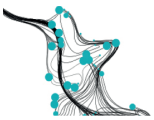

Academic Research Skills
Conference Proceedings

Classification of website screenshot data using Neural Networks

Elitsa Marinova, Camiel Verdult and Raffaele Sommese

Pre-master track: CS and IST

Submisstion date: December 15, 2023

This paper examines the training a neural network on a website's front page and its categories through the use of a Neural Networks. Currently, there are ways of classifying websites such as scraping text and classifying it with a Large Language Model, scraping media and classifying it with a Convolutional Neural Network or a Recurrent Neural Network for video or audio. As for now, there is missing research on classifying websites entirely based on their front page. This paper focuses on experimenting with classifying websites based on their front page screenshots using a Convolutional Neural Network, describes the tools that were used and the results from a trained model with a discussion and interpretation.
Keywords: classification, CNN, screenshot, website, neural network

1. Introduction

The internet is a vast space in today's digital world where millions of websites compete for the attention of the user. In the future, these websites could be categorized not just by text or metadata, but also by the visual fingerprint that is captured in a single front-page screenshot.

This paper researches the use of neural network models to classify website topics by use of a screenshot of their front page. Training a neural network with screenshots of domains with known topics will allow the network to find correlations between website appearances and website topics. When giving the network a domain that was not used in training, it will try to recognize correlations found during training. Detected correlations would be the classification result returned by the neural network. Classification results from the neural network are the labeled topics assigned to the screenshot training data. The most probable classifications are topics inferred from a screenshot of the domain (that the neural network has learned to recognize during training). Classifying websites based on browser appearance allows circumvention of obfuscation and search engine ranking[1] techniques used by websites. Using a browser to take the screenshot provides a image of the website as a user would see it.

2. Background

A literature review has been conducted before starting the writing of this article in order to create a better understanding of the existing research. There are many articles that allow us to learn about understanding the concept of neural networks, their boundaries and limitations. One of the points concluded from the literature review is that most often websites are classified based on textual data rather than on visual data[2]. This research attempts to gain perspective on the other side of the aisle. Research on Convolutional Neural Networks has shown that this deep neural network architecture is a good candidate for processing images[3] due to the ability "to handle a huge amount of data"[4]. Understanding deep learning models is difficult, since they have so many layers. One good explanation is given by the article that promotes the use of convolutional neural networks in large data scenarios:

> One of the most popular deep neural networks is the Convolutional Neural Network (CNN). It take this name from mathematical linear operation between matrixes called convolution.

The convolution operation by definition is an integral transform which shows the influence of one function on another function. The definition for the transform is given as the following definite integral:

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)\,d\tau$$

Looking at the operation from a high level and beyond the integral definition provides more context on why this technique is important for machine learning. The convolution transformation moves a filter over input data, such as a screenshot, and applies the mentioned integral systematically. This summing process is called convolution and can be used to smooth, blend and detect edges. Pieces of the input are multiplied element-wise by a filter matrix. All values are summed up to create feature maps. Feature maps represent attributes spotted in input like edges or textures. Deep learning networks applying the CNN architecture consist of several convolutional layers stacked together. Each layer extracts more complex features than the last one. The early layers capture simple attributes such as lines or gradients. Deeper layers can recognize more complicated patterns and shapes within images. Convolutional Neural Networks (CNNs) use a technique called pooling after processing images. Pooling reduces the image size and complexity, making it easier for the image to be analyzed. Following pooling, CNNs use fully connected layers, which consider all these simplified features together, to accurately classify or identify what's in the image. The architecture performs well in recognizing faces or objects. Results will review how accurate a CNN can be in finding a correlation between characteristics of a website based on a screenshot and the corresponding topics.

3. Methodology

The discrete research task is the classification of screenshots of websites. This task was conducted on thousands of website (exact number to follow) screenshots to build a set of training data. CNN models are trained on website screenshots in order to infer a neural network that could classify websites into topics. The data was gathered by a proof of concept program that creates screenshots of the most visited websites.

3.1. Data set

The final choice of a labeling tool was made after careful consideration, a few trials and with advice from the researcher. We ended up using Cloudflare[5] which provides the top 100 visited domains in a CSV file per country. The CSV file contains a list of entries with the domain and website category and is used as a data source gathering screenshots and training with the corresponding screenshot label. We chose 27 different countries to get the top ranked websites and labels of. The countries were chosen with the intent to include countries from all continents with various alphabets. This could hopefully result in a more robust model. Our proof of concept program was updated to combine all of the Cloudflare CSV files into one and remove reoccurring websites. The distinct domains and labels are inserted into an SQLite database. These websites are then processed by the proof of concept as described in the next subsection.

Before settling with the Cloudflare CSV, other options were discussed and attempted. Tranco is a website that provides a CSV file with popularity ranking and domain entries that provides "a Research-Oriented Top Sites Ranking Hardened Against Manipulation"[6]. The Tranco list was used as the first option to obtain the top ranked websites worldwide. Initially, the labeling of data was performed manually, since the Tranco list provides no

categories per website. This was done by creating a 'categories' file with category numbers mapped to the category name. We would have another 'labels' file with domain entries and one or multiple topics per domain. This approach was not preferred since the process was very time inefficient and susceptible to bias by the person adding the label.

In order to combat the time inefficiency issue, the Cyren API[7] seemed to provide an alternative to manual labelling. This API can provide topics for the websites in the training dataset. Upon investigating the usability of the API in this research, the platform appeared to be usable in this project. As it appeared, the API from Cyren had a limited number of free requests, more specifically - 1000. This is the bare minimum for a well trained model so we attempted to acquire an API key to automate the labelling process in the proof of concept. After making an account, we received an inquiry from the sales team and were informed that we could not use their platform for our application. Furthermore, we investigated more options of API's to provide us with labels but nothing was suitable to our requirements - at least 1000 free requests, thus the choice of Cloudflare's labeled CSV files. The Cloudflare CSV's ended up in 410 labeled domains, which is less than the amount we aimed for. Due to time constraints, we ended up using this as a data set.

3.2. Gathering screenshots for training

A proof of concept program that retrieves screenshots of the most popular websites is written in Python. Python provides and has access to many libraries that allow for quick prototyping and abstraction. Data gathering happens on processes that control a Firefox browser instance. Processes are ran in parallel to decrease the overall time spent waiting for data gathering and to make use of the available hardware resources. Each process controls the browser to visit a domain and to take a screenshot of the rendered page. Screenshot-gathering metrics are saved in an SQLite database for each domain entry in top 100 most popular domains per each country. The program results in a folder with screenshots and a database file containing information about each screenshot. The information includes the domain. Domain request failures are tracked and unreachable domains do not result in a screenshot file.

It is important to mention that not all of the website domains were appropriate to classify. Domains that are compromised, contain Riskware, Trojans, or Malvertising are filtered out before visiting. Malwarebytes has a database of endpoints that are unsafe to visit because of the aforementioned risks. Unsafe endpoints found in the list of domains from the Cloudflare CSV's are ignored. Firefox instances used by the screenshot program have two added extensions, "uBlock0_1.54.1b6" and "i_dont_care_about_cookies-3.5.0" (this will be addressed as "NoCookies" for simplification). NoCookies removes cookie pop-ups on websites that need to be accepted manually. Cookie pop-ups could obscure parts of the web page, distorting the resulting training data. Ublock removes and blocks ads. Another feature of Ublock is preventing navigation to malicious websites. Malicious websites are stored in optional block lists, the program has the default block list options enabled. When the program visits the website, a check happens for a warning from the Ublock extension about the visited domain being on one of the default block lists. If this is the case, the domain's web page will not be captured and ignored (in the future). Domains ignored by the program either do not have a root DNS record (such as content delivery network domains) or are inappropriate due to safety reasons. When such a domain is encountered, it is ignored and the web page is not captured.

3.3. Neural network

We want to use a neural networks to find correlations in the appearance of a website and its topics to the domain. Correlations would be given as a classification by the neural network as the probability of a website being of a certain topic given its rendered appearance in the browser. The data would finally be classified by the neural network and manually checked by the researchers for validation purposes. After gathering the training data by creating screenshots of all domains that did not throw an error upon visiting, we are able to use the images of websites and their labels to train a Convolutional Neural Network. The layers of the CNN proposed for use in this research can be seen in the figure below.

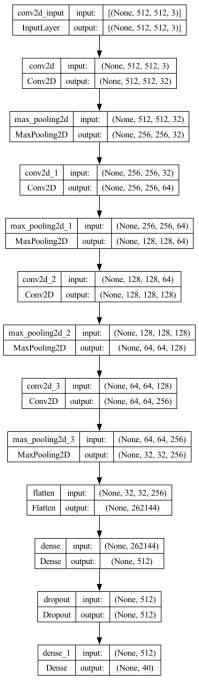

Figure 1.; Schematic representation of the CNN used for categorizing website content. This model uses convolutional filter layers to detect and combine visual cues into patterns for topic classification. The "None" in each layer's input and output refers to the batch size in Keras/TensorFlow models. It means that the model can accept a batch of any size.

The program interacting with the model is written in Python and is part of the described proof of concept in chapter 3.2. As stated before, Python provides and has access to many libraries that allow for quick prototyping and abstraction. Standardizing the arithmetic that goes into working with machine learning models is something that Tensorflow[8] provides as a Python package. The model is defined, compiled and trained with the use the Keras[9] subpackage, which is part of Tensorflow. Keras allows this process to be consistent across a range of computer hardware. Another advantage of using Keras is the combined time and effort of the development team that can be made advantage of in this research. Optimizations to the mathematical arithmetic required to work with a machine learning model are therefore a problem that can be omitted in this research. Hardware that was used to work with the CNN models have the following specifications:

- Testing & results: Apple M1 SOC with 8-Core CPU and 16 GB LPDDR4-4266 Memory
- Training & results: Apple M2 Pro SoC with 10-Core CPU and 16GB LPDDR5-6400 Memory

4. Results

The proof of concept program allowed the capturing and automated labelling of the training data. Quality and quantity of the screenshots and the corresponding labels eventually decided the accuracy of the classifications by the neural network. Results from the described methods applied with the proof of concept would be analyzed. Reviewing this data gave more insights about the events that happen with the CNN when training the model or during the event of a classification.

4.1. Data gathering

Gathered data represents screenshots of website front pages, the approach to taking these screenshots is described in detail in the Methodology chapter 2.1. Ads, inappropriate or unsafe websites and domains not meant for people (such as a content delivery network domains) did not end up in the training data set. The proof of concept generated metrics on each visited domain. It is in the same step in which the domains (that were filtered to exclude malware) were validated to be appropriate in training by building a database of metrics. Metrics included response time, exceptions during making a screenshot and the overall time spent on a domain. Exceptions were

be thrown by one of multiple events occurring during an attempt to capture the website page:

- Client error responses (HTTP codes 404 or 403)
- Request timeout (timeout)
- Blocked by uBlock (ublock)
- Domain DNS issue(s) (neterror)

Filtering on domains without an exception yields the websites that were captured successfully. The domains that did not throw an error were the ones used for the training.

The following figure shows the amount of successful screenshots together with exceptions for 410 labeled domains from the Cloudflare. Additionally, the graph displays the exception types with successes holding the majority of the count. That way they are compared to each other and can be seen that the successful screenshots are the majority.

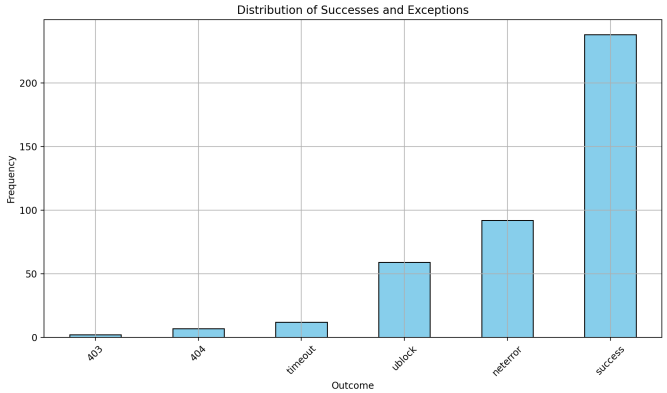

Figure 2.; Distribution of successful screenshots and exceptions of 410 Cloudflare labeled domains.

The 'neterror' bar shows the amount of websites that resulted in a network error when visited, meaning they are not able to be reached by a user. Some websites were blocked because they are labeled as malicious by uBlock, shown in the 'ublock' bar. The rest of the exceptions consists of timeout and HTTP errors. A substantial amount of screenshots failed. Screenshots of the top ranked domains yielded in 238 domains being successfully captured out of 410 total domains. Therefore, the success rate of capturing the top visited domains is 58%. The rest of the domains (42%) do not represent a website intended to be visited by a person, such as CDN domains (Content Delivery Network).

4.2. Label distribution

Figure 3. shows the distribution of categories for websites. It is noticeable that there is a trend where most websites that are the top visited per country are primarily of the category 'Technology'. Other popular categories are 'Search Engine', 'Information Technology', 'E-commerce', 'News and Media' and 'Business'.

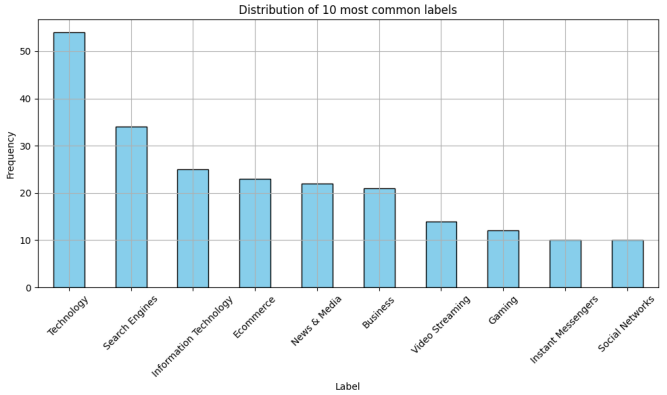

Figure 3.; Distribution of the 10 most common labels in successfully screenshots.

4.3. Model performance

Below is a confusion matrix diagram of the model with the highest accuracy that we received after we were able to train on the limited dataset. The confusion matrix shows how well (or not) the model classified the validation data (on the x-axis) when compared to the true categories of said validation data (on the y-axis). A perfect classifying model is supposed to have a confusion matrix that is a diagonal straight line.

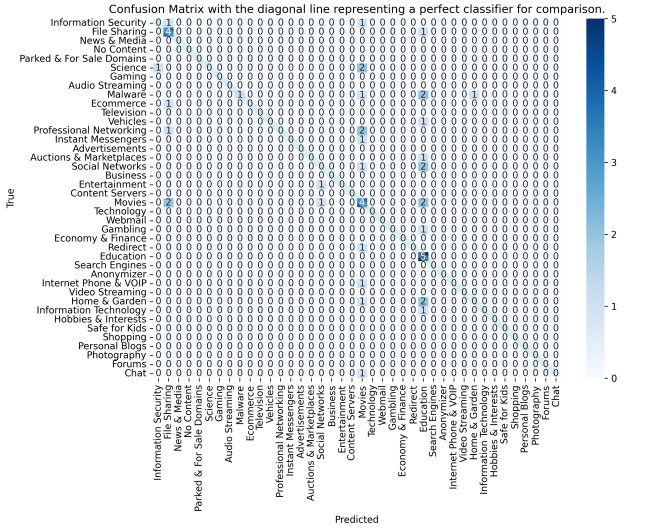

Figure 4.; Confusion matrix of the model's predictions on 20% of the screenshots that were not used in training.

Some categories are correctly classified when looking at the confusion matrix. These categories are 'File Sharing', 'Movies' and 'Education'. The prediction columns for these categories contain most of the predictions, shown as positive values in the confusion matrix. Analysing the predictions on our validation data, 45.83% of all predictions fall in only 3 out of the 39 total categories. To reiterate, 45.8% of the predictions on validation data by the model we trained on the Cloudflare data fall into 7.7% of the categories.

5. Discussion

The top 10 common labels shown in the label distribution from Figure 3. appear to follow a logarithmic function. The large difference between the most common labels and less common labels indicate a non-uniform distribution of labels.

Additionally, from the confusion matrix it is observed that the model is not conclusive, thus it cannot always make correct predictions, which is also seen when the model is run on the validation data. This is indicated in the confusion matrix of Figure 4. There is seen that some deviations from the 'perfect' prediction are available.

5.1. Interpretations and Implications

Our hypothesis was that the performance of the model trained on the Cloudflare data might not perform well on validation data due to the extremely small training dataset. Looking at the distribution of the label categories, we see that it is not uniform. Combining the extremely small training dataset and non-uniformity of the labels, our suspicion of the model producing inconclusive predictions increased.

The confusion diagram showed a poor correlation between predicted and true website categories. Poor correlation implies that the model fails to correctly classify a website by its rendered front-page appearance. We only saw high precision and recall values for categories in predictions on validation data that were almost not present in the dataset. This contradicts the accuracy values given by Keras during training.

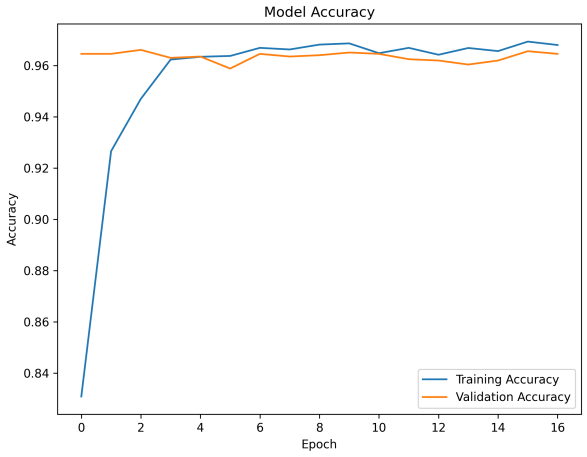

Figure 5.; The training and validation accuracy during training of the model, given by Keras. The graph shows a high training and validation data performance.

It was important for us to find confirmation on why the model was producing predictions that were not in line with the validation labels but was performing well in training accuracy. We investigated the model predictions on our validation data to determine if the model was over-fitted during training. This would give us an indication if our hypothesis was correct.

"Overfitting is a fundamental issue in supervised machine learning which prevents us from perfectly generalizing the models to well fit observed data on training data, as well as unseen data on testing set."[10]

We saw the above described phenomenon of poor generalisation on training data in the results section, which furthermore supports our hypothesis and introduces an explanation for the observations. The phenomenon we are experiencing with our model trained on an extremely small and non uniform labeled dataset matches the definition of overfitting, which can be described by the following two events:

$$\text{Overfitting} \Leftrightarrow (\text{High Training Data Performance}) \wedge (\text{Poor Validation Data Generalization})$$

The high training data accuracies match with the first part of the overfitting definition, concluding the last part of the arguments we need to confirm our hypothesis.

5.2. Limitations

The dataset that was used in the training was insufficient quantity wise. This resulted in the model to overfit and no conclusive results could be acquired. The results of the research cannot be used to safely draw conclusion on how well the proposed method for training a neural network on website screenshots works.

5.3. Conclusion and Recommendations

The research focused on classifying website front pages through a convolutional neural network. We hoped that the results would indicate that predictions of a website category can be used for website classification if performed properly.

The extremely small dataset resulted in inaccurate predictions because of overfitting. Seeing that the predictions were fitted towards a small part of the label category population confirmed that the dataset introduced over-fitting in the trained model. It could have been the case that the validation labels do not have the same distribution as the training labels. This could happen since we randomly split the dataset into 80% training data and 20% validation data, without taking label distributions into account.

Our random sampling implementation for splitting the dataset into training and validation data may have had an influence in overfitting. The extremely small dataset and non-uniformity in the label category distribution seem to be a more likely explanation. Furthermore, we assumed the results from our model were inconclusive by over-fitting.

Future research has the possibility to improve the classification of websites by their front pages and compare it to classification methods based on scraped text or multimedia.

Another recommendation for future researches is to focus on training a neural network on a wider variety of websites with different alphabets and examine how the robustness of the model changes.

6. AI/ChatGPT usage declaration

During the execution of this experiment and the research on it, the authors used ChatGPT, an artificial intelligence tool. Everything that has been produced by ChatGPT has been reviewed by the authors and tailored to the experiment as needed.

ChatGPT has been used for the following parts of this research:

- The development of the programs for making screenshots and using the neural network. Specifically for writing Python boilerplate code and SQL queries.
- Discuss phrasing and order of sentences.
- Generate latex commands for inserting figures and equations/definitions.
- Learn about various evaluation metrics and overfitting.

The authors have not used ChatGPT to generate any of the content of this paper. The authors take full responsibility for the content of this work.

References

[1] N. Nazar, "Exploring seo techniques for web 2.0 websites", English, Department: Chalmers tekniska högskola / Institutionen för data- och informationsteknik (Chalmers), M.S. thesis, Chalmers University of Technology / Department of Computer Science and Engineering, Jul. 2009. [Online]. Available: https://hdl.handle.net/20.500.12380/96559.

[2] R. Du, R. Safavi-Naini and W. Susilo, "Web filtering using text classification", in *The 11th IEEE International Conference on Networks, 2003. ICON2003.*, 2003, pp. 325–330. DOI: 10.1109/ICON.2003.1266211.

[3] S. H. Apandi, J. Sallim and R. Mohamed, "A convolutional neural network (cnn) classification model for web page: A tool for improving web page category detection accuracy", *JITSI : Jurnal Ilmiah Teknologi Sistem Informasi*, Sep. 2023, ISSN: 2722-4600.

[4] S. Albawi, T. A. Mohammed and S. Al-Zawi, "Understanding of a convolutional neural network", in *2017 International Conference on Engineering and Technology (ICET)*, 2017, pp. 1–6. DOI: 10.1109/ICEngTechnol.2017.8308186.

[5] Cloudflare. "Cloudflare". (2024), [Online]. Available: https://www.cloudflare.com/.

[6] V. Le Pochat, T. Van Goethem, S. Tajalizadehkhoob, M. Korczyński and W. Joosen, "Tranco: A research-oriented top sites ranking hardened against manipulation", in *Proceedings of the 26th Annual Network and Distributed System Security Symposium*, ser. NDSS 2019, Feb. 2019. DOI: 10.14722/ndss.2019.23386.

[7] Cyren. "Website URL category checker". Accessed on 13-12. (2023).

[8] Tensorflow. "Tensorflow Python". (2024), [Online]. Available: https://www.tensorflow.org/learn.

[9] Keras. "Keras". (2024), [Online]. Available: https://keras.io/.

[10] X. Ying, "An overview of overfitting and its solutions", *Journal of Physics: Conference Series*, vol. 1168, no. 2, p. 022 022, Feb. 2019. DOI: 10.1088/1742-6596/1168/2/022022. [Online]. Available: https://dx.doi.org/10.1088/1742-6596/1168/2/022022.