When Vision Fails: Text Attacks Against ViT and OCR

Anonymous authors
Paper under double-blind review

Abstract

Text-based machine learning models rely on visual inputs of rendered text as a defense strategy against an emerging class of Unicode attacks (e.g., pre-processing text using optical character recognition). In this work, we show that these visual defenses are inadequate. We use a genetic algorithm to generate visual adversarial examples in a black-box setting and demonstrate a highly effective rendering attack that leverages these adversarial examples encoded as text. Specifically, we use the Unicode functionality of combining diacritical marks to manipulate text inputs so that small visual perturbations appear when the text is rendered. We additionally conduct a user study to establish that the model-fooling adversarial examples do not affect human comprehension of the text. We demonstrate the effectiveness of these attacks in the real world by creating adversarial examples against production models published by Facebook, Microsoft, IBM, and Google.

1 Introduction

Advances in adversarial examples (Szegedy et al., 2014; Biggio et al., 2013; Goodfellow et al., 2014) for text-based machine learning systems have led to imperceptible perturbation techniques targeting the encoding of text (Boucher et al., 2022; Pajola & Conti, 2021). These attacks leverage uncommon Unicode encodings to subvert text models without visual indication to users (The Unicode Consortium, 2021a). Existing defenses against encoding attacks achieve protection by unifying the visual and encoding pipelines; specifically, the Vision Transformers (ViTs) architecture can be used to build robust new models (Salesky et al., 2021), and Optical Character Recognition (OCR) can be used to retrofit existing models with defenses (Boucher et al., 2022). Such defenses seek to ensure that text inputs which are visually-identical when rendered produce equivalent outputs in defended models.

We show that these defenses are inadequate. We propose a technique to encode text perturbations in a way that, once rendered, the image of the rendered text will contain adversarial perturbations that defeat visual defenses. These adversarial examples operate in the visual text domain, meaning that visual inputs are generated exclusively from rendered text such that it is impossible to perturb arbitrary pixel values. However, by leveraging combining marks from the Unicode specification, we can craft small, targeted visual perturbations that appear on the rendered image of text. While these marks are not visually significant enough to impact a human reader's understanding of text, the manipulated pixels in the image domain of rendered text enable targeted attacks on model outputs.

In particular, we build on existing adversarial text perturbation techniques to introduce a new class of Unicode-based adversarial examples that leverages Unicode's combining diacritical mark functionality. Prior work claiming ViT and OCR are robust against text-based perturbations (Boucher et al., 2022; Salesky et al., 2021) did not consider text containing combining diacritics prior to visual rendering, but in this work we demonstrate that these barely perceptible marks have a substantial impact on ViT and OCR model performance, causing performance drops of around 60% to as high as 92.6%.

Text attacks on ViT and OCR have broad implications. For example, we demonstrate extensions to attacks on toxic content detection (Hosseini et al., 2017) and machine translation (Belinkov & Bisk, 2018) that succeed despite ViT and OCR defenses against encoding attacks. The techniques we propose can even be

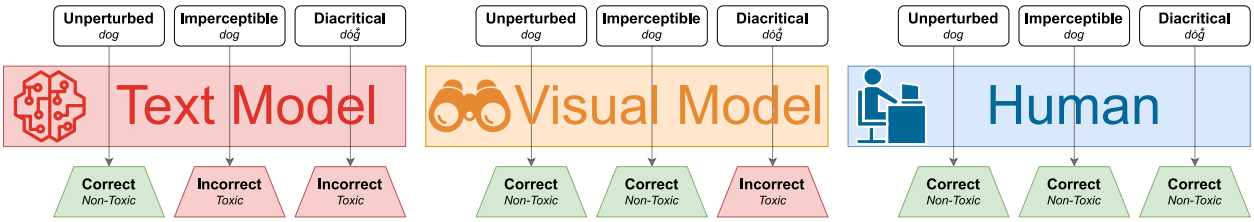

Figure 1: The visualization gap, against which visual models claim to be robust, continues to be a source of adversarial examples via diacritics as in this toxic content classifier example.

used to affect the physical domain, such as disclosures submitted as part of a legal case. If the disclosing legal team adds targeted text perturbations to printed disclosure documents, the text can form visual adversarial examples that are incorrectly processed by OCR systems despite being readable by humans. Large-scale document scanning in particular is often automated and performed with little to no human review of the original paper documents—and so it is entirely plausible that a system may scan encoded text (and therefore produce incorrect output) without detection. This could, for instance, prevent searchability, thus impeding the opposing legal team in the discovery process.

In summary, we make the following contributions in this paper:

- We introduce a novel form of adversarial example against machine-learning models that process rendered text, including Vision Transformers and Optical Character Recognition models, which are encoded in the textual domain but operate in the visual domain.
- We demonstrate that previously-proposed defenses (Boucher et al., 2022; Salesky et al., 2021; Clark et al., 2022) for encoding attacks against text models including ViT, OCR, and neural encoders are insufficient.
- We conduct a user study to validate that our adversarial examples do not impact human readability.
- We propose definitive defenses for diacritical attacks against Unicode-based visual models.

2 Background

2.1 The Visualization Gap

Traditionally, machine learning models processing text such as natural language operate directly upon some encoding of the input text. This may take the form of input embeddings as vectors representing words, characters, or learned subword components created by parsing Unicode inputs (The Unicode Consortium, 2021a). However, unlike models, humans do not directly consume encoded text. Rather, text is rendered and then visually presented to human users.

It is here that a security design flaw arises: the relationship between encoded text and rendered text is not bijective. That is, a visual rendering could be represented by many unique text encodings. Formally,

$$\forall t \in T, \quad U(t) \not\Leftrightarrow \{v(t)\} \tag{1}$$

where T is the set of all possible text sequences, U is the function generating the set of all possible Unicode representations of a text, and v is the visual rendering of a text.

Consider, for example, the presence of invisible characters such as Unicode's Zero-Width Space (ZWSP); these characters have no effect on the rendering of most text and yet change the encoded representation. Visually-identical characters, known as homoglyphs, can also be used interchangeably, and control characters can be used to delete and reorder characters.

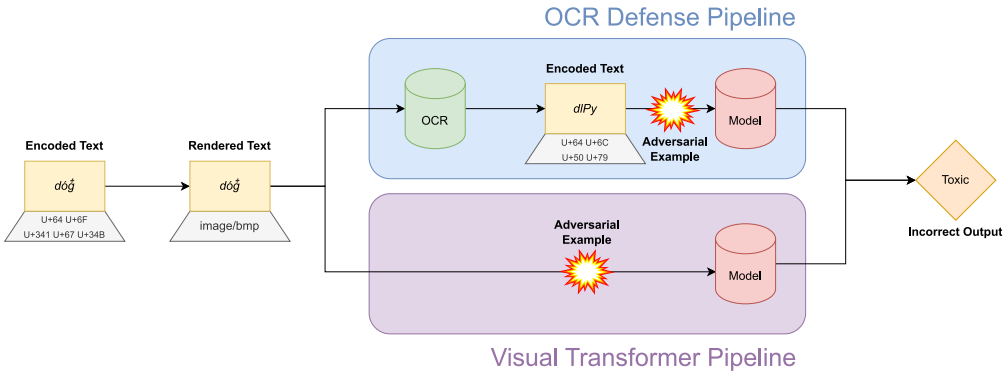

Figure 2: Our attack: adversarial examples in the visual domain encoded in the textual domain.

The difference between the encoding and visualization of text can be used to create adversarial examples against models that operate directly upon some form of textual input (Boucher et al., 2022; Pajola & Conti, 2021), improving the stealth of earlier techniques leveraging misspelling or paraphrasing (Gao et al., 2018; Li et al., 2018; Belinkov & Bisk, 2017; Khayrallah & Koehn, 2018). The visualization gap is depicted in Figure 1.

2.2 Defense Through Vision

To defend against adversarial examples that exploit the visualization gap, model designers must seek to unify the text encoding and visualization pipelines. That is, designers must seek to build or augment models such that:

$$\forall t \in T, \quad E(U(t)) = \{t'\} \Leftrightarrow \{v(t)\} \tag{2}$$

where E generates the set of embeddings for the encoded values taken as input.

One simple but effective way to accomplish this on existing models is to render text inputs and process the resulting images through OCR as a pre-processing step prior to model inference (Boucher et al., 2022). In effect, this provides an automated system that maps fixed visual renderings to a common encoded input. The inference pipeline in this setting is: *encoded input* $\rightarrow$ *rendered image* $\rightarrow$ *text* $\rightarrow$ *model*.

For greenfield models, Vision Transformers may be the preferred defense approach as no compute-intensive pre-processing model is required. ViTs operate upon images as input, and operating directly upon rendered images as embeddings yields both good performance and defense-by-design against attacks exploiting the visualization gap (Salesky et al., 2021). The inference pipeline in this setting is: *encoded input* $\rightarrow$ *rendered image* $\rightarrow$ *model*.

Finally, neural encoders offer new NLP models some robustness against Unicode perturbations (Clark et al., 2022). Although they do not operate in the visual domain, neural encoders are a form of learned embedding that map relationships between Unicode characters such that encoded values that would look similar if rendered should result in similar embeddings. The inference pipeline in this setting is: *encoded input* $\rightarrow$ *neural embedding* $\rightarrow$ *model*.

3 Methods

3.1 Exploiting Diacritics

In the image domain, adversarial examples are typically crafted by slightly perturbing the values of key pixels often identified through a gradient-based approach (Goodfellow et al., 2014). While such an approach would in theory work against ViTs and OCR models, the visual text domain has the added constraint that the input images are generated through rendering text. Therefore, it is not possible to arbitrarily perturb pixel values, as pixel values are not directly encoded; rather, the pixels are generated from encoded text.

Diacritics – also known as diacritical marks or accents – are small marks that can be placed on top of other written letters. These marks serve different purposes across different linguistic families, but often serve to modify the pronunciation or meaning of words. Examples of diacritics include the Spanish ñ, the German ä, and the French è. Diacritics are most common in European languages, but are also used to aid in the romanization of non-Latin scripts such as the case of pinyin for Mandarin.

In Unicode, characters commonly used with diacritical marks typically have a dedicated character – or code point – in the Unicode specification. However, the concept of *combining diacritical marks* also exists. These characters modify the character immediately preceding them to add the specified diacritical mark. The Unicode specification defines 256 such combining characters (The Unicode Consortium, 2021b;d;f;e;c).

Combining diacritical marks provide a method by which arbitrary characters can be subtly perturbed in the image domain. These marks will produce a noticeable visual effect, but as we will later demonstrate through a user study these marks do not affect a human's ability to read the text. However, just as changing particular pixel values can cause general image classifiers to fail, so too can the perturbed pixel values created by diacritical marks affect ViTs and OCR models. Although the size of the perturbation space is smaller than the more continuous space provided by color pixels in unconstrained images, we will later demonstrate through a series of experiments that this space is sufficiently large to generate adversarial examples for text. This attack is visualized for ViT and OCR pipelines in Figure 2.

3.2 Attack Technique

Visual adversarial examples encoded as text can be generated in a black-box model using a gradient-free optimization technique. We built our technique using differential evolution, a genetic optimization algorithm (Storn & Price, 1997). Adopting a similar approach previously proposed for imperceptible perturbations (Boucher et al., 2022; Shumailov et al., 2021), we use differential evolution to minimize similarity with the reference output for sequence-to-sequence tasks, and to minimize the output probability of the target class for classification tasks.

The algorithm to generate visual adversarial examples encoded as text is provided as Algorithm 1 in the Appendix. In summary, this algorithm takes as input a string to perturb, the target output class or reference output sequence, a visual text model, a perturbation budget representing the maximum number of injected diacritics, and a set of diacritics from which to select. A set of optimization parameters specific to differential evolution are also required by the algorithm (Storn & Price, 1997), although reasonable defaults may be used across repeated invocations. We provide the parameters chosen for our implementation in the following section. In the case of classification models, this algorithm will return a version of the input string perturbed with diacritics up to the allowed budget that minimizes the output probability of the supplied target class. In the case of sequence-to-sequence models – such as the machine translation task – the algorithm minimizes the supplied similarity metric with the supplied reference output.

The threat model for this attack contains an adversary who is able to submit inputs to a model, either via an API or locally. The adversary does not have access to the model's weights (i.e. a black-box setting) and the target model leverages *vision-driven defenses* for encoding attacks such as OCR pre-processing or a ViT architecture. The adversary's goal is to craft an input to the model that will cause an incorrect output.

4 Evaluation

The experimental evaluation of visual text adversarial examples requires examining two claims: first, that the adversarial examples effectively degrade the performance of a broad set of models, and second, that the adversarial examples do not affect human comprehension.

To examine the first claim, we will generate adversarial examples for an OCR model and measure performance degradation. We will then place this model into a pipeline that renders text input, performs OCR, and then calls the downstream model for both machine translation and toxic content detection and evaluate the performance of adversarial examples. Next, we will craft and measure adversarial examples for a ViT performing machine translation directly on rendered text. Finally, we will demonstrate an extension of these

Table 1: Model performance against adversarial examples of diacritical mark budget 5.

Model	**Baseline Perf**	**Adv. Ex. Perf** $\beta = 5$	**Attack Perf Drop**
TrOCR (Li et al., 2021)	0	12	n/a
TrOCR-FairSeq (Ott et al., 2019)	60.9	24.2	60.3%
TrOCR-IBM Toxic (IBM, 2020)	87.6	31.8	63.7%
ViT FairSeq (Salesky et al., 2021)	57.9	24.5	57.7%
CANINE SQuAD (Clark et al., 2022)	67.6	5.0	92.6%

attacks to a neural encoder model performing question answering. A summary of the experimental results is given in Table 1. These results are presented over varying budgets in Figure 3, with additional metric-specific visualizations given in Figures 4 to 8 in the Appendix.

All adversarial examples were generated on a cluster of machines each equipped with a Tesla P100 GPU and Intel Xeon Silver 4110 CPU running Ubuntu. We followed Algorithm 1 for example generation, selecting a a population size of 32, 10 iterations, a crossover probability of 0.7, a differential weight dithering from 0.5 to 1, and a varying budget ranging from 0 to 5. Where applicable, we select chrF (Popović, 2015) as the similarity metric $\mathcal{S}$. For each experiment, we generate 500 adversarial examples for each budget value. Perturbations are generated from the subset of diacritical marks in Unicode supported by the Microsoft Arial Unicode font (Microsoft, 2021), which is `U+300-U+346` and `U+360-U+361`. All datasets and models used are publicly available for research purposes. All experimental code and results are made available for future research at anonymous.4open.science/r/diacritics.

4.1 TrOCR

TrOCR is a transformer model (Vaswani et al., 2017) published by Microsoft implementing OCR (Li et al., 2021). TrOCR achieves state-of-the art performance on text recognition and leverages the modern transformer architecture.

Our first experiment evaluates the performance of TrOCR against diacritics injected via Algorithm 1. For each example, we render the input using the Microsoft Arial Unicode font (Microsoft, 2021) and pass the rendered image to the TrOCR model for inference. In this experiment, we use negative Levenshtein distance of the model output with the adversarial input as the similarity metric. If the TrOCR model is highly performant, we would expect - and indeed do see - a small average distance between the input and output when the attack budget is zero (representing no diacritic injections). The experiments seek to generate adversarial examples using diacritics injected into inputs sampled from the grammatically-correct validation split of the CoLA dataset (Warstadt et al., 2018), selected as a sample of short English-language inputs.

In Figure 4 we plot the distance between the output and both the perturbed and unperturbed input. Both measures grow with the budget, indicating that the model neither removes diacritical marks nor recognizes them as diacritics.

4.2 Defended FairSeq

In this set of experiments, we evaluated an English to French machine translation model published with Facebook's FairSeq toolkit (Ott et al., 2019). The model is a transformer architecture (Vaswani et al., 2017) trained on the WMT14 EN→FR corpus (Ott et al., 2018).

Following the retrofitted defense tactic previously proposed for Unicode attacks (Boucher et al., 2022), we placed the FairSeq EN→FR model in a pipeline that first renders text input and performs OCR via TrOCR.

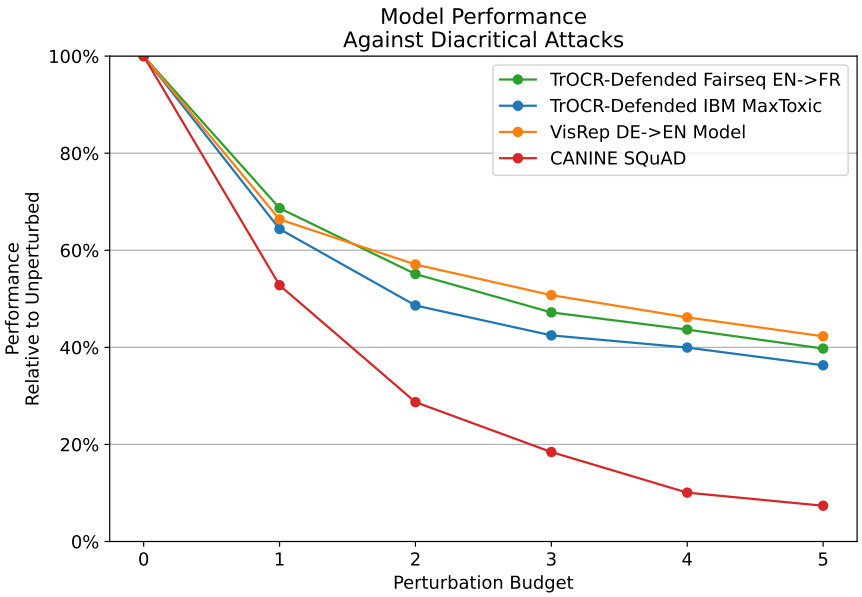

Figure 3: Performance of evaluated models across perturbation budgets relative to no perturbations.

The output of TrOCR is then passed as input to the FairSeq model. The experiments generate adversarial examples using diacritics for inputs drawn from the test split of the WMT14 EN→FR dataset.

In Figure 5 we plot the average chrF score (Popović, 2015) – a measure of translation quality selected as the similarity metric for this experiment – relative to the reference translation as computed by sacreBLEU (Post, 2018). The results indicate a decreasing translation quality with increasing diacritics budget.

4.3 Defended Toxic Classifier

Similarly, we evaluated another model defended from Unicode attacks via TrOCR. In this experiment, the downstream model is IBM's toxic content classifier (IBM, 2020). We use the Wikipedia Detox Dataset (Thain et al., 2017) as the inputs from which we craft adversarial examples.

The classification performance of the defended model over varying attack budgets is shown in Figure 6. There is, again, a negative relationship between the performance of the model and the attack budget. In fact, with a budget of just two diacritical marks, the performance of the model falls below random (i.e. 50% correct classification).

4.4 Visual FairSeq

In this experiment, we evaluate the performance of ViTs against diacritic attacks. Specifically, we target the fork of FairSeq (Ott et al., 2019) implementing German to English machine translation operating directly on visual inputs. This is the model used by Salesky et al. (2021) in their Visual Transformer proposal. Since this model operated upon inputs encoded as images, no OCR defense is necessary. We used the test split of the WMT20 DE→EN news dataset as to generate adversarial examples (Mathur et al., 2020).

In Figure 7 we plot the average chrF score over the perturbation budget. Similar to the TrOCR-defended machine translation experiment, we see a significant negative correlation between translation quality and diacritics perturbation budget.

4.5 CANINE Question Answering

Neural encoders are a recent NLP tool to replace dictionaries with embeddings learned directly from Unicode code points. Although these models do not unify the visual and encoded pipelines for text input, the learned embeddings claim to offer better flexibility for unknown characters. In this experiment, we evaluate whether diacritic attacks can subvert models leveraging neural encoders.

We evaluated a question answering model leveraging the CANINE neural encoder proposed by Google (Clark et al., 2022). Specifically, we evaluated a transformer model (Hsu, 2022) trained on the SQuAD dataset (Rajpurkar et al., 2016). Consistent with benchmarks against this dataset, we select F1 score as the similarity metric. The results, shown in Figure 8, show a sharp decline in F1 score with increasing diacritical perturbations.

5 User Study

Having shown that OCR and ViT models perform poorly when interpreting text containing visual perturbations and that these failures can be used to craft adversarial examples for downstream tasks, we conduct a user survey to understand how diacritical perturbations affect human comprehension.

5.1 User Study Methodology

The survey contained two main questions which are shown in Appendix A.3.

First, to measure human ability to read text containing diacritics, we ask respondents to retype sentences containing diacritics, specifying that they should omit diacritics or other marks. We selected 5 short examples to measure comprehension.

Second, to measure human comprehension of text containing diacritics, we asked respondents to identify whether a short sentence is toxic using the toxicity definition of Google's Perspective API (Google, 2021). For the sentences with a ground-truth toxic label, we filtered to select milder language. We used 12 sentences in total, evenly split between six toxic and six non-toxic examples. The sentences were randomly ordered within the survey question to avoid biasing respondents.

We surveyed 200 people using Amazon Mechanical Turk, limiting the survey to include only workers with at least a U.S. high school education since our questions concerned critical interpretation of text. The survey was advertised as "Answer a short (estimated 3-4 min) survey about reading and interpreting short sentences", intentionally avoiding any mention of diacritics, accents, or unusual marks.

In analyzing the results, we identify six identical low-quality responses that failed quality-control screening, leaving a total of $n = 194$ responses. We paid the remaining respondents $1.50 USD for completing the survey, an amount chosen to reflect a fair hourly wage.

5.2 User Study Results

Overall, the results demonstrate that diacritical marks do not significantly affect a human user's ability to comprehend the adversarial examples generated in the prior experiments. User responses were highly accurate in both reading and interpreting text containing diacritical marks, indicating that OCR and ViT models struggle with a text interpretation task humans find simple.

5.2.1 Text Legibility

We find that the overwhelming majority of human readers are able to correctly read and repeat the example sentences, with respondent accuracy rates ranging from 93.3% to 97.9%. In evaluating what constitutes an 'accurate' response, we ignore minor punctuation differences that seem unlikely to have been caused by diacritical marks (e.g., lack of capitalization, quotation marks around the sentence, a lack of a period at the end of the sentence, etc.).

Inaccurate responses mostly substituted a word that contained a diacritical mark for a similar but incorrect word (e.g., "tube" instead of "tub", or "John laid" instead of "John lay"). In such cases, we take a conservative approach and err on the side of marking it as incorrect since we cannot determine whether the substitution was due to the presence of diacritical marks or to ordinary typographical human errors. We conclude that while a small minority of respondents struggled to identify certain words containing diacritics, the vast majority of human readers are readily capable of comprehending text regardless of diacritical marks.

5.2.2 Text Comprehension

In the second question, we test individuals' ability to identify toxic text as a proxy for human comprehension of the text. Appendix A.4 contains the 12 examples used in the survey, evenly divided between six examples the TrOCR toxic content classifier labeled correctly (based on the ground-truth toxicity label from the Wikipedia Detox Dataset) and six it did not. Human readers were able to correctly classify all 12 sentences, including the six examples that the TrOCR model classified incorrectly, with respondent accuracy ranging from 80.4% to 99.0% for a given sentence.

In some cases, the percentage of human readers identifying a sentence as toxic was seemingly affected more by individual perceptions of toxicity than by the diacritical marks. For instance, the adversarial example "this band sucks anyways" was correctly identified as toxic by only 80.4% of respondents, a noticeably lower consensus than the other examples (the sentence with the second-lowest performance had an accuracy of 93.3%) though still comprising a clear majority. Given that human accuracy was easily maintained through all examples, we conclude that human performance is unaffected by the presence of diacritical marks even as OCR-defended model performance drops.

6 Discussion

6.1 Ethics

All attacks performed were run against local models and did not target any online services. We leveraged laboratory GPU resources for attack generation and did not use cloud services. Our user study was reviewed and approved by our Institutional Review Board, and no personal information was tracked from survey participants. We compensated users via Amazon Mechanical Turk at fair market prices. Since the user study involved viewing toxic content, we included a warning in the survey title that the assignment contains "language that may be considered rude or disrespectful."

6.2 Limitations

The adversarial examples in this work produce minor visual artifacts by design. These examples contrast the assumptions made by Boucher et al. (2022) in which no visual artifacts were permitted in encoding attacks. However, the results of our user study indicate that the diacritic perturbations produced do not affect the ability of humans reading text, thus giving motivation to break the imperceptibility assumption of encoding attacks.

6.3 Defenses

The attacks described in this paper all use Unicode diacritical marks to encode visual perturbations in the textual domain. Diacritics carry important linguistic value – particularly for pronunciation purposes – and cannot be disallowed in inputs without breaking internationalization.

However, it is reasonable to simply remove combining diacritical marks from the encoded text input to models prior to rendering for inference. In many settings, diacritical marks aren't used to encode the base meaning of language and removing them inside the model pipeline will effectively mitigate this attack without removing information necessary to the performance of the natural language model. Removing combining diacritical marks from visual text model inputs fully mitigates this attack vector, and can be accomplished by pre-processing inputs to remove all matches to the following Java-syntax RegEx: `[\u0300-\u036f]`.

In settings where diacritical marks can't be removed without hurting the performance of the model, the next best option is to perform adversarial training on the ViT, OCR, or neural encoder processing textual inputs. Yet, this will not come without additional risks via invariance-based adversarial examples where the model will learn to detect characters that are no longer human decipherable (Tramèr et al., 2020).

6.4 Rendering Design

In the visual text domain, setting-specific engineering details become pseudo-hyperparameters of the model. In this setting, it is no longer sufficient to communicate the model architecture and weights, but rather model providers must provide a thorough implementation or explanation of text rendering, chunking, and canvas handling along with their trained models.

One engineering challenge introduced by visual models is the need to implement text rendering onto fixed-sized canvases. By the nature of machine learning in the visual domain, model inputs must take the form of a single, fixed size image as specified by the model. This image size, which we will call the canvas dimension, introduces a key hyperparameter of the model that must be optimized. Additionally, one must also specify the font and font size that will be used for rendering. The implementation must also account for how to fit text of variable length onto the canvas, which may be further complicated if the font is not fixed width.

When there is too much text to fit onto the canvas, the canvas cannot be resized to make it fit; this would necessitate the image being further resized before inference, and the text becoming too small for the model to optimally recognize. Instead, long text inputs must be chunked into appropriately sized sections that each fit on their own canvas. These inputs must be processed separately and combined after inference (a process which must account for potential false whitespace caused by the edge of the canvas). Further, if words are broken across canvases the model will likely lose the performance benefits of recognizing adjacent characters of higher likelihood. We provide a visualization of this in Appendix A.5.

7 Related Work

7.1 Text-Based Adversarial Examples

While most adversarial examples proposed in prior work have been image classification attacks (Szegedy et al., 2014; Goodfellow et al., 2014; Athalye et al., 2018), recent work has demonstrated the existence of adversarial examples against text-based ML models. In particular, the accuracy of text classification models decreases on noisy inputs and it is possible to deliberately craft adversarial texts targeting systems reliant on text understanding (Gao et al., 2018; Li et al., 2018; Belinkov & Bisk, 2017; Khayrallah & Koehn, 2018; Alzantot et al., 2018; Zou et al., 2019).

In contrast to prior work presenting visibly perceptible perturbations, Boucher et al. (2022) previously showed that the discrepancy between text encoding and text visualization can be exploited as an attack vector against text-based machine learning models. They proposed optical character recognition (OCR) models as a "catch-all defense" against such attacks, arguing that the visual nature of the input would mitigate differences in visual and text encoding representation that left text-based processing systems vulnerable.

While Boucher et al. (2022) evaluated OCR as a defense against imperceptible perturbations, diacritics present a new class of attacks that are often scarcely more perceptible than invisible perturbations but have a substantial impact on OCR accuracy. Salesky et al. (2021) showed that Arabic diacritics and other minor text modifications substantially degrade the performance of text-based models, but concluded that ViTs achieve "relatively robust" performance against diacritized text. In this paper we demonstrate that both OCR systems and ViTs are indeed vulnerable to the Unicode-based adversarial examples introduced by Boucher et al. (2022), and that such examples significantly impact the performance of these models.

7.2 Attacks on OCR

Most prior work in breaking OCR systems has focused on visual CAPTCHAs (Yan & El Ahmad, 2007; Azad & Jain, 2013), but OCR is commonly used in a variety of preprocessing tasks. Recently, Chen et al.

(2020) demonstrated a class of attacks on OCR using adversarial watermarks in a twist on classic adversarial image examples. The commonality of watermarks on inputs to OCR systems makes this attack effectively imperceptible to humans, though this attack relies on non-text image data in inputs. In contrast, in this work we present a class of attacks in which the adversarial examples are encoded directly into text rather than perturbing an image's background. To the best of our knowledge, ours is the first such attack that manipulates pre-rendered text directly, using the resulting image as an adversarial example.

7.3 Toxic Content Detection

Identifying toxic content online has proven to be one of the most pervasive yet elusive research questions in natural language processing. A rapidly growing area of research, toxic content detection is challenging even when classifying standard Unicode characters in the English language, and toxic content classifiers are generally not robust against adversarial examples (Kurita et al., 2019; Risch & Krestel, 2020). Online users commonly deploy a variety of simple text modifications to bypass platform filters: Kurita et al. (2019) gives the adversarial example of writing "s*ut up" in the place of 'shut up'. While most adversarial examples against toxic content detection are visually perceptible, Boucher et al. (2022) showed it is also possible to modify the Unicode text encoding of toxic content in a way that is visually imperceptible but fools content detection models.

8 Conclusion

We have presented a novel method of attacking language models operating in the visual domain by encoding adversarial examples in the textual domain. Once rendered, these examples will generate small visual artifacts that drastically decrease model performance. We accomplish this through the injection of Unicode's combining diacritical marks via a gradient-free optimization designed to minimize target model performance over the perturbed input. We generate such adversarial examples for real-world models published by Microsoft, Facebook, IBM, and Google finding the attacks to significantly degrade model performance. We also conduct a user study which concludes that the comprehension of human users is not affected by the diacritic injections used in our attacks. In order to defend against such attacks, we conclude that model designers should remove combining diacritical marks from inputs prior to inference.

References

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. Generating natural language adversarial examples. *arXiv preprint arXiv:1804.07998*, 2018.

Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. Synthesizing robust adversarial examples. In *International conference on machine learning*, pp. 284–293. PMLR, 2018.

Silky Azad and Kiran Jain. Captcha: Attacks and weaknesses against ocr technology. *Global Journal of Computer Science and Technology*, 2013.

Yonatan Belinkov and Yonatan Bisk. Synthetic and natural noise both break neural machine translation. *arXiv preprint arXiv:1711.02173*, 2017.

Yonatan Belinkov and Yonatan Bisk. Synthetic and natural noise both break neural machine translation. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=BJ8vJebC-`.

Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In *Joint European conference on machine learning and knowledge discovery in databases*, pp. 387–402. Springer, 2013.

Nicholas Boucher, Ilia Shumailov, Ross Anderson, and Nicolas Papernot. Bad Characters: Imperceptible NLP Attacks. In *43rd IEEE Symposium on Security and Privacy*. IEEE, 2022.

Lu Chen, Jiao Sun, and Wei Xu. Fawa: Fast adversarial watermark attack on optical character recognition (ocr) systems. *arXiv preprint arXiv:2012.08096*, 2020.

Jonathan H. Clark, Dan Garrette, Iulia Turc, and John Wieting. Canine: Pre-training an efficient tokenization-free encoder for language representation. *Transactions of the Association for Computational Linguistics*, 10:73–91, 2022. doi: 10.1162/tacl_a_00448. URL https://aclanthology.org/2022.tacl-1.5.

Ji Gao, Jack Lanchantin, Mary Lou Soffa, and Yanjun Qi. Black-box generation of adversarial text sequences to evade deep learning classifiers. In *2018 IEEE Security and Privacy Workshops (SPW)*, pp. 50–56. IEEE, 2018.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

Google. Perspective API, 2021. URL https://www.perspectiveapi.com/.

Hossein Hosseini, Sreeram Kannan, Baosen Zhang, and Radha Poovendran. Deceiving google's perspective api built for detecting toxic comments, 2017.

Jeff Hsu. Splend1dchan/canine-s-squad, 2022. URL https://huggingface.co/Splend1dchan/canine-s-squad.

IBM. Toxic comment classifier, December 2020. URL https://github.com/IBM/MAX-Toxic-Comment-Classifier.

Huda Khayrallah and Philipp Koehn. On the impact of various types of noise on neural machine translation. *arXiv preprint arXiv:1805.12282*, 2018.

Keita Kurita, Anna Belova, and Antonios Anastasopoulos. Towards robust toxic content classification. *arXiv preprint arXiv:1912.06872*, 2019.

Jinfeng Li, Shouling Ji, Tianyu Du, Bo Li, and Ting Wang. Textbugger: Generating adversarial text against real-world applications. *arXiv preprint arXiv:1812.05271*, 2018.

Minghao Li, Tengchao Lv, Lei Cui, Yijuan Lu, Dinei Florencio, Cha Zhang, Zhoujun Li, and Furu Wei. TrOCR: Transformer-based Optical Character Recognition with Pre-trained Models. *arXiv preprint arXiv:2109.10282*, 2021.

Nitika Mathur, Johnny Wei, Markus Freitag, Qingsong Ma, and Ondřej Bojar. Results of the WMT20 metrics shared task. In *Proceedings of the Fifth Conference on Machine Translation*, pp. 688–725, Online, November 2020. Association for Computational Linguistics. URL https://aclanthology.org/2020.wmt-1.77.

Microsoft. Arial Unicode MS font family, November 2021. URL https://www.unicode.org/charts/PDF/UFE20.pdf.

Myle Ott, Sergey Edunov, David Grangier, and Michael Auli. Scaling neural machine translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pp. 1–9, Brussels, Belgium, October 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-6301. URL https://www.aclweb.org/anthology/W18-6301.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*, 2019.

Luca Pajola and Mauro Conti. Fall of giants: How popular text-based mlaas fall against a simple evasion attack. In *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*, pp. 198–211, 2021. doi: 10.1109/EuroSP51992.2021.00023.

Maja Popović. chrF: character n-gram F-score for automatic MT evaluation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pp. 392–395, Lisbon, Portugal, September 2015. Association for Computational Linguistics. doi: 10.18653/v1/W15-3049. URL https://aclanthology.org/W15-3049.

Matt Post. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pp. 186–191, Belgium, Brussels, October 2018. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W18-6319.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 2383–2392, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1264. URL https://aclanthology.org/D16-1264.

Julian Risch and Ralf Krestel. Toxic comment detection in online discussions. In *Deep Learning-Based Approaches for Sentiment Analysis*, pp. 85–109. Springer, 2020.

Elizabeth Salesky, David Etter, and Matt Post. Robust Open-Vocabulary Translation from Visual Text Representations. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Online, November 2021. Association for Computational Linguistics. URL https://arxiv.org/abs/2104.08211.

Ilia Shumailov, Yiren Zhao, Daniel Bates, Nicolas Papernot, Robert Mullins, and Ross Anderson. Sponge examples: Energy-latency attacks on neural networks. In *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*, pp. 212–231, 2021. doi: 10.1109/EuroSP51992.2021.00024.

Rainer Storn and Kenneth Price. Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces. *Journal of Global Optimization*, 11(4):341–359, December 1997. ISSN 1573-2916. doi: 10.1023/A:1008202821328. URL https://doi.org/10.1023/A:1008202821328.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *The Second International Conference on Learning Representations*. ICLR, 2014.

Nithum Thain, Lucas Dixon, and Ellery Wulczyn. Wikipedia talk labels: Toxicity, Feb 2017. URL https://figshare.com/articles/dataset/Wikipedia_Talk_Labels_Toxicity/4563973/2.

The Unicode Consortium. The Unicode Standard, Version 14.0, September 2021a. URL https://www.unicode.org/versions/Unicode14.0.0.

The Unicode Consortium. Combining Diacritical Marks, 2021b. URL https://www.unicode.org/charts/PDF/U0300.pdf.

The Unicode Consortium. Combining Half Marks, 2021c. URL https://www.unicode.org/charts/PDF/UFE20.pdf.

The Unicode Consortium. Combining Diacritical Marks Extended, 2021d. URL https://www.unicode.org/charts/PDF/U1AB0.pdf.

The Unicode Consortium. Combining Diacritical Marks Supplement, 2021e. URL https://www.unicode.org/charts/PDF/U1DC0.pdf.

The Unicode Consortium. Combining Diacritical Marks for Symbols, 2021f. URL https://www.unicode.org/charts/PDF/U20D0.pdf.

Florian Tramèr, Jens Behrmann, Nicholas Carlini, Nicolas Papernot, and Jörn-Henrik Jacobsen. Fundamental tradeoffs between invariance and sensitivity to adversarial perturbations, 2020.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. Neural network acceptability judgments. *arXiv preprint arXiv:1805.12471*, 2018.

Jeff Yan and Ahmad Salah El Ahmad. Breaking visual captchas with naive pattern recognition algorithms. In *Twenty-Third annual computer security applications conference (ACSAC 2007)*, pp. 279–291. IEEE, 2007.

Wei Zou, Shujian Huang, Jun Xie, Xinyu Dai, and Jiajun Chen. A reinforced generation of adversarial examples for neural machine translation. *arXiv preprint arXiv:1911.03677*, 2019.

A Appendix

A.1 Attack Algorithm

Algorithm 1: Black-box generation of visual text adversarial example

Input: text $\mathbf{x}$, reference output $\mathbf{y}$, model $\mathcal{M}$, perturbation budget β, diacritics list $\mathbf{D}$
Optimization Params: similarity metric $\mathcal{S}$, population size s, evolution iterations m, differential weight $F \in [0, 2]$, crossover probability $CR \in [0, 1]$
Result: Encoded text visually similar to $\mathbf{x}$ which is an adversarial example against $\mathcal{M}$ when rendered

```
procedure PERTURB (p)
  x̄ := x
  for n := 0 to β do
    d, i := p_n
    if round(i) ≥ 0 then
      x̄ = insert(x̄, D_d, round(i))
    end if
  end for
  return x̄
end procedure

Randomly initialize   P := {p_0, ..., p_s},
        where p_n := [(d_0, i_0), ..., (d_β, i_β)],
  where d_n ~ U(0, |D|),  i_n ~ U(−1, |x|)
                    ▷ U is uniform dist.
if M is classifier then
  F(x̂) = M(x̂)        ▷ logit of target class
else
  F(x̂) = S(M(x̂), y)
end if

for i := 0 to m do
  for j := 0 to s do
    p_a, p_b, p_c ←rand P s.t. j ≠ a ≠ b ≠ c
    R ~ U(0, |p_j|)
    p̂_j := p_j
    for k := 0 to |p_j| do
      r_j ~ U(0, 1)
      if r_j < CR or R = k then
        p̂_{j_k} = p_{a_k} + F × (p_{b_k} − p_{b_k})
      end if
    end for
    if F(PERTURB(p̂_j)) < F(PERTURB(p_j)) then
      p_j = p̂_j
    end if
  end for
end for
f̄ := {F(PERTURB(p_0)),
      ..., F(PERTURB(p_s))}
return PERTURB(P_argmax(f̄))
```

A.2 Metric-Specific Experimental Results

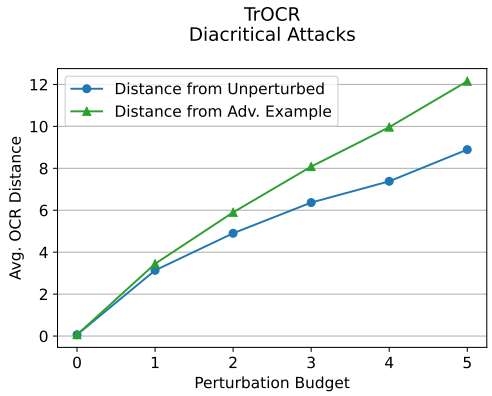

Figure 4: Evaluation of diacritic adversarial examples against TrOCR.

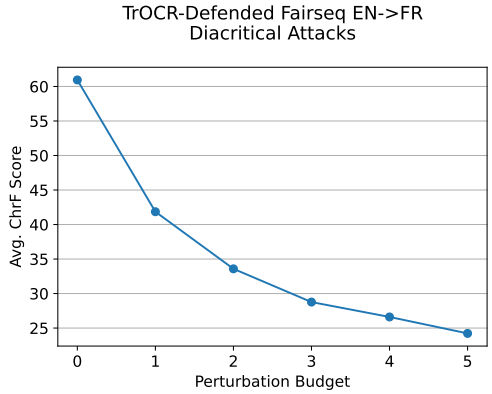

Figure 5: Eval of diacritic adv. examples against FairSeq EN->FR translation model defended by TrOCR.

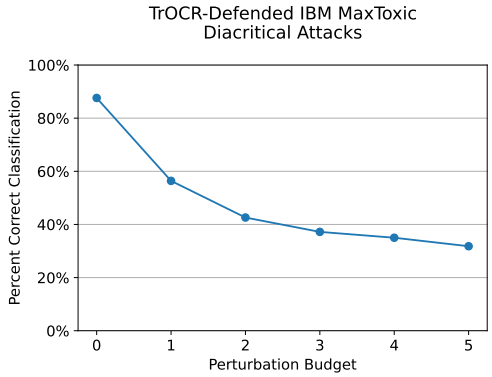

Figure 6: Eval of diacritic adv. examples against IBM's toxic content detection model defended by TrOCR.

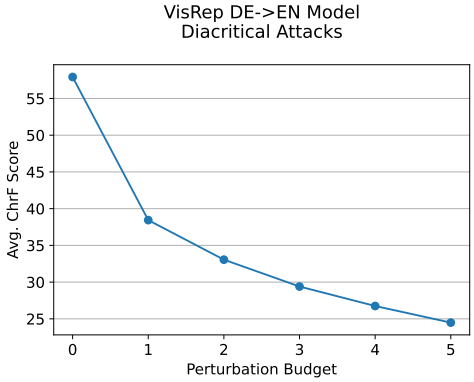

Figure 7: Eval of diacritic adv. examples against Visual Representation fork of FairSeq for DE->EN translation.

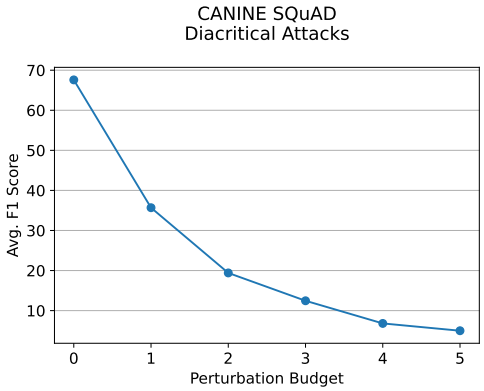

Figure 8: Eval of diacritic adv. examples against SQuAD transformer using Google's CANINE neural encoder.

A.3 User Study Questions

A) For each of the following five sentences, retype the sentence below using only standard English characters. For example, if the sentence given is "The wåll įš red", type "The wall is red".

1. The tüb leakęd water.

2. It annoys me tḥaṯ̈ Fido barks.

3. I'll work on it if I can.

4. Aṅ evil thọught struck Dave.

5. John lay the ball in the box.

Figure 9: First survey question measuring human readers' comprehension of text containing diacritics.

B) Toxicity is defined as "a rude, disrespectful, or unreasonable comment that is likely to make someone leave a discussion." For each of the following sentences, check the box next to the sentence if the sentence is toxic.

1. You are a coward ąňd a fool. ☐
2. Pìç of the day Wednesday ☐
3. It is a golden hair. ☐
4. It ḑid not rain. ☐
5. Ahå - fọunḍ it . - ☐
6. Your mother is illiterate! ☐
7. Yįur house is ūg̏ly, too. ☐
8. this band sucks anyways ☐
9. John often meęts Mary. ☐
10. :I'Îl reply on your talk page. ☐
11. ::Ah yes, aň ȁsshole ☐
12. I didn'ṭ v̤andalize your user page, idiot. ☐

Figure 10: Second survey question measuring human readers' ability to identify toxic text containing diacritics.

A.4 Attack Examples

Example	Ground Truth Toxicity Label	Model Label	Human Reader Label	Human Reader Performance	Model Correct	Humans Correct
Yi̧ur house is ūg̈ly, too.	True	False	True	93.8%	✗	✓
I didn't ṭ ṷ̈andalize your user page, ı͡diot.	True	False	True	94.8%	✗	✓
::Ah͡ ̧yes, an̓ ̀asshole	True	False	True	96.4%	✗	✓
Ahå ̣ ̣ fo̧und̩ it . –	False	True	False	99.0%	✗	✓
:I'Îl^rep̏ly on youř talk page.	False	True	False	99.0%	✗	✓
Pìç of ṯhe day‾Wednesday	False	True	False	99.0%	✗	✓
thiś band sucks anyways	True	True	True	80.4%	✓	✓
You are a˝ċoward ạňd a fool.	True	True	True	97.9%	✓	✓
Your motheṙ is illiterate!	True	True	True	93.3%	✓	✓
It¸is¸a golden hair.	False	False	False	98.5%	✓	✓
It ḍid not rain.	False	False	False	99.5%	✓	✓
John oftẹn¨meẹts Mary.	False	False	False	99.0%	✓	✓

Table 2: Comparison of OCR and ViT model and human performance detecting toxicity in adversarial examples. 'True' means the adversarial example was considered toxic, while 'False' indicates the example was non-toxic.

A.5 Rendering Design

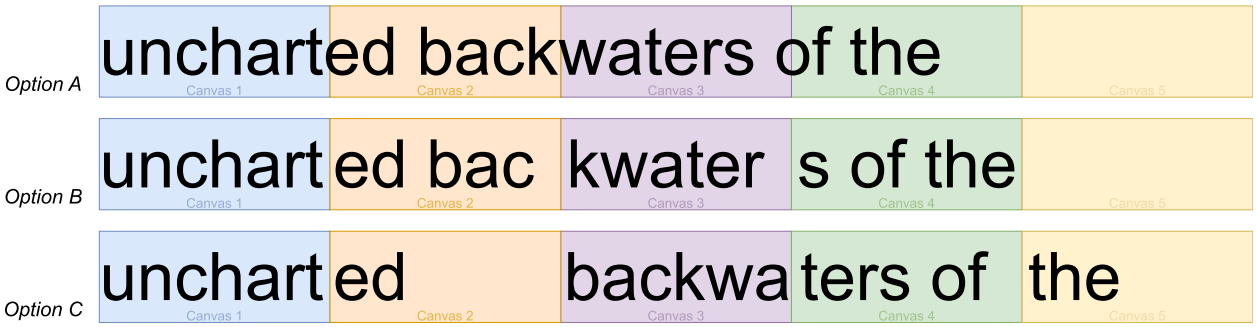

Figure 11: An example of three chunking options to render the text "uncharted backwaters of the" onto a series of fixed-size canvases for inference. Option A breaks letters across canvases, which will cause failures during inference. Option B breaks the word *backwaters* across more canvases than necessary, decreasing the model's ability to recognize likely adjacencies. Option C is likely the strongest choice in this example, but as with all other options the model pipeline must account for whitespace added by chunking after inference.