Identifying Misinformation from Website Screenshots

Sara Abdali[1], Rutuja Gurav[1], Siddharth Menon[1], Daniel Fonseca[1], Negin Entezari[1], Neil Shah[2], Evangelos E. Papalexakis [1]

[1] Department of Computer Science and Engineering University of California, Riverside
900 University Avenue, Riverside, CA, USA
[2] Snap Inc.
{sabda005,rgura001,smeno004,dfons007,nente001}@ucr.edu
epapalex@cs.ucr.edu
nshah@snap.com

Abstract

Can the look and the feel of a website give information about the trustworthiness of an article? In this paper, we propose to use a promising, yet neglected aspect in detecting the misinformativeness: the overall look of the domain webpage. To capture this overall look, we take screenshots of news articles served by either misinformative or trustworthy web domains and leverage a tensor decomposition based semi-supervised classification technique. The proposed approach i.e., `VizFake` is insensitive to a number of image transformations such as converting the image to grayscale, vectorizing the image and losing some parts of the screenshots. `VizFake` leverages a very small amount of known labels, mirroring realistic and practical scenarios, where labels (especially for known misinformative articles), are scarce and quickly become dated. The F1 score of `VizFake` on a dataset of 50k screenshots of news articles spanning more than 500 domains is roughly 85% using only 5% of ground truth labels. Furthermore, tensor representations of `VizFake`, obtained in an unsupervised manner, allow for exploratory analysis of the data that provides valuable insights into the problem. Finally, we compare `VizFake` with deep transfer learning, since it is a very popular black-box approach for image classification and also well-known text text-based methods. `VizFake` achieves competitive accuracy with deep transfer learning models while being two orders of magnitude faster and not requiring laborious hyper-parameter tuning.

Introduction

Despite the benefits that the emergence of web-based technologies has created for news and information spread, the increasing spread of fake news and misinformation due to access and public dissemination functionalities of these technologies has become increasingly apparent in recent years. Given the growing importance of the fake news detection task on web-based outlets, researchers have placed considerable effort into design and implementation of efficient methods for finding misinformation on the web, most notably via natural language processing methods (Ciampaglia et al. 2015; Rubin et al. 2016; Horne and Adali 2017; Shu et al. 2017). intended to discover misinformation via nuances in article text. article's text Although utilizing textual information is a natural approach, there are few drawbacks: most notably, such approaches require complicated and time-consuming analysis to extract linguistic, lexical, or psychological features such as sentiment, entity usage, phrasing, stance, knowledge-base grounding, etc. Moreover, the problem of identifying misinformativeness using textual cues is challenging to define well, given that each article is composed of many dependent statements (not all of which are fact-based) and editorialization. Finally, most such approaches require extremely large labeled sets of misinformative articles, which are often unavailable in practice due to lack of reliable human annotators, as well as quickly become "dated" due to shift in topics, sentiment, and reality and time itself. These article-based labels inherently result in event-specificity and bias in resulting models, which can lead to poor generalization in the future for different article

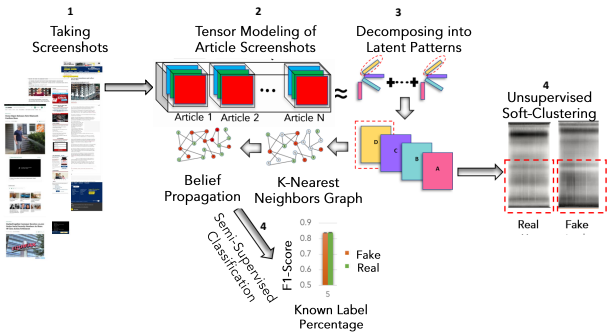


Figure 1: Creating a tensor-based model out of news articles' screenshots and decomposing the tensor using CP/PARAFAC into latent factors and then creating a nearest neighbor graph based on the similarity of latent patterns and leveraging belief propagation to propagate very few known labels throughout the graph. As illustrated, the F1 score of both real and fake classes is roughly 85% using just 5% of known labels. Moreover, VizFake has exploratory capabilities for unsupervised clustering of screenshots.

types.

In this work, we take a step back to tackle the problem with a human, rather than an algorithmic perspective. We make two choices that are not made jointly in prior work. Firstly, we tackle misinformation detection by leveraging a *domain-level* feature. Secondly, we focus on the discovery of misinformation using *visual cues* rather than textual ones. We expand upon these two points below. Firstly, leveraging domain features for misinformation detection is not only an easier but also a likely more fruitful/applicable problem setting in practice. In reality, most highly reputed news sources do not report misinformative articles due to high editorial standards, scrutiny, and expectations. For example, the public fallout from misinformation being spread through famous organizations like CNN or BBC would be disastrous. However, there are many misinformation farms and third-parties which create new domains with the intent of deceiving the public (Boatwright, Linvill, and Warren 2018). Moreover, these actors have little incentive to spread real articles in addition to fake ones. Thus, in most cases, domain feature could prove to be a better target to stymie the spread of misinformation. Conveniently, several crowd-sourced tools and fact-checkers like BS Detector [1] or Newsguard [2] provide domain-level labels rather than article-level, which we utilize here.

Secondly, visual cues are a promising, yet underserved research area, especially in the context of misinformation detection. While past literature in text-based methods in this space is rich (see (Oshikawa, Qian, and Wang 2018) for an overview), prior work on visual cues is sparse. Past works (Jin et al. 2017; Gupta et al. 2013; Sun et al. 2013) primarily focus on doctored/fake-news associated images and visual coherence of images with article text. However, since these works are limited to fake news which spreads with images, they are inapplicable for articles which do not incorporate multimedia. Moreover, these works all have inherent article specificity, and none consider the overall visual look and representation of the hosting domain or website for a given article. Intuitively and anecdotally, in contrast to unreliable sources that tend to be visually messy and full of advertisements and popups, trustworthy domains often look professional and ordered. For example, real domains often request users to agree to privacy policies, have login/signup/subscription functionalities, have multiple featured news articles clearly visible, etc. Conversely, strong tells for fake domains tend to include errors, negative space, unprofessional/hard-to-read fonts, and blog-post style (Cyr 2013; Yan, Yurchisin, and Watchravesringkan 2011; Wells, Valacich, and Hess 2011). Figure 1 demonstrates this dichotomy with a few examples. While we as humans use these signals to quickly discern the quality and reliability of news sources without delving into the depth of the text, prior works have not directly considered them. Thus, we focus on bridging this gap with the assumption that many misinformative articles do not need to be read to be suspected.

Given these two facets, we ask: *"can we identify misinformation by leveraging the visual characteristics of their domains?"* In this work, we propose an approach for classification of article screenshots using image processing approaches. In contrast to deep learning approaches such as convolutional neural networks (CNNs) which take a relatively long time to train, are data-hungry, and require careful hyperparameter tuning, we propose a novel tensor-based semi-supervised classification approach which is fast, efficient, robust to image resolution, and missing image segments, and data-limited. We demonstrate that our approach henceforth refereed to as `VizFake`, can successfully classify articles into fake or real classes with an F1 score of 85% using very few (i.e., $< 5\%$ of available labels). Summarily, our major contributions are as follows:

- **Using visual signal for modeling domain structure**: We propose to model article screenshots from different domains using a tensor-based formulation.
- **Fast and robust tensor decomposition approach for classification of visual information**: We propose a tensor-based model to find latent article patterns. We compare it against typical deep learning models. `VizFake` performs on par, while being significantly faster and needless to laborious hyperparameter tuning.
- **Unsupervised exploratory analysis**: Tensor-based representations of `VizFake` derived in an unsupervised manner, allow for interpretable exploratory analysis of the data which correlate with existing ground truth.
- **Performance in label-scarce settings**: In contrast to deep learning approaches, `VizFake` is able to classify news articles with high performance using very few labels, due to a semi-supervised belief propagation formulation.
- **Experimenting on real-world data**: We evaluate `VizFake` on a real-world dataset we constructed with over 50K news article screenshots from more than 500 domains, by extracting tweets with news article links. Our experiments suggest strong classification results (85% F1 score) with very few labels ($< 5\%$) and over two orders of speedup compared to CNN-based methods.

The remainder of this paper is organized as follows: First, the proposed `VizFake` is described. Next, we discuss the implementation details and the dataset. Afterwards, the experimental evaluation of the `VizFake` as well as variants and baselines is presented. Then, we discuss the related work, and finally we draw the conclusions.

Proposed Method

Here, we discuss our formulation and proposed semi-supervised tensor-based approach i.e., `VizFake` method.

Problem Formulation

We solve the following problem:

> **Given** (i) a collection of news domains and a number of full-page screenshots of news articles published by each domain and (ii) a small number of labels.
> **Classify** the unlabeled screenshots as misinformation or not.

[1] http://bsdetector.tech/

[2] https://www.newsguardtech.com

Semi-supervised Tensor-based Method i.e VizFake

VizFake aims to explore the predictive power of visual information about articles published by domains. As we argued above, there is empirical evidence that suggests this proposition is plausible.Thus, we introduce a novel model to leverage this visual information. We propose a tensor-based semi-supervised approach that is able to effectively extract and use the visual cue which yields highly predictive representations of screenshots, even with limited supervision, also, due to its elegant and simple nature, allows for interpretable exploration. VizFake has the following steps:

Tensor-based modeling The first step of VizFake refers to constructing a tensor-based model out of articles' screenshots. RGB digital images are made of pixels each of which is represented by three channels, i.e., red, green, and blue. So each image channel shows the intensity of the corresponding color for each pixel of the image.

A tensor is a multi-way array. We use a 4-mode tensor embedding for modeling news articles' screenshots. since each channel of an RGB digital image is a matrix, by stacking all three channels we create a 3-mode tensor for each screenshot and if we put all 3-mode tensor together, we create a 4-mode tensor out of all screenshots.

Tensor Decomposition As we mentioned above, a tensor is a multi-way array, i.e., an array with three or more dimensions. The Canonical Polyadic (CP) or PARAFAC decomposition, factorizes a tensor into a summation of R rank-one tensors. For instance, a 4-mode tensor $\boldsymbol{\mathcal{X}}$ of dimensions $I \times J \times K \times L$ is decomposed into a sum of outer products of four vectors as follows:

$$\boldsymbol{\mathcal{X}} \approx \Sigma_{r=1}^{R} \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r \circ \mathbf{d}_r$$

where $\mathbf{a}_r \in \mathbb{R}^I$, $\mathbf{b}_r \in \mathbb{R}^J$, $\mathbf{c}_r \in \mathbb{R}^K$ $\mathbf{d}_r \in \mathbb{R}^l$ and the outer product is given by (Papalexakis, Faloutsos, and Sidiropoulos 2016; Sidiropoulos et al. 2016):

$$(\mathbf{a}_r, \mathbf{b}_r, \mathbf{c}_r, \mathbf{d}_r)(i, j, k, l) = \mathbf{a}_r(i)\mathbf{b}_r(j)\mathbf{c}_r(k)\mathbf{d}_r(l) \forall i, j, k, l$$

We define the factor matrices as $\mathbf{A} = [\mathbf{a}_1\ \mathbf{a}_2 \dots \mathbf{a}_R]$, $\mathbf{B} = [\mathbf{b}_1\ \mathbf{b}_2 \dots \mathbf{b}_R]$, $\mathbf{C} = [\mathbf{c}_1\ \mathbf{c}_2 \dots \mathbf{c}_R]$ and $\mathbf{D} = [\mathbf{d}_1\ \mathbf{d}_2 \dots \mathbf{d}_R]$ where $\mathbf{A} \in \mathbb{R}^{I \times R}$, $\mathbf{B} \in \mathbb{R}^{J \times R}$, $\mathbf{C} \in \mathbb{R}^{K \times R}$ and $\mathbf{D} \in \mathbb{R}^{L \times R}$ denote the factor matrices and R is the rank of decomposition or the number of columns in the factor matrices. Moreover, the optimization problem for estimating the factor matrices is defined as follows:

$$\min_{\mathbf{A},\mathbf{B},\mathbf{C},\mathbf{D}} = \|\boldsymbol{\mathcal{X}} - \Sigma_{r=1}^{R} \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r \circ \mathbf{d}_r\|^2$$

For solving the optimization problem above we use Alternating Least Squares (ALS) which solves for any of the factor matrices by fixing the others due to simplicity and the speed of this algorithm (Papalexakis, Faloutsos, and Sidiropoulos 2016; Sidiropoulos et al. 2016).

Having the mathematical explanation above in mind, the second step of VizFake is decomposition of the proposed tensor-based model for finding the factor matrix corresponding to article mode, i.e., factor matrix $\mathbf{D}$ which comprises latent patterns of screenshots. We will leverage these latent patterns for screenshot classification.

.

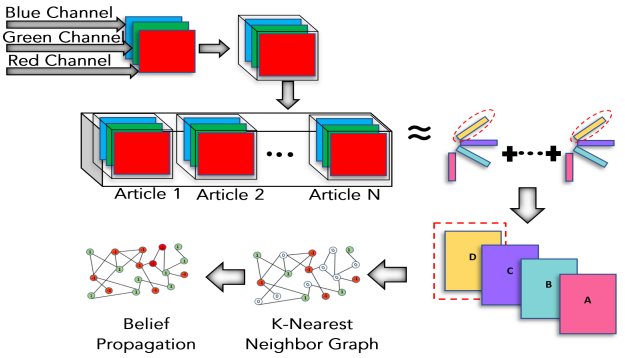


Figure 2: Proposed tensor-based modeling and semi-supervised classification of the screenshots i.e. VizFake.

Semi-supervised classification The third and last step of VizFake is the classification of news articles using the factor matrix $\mathbf{D}$ corresponding to article mode resulted from the decomposition of the tensor-based model.

As we mentioned before, each factor matrix comprises the latent patterns of the corresponding mode in $\mathbf{R}$ dimensional space. Therefore, each row of factor matrix $\mathbf{D}$ is an $\mathbf{R}$ dimensional representation of the corresponding screenshot. So, we can consider each screenshot as a data point in $\mathbf{R}$ dimensional space. We create a K-nearest neighbor graph (K-NN) Graph by considering data points as nodes, and the Euclidean distance between the nodes as edges of the graph.

Belief propagation (Braunstein, Mézard, and Zecchina 2005; Yedidia, Freeman, and Weiss 2005) is a message passing-based algorithm that is usually used for calculating the marginal distribution on graph-based models such as Bayesian networks, Markov, or K-NN graph. In this algorithm, each node of a given graph leverages the messages received from neighboring nodes to compute its belief (label) using the following iterative update rule:

$$b_i(x_i) \propto \prod_{j \in N_i} m_{j \hookrightarrow i}(x_i)$$

where $b_i(x_i)$ denotes the belief of node i, $m_{j \hookrightarrow i}(x_i)$ is a message sent from node j to node i and conveys the opinion of node j about the belief of node i, and N_i denotes all the neighboring nodes of node i (Braunstein, Mézard, and Zecchina 2005; Yedidia, Freeman, and Weiss 2005).

Since we model homophily (similarity) of screenshots patterns using a K-NN graph as explained above, we can leverage Belief Propagation in a semi-supervised manner to propagate very few available labels throughout the graph. A fast and linearized implementation of Belief propagation is proposed in (Koutra et al. 2011) which solves the following linear system:

$$[\mathbf{I} + a\mathbf{D} - c'\mathbf{A}]b_h = \phi_h$$

where ϕ_h and b_h stand for the prior and the final beliefs, respectively. $\mathbf{A}$ denotes the $n \times n$ adjacency matrix of the K-NN graph, $\mathbf{I}$ denotes the $n \times n$ identity matrix, and $\mathbf{D}$ is a $n \times n$ matrix where $\mathbf{D}_{ii} = \sum_j \mathbf{A}_{ij}$ and $\mathbf{D}_{ij} = 0$ for $i \neq j$.

a and $c^{'}$ are also defined as: $a = \frac{4h_h^2}{1-4h_h^2}$, $c' = \frac{2h_h}{(1-4h_h^2)}$ where h_h denotes the homophily factor between nodes. In fact, higher homophily corresponds to having more similar labels. Readers are referred to (Koutra et al. 2011) for more details. An overview of VizFake is depicted in Figure 2.

Experimental Evaluation

In this section, we first discuss implementation and dataset details and then report a set of experiments to investigate the effect of changing rank, resolution, and some image transformation on the performance of VizFake and then we compare it against CNN deep-learning model, text-base text-based approaches and webpage structure features.

Dataset Description

Although collecting human annotation for misinformation detection is a complicated and time-consuming task, there exist some crowd-sourced schemes such as the browser extension "BS Detector" which provide a number of label options, allowing users to label domains into different categories such as biased, clickbait, conspiracy, fake, hate, junk science, rumor, satire, unreliable, and real. We use BS Detector as our ground truth and consider all of the nine categories above but "real" as "fake" class. We reserve a more fine-grained analysis of different "fake" categories for future work (henceforth collectively refer to all of those categories of misinformation as "fake").

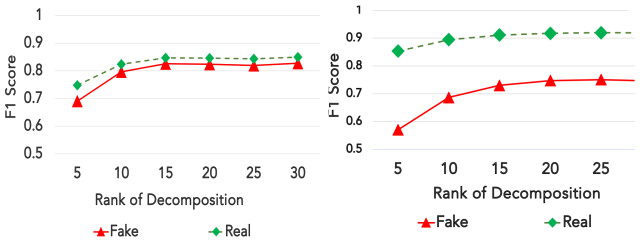
We describe our crawling process in order to promote reproducibility, as we are unable to share the data because of copyright considerations. We crawled Twitter to create a dataset out of tweets published between June and August 2017 which included links to news articles. Then, we implemented a javascript code using Node.js open source server environment and Puppeteer library for automatically taking screenshots of scrolled news articles of our collected dataset.

- we took screenshots of 50K news articles equally from more than 500 fake and real domains i.e., a balanced dataset including 50% from fake and 50% real domains.
- To investigate the effect of class imbalance, we created an imbalanced dataset of the same size, i.e., 50k but this time we selected $\frac{2}{3}$ of the screenshots from real domains and $\frac{1}{3}$ of the data from fake ones.
- Although we tried to select an equal number of articles per each domain, sometimes fake domains do not last long and the number of fake articles published by them is limited. However, we show that this limitation does not affect the classification, because the result of the fake discrimination is almost same as the real class.

Implementation Details

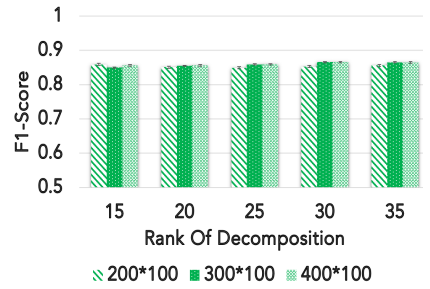
We used Matlab for implementing VizFake approach and for CP/PARAFAC decomposition we used Tensor Toolbox version 2.6 [3](Bader and Kolda 2006). For Belief Propagation, we used Fast Belief Propagation (FaBP) (Koutra et al. 2011) which is linear in the number of edges. For finding the best rank of decomposition R and the number of nearest neighbors K for both balanced and imbalanced datasets, we grid searched the values between range 5-30 for R and 1-50 for K. Based on our experiments, we set R to 15 and 25 for balanced and imbalanced datasets, respectively and set K to 20 for both datasets. We measured the effectiveness of VizFake using widely used F1 score, precision, and recall metrics. We run all of the experiments 25 times and we report the average and standard deviation of the results for all mentioned metrics. The F1 score of different ranks for balanced and imbalanced datasets and both real and fake classes is shown in Figure 3.

[3]https://www.sandia.gov/ tgkolda/TensorToolbox/index-2.6.html

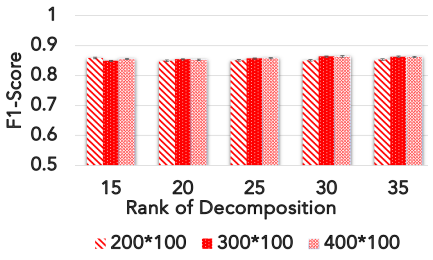


(a) F1-Balanced (b) F1-Imbalanced

Figure 3: F1 score of VizFake for different ranks when experimenting on balanced and imbalanced datasets. The best ranks for these datasets are 15 and 25 respectively.



(a) F1-Real



(b) F1-Fake

Figure 4: F1 score of VizFake for different resolutions. F1 score increases slightly when experimenting on higher resolution images.

Investigating Detection Performance

First, we aim at investigating the detection performance of VizFake in discovering misinformative articles. A caveat

in experimentation is that different articles even from the same domains may have different lengths, and thus screenshots of a fixed resolution may capture more or less information from different articles. However, fixed-resolution is an important prerogative for VizFake (and many others), thus we must use the same length for all screenshots.

Thus, we first evaluate the effect of resolution to choose a fixed setting for our model in further experiments. We experiment on screenshots of size 200×100, 300×100, and 400×100, and simultaneously evaluate the effect of different decomposition rank given the association with different amounts of information across resolutions. Figure 4 shows the detection performance (F1 scores) across the above resolution settings and differing ranks from 15-35, using 10% seed labels in the belief propagation step.

Our experiments suggest that F1 score does increase slightly with higher resolutions and decomposition ranks, but the increases are not significant. We hypothesize that the invariance to changes in resolution is due to the fact that coarse-grained features like number of ads, positions of images in the article, and the overall format of the writing is still captured even at lower resolutions and the detection is not heavily reliant on the fine-grained features of the articles as shown in Figure 5. This finding is promising, as it suggests valuable practical advantages in achieving high performance (88% F1 score) even using very low resolution or even icon size images and significant associated computational benefits. Thus, unless specified, in further experiments, we use 200×100 images.

Investigating Sensitivity to Image Transformation

Next, we investigate different image-level Transformation to evaluate performance under such settings. Firstly, we consider the importance of colors in the creation of latent patterns and the role they play in the classification task via grayscaling. Next, we explore how vectorizing the channels of color screenshots improves the performance.

We first try to convert the color screenshots into grayscale ones using the below commonly used formula in image processing tasks (Kanan and Cottrell 2012):

$$P = R \times (299/1000) + G \times (578/1000) + B \times (114/1000)$$

where P, R, G, and B are grayscale, red, green, and blue pixel values, respectively. Next, we create a 3-mode tensor from all grayscale screenshots and apply VizFake.

Likewise, to investigate the effect of vectorizing channels of color screenshots, we created another 3-mode tensor by vectorizing each channel matrix. The detection performance using grayscale and vectorized channel tensors in comparison to our standard 4-mode tensor (from color screenshots) are shown in Figure 6. Given these different input representations, we again evaluate VizFake on different rank decompositions. As shown, in contrast to grayscaling, vectorizing the channels slightly improves the F1 scores.

We hypothesize the rationale for similar grayscale performance to the base 4-mode color model is that several important aspects like number of ads, image positions, writing styles (e.g., number of columns, font) are unaffected

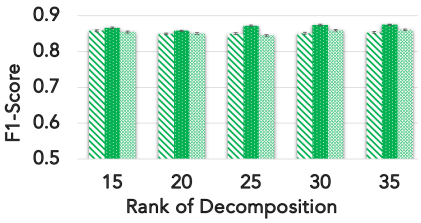
Figure 5: An example of grayscaling and changing the resolution on overall look of screenshots.

and still capture the overall look of the webpage (see Figure 5) and thus producing consistent performance. The performance improvement for vectorizaing can be explained as follows: By vectorizing an image, we treat an image as a single observation, or a point in high dimensional pixel space. As a result, we are calculating all possible combinations of pixel statistics, both near and faraway statistics. On the other hand, when we consider an image as a matrix, every different image column is treated as an independent observation, and each pixel only covaries with pixels in the rows and the columns and we are not able to capture all possible pairwise statistics. (Vasilescu 2012) offers a relevant discussion on vectorization, albeit using subspace arguments rather than latent factor imposed constraints. Overall, the minor changes in the F1 score show that VizFake is robust against common image transformations, suggesting practical performance across various color configurations and image representation schemes.

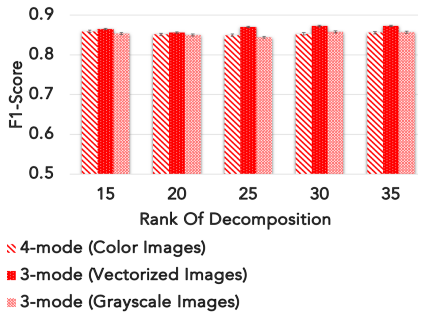
Investigating Sensitivity to Class Imbalance

Next, we investigate sensitivity of VizFake to class imbalance, as is often the case in practical settings. We create a dataset of size 50k with a 1:2 fake to real article split. We then assume that the known labels are reflective of the class distribution, and use stratified sampling to designate known labels for the belief propagation step. Figure. 7 shows the F1 scores on both balanced and imbalanced data for different percentages of known labels.

As we expect, the F1 score of the fake class drops when we have a scarcity of fake screenshots in the seed label population. Conversely, the F1 score of the real class increases in comparison to a balanced dataset due to more real samples. However, even under the scarcity of fake samples, the F1 score using just 5% of the data is around 70% and using 20% the F1 score is almost 78%, suggesting considerably strong results for this challenging task. Overall, changing the proportion of fake to real articles does expectedly impact classification performance. However, performance on the real class is actually not significantly affected.

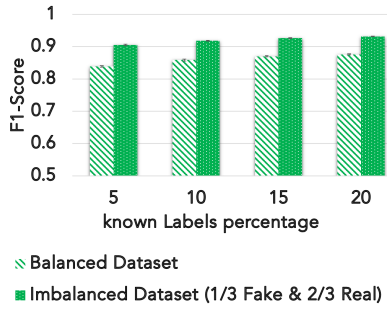


(a) F1-Real

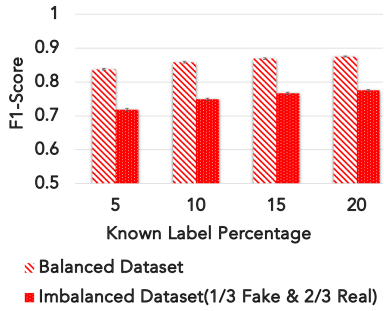


(b) F1-Fake

Figure 6: F1 score of 4-mode tensor modeling created out of color screenshots against 3-mode tensors out of vectorized and grayscale screenshots for different ranks.



(a) F1-Real



(b) F1-Fake

Figure 7: F1 score of using VizFake on an imbalanced dataset (The ratio of screenshots published by fake domains to real ones is $1:2$). On the contrary to fake class, the F1 score of real class increases due to having more samples.

Investigating Importance of Website Sections

One might ask, "which parts of the screenshots are more informative?" In other words, in which sections are the latent patterns formed? To answer these questions, we propose to cut screenshots into four sections as demonstrated in Figure 8 and use different sections or their combinations while excluding others to create the tensor model (a type of feature ablation study). We propose to create four tensors out of the top, bottom, 2 middle sections, excluding the banner and the concatenation of the top and bottom sections, respectively. For this experiment, we used the 4-mode color tensor and screenshots of size 200×100. Thus, each section is of size 50×100. Figure 9 shows F1 scores of `VizFake` on the aforementioned tensors in comparison to using complete screenshots.

The results show that by cutting the top or bottom sections of the screenshots the F1 score drops by roughly 6% and 8%, respectively. Moreover, if we cut both top and bottom sections the F1 scores decrease significantly by almost 15%. These two sections convey important information including banners, copyright signatures, sign-in forms, headline images, ads, popups, etc. We noted a considerable portion of the informativeness is included outside the banners, as the banners comprise only 10-20% of the top/bottom sections and the F1 scores when only excluding the banners are considerably better than when excluding top and bottom both. The middle sections typically consist of the text of the articles, while other article aspects like pictures, ads, and webpage boilerplate tend to be located at the top/bottom sections. Although the top/bottom sections are more informative, the two middle sections still contain important information such as the number of columns, font style, etc. because the middle sections solely, can still classify screenshots with the F1 score of 67% using just 5% of labels. By capturing all sections, we achieve significantly stronger results i.e., 83% F1 using just 5% labels. This experiment suggests that even if the screenshots are corrupted or censored for privacy considerations e.g., excluding headers and other obvious website tells, we are still capable of identifying fake/real domains using as little as 50% of the underlying images.

Comparing Against Deep-learning Models

A very reasonable first attempt at classification of screenshots, given their wide success in many computer vision tasks, is the use of Deep Convolutional Neural Networks (CNNs). To understand whether or not CNNs are able to capture hidden features that `VizFake` scheme cannot extract, we also try CNNs for classification of screenshots. From a pragmatic point of view, we compare i) the classification results each method achieves, and ii) the runtime required to train the model in each case. In what follows, we discuss the implementation details.

`VizFake` configuration We showed that the vectorized tensor outperforms 3-mode grayscale and 4-mode color ten-

	Fake Class					
	VizFake			VGG16 deep network		
%labels	F1	Precision	Recall	F1	Precision	Recall
5	**0.852±0.002**	0.860±0.005	0.844±0.004	0.799±0.008	0.823±0.027	0.779±0.039
10	**0.871±0.001**	0.880±0.003	0.863±0.005	0.816±0.003	0.842±0.014	0.793±0.018
15	**0.881±0.001**	0.890±0.002	0.873±0.003	0.837±0.001	0.883±0.009	0.795±0.009
20	**0.888±0.001**	0.896±0.002	0.880±0.003	0.849±0.009	0.884±0.023	0.818±0.034

Table 1: VizFake outperforms VGG16 when classifying fake class e.g., F1 score (> 0.85) with only 5% of labels.

	Real Class					
	VizFake			VGG16 deep network		
%labels	F1	Precision	Recall	F1	Precision	Recall
5	**0.854±0.003**	0.847±0.003	0.862±0.006	0.809±0.007	0.790±0.021	0.830±0.039
10	**0.874±0.001**	0.865±0.004	0.882±0.004	0.827±0.003	0.804±0.010	0.851±0.019
15	**0.884±0.001**	0.876±0.002	0.892±0.003	0.852±0.002	0.813±0.005	0.894±0.010
20	**0.890±0.001**	0.882±0.003	0.898±0.003	0.860±0.005	0.831±0.021	0.892±0.029

Table 2: VizFake outperforms VGG16 when classifying real class e.g., F1 score (> 0.85) with only 5% of labels.

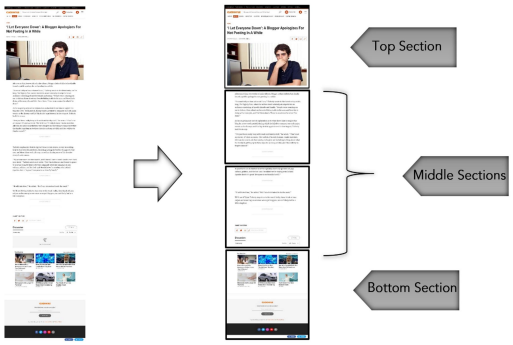


Figure 8: Cutting a screenshot into four sections.

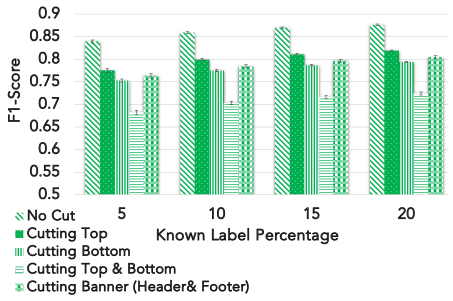
sors. So, we choose the 3-mode tensor as tensor model. We use the balanced dataset comprising 50k screenshots with resolution of 200×100 and finally we set the rank to 35 based on what is in Figure. 6.

Deep learning configuration Although our modest-sized dataset has considerable examples per class (25k), it is not of the required scale for current deep models; thus, we resort to deep transfer learning (Pan and Yang 2009).

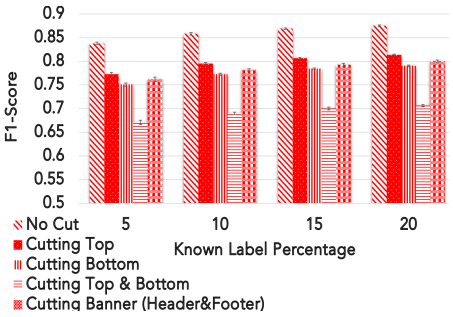
We choose VGG16 (Simonyan and Zisserman 2014) pre-trained on ImageNet (Deng et al. 2009) as our base convolutional network and modify the final fully connected layers to suite our binary classification task.We also tried some other models, they all basically perform similarly. The performance we got was indicative and also was on par with other models. So, we just report the results for VGG16 which is robust enough and the hyper-parameter optimization process is feasible in terms of time and available resources. The network is subsequently fine-tuned on screenshot images. Due to label scarcity, we want to see if the deep network performs as well as `VizFake` when there is a limited amount of labels. Thus, we experiment by fine-tuning on the same label percentage we use for `VizFake`. The remaining images are used for validation and testing.

We use the Adam optimizer (Kingma and Ba 2014) and search between 0.0001 and 0.01 for the initial learning rate. We apply sigmoid activation in the output layer of the network and the binary cross-entropy as the loss function. The batch sizes we experiment with ranged from 32 to 512 and we finally fixed the batch size for all experiments to 512. Batch size significantly impacts learning as a large enough batch size provides a stable estimate of the gradient for the whole dataset. (Smith et al. 2017; Hoffer, Hubara, and Soudry 2017). The convergence takes approximately 50 epochs. We note that the effort required to fine-tune a deep network for this task was tedious and included manual trial-and-error, while `VizFake` requires the determination of just 2 parameters, both of which produce stable performance across a reasonable range.

Comparing classification performance Next, we compare the classification performance of `VizFake` against the CNN method we explained above in terms of precision, recall, and F1 score. Tables 1 and 2 show the achieved results of these metrics for `VizFake` and CNN model. As demonstrated, `VizFake` outperforms CNN especially given less labeled data. For instance, the F1 scores of `VizFake` for the fake class when we use only 5%-10% of the labels are 85%-87%, respectively which is 5-6% higher than the 80%-81% F1 scores from the CNN model. Thus, `VizFake` achieves better performance while avoiding considerable time in finding optimal hyperparameters required for tuning VGG16.



(a) F1-Real



(b) F1-Fake

Figure 9: Changes in the F1 score when cutting different sections of the screenshots. In contrast to 2 middle sections, Cutting the top and bottom sections causes a considerable decrease in F1 scores. It seems that style defining style-defining events of the webpages are mostly focused in the top and bottom sections of the webpages.

Comparing the time efficiency We evaluate time efficiency by measuring the runtime each method requires to achieve the best results. We experiment on two settings:

The first one uses a GPU since CNN training is an intensive and time-consuming phase which typically requires performant hardware. Although using a GPU-based framework is not necessary for `VizFake`, we re-implemented `VizFake` on the same setting we use for the deep learning model to leverage the same scheme, i.e. Python using TensorLy library (Kossaifi et al. 2019) with TensorFlow backend. Thus, we avoid influence from factors like programming language, hardware configuration, etc.

The second configuration uses a CPU and is the one we used in prior experiments and discussed in the Implementation section. Since we are not able to train the CNN model with this configuration due to excessively long runtime, we only report them for `VizFake`.

For both experiments, we measure the runtime of bottlenecks, i.e., decomposition of `VizFake` and training phase of the deep learning method. Other steps such as: K-NN graph construction, belief propagation, and test phase for CNN method are relatively fast and have negligible runtimes (e.g. construction and propagation for the K-NN graph with 50K screenshots take just 3-4 seconds). Due to our limited GPU memory, we experiment using a 5% fraction of the dataset for the GPU configuration. By doing so, we also reduce the I/O overhead that may be counted as execution time when we have to read the dataset in bashes. However, we use 100% of the dataset for the CPU setting. The technical aspects of each configuration are as follows:

Resolution	Avg.# Iter.	Avg. Time/Iter.	Avg. Time
200×100	7.64	23.76s	181.55s
300×100	7.88	35.52s	279.95s
400×100	7.72	47.82s	369.22s

Table 3: Execution time (Sec.) of VizFake for different resolutions on configuration 2

Method	Avg. # Iter.	Avg. time/Iter.	Avg. Time
VizFake	**7.08**	**1.05s**	**7.64s**
CNN	50	33.08s	1654s

Table 4: Execution times (Sec.) of VizFake and CNN deep learning model on configuration 1.

Configuration 1:

- Keras API for Tensorflow in Python to train the deep network and Python using Tensorly with TensorFlow backend for `VizFake`.
- 2 Nvidia Titan Xp GPUs (12 GB)
- Training: 5% (2500 screenshots of size 200×100), validation: 4% (2000 screenshots)
- Decomposition: 5% (2500 screenshots of size 200×100)

Configuration 2:

- Matlab Tensor Toolbox 2.6
- CPU: Intel(R) Core(TM) i5-8600K CPU @ 3.60GHz
- Decomposition: 100% (50K screenshots)

The average number of iterations, time per iteration, and average total time for 10 runs of both methods on Configuration 1 and the same metrics for `VizFake` on Configuration 2 are reported in Tables 4 and 3, respectively.

Based on execution times demonstrated in Table 4, the tensor-based method is roughly 216 and 31.5 times faster than the deep learning method in terms of average time and average time per iteration, respectively. Moreover, the iterations required for `VizFake` is almost 7 times less than the epochs required for the CNN method. Note that these results are very conservative estimates since we do not consider time spent tuning CNN hyperparameters in this evaluation. Table 3 shows the execution time for `VizFake` on Configuration 2. Decomposing a tensor of 50k color screenshots using CPU is roughly 3 Mins for screenshots of size 200×100, increasing to 6 Mins for larger tensors.

Overall, the results suggest that `VizFake` is *2 orders of magnitude faster* than the state-of-the-art deep transfer learning method for the application at hand, and generally more "user-friendly" for real-world deployment.

	Fake Class				
%labels	**TF-IDF/SVM**	**Doc2Vec/SVM**	**GloVe/LSTM**	**FastText**	**VizFake**
5	0.812±0.005	0.511±0.000	0.651±0.019	0.717±0.010	**0.844±0.004**
10	0.828±0.001	0.530±0.004	0.672±0.024	0.748±0.007	**0.863±0.005**
15	0.836±0.002	0.540±0.004	0.699±0.020	0.757±0.006	**0.873±0.003**
20	0.841±0.001	0.546±0.002	0.718±0.002	0.758±0.004	**0.880±0.003**

Table 5: The F1 score of VizFake for fake class, outperforms the F1 score of state of the art text-based approaches.

	Real Class				
%labels	**TF-IDF/SVM**	**Doc2Vec/SVM**	**GloVe/LSTM**	**FastText**	**VizFake**
5	0.814±0.004	0.511±0.000	0.650± 0.028	0.650± 0.030	**0.862±0.006**
10	0.829±0.005	0.520±0.001	0.680±0.005	0.707± 0.016	**0.882±0.004**
15	0.836±0.003	0.526±0.002	0.698±0.013	0.712±0.010	**0.892±0.003**
20	0.842±0.001	0.534±0.006	0.712±0.009	0.728±0.009	**0.898±0.003**

Table 6: The F1 score of VizFake for real class, outperforms the F1 score of state of the art text-based approaches.

Comparing Against Text-based Methods

Even though the main goal of this work is to explore whether or not we can leverage the overall look of the serving webpage to discriminate misinformation, we compare the classification performance of `VizFake` with some well-known text-based approaches to investigate how successful is the proposed approach in comparison to these widely used methods. We compare against:

- **`TF-IDF`** term frequency–inverse document frequency method is one of the widely used methods for document classification. `TF-IDF` models the importance of words in documents. We create a `TF-IDF` model out of screenshots text and then we leverage SVM for classification.
- **`Doc2Vec`** a shallow 2-layers neural network proposed by Google (Le and Mikolov 2014). `Doc2Vec` is an extension to word2vec and generate vectors for documents. Again, we use SVM classifier.[4]
- **`FastText`** a proposed NLP library by Facebook Research. `FastText` learns the word representations which can be used for text classification. It is shown that the accuracy of `FastText` is comparable to deep learning models but is considerably faster than deep competitors[5](Bojanowski et al. 2016).
- **`GloVe/LSTM`** a linear vector representation of the words using an aggregated global word-word co-occurrence. We create a dictionary of unique words and leverage Glove to map indices of words into a pre-trained word embedding(Lin et al. 2017). Finally, we leverage a LSTM classifier[6] pre-trained on IMDB and fine-tune it on our dataset. We examined embedding length in range 50-300 and finally set it to 300. The tuned batch size and hidden size are 256, 64 respectively.

The experimental results of the aforementioned methods are given in Table. 5 and Table. 6. As demonstrated, the classification performance of `VizFake` reported in these tables, outperforms the performance of the shallow network approaches i.e., `Doc2Vec` and `FastText` as well as the deep network approach i.e., `GloVe/LSTM` which shows the capability of `VizFake` in comparison to neural network methods in settings that there is a scarcity of labels. The `TF-IDF` representation along with SVM classifier leads to classification performance close to the proposed visual approach which illustrates that visual information of the publishers is as discriminative as the best text-based approaches.

Comparing Against Website Structure Features

A question that may come to mind is "why not using website features instead of screenshots?" To address this question, we repeat the proposed pipeline i.e., decomposition, K-NN graph, and belief propagation this time using HTML tags crawled from the serving webpages. To this end, we create an article/tags matrix then we decompose this matrix using Singular Value Decomposition ($\mathbf{X} \approx \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$) and leverage matrix $\mathbf{U}$ which corresponds to articles pattern to create a K-NN graph and propagate the labels using FaBP. The result of this experiment is given in Table.7. As illustrated in Table.7, using HTML tags is highly predictive which is another justification for using the overall look of the webpages. The question raises now is that "Why not just using website features for capturing the overall look, especially when the classification performance is better?" Here is some reasons for using screenshots instead of website features:

- HTML source of the domain is not always available or even if we gain access to the source, the page may be generated dynamically and as a result, the features that can be informative are probably non-accessible scripted content. This is why the HTML source of our dataset provided us with features mainly related to the high-level structure of the domain shared between different screenshots.
- HTML feature extraction requires tedious web crawling and data cleaning processes and is difficult to separate

[4] https://github.com/seyedsaeidmasoumzadeh/Binary-Text-Classification-Doc2vec-SVM

[5] https://github.com/facebookresearch/fastText

[6] https://github.com/prakashpandey9/Text-Classification-Pytorch

%labels	5	10	15
Fake	0.977±0.0004	0.983±0.0002	0.985±0.0002
Real	0.977±0.0004	0.983±0.0003	0.985±0.0002

Table 7: Performing proposed pipeline on HTML-Tags of articles. The result justifies that HTMLs only contain domain features which is shared between all articles of that domain.

useful features from useless ones. Taking screenshots is easy and can be done fast and online needless to extra resources or expert knowledge for web crawling.

- Even if we have access to the HTML source and be able to separate useful features in an efficient way, these features do not give us any information about the content of the web events such as images, videos, ads, etc. If we are to conduct article-level labeling or even section level labeling (usually just some part of an article is misinformative) we will miss a lot of useful information when we use HTML features while screenshots capture such details.

Given the reasons above, the screenshots are not only as informative as textual content, but also are preferred over time-consuming and often less informative HTML features.

Exploratory Analysis

The tensor representation of `VizFake` is not only highly predictive in semi-supervised settings, but also lends itself to exploratory analysis, due to the ease of interpretability of the decomposition factors. In this section, we leverage those factors in order to cluster domains into coherent categories (misinformative or not), in an unsupervised fashion. Each column of the screenshot embedding $\mathbf{C}$ indicates the membership of each screenshot to a cluster, defined by each of the rank-one components (for details on how to generally interpret CP factors as clustering, see (Papalexakis, Faloutsos, and Sidiropoulos 2016)). Each one of the clusters has a representative latent image, which captures the overall intensity in different parts of the image indicating regions of interest that are participating in generating that cluster. To obtain this image, we compute the outer product of column vectors of matrices corresponding to pixels and channels i.e., $\mathbf{A}$ and $\mathbf{B}$ for the vectorized tensor and scale it to range 0-255 which provides us with R latent images. We then annotate the images based on the ground truth only to verify that the coherent clusters correspond to fake or real examples. We investigate the interpretability of these latent images by taking the 90th percentile majority vote from the labels of articles with high score in that latent factor. The details of clustering approach is demonstrated in Algorithm. 1.

Examples of latent images corresponding to misinformative and real classes are illustrated in Figure 10. The darker a location of an image, the higher degree of "activity" it exhibits with respect to that latent pattern. We may view those latent images as "masks" that identify locations of interest within the screenshots in the original pixel space. In Figure 10, we observe that latent images corresponding to real clusters appear to have lighter pixels, indicating little "activity" in those locations. For example, the two latent images

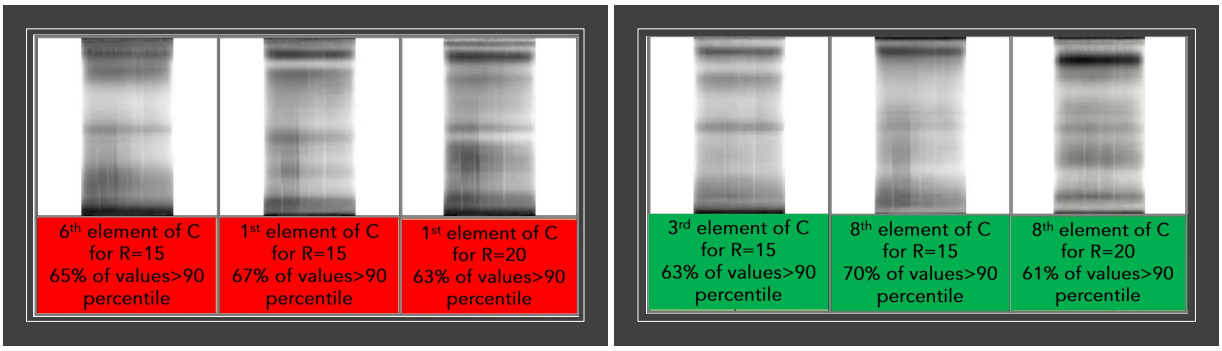
Algorithm 1: Exploratory analysis

Input: A, B and C Factor Matrices
Result: Latent pattern images
\\ scale the result to values between 0-255
$min = 0; max = 255$
$a_{ij} = \frac{(a_{ij} - min(a_{ij}) \times (max - min)}{(max(a_{ij}) - min(aij))} + min$
$b_{ij} = \frac{(b_{ij} - min(b_{ij}) \times (max - min)}{(max(b_{ij}) - min(bij))} + min$
for $i = 1 \cdots R$ **do**
    $\mathbf{X}^i_{cumulative} \approx \mathbf{a}_i \circ \mathbf{b}_i$
    ${top_n}^i$ = top $(100 - \alpha)$ percentile values c_i
    $\mathbf{X}^i_{cumulative}$ =Label-majority-Vote(${top_n}^i$)
end

resulted from rank 15 decomposition are lighter than latent images for the fake class, also the same holds for rank 20. Moreover, as illustrated in Figure 10, darker pixels are more concentrated at the top and the bottom parts of the images which are wider for misinformative patterns and corroborate our assumption about having more objects, such as ads and pop-ups, in fake news websites. As mentioned, such objects are more prevalent at the top and the bottom of the websites which matches our observation here and the cutting observation we discussed earlier. As shown in Figure 9, cutting the bottom and top sections lead to more significant changes in performance than cutting just the banner which also confirms our assumption about informativeness of these sections. This experiment not only provides us with a clustering approach which is obtained without labels and correlates with existing ground truth but also enables us to define filters for misinformation pattern recognition tasks in form of binary masks, that identify locations of interest within a screenshot, which can further focus our analysis.

Limitations of the Work

As discussed earlier, collecting annotation for misinformation detection is a complicated and time-consuming task and as we increase the granularity of the labels from domain level to articles level and even article sections it becomes harder and harder. Moreover, the majority of available ground truth resources like "BS Detector" or "NewsGuard" provide labels pertain to domains rather than articles. Despite this disparity, it is shown in several works (Helmstetter and Paulheim 2018; Zhou 2017) that the weakly-supervised task of using labels pertaining to domains, and subsequently testing on labels pertaining to articles, yields negligible accuracy loss due to the strong correlation between the two targets. However, as mentioned in the webpage structure section, there are useful article-level information like web events content that can be taken advantage of when we have grainier labels and capturing them causes a drop in performance because they may be considered as noise when working with domain level labels. We defer the study of obtaining and using finer-grained labels for future work.



(a) Misinformative latent pattern images

(b) Real latent pattern images

Figure 10: Examples of the cumulative structures of all articles corresponding to factors with the majority of misinformative/real labels. Contrary to the real class, images of misinformative class have darker pixels i.e., the dark portion of the image is wider.

Related Work

Visual-based Misinformation Detection

The majority of work proposed so far focus on content-based or social-based information. However, there are few studies on visual information of articles. For instance, in (Ringel Morris et al. 2012; Gupta, Zhao, and Han 2012) the authors consider user image as a feature to investigate the credibility of the tweets. In another work, Jin et al. (Jin et al. 2017) define clarity, coherence, similarity distribution, diversity, and visual clustering scores to verify microblogs news, based on the distribution, coherency, similarity, and diversity of images within microblog posts. In (Sun et al. 2013) authors find outdated images for the detection of unmatched text and pictures of rumors. Gupta et al. in (Gupta et al. 2013) classify fake images on Twitter using a characterization analysis to understand the temporal, social reputation of images. On the contrary, we do not focus on the user aspect, i.e., profile image or metadata within a post e.g., image, video, etc. Thus, no matter if there is any images or not, `VizFake` captures the overall look of the article.

Tensor-based Misinformation Detection

There are some studies on fake news detection which leverage tensor-based models. For example, in (Hosseinimotlagh and Papalexakis 2017; Guacho et al. 2018) the authors model content-based information using tensor embedding and try to discriminate misinformation in an unsupervised or semi-supervised regime. In this paper, rather than using article text, we leverage tensors to model article images. Although `VizFake` is able to capture the textual look of the article, we are not using time-consuming text analysis and we leverage all features of the article such as text, metadata, domain, etc., when we capture the screenshot of the page.

Conclusions

In this paper, we leverage a very important yet neglected feature for detecting misinformation, i.e., the overall look of serving domain. We propose a tensor-based model and semi-supervised classification pipeline i.e., `VizFake` which outperforms text-based methods and state-of-the-art deep learning models and is over 200 times faster, while also being easier to fine-tune and more practical. Moreover, `VizFake` is resistant to some common image transformations like grayscaling and changing the resolution, as well as partial corruptions of the image. Furthermore, `VizFake` has exploratory capabilities i.e., it can be used for unsupervised soft-clustering of the articles. `VizFake` achieves F1 score of roughly 85% using only 5% of labels for both real and fake classes on a balanced dataset and an F1 score of roughly 95% for real class and 78% for the fake class using only 20% of ground truth on a highly imbalanced dataset.

Acknowledgements

The authors would like to thank Gisel Bastidas for her invaluable help with data collection. Research was supported by a UCR Regents Faculty Fellowship, a gift from Snap Inc., the Department of the Navy, Naval Engineering Education Consortium under award no. N00174-17-1-0005, and the National Science Foundation CDS&E Grant no. OAC-1808591. The GPUs used for this research were donated by the NVIDIA Corp. Any opinions, findings, and conclusions expressed in this paper are those of the author(s) and do not necessarily reflect the views of the funding parties.

References

Bader, B. W., and Kolda, T. G. 2006. Algorithm 862: MATLAB tensor classes for fast algorithm prototyping. *Transactions on Mathematical Software* 32(4):635–653.

Boatwright, B. C.; Linvill, D. L.; and Warren, P. L. 2018. Troll factories: The internet research agency and state-sponsored agenda building. *Resource Centre on Media Freedom in Europe*.

Bojanowski, P.; Grave, E.; Joulin, A.; and Mikolov, T. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Braunstein, A.; Mézard, M.; and Zecchina, R. 2005. Survey propagation: An algorithm for satisfiability. *Random Struct. Algorithms* 27(2):201–226.

Ciampaglia, G. L.; Shiralkar, P.; Rocha, L. M.; Bollen, J.; Menczer, F.; and Flammini, A. 2015. Computational fact checking from knowledge networks. *PloS one* 10(6):e0128193.

Cyr, D. 2013. Website design, trust and culture: An eight country investigation. *ECRA* 12.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. Imagenet: A large-scale hierarchical image database. 248–255.

Guacho, G. B.; Abdali, S.; Shah, N.; and Papalexakis, E. E. 2018. Semi-supervised content-based detection of misinformation via tensor embeddings. 322–325.

Gupta, A.; Lamba, H.; Kumaraguru, P.; and Joshi, A. 2013. Faking sandy: characterizing and identifying fake images on twitter during hurricane sandy. *WWW 2013 Companion - Proceedings of the 22nd International Conference on World Wide Web* 729–736.

Gupta, M.; Zhao, P.; and Han, J. 2012. Evaluating event credibility on twitter. *SIAM International Conference In Data Mining* 153–164.

Helmstetter, S., and Paulheim, H. 2018. Weakly supervised learning for fake news detection on twitter. 274–277.

Hoffer, E.; Hubara, I.; and Soudry, D. 2017. Train longer, generalize better: closing the generalization gap in large batch training of neural networks. 1731–1741.

Horne, B. D., and Adali, S. 2017. This just in: Fake news packs a lot in title, uses simpler, repetitive content in text body, more similar to satire than real news. *CoRR* abs/1703.09398.

Hosseinimotlagh, S., and Papalexakis, E. E. 2017. Unsupervised content-based identification of fake news articles with tensor decomposition ensembles.

Jin, Z.; Cao, J.; Zhang, Y.; Zhou, J.; and Tian, Q. 2017. Novel visual and statistical image features for microblogs news verification. *Transactions on Multimedia* 19(3):598–608.

Kanan, C., and Cottrell, G. 2012. Color-to-grayscale: Does the method matter in image recognition? *PloS* 7:e29740.

Kingma, D. P., and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv:1412.6980*.

Kossaifi, J.; Panagakis, Y.; Anandkumar, A.; and Pantic, M. 2019. Tensorly: Tensor learning in python. *The Journal of Machine Learning Research* 20(1):925–930.

Koutra, D.; Ke, T.-Y.; Kang, U.; Chau, D.; Pao, H.-K.; and Faloutsos, C. 2011. Unifying Guilt-by-Association Approaches: Theorems and Fast Algorithms. In *ECML/PKDD*, volume 6912 of *Lecture Notes in Computer Science*. 245–260.

Le, Q., and Mikolov, T. 2014. Distributed representations of sentences and documents. *ICML 2014* 4.

Lin, Z.; Feng, M.; dos Santos, C. N.; Yu, M.; Xiang, B.; Zhou, B.; and Bengio, Y. 2017. A structured self-attentive sentence embedding. *ICLR* abs/1703.03130.

Oshikawa, R.; Qian, J.; and Wang, W. Y. 2018. A survey on natural language processing for fake news detection. *arXiv:1811.00770*.

Pan, S. J., and Yang, Q. 2009. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering* 22(10):1345–1359.

Papalexakis, E. E.; Faloutsos, C.; and Sidiropoulos, N. D. 2016. Tensors for data mining and data fusion: Models, applications, and scalable algorithms. *ACM Trans. Intell. Syst. Technol.* 8(2):16:1–16:44.

Ringel Morris, M.; Counts, S.; Roseway, A.; Hoff, A.; and Schwarz, J. 2012. Tweeting is believing?: Understanding microblog credibility perceptions. *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work* 441–450.

Rubin, V. L.; Conroy, N. J.; Chen, Y.; and Cornwell, S. 2016. Fake news or truth? using satirical cues to detect potentially misleading news.

Shu, K.; Sliva, A.; Wang, S.; Tang, J.; and Liu, H. 2017. Fake news detection on social media: A data mining perspective. *KDD*.

Sidiropoulos, N.; De Lathauwer, L.; Fu, X.; Huang, K.; Papalexakis, E.; and Faloutsos, C. 2016. Tensor decomposition for signal processing and machine learning. *IEEE Transactions on Signal Processing*.

Simonyan, K., and Zisserman, A. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv:1409.1556*.

Smith, S. L.; Kindermans, P.-J.; Ying, C.; and Le, Q. V. 2017. Don't decay the learning rate, increase the batch size. *arXiv:1711.00489*.

Sun, S.; Liu, H.; He, J.; and Du, X. 2013. Detecting event rumors on sina weibo automatically. 120–131.

Vasilescu, M. A. O. 2012. *A Multilinear (Tensor) Algebraic Framework for Computer Graphics, Computer Vision and Machine Learning*. Ph.D. Dissertation, Citeseer.

Wells, J.; Valacich, J.; and Hess, T. 2011. What signal are you sending? how website quality influences perceptions of product quality and purchase intentions. *MIS Quarterly* 35:373–396.

Yan, R.-N.; Yurchisin, J.; and Watchravesringkan, K. 2011. Does formality matter?: Effects of employee clothing formality on consumers' service quality expectations and store image perceptions. *Retail & Distribution Management* 39:346–362.

Yedidia, J.; Freeman, W.; and Weiss, Y. 2005. Constructing free-energy approximations and generalized belief propagation algorithms. *Information Theory* 51:2282 – 2312.

Zhou, Z.-H. 2017. A brief introduction to weakly supervised learning. *National Science Review* 5.