc* performs slightly better as the baseline classifier

in all cases except for visual change on shopping sites where it performs even worse. Furthermore, we can see that in most cases our *CNNc*'s F_1scores can not keep up with the support vector classifier or random forest classifier except on health sites where it is beating their visual same F_1score. Our *CNNc* has a harder time to keep up with state-of-the-art classifiers than our *CNN3*. We notice that our classifier appears to be more consistent than the other classifiers. The F_1scores of both classes of *SVC* and *R.F* have more fluctuations than the scores of our classifier.

The exception of the health sites category might showcase some problems with hand-picked features that we do not have. The strengths and weaknesses of *SVC* and *R.F* seem pretty similar. Both classifiers score above and below-average on the same website categories. We can assume that behavior is caused by the fact that both classifiers are using the same features as input. While this might work most of the time it does not for the health category. The hand-picked features are either not enough, the wrong ones or are processed wrongly by learning to detect visual change with features that do not work on health sites. The use of a convolutional neural network removes human interference which might have caused the subpar performance on health sites. The smaller fluctuations of the scores of our *CNNc* support this theory. Nonetheless, our classifier scored worse overall. Our *CNNc* can be seen as a situational alternative again. But if a convolutional neural network classifier can reach the performance of the other classifiers through more training data or another architecture it will be an alternative that should be preferred to the state-of-the-art classifiers because of the aforementioned inconsistency.

We conclude that the task of across-category classification seems to be much harder for our CNN than for state-of-the-art classifiers. But we also record that convolutional neural networks have the potential to be more than a reasonable alternative.

Now we want to take a quick look at some problems with our CNN that were encountered while working on this thesis. The execution time of the training as well as of the testing of a classifier was extremely high and ultimately the biggest obstacle. This as well as the needed hardware to run our scripts crippled any opportunity to try out a variety of different things. As soon as we found a network architecture that worked and produced results we had to stick with it. Hours of running the scripts were wasted by small coding errors that would only show on the results. The large image size as input was the main culprit of the long execution times. To avoid data loss by downsampling the images before using them as inputs we needed a very deep network to address the high dimensions. As we presented the use of our CNN as a reasonable alternative to other classifiers we referred only to the results. As long as no changes are made to reduce the execution times we can in no way suggest our CNN for the tasks it was given.

7 Conclusion

Detecting visual changes in recordings of interactions with websites is an important aspect of usability studies. For this, we introduced a convolutional neural network

to classify visual changes on a data set of user interaction data. We looked at related literature regarding the use of CNNs to detect changes in real-world videos and applied that knowledge on screencasts of users interacting with websites. We gave an overview of the data set that was used in our work and described how we prepared the data to use it for classification. We showcased the architecture of our CNN and trained a classifier with its implementation to detect visual changes in several scenarios with the aspect of similar use-cases that might occur in a production environment. The results of our classifiers were analyzed and finally compared to other state-of-the-art visual change detection methods on recordings of user interactions with a dynamic interface. Our seemingly novel approach to use a CNN on screencasts did not perform as well as the established methods but showed great potential to perform better under other circumstances.

We want to go over several ideas that might improve the performance of our CNN significantly. We start with the lack of enough training data was evident several times. Instead of creating more data by recording users on websites and labeling the observations by hand afterward, we propose the use of automatically generated data. This could be done in two ways. One or more websites could be created for labeling visual change automatically while the interactions of users with the websites are recorded, as the websites "know" when visual changes happen since the websites produce the visual change themselves. These dummy websites should cover a wide range of different looks, elements, and kinds of visual changes like they would exist on real websites. The data generated on dummy websites could be used to train a CNN in the early stages and finish its training on a smaller data set of a specific website where the classifier will be used on. In the second way to generate data, the creation of a website can be skipped completely. As we know how websites look and how visual change is defined we should be able to generate image pairs that look like a website, decide to add some elements that look the same or visually different, and label them accordingly. This also removes the need for any test users and video recordings. The only drawback is the lack of user interaction data that would no longer be assessable. That brings us to our next idea. User interaction data as additional input might improve the performance of our CNN as visual change is often a consequence of user interaction. We see two ways to include user interaction data into the convolutional neural network. The first way would be as additional inputs to the fully connected layers right after the convolutional and pooling layers. The other option is the inclusion as an additional visual input together with the frame pair. This might be done with heatmaps of mouse movements or fixations and visualized scanpaths as additional image channels. The last idea for improvement is about the CNN's architecture. The architecture we used is very slow and we were not able to try different setups. Modifications of layers, depth, and parameters as well as different activation and loss functions and optimizers could cause better results than what we were able to achieve.

References

[1] D. Mitrovic, S. Hartlieb, M. Zeppelzauer, and C. Breiteneder, "Scene segmentation in artistic archive documentaries," in *HCI in Work and Learning, Life and Leisure*, (6389), pp. 400–410, Springer-Verlag Berlin Heidelberg,, 2010. Vortrag: 6th Symposium of the Workgroup Human-Computer Interaction and Usability Engineering, USAB 2010, Klagenfurt; 2010-11-04 – 2010-11-05.

[2] M. Del Fabro and L. Böszörmenyi, "State-of-the-art and future challenges in video scene detection: a survey," *Multimedia Systems*, vol. 19, pp. 427–454, October 2013.

[3] R. Kasturi, S. H. Strayer, U. Gargi, and S. K. Antani, "An evaluation of color histogram based methods in video indexing," 1996.

[4] R. Zabih, J. Miller, and K. Mai, "A feature-based algorithm for detecting and classifying production effects," *Multimedia Systems*, vol. 7, pp. 119–128, Mar 1999.

[5] A. G. Haupmann and M. J. Witbrock, "Story segmentation and detection of commercials in broadcast news video," in *Proceedings of the Advances in Digital Libraries Conference*, ADL '98, (Washington, DC, USA), pp. 168–, IEEE Computer Society, 1998.

[6] L. Baraldi, C. Grana, and R. Cucchiara, "A deep siamese network for scene detection in broadcast videos," *Proceedings of the 23rd ACM international conference on Multimedia - MM '15*, 2015.

[7] P. Over, G. Awad, W. Kraaij, and A. Smeaton, "TRECVID 2007 - overview," in *Proceedings of TRECVID 2007*, NIST, USA, 2007.

[8] A. F. Smeaton, P. Over, and A. R. Doherty, "Video shot boundary detection: Seven years of trecvid activity," *Computer Vision and Image Understanding*, vol. 114, no. 4, pp. 411 – 418, 2010. Special issue on Image and Video Retrieval Evaluation.

[9] M. Gygli, "Ridiculously fast shot boundary detection with fully convolutional neural networks," *CoRR*, vol. abs/1705.08214, 2017.

[10] R. Menges, C. Kumar, C. Schaefer, T. Walber, and S. Staab, "What Did My Users Experience? Discovering Visual Stimuli of Dynamic User Interfaces," *ACM Trans. Comput.-Hum. Interact.* Under preparation.

[11] S. Protasov, A. M. Khan, K. Sozykin, and M. Ahmad, "Using deep features for video scene detection and annotation," *Signal, Image and Video Processing*, vol. 12, pp. 991–999, Jul 2018.

[12] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," vol. 15 of *Proceedings of Machine Learning Research*, (Fort Lauderdale, FL, USA), pp. 315–323, JMLR Workshop and Conference Proceedings, 11–13 Apr 2011.

[13] M. Valueva, N. Nagornov, P. Lyakhov, G. Valuev, and N. Chervyakov, "Application of the residue number system to reduce hardware costs of the convolutional neural network implementation," *Mathematics and Computers in Simulation*, vol. 177, pp. 232 – 243, 2020.

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, p. 84–90, May 2017.

[15] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014.

[16] R. Y. Rubinstein and D. P. Kroese, *Applications of CE to Machine Learning*, pp. 251–270. New York, NY: Springer New York, 2004.

[17] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014.

[18] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, and et al., "Scikit-learn: Machine learning in python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Nov. 2011.

8 Appendix

Table 6: Evaluated features to estimate visual change. Postfixes b, g, and r stand for the blue, green, and red color channel, respectively. Postfixes h, s, and l stand for hue, saturation, and lightness. The postfixes of the SIFT-based matches describe the applied threshold on the SIFT-feature similarity score. The postfix spatial stands for an additional check for a similar image coordinate.

Type	Value-based		Histogram-based
Feature	*Aggregation*	*Count*	*Correlation*
Variations	agg_bgr/b/g/r	count_bgr/b/g/r	corr_b/g/r
	agg_h/s/l	count_h/s/l	corr_h/s/l
	agg_gray	count_gray	corr_gray

Type	Edge-based	Signal-based	
Feature	*Change Fraction*	*PSNR*	*MSSIM*
Variations	change_fraction	psnr	mssim_b/g/r

Type	Optical-flow-based		SIFT-based
Feature	*Angle*	*Magnitude*	*Match*
Variations	angle_mean/std	mag_max/mean/std	match/_0/4/16/64/256/512
			match_spatial
			match_dist_min/max/mean/std

Type	Text-based	
Feature	*Bag of Words*	*n-Grams*
Variations	diff_words_count	n_grams_jaccard/match_count/ratio
	unique_term_count	n_grams_match_count/ratio
		n_grams_vocabulary_size