Detecting Generated Media: A Case Study on Twitter Data

Johan Sabel and Harald Stiff
Swedish Defence Research Agency (FOI)
SWEDEN

johan.sabel@foi.se harald.stiff@foi.se

ABSTRACT

Recent advancements in generative modeling with deep neural networks have made it easier than ever before to fabricate digital media that can be used to facilitate the spread of propaganda and disinformation on the internet. As a consequence, intelligence gathering on social media has become increasingly important. In this paper, we evaluate methods for detecting images and texts that have been automatically generated and uploaded to Twitter. Our findings suggest that although the detectors are able to achieve high precision under some conditions and do have potential to aid intelligence analysts in their work, it remains a challenge to build comprehensive detection systems that are reliable enough to be deployed in the wild.

1.0 INTRODUCTION AND PRIOR WORK

Being able to verify the credibility of information online is becoming increasingly difficult as large-scale cyber influence operations become more sophisticated. Advancements in generative modeling have made it possible for adversaries to generate large amounts of digital media that is perceived as authentic. For instance, publicly available AI tools [1, 2] can be exploited to set up bot networks of genuine-looking social media profiles, where both user profile content (e.g., profile images [3]) and information intended to be spread (e.g., disinformation tweets) are automatically generated. Therefore, it is important to develop tools that are able to detect generated media; not only to take action against it, but also to investigate the possibilities of incorporating detectors into systems used for intelligence gathering as a step towards improving situational awareness. The ability to automatically extract information of interest from large amounts of unstructured web data might ultimately help streamline the workflow of intelligence analysts and decision makers.

Prior work has shown that neural-based detectors can reliably detect generated media in controlled settings where the generative models are known to the defender [4, 5, 6, 7, 8]. However, there has only been limited work on evaluating the detectors on data from unknown sources in the wild. As detectors are known to be brittle to post-processing and model variations [4, 5, 6, 9] it is not evident that they are reliable enough to be used in real-world systems. In light of this, we evaluate state-of-the-art detectors on Twitter data consisting of profile images and texts from tweets. We train XceptionNet [10] to detect images of human faces synthesized with Generative Adversarial Networks (GANs), and fine-tune the transformer-based language model RoBERTa [11] to detect texts generated with language models. Our experiments are closely tied to our prototype which can be used to perform the analyses and extract information of interest without requiring deep technical knowledge from the end user.

2.0 METHOD

In this section, we describe the methodology of our experiments. The overall procedure is as follows: ***Step 1.*** We train detectors to distinguish between real and generated data. This is accomplished using annotated training datasets. ***Step 2.*** We collect data in the wild (i.e., data from Twitter) and feed it to the trained detectors which classify input samples as either *real* or *generated*. This is accomplished using our prototype which provides functionality to both capture and analyze (e.g., classify) data streams in real-time.

Step 3. We evaluate the performance of the detectors after manually annotating Twitter data fed to the detectors in the previous step.

2.1 Image Detector Training

Similar to what has been done in previous work [4, 5, 6], we use an XceptionNet [10] model pre-trained on the ImageNet [12] dataset to detect images of faces that have been generated by GANs. Specifically, we fine-tune XceptionNet on real images from the publicly available FFHQ [13] and CelebA-HQ [14] datasets as well as GAN-generated images from publicly available ProGAN [14], StyleGAN [13], and StyleGAN2 [1] models. This is illustrated in Figure 2-1. All training images are augmented with random combinations of Gaussian noise, Gaussian blur, JPEG compression, and resizing to increase model robustness. We also crop the images before feeding them to the detector, as shown in Figure 2-2, since real images tend to incorporate more detail and variety in their background compared to generated images.

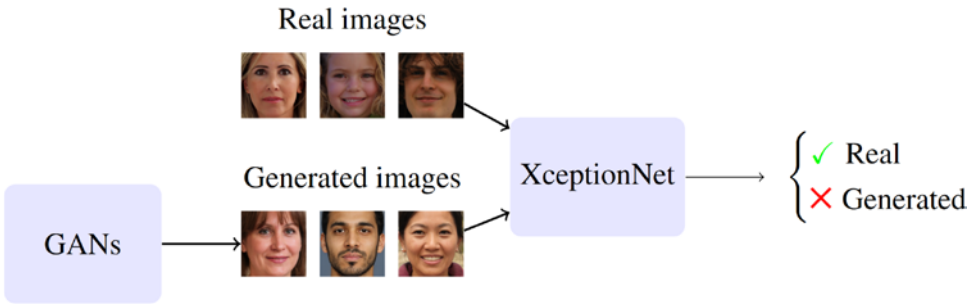


Figure 2-1: Training of the XceptionNet image detector. The detector learns to distinguish between real and generated images of faces when being fed with labeled samples from the training datasets. In practice, the detector works as a classifier which outputs a label (*real* or *generated*) for each input sample.

Figure 2-2: Example images from the training datasets. Top row: Original images. Bottom row: Cropped images. Our training datasets contain 24,000 CelebA-HQ images, 56,000 FFHQ images, 26,666 ProGAN images, 26,667 StyleGAN images, and 26,667 StyleGAN2 images.

2.2 Text Detector Training

The detector used to distinguish real texts from texts generated with language models is based on a RoBERTa [7] neural language model fine-tuned to detect generated text from a large dataset synthesized with GPT-2 [2]. GPT-2 was trained on the WebText [2] dataset, containing 40 GB of text scraped from the internet, and is therefore heavily biased towards forum posts and news articles, rather than Twitter posts which are of interest in our study. Hence, the fine-tuned RoBERTa model is not suitable to be used out of the box. Furthermore, as there is a plethora of different generator models, it is important to train the detector on

several generators before testing it on real world data. In order to make the detector more appropriate on Twitter data and more robust to model variations, we fine-tune it on two additional datasets:

- **GPT-2 Twitter:** A dataset of 5,000 generated tweets from a fine-tuned version of GPT-2, as well as an equal number of real tweets. The data we used to fine-tune GPT-2 was obtained from the Sentiment140 [15] dataset of tweets originally created for sentiment classification. The real tweets used to train the detector originate from the same dataset.
- **TweepFake:** A dataset containing a total of 25,836 tweets, with an equal number of generated tweets as real tweets [9]. The generated tweets originate from real bot accounts on Twitter and have been synthesized with several different language models such as GPT-2, RNNs, and LSTMs.

Figure 2-3 summarizes the fine-tuning of RoBERTa.

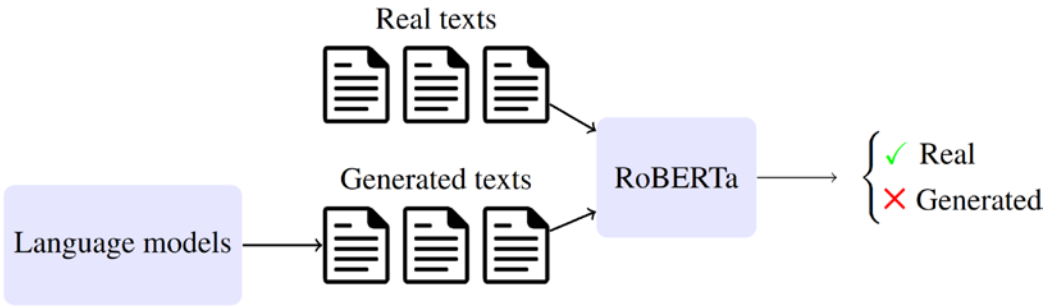


Figure 2-3: Training of the RoBERTa text detector. The detector learns to distinguish between real and generated texts when being fed with labeled samples from the training datasets. In practice, the detector works as a classifier which outputs a label (*real* or *generated*) for each input sample.

2.3 Prototype

Our prototype provides functionality to download Twitter data and optionally feed it through one or multiple components used to perform data analyses. Therefore, we incorporate the trained detectors into the prototype to evaluate their ability to detect generated media on Twitter. In this case, the image and text detectors can be seen as corresponding to two separate components. Hence, we choose to perform the evaluations using two different instances of the prototype: one incorporating the image detector (XceptionNet) and another incorporating the text detector (RoBERTa) as shown in Figure 2-4.

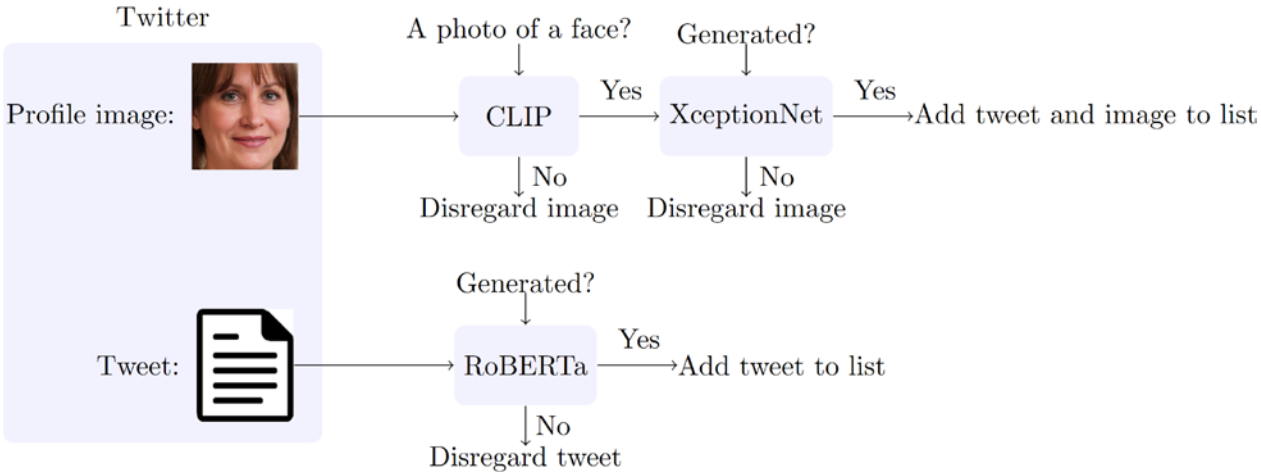


Figure 2-4: Prototype flowchart that shows how the input (a profile image or Twitter post) is fed to the detectors and, if believed to have been automatically generated, added to a list which is presented to the end user.

Note that an additional component called CLIP precedes the XceptionNet image detector. CLIP [16] (Contrastive Language–Image Pre-training) is a pre-trained computer vision model which can be applied to many types of visual classification tasks without having to be fine-tuned or re-trained from scratch. Since we train XceptionNet to distinguish between real and generated images of faces, we use CLIP in an attempt to ensure that the detector only receives facial images as input. Specifically, we define the two classes “a photo of a face” and “not a photo of a face”.

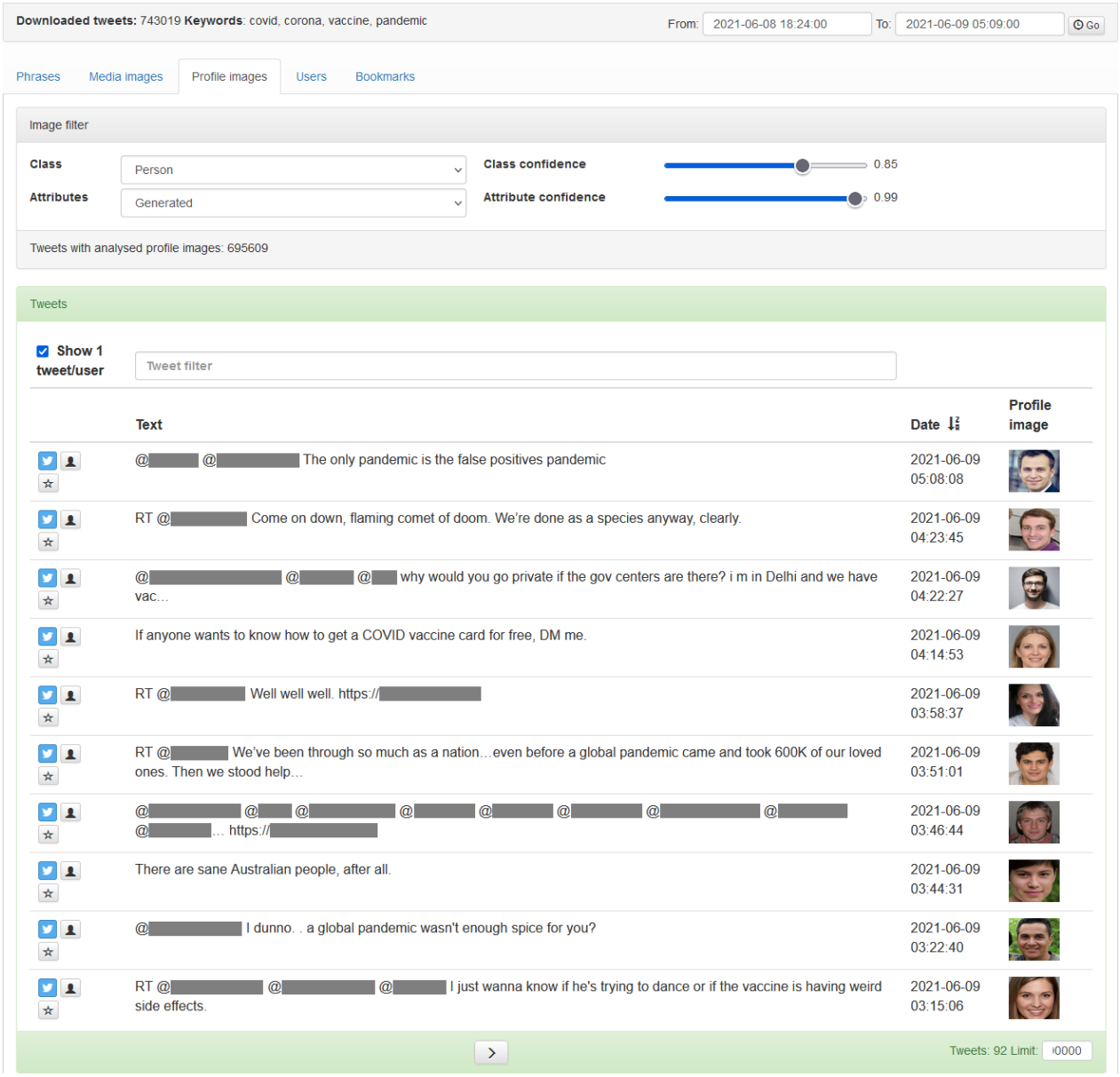


Figure 2-5: Screenshot of the graphical user interface of the prototype used for detecting tweets that have been posted or shared by fake Twitter accounts (in this case accounts that are using GAN-generated profile images). By clicking on the buttons to the left of a tweet one can view the original tweet on Twitter as well as the user account associated with it. It is possible to enlarge the profile images by clicking on them. Some information such as account names have been removed from the screenshot.

The prototype allows the user to specify keywords related to topics of interest and then collects recently posted tweets which contain at least one of the keywords. More precisely, the prototype collects a tweet as long as it contains at least one of the keywords, is a shared (re-tweeted) post which contains at least one of the keywords, or is a comment on another tweet which contains at least one of the keywords. The image detector analyses the profile image of the account which posted or shared the tweet, while the text detector analyses the text content of the tweet.

Figure 2-5 shows a screenshot of the interface of the prototype used to detect GAN-generated profile images. At runtime the user is presented with a growing list of tweets believed to be posted or shared by accounts

using fake profile images. The profile images are shown next to the tweets. Both CLIP and XceptionNet output a score in the [0, 1] range. The score from CLIP indicates the probability that an input image contains a face, while the score from XceptionNet indicates the probability that the image has been generated. It is possible to adjust both the class confidence threshold (i.e., the minimum score required for CLIP to classify an image as "a photo of a face") and the attribute confidence threshold (i.e., the minimum score required for XceptionNet to classify the very same image as "generated"). When using the text detector, the user is only presented with the text content of the tweets and a confidence threshold meter for RoBERTa which can be used to specify the minimum score required to classify a text as "generated".

2.4 Evaluation Datasets

We consider tweets related to three different topics: (1) COVID-19, (2) the Israeli-Palestinian conflict which grew more intense during the spring of 2021, and (3) the G7 and NATO summit meetings of 2021 which were held in Cornwall and Brussels. Details about the collected data are presented in Table 2-1.

Table 2-1: Number of tweets collected for different topics used to evaluate the image and text detectors. The topic keywords are italicized. All collected tweets are written in English.

Number of Tweets	Topic: COVID-19 (*covid, corona, vaccine, pandemic*)	Topic: Israel-Palestine (*israel, palestine, hamas, idf*)	Topic: G7 & NATO (*g7, nato, summit, cornwall, brussels*)
Image Detector	2,108,569	1,301,000	515,447
Text Detector	2,674,878	837,510	328,650

2.5 Data Annotation

The fact that the evaluation data is collected in the wild means that there are no ground truth class labels available. Therefore, manual annotation is necessary. During evaluation we only consider samples classified as *generated* by the detectors since the large amount of collected data makes manual labeling of the full datasets difficult.

2.5.1 Profile Image Annotation

As mentioned, when using the image detector, all samples (tweets and corresponding profile images) that have been classified as generated are presented in a list. Since the number of detected samples is relatively small compared to the full size of the downloaded datasets it is possible to label each detected sample in the list as either *real* (false positive) or *generated* (true positive). This is done by going through the list manually and for each profile image, based on its visual characteristics, determine whether or not it is likely to have been generated by a GAN. Since this can be a difficult task for the average person, we leave it to an expert with experience in analyzing GAN-generated images.

In order to further verify that the labels obtained from the manual annotation are correct, we attempt to use a StyleGAN2 model to reconstruct a random subset of the images labeled as generated by the expert. If an

accurate reconstruction can be made for an image, it is likely that it has been generated with the same model. The reconstruction is done following the official instructions provided by the authors of StyleGAN2 [1]. A possibility would be to use the reconstruction tool itself as a detector. However, since it is only able to handle StyleGAN2 images, while also requiring large computational resources, we do not consider this approach suitable.

2.5.2 Twitter Post Annotation

We resort to a semi-automatic process when annotating the Twitter posts that have been classified as generated by the text detector. The annotations are done user-wise and are mainly based on the description of the profile and in a few cases, the profile name. Specifically, a post is classified as *generated* if the user that sent the post explicitly states that it is a bot that posts machine generated texts. For instance, the description might say that the user is a bot that posts tweets generated with a specific language model that has been trained on tweets from a celebrity.

Although an account is a bot, the texts it posts might not necessarily have been generated by a language model, but could rather be, for instance, automatic summarizations of statistics (e.g., daily summaries of COVID-19 deaths). Therefore, in addition to checking the user description, we scan the posts of the users and determine whether the posts are diverse enough to actually have been generated with a language model.

It should be noted that it is likely that many users which post machine generated text do not disclose it in their profile descriptions or profile names. Unfortunately, we find no other quick reliable way of determining whether a user posts generated text. Consequently, the true performance of the text detector model is likely underestimated as some of the false positives might indeed be true positives in reality.

2.6 Evaluation Metrics

After finishing the manual data annotation, we proceed to compute evaluation metrics to quantify the performance of the detectors. Note that we do not compute metrics such as accuracy (the fraction of samples that are correctly classified by the detector) and recall (the fraction of all existing generated samples that are actually detected), since this would require fully annotated datasets. Furthermore, the detectors could in theory achieve near-perfect accuracy by simply classifying all samples as *real*, since one can assume that only a small fraction of the downloaded Twitter data has been generated. Hence, we mainly focus on precision (the fraction of detected samples that are correctly classified):

Precision = *Number of True Positives* / **(***Number of True Positives* + *Number of False Positives***)**.

When it comes to the graphical user interface of the prototype, the precision tells us how large portion of the list actually contains automatically generated samples (i.e., fake profile images or fake tweets). We believe that precision is important in our context because an intelligence analyst should ideally be able to trust the output of the prototype to some extent.

2.6.1 Image Detector

For the image detector, we compute the precision for various class confidence thresholds and attribute confidence thresholds. Therefore, we also report the number of true positives to analyze whether it is possible to improve precision through threshold adjustments without significantly decreasing the number of detected true positives. Furthermore, the evaluation is performed on both a tweet level and user level, where the latter means that we configure the prototype to output a maximum of one tweet from each user account. This functionality might be useful if the end user of the prototype only wants to receive a list of unique accounts with fake profile images rather than the full list of tweets posted by such accounts. User level analysis could also help prevent accounts with unusually high activity from potentially influencing the

results, since multiple tweets from a single account will result in the same profile image being included in the analysis multiple times.

2.6.2 Text Detector

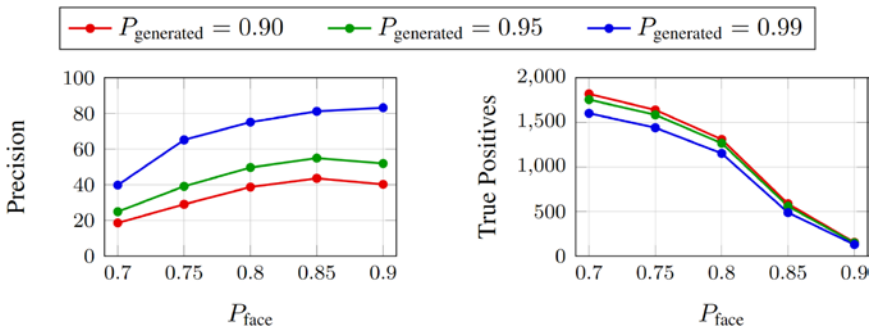
As the length of a single tweet is relatively short, the text detector cannot accurately predict whether it is real or generated [8]. In order to increase the accuracy we take the average prediction score of N tweets and assign each of the N tweets the class of the average prediction. We only take the average prediction of tweets belonging to the same Twitter user, relying on the assumption that a majority of the users exclusively posts either real or generated tweets; not a combination of both. The results are reported for different values of N showing how the precision varies with the amount of tweets used in the predictions. Moreover, the precision is measured for different class confidence values, and we also report the number of true positives for each value. Lastly, the results are both reported on a tweet level where all the tweets are included in the analysis and on a user level where an equal number of posts from each user is included. We include the user level results in order to make the results less biased to the few number of users that make up a large fraction of the tweets in the datasets.

3.0 RESULTS

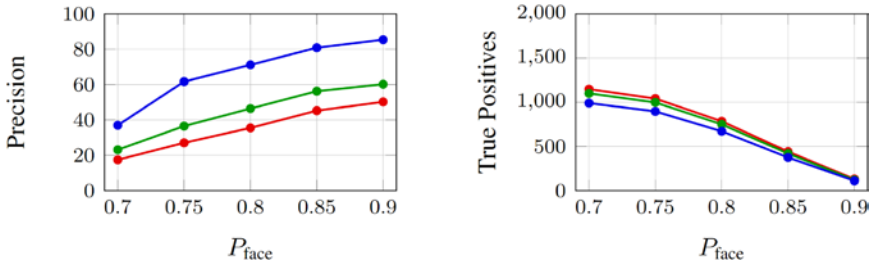
In this section, we present the results of the experiments described in Section 2.0. The performance of the image detector is shown in Figures 3-1, 3-2, and 3-3, while the performance of the text detector is shown in Figures 3-4, 3-5, and 3-6. For the image detector, the precision and number of true positives are plotted with respect to different values of P_{face} (minimum score required for an image to pass through CLIP). This is done using three different values of $P_{\text{generated}}$ (minimum score required for XceptionNet to classify the image as generated).

For the text detector, the precision and number of true positives are plotted with respect to $P_{\text{generated}}$ (minimum score required for RoBERTa to classify a tweet as generated). We present three different graphs where we vary the number of tweets (2, 3, or 5) used by the text detector to make a classification, as described in Section 2.6.2. Furthermore, note that we do not report the user level results on the Israel-Palestine dataset in Figure 3-5 as the number of unique users classified to have posted generated text is too small.

Finally, examples of reconstructed profile images are presented in Figure 3-7.

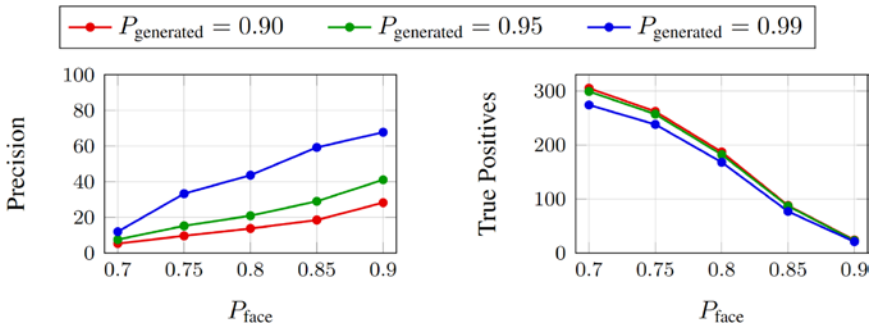


(a) Tweet level performance (all tweets are included in the analysis).

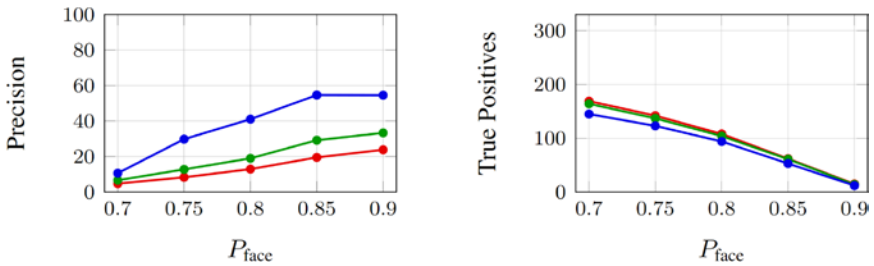


(b) User level performance (one tweet per user account is included in the analysis).

Figure 3-1: Image detector performance when analyzing 2.1M tweets about COVID-19.

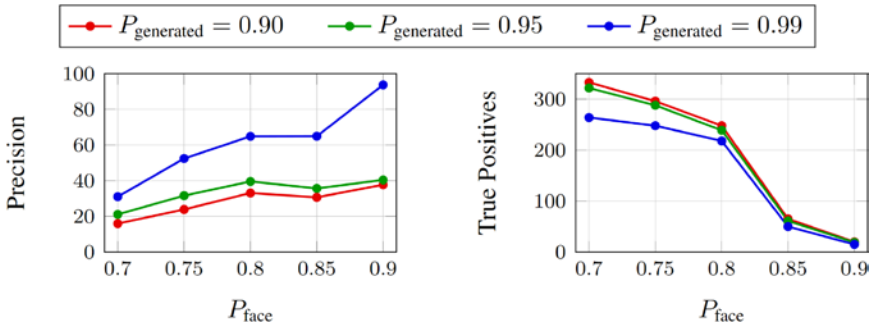


(a) Tweet level performance (all tweets are included in the analysis).

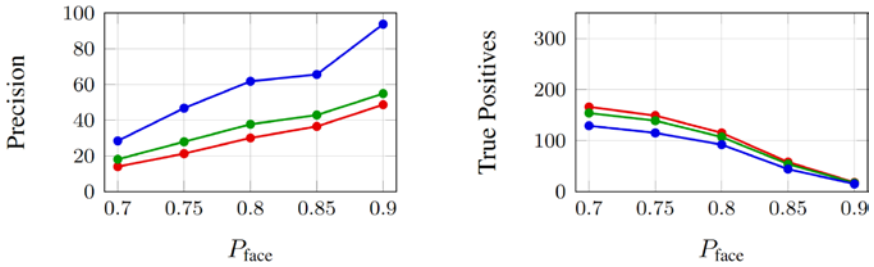


(b) User level performance (one tweet per user account is included in the analysis).

Figure 3-2: Image detector performance when analyzing 1.3M tweets about the Israeli-Palestinian conflict.

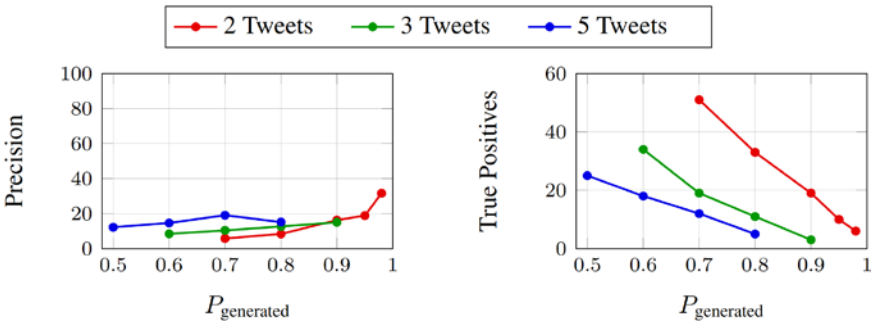


(a) Tweet level performance (all tweets are included in the analysis).

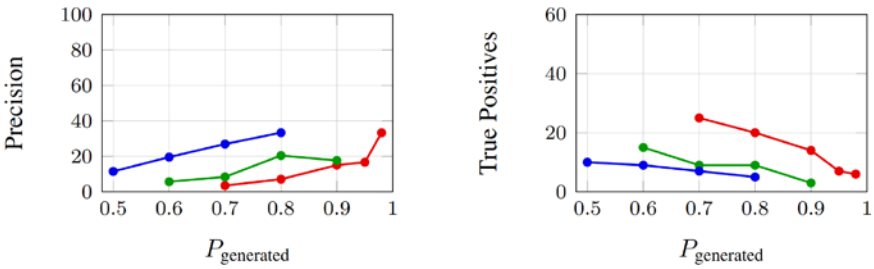


(b) User level performance (one tweet per user account is included in the analysis).

Figure 3-3: Image detector performance when analyzing 515K tweets about G7 and NATO summit meetings.



(a) Tweet level performance (all tweets are included in the analysis).



(b) User level performance (a fixed equal number of tweets from each user account are used).

Figure 3-4: Text detector performance when analyzing 2.6M tweets about COVID-19.

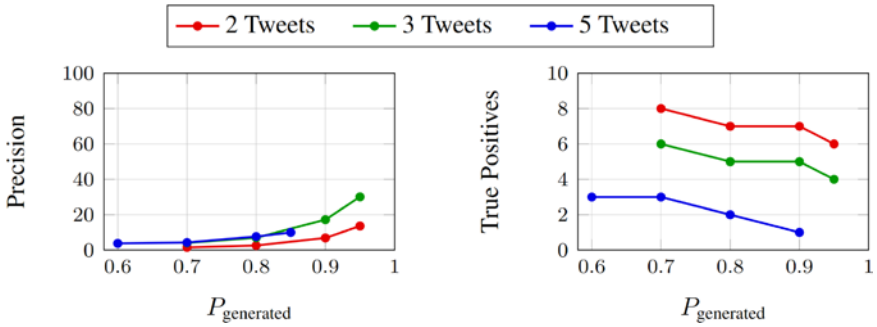
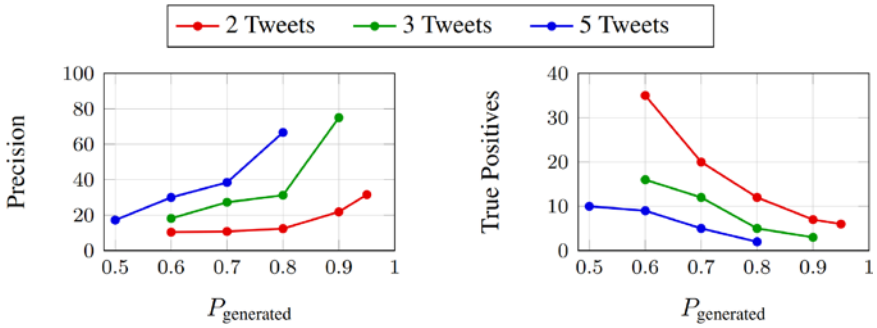
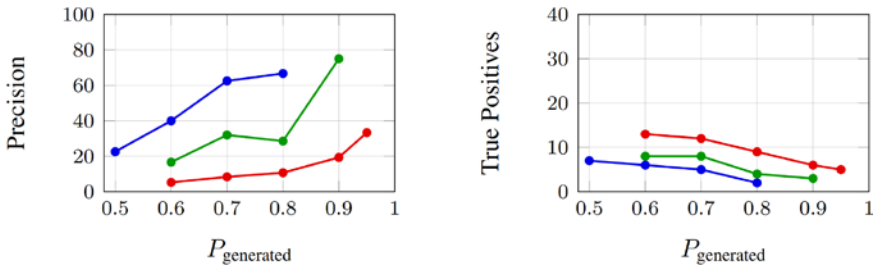


Figure 3-5: Text detector performance when analyzing 837K tweets about the Israeli-Palestinian conflict. The figure shows the tweet level performance where all the tweets are included in the analysis.



(a) Tweet level performance (all tweets are included in the analysis).



(b) User level performance (a fixed equal number of tweets from each user account are used).

Figure 3-6: Text detector performance when analyzing 328K tweets about G7 and NATO summit meetings.

Figure 3-7: Top row: Returned positives from the image detector. Bottom row: StyleGAN2 reconstructions of the positives. An accurate reconstruction implies that the returned positive indeed has been generated (in this case with StyleGAN2).

4.0 DISCUSSION AND CONCLUSION

By observing Figures 3-1, 3-2, and 3-3, it is clear that the image detector is able to achieve high precision for certain values of the P_{face} and $P_{generated}$ thresholds. For instance, Figure 3-1 related to the COVID-19 topic shows that when CLIP requires a minimum of $P_{face} = 0.8$ to let images pass through to XceptionNet, the precision approaches 80% provided that XceptionNet requires a minimum of $P_{generated} = 0.99$ to classify images as generated (see the blue graphs). Hence, in this case the prototype will provide the end user with a list in which 80% of the images actually have been generated by a GAN. One should also be aware that some of the samples misclassified by XceptionNet are non-facial images that somehow bypasses CLIP. Although this is positive in the sense that the list contains less real facial images than indicated by the precision, a noisy list still worsens the overall user experience.

There is also another issue. When precision increases as a result of increasing P_{face}, the number of true positives tends to decrease quite dramatically (see the right column of Figure 3-1). In other words, many images that have been generated are not going to be included in the list when using high threshold values. There is a clear trade-off between obtaining high precision and detecting a large number of true positives. It would probably be possible to detect more generated images if we could make sure that less non-facial images reach XceptionNet, since this issue currently forces us to use high values of both P_{face} and $P_{generated}$ in order to obtain reasonable precision. Although the precision varies quite a bit across different topics, we still observe the same pattern in Figures 3-2 and 3-3. One can also observe that in general there seems to be no significant difference in precision when performing the analysis on a tweet level versus a user level for each topic, while the number of true positives is smaller on a user level as expected.

It is well known that detectors often struggle to detect samples from unknown generators. Hence, it is difficult to estimate the extent to which data bypasses detection since we do not work with fully annotated datasets. However, it is not unreasonable to assume that a large portion of fake images on Twitter originate from well known pre-trained GAN models, such as StyleGAN2, which are commonly used on websites such as ThisPersonDoesNotExist.com. Figure 3-7 indicates that our detector is able to spot such images. All images in the figure are almost perfectly reconstructed, except for the one to the far right.

When it comes to the text detector, there is a notably smaller number of true positives in comparison to the image detector as shown in Figures 3-4, 3-5, and 3-6. There are many reasons to why this is the case. Firstly, it is harder for the average user to set up a bot that automatically posts generated tweets than it is to download and use a generated profile image, leading to a smaller amount of generated text in the wild. Secondly, the text detection problem is likely harder, not only because short texts are difficult to classify, but also since there is a wide spread of different language models in use that the detector has not been trained on. Lastly, another reason for the smaller amount of true positives is that we only label a post as *generated* if the user behind the post explicitly states so, as mentioned in Section 2.0. The small number of true positives makes it difficult to generalize about the results. However, similar to the image detector, the number of true positives decreases when increasing the probability threshold (in this case $P_{generated}$). Unfortunately, keeping the threshold high is needed in order to reach a reasonable precision. Otherwise, manual work would be required from the end user to filter out a large number of false positives. Finally, the precision tends to increase when including more tweets in each classification, making it a simple way to improve the performance of the detector.

While the overall results are far from perfect, we still conclude that neural-based detectors do have potential to aid intelligence analysts and decision makers in their work. Our prototype provides easy to use tools for analyzing and extracting information of interest from huge volumes of data that are unmanageable by humans. However, the detectors and prototype should not be regarded as readily deployable solutions; and using methods such as these requires an understanding of the inherent limitations of AI-based tools. A possible next step would be to involve potential end users in the evaluation, similar to what has already been

done for another tool and use case of our prototype [17]. This could provide us with additional insights that our quantitative metrics are unable to convey.

5.0 REFERENCES

[1] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila. Analyzing and Improving the Image Quality of StyleGAN. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[2] A. Radford, J. Wu, D. Amodei, D. Amodei, J. Clark, M. Brundage, and I. Sutskever. Better Language Models and Their Implications. *OpenAI [Online]. Available: https://openai.com/blog/better-language-models/*, 2019.

[3] T. Hatmaker. Chinese Propaganda Network on Facebook Used AI-Generated Faces. *TechCrunch [Online]. Available: https://tcrn.ch/3cynGql*, 2020.

[4] J. C. Neves, R. Tolosana, R. Vera-Rodriguez, V. Lopes, H. Proença, and J. Fierrez. GANprintR: Improved Fakes and Evaluation of the State of the Art in Face Manipulation Detection. *IEEE Journal of Selected Topics in Signal Processing (JSTSP)*, 14(5):1038–1048, 2020.

[5] N. Hulzebosch, S. Ibrahimi, and M. Worring. Detecting CNN-Generated Facial Images in Real-World Scenarios. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2020.

[6] J. Sabel and F. Johansson. On the Robustness and Generalizability of Face Synthesis Detection Methods. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2021.

[7] I. Solaiman, M. Brundage, J. Clark, A. Askell, A. Herbert-Voss, J. Wu, A. Radford, G. Krueger, J. W. Kim, S. Kreps, M. McCain, A. Newhouse, J. Blazakis, K. McGuffie, and J. Wang. Release Strategies and the Social Impacts of Language Models. *arXiv preprint arXiv:1908.09203*, 2019.

[8] D. Ippolito, D. Duckworth, C. Callison-Burch, and D. Eck. Automatic Detection of Generated Text is Easiest when Humans are Fooled. *arXiv preprint arXiv:1911.00650*, 2019.

[9] T. Fagni, F. Falchi, M. Gambini, A. Martella, and M. Tesconi. TweepFake: About detecting deepfake tweets. *PLOS ONE*, 16(5):e0251415, 2021.

[10] F. Chollet. Xception: Deep Learning with Depthwise Separable Convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[11] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv preprint arXiv:1907.11692*, 2019.

[12] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[13] T. Karras, S. Laine, and T. Aila. A Style-Based Generator Architecture for Generative Adversarial Networks. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[14] T. Karras, T. Aila, S. Laine, and J. Lehtinen. Progressive Growing of GANs for Improved Quality, Stability, and Variation. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.

[15] A. Go, R. Bhayani, and L. Huang. Twitter Sentiment Classification using Distant Supervision. *CS224N Project Report Stanford*, 1(12), 2009.

[16] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever. Learning Transferable Visual Models From Natural Language Supervision. *arXiv preprint arXiv:2103.00020*, 2021.

[17] S. Varga, J. Brynielsson, A. Horndahl, and M. Rosell. Automated Text Analysis for Intelligence Purposes: A Psychological Operations Case Study. In *Open Source Intelligence and Cyber Crime*, pages 221–251. Springer, 2020.