Multimodal Embedding Alignment for Homoglyph Detection

by

Connie Cao

B.S. Computer Science & Engineering and Finance
MIT, 2025

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

MASTER OF ENGINEERING IN ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2026

Authored by: Connie Cao
Department of Electrical Engineering and Computer Science
January 16, 2026

Certified by: Amar Gupta
Research Scientist, Thesis Supervisor

Accepted by: Katrina LaCurts
Chair, Master of Engineering Thesis Committee

Multimodal Embedding Alignment for Homoglyph Detection

by

Connie Cao

Submitted to the Department of Electrical Engineering and Computer Science
on January 16, 2026 in partial fulfillment of the requirements for the degree of

MASTER OF ENGINEERING IN ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

ABSTRACT

With the increasing use of digital platforms for business registration and online payments, the need for technology that can detect visually deceptive or intentionally manipulated entity names or homoglyphs grows. Homoglyph attacks occur when adversaries replace characters with visually similar alternatives in known entity names in an attempt to spoof users. Exisiting string-matching metrics such as edit distance or token-based methods lack the ability to capture the visual similarities of homoglyphic attacks while Optical Character Recognition (OCR) systems have significant computational overhead and are not robust to variations in font or image quality. To address these limitations, this thesis proposes a multimodal alignment approach that jointly learns a shared embedding space for text and glyphs (images of rendered text). By explicitly optimizing for the detection of visual similarity in text, the model learns to align strings that look similar and push strings that are visually dissimilar apart. In training, the model sees pairs consisting of a real entity name and either a visually similar spoof or an unrelated negative example, allowing it to learn which strings should be categorized as spoofs and which to separate. Evaluated on a large-scale homoglyph attack dataset of domain names, the proposed approach outperforms traditional string-matching methods and OCR systems for spoof detection. By integrating this multimodal embedding into homoglypic attack detection, we can increase the accuracy and reduce the false negatives in existing spoof detection methods and provide more reliable identification of fraudulent entities within the financial industry.

Thesis supervisor: Amar Gupta
Title: Research Scientist

Acknowledgments

First, I would like to thank Dr. Gupta, my thesis advisor, for all his support, guidance, and encouragement throughout this entire research process. His insight and mentorship helped shape this work, and I am very grateful for the time and care he has invested in helping me grow as a researcher and person.

I would also like to thank Hailey Boriel, who I've been able to work closely with in designing experiments and generate large-scale embedding datasets. Her technical expertise and dedication made the research process both more rigorous and enjoyable.

I also want to acknowledge Sasha Jovanovic-Hacon, whose prior work on VA-TE laid a conceptual and technical foundation that this thesis builds upon. Sasha's contributions in both developing the original system and in sharing guidance throughout this process were essential in shaping the direction of this research.

This research would also not be possible without the guidance and collaboration of our sponsors, Miguel Domingos Wanderley, Guilhereme Rinaldo, Gustavo Bicalho, and Jair Carvalho at Itaú Unibanco.

Finally, I am the most grateful for the support of my family, especially my parents, who have been with me through every challenge and milestone during my time at MIT.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Digital systems, such as payment or financial account registration systems, rely on automated identity verification by processing textual keys or entity names such as usernames, account names, domain names, and brand names. As these systems expand across languages and use cases, they become vulnerable to homoglyph attacks and fraudulent entities. Homoglyph attacks are manipulations of strings created by replacing characters in a given string with visually similar characters so that when rendered, both the original string and manipulated string look nearly identically despite being textually different [1,2]. These attacks undermine the reliability of text-based string-matching techniques and machine learning models used by many systems to verify strings.

Homoglyph attacks present a unique multimodal challenge in that the spoofed name may be drastically different from the legitimate name textually but look almost identical visually. Traditional string-similarity or edit-distance techniques such as Levenshtein distance [3] or Jaro similarity [4] and token-level cosine similarity use characters or tokens to determine whether strings are homoglyph attacks or not but they are not able to capture the visual similarity of more complex spoofs. As a result, they fail to distinguish harmless textual differences from visually deceptive homoglyph attacks, leading to high false negatives for these more nuanced and complex spoofs [2].

OCR systems attempt to address the visual component of homoglyph attacks by converting rendered images into text before performing similarity scoring or applying spoof-detection techniques. However, OCR systems lack robustness since they are sensitive to variations in font, image quality, and noise. Furthermore, OCR pipelines add a large computational overhead which is not feasible or scalable for large-scale spoof detection systems in industries such as financial services where millions of textual identifiers like account names might be processed and verified every day [5].

Recent work on Visually Aligned Text Embeddings (VATE) [6] illustrates that visually grounded text embeddings can better detect homoglyph attacks by using embeddings aligned with human visual perception. However, these models are text-based and do not build a multimodal shared space between image and text. As a result, they do not fully utilize the perceptual structure captured by vision language encoders, and they use visual signals in training rather than as a component of the representation space. These limitations motivate a multimodal approach to spoof detection that translates textual and visual signals into a joint space.

As such, this thesis proposes a multimodal embedding alignment framework that directly joins rendered images of strings (glyphs) and their textual representation into a single embedding space. Rather than relying on text embeddings alone, this proposed method uses an image encoder as a perceptual anchor and trains a lightweight MLP projection to map text embeddings into the image embedding space. This framework's design is motivated by the idea that homoglyph detection is a visual task and involves analyzing how a string looks rather than relying entirely on what tokens it is comprised of.

The model is trained contrastively using pairs in which a legitimate name is matched either with a visually similar homoglyph (positive pair) or with an unrelated negative example (negative pair). This contrastive learning strategy allows the system to pull visually similar strings closer together in the embedding space while pushing strings that are not visually similar far apart. Curriculum learning further improved the robustness of the system by gradually exposing the model to increasingly challenging spoof pairs.

In this thesis, we begin by reviewing prior work on homoglyph attacks and text-based and image-based string similarity models. We then introduce the proposed multimodal alignment framework and describe the system's architecture. Next, we present the experimental setup, including training objectives, triplet construction, and the MLP projection mechanisms used to map text embeddings into an image-embedding space. Finally, we demonstrate that aligning text representations to a visually grounded embedding space yields improvements in homoglyph attack detection, illustrating how multimodal alignment can offer a scalable and robust solution to spoof detection.

Chapter 2

Related Works

Homoglyph attack detection combines the methods and research of computer vision and natural language processing. Because attackers exploit the visual similarity of characters, effective homoglyph detection techniques require capturing both the textual structure of strings and their rendered visual structure. As a result, prior work that supports homoglyph detection involves several research areas: vision–language models that learn joint embedding spaces, image encoders capable of capturing fine-grained visual features, OCR systems that transform images into text, perceptual similarity metrics designed to model human visual perception, traditional string similarity metrics, and contrastive and curriculum learning as model training strategies. Together this existing literature helps to motivate the design choices behind the proposed multimodal alignment framework.

2.1 Vision-Language Models

Recent progress in large-scale vision-language pre-training has produced models that learn joint embedding spaces for images and text. CLIP [7] demonstrated that contrastive training on hundreds of millions of image-caption pairs can produce representations with high recognition performance. CLIP-type models embed images and text into a shared space and use contrastive loss training to push matching image-text pairs closer while pushing mismatched pairs apart [7].

Further work in this area has scaled both the data used and the size of models developed and led researchers to explore more complex text and image alignment strategies. ALIGN [8] showed that VLM models trained on noisy but large-scale image and alt-text data can also lead to high recognition performance, while "Align Before Fuse" [9] introduced momentum distillation to separately align image and text representations before joining them for multimodal tasks. The LiT [10] model proposed holding a pre-trained image encoder static and training a text tower with contrastive loss training, illustrating how keeping the image backbone steady can improve the mapping between image and text embeddings. More recent VLMs such as Flamingo [11], PaLI-X [12], and visually instructed models [13] combine large language models with visual encoders for few-shot reasoning and instruction following.

The work in this thesis relates most closely to SigLIP [14], which replaces the softmax contrastive loss used in CLIP with a sigmoid-based loss function to better handle large batch sizes and negative sampling. SigLIP saw strong classification performance and retrieval

benchmarks. Its vision tower was also trained on images of objects instead of glyphs as this thesis does.

2.2 Image Encoders

VLMs leverage strong image backbones that have been pre-trained on large-scale image classification or self-supervised tasks. Vision Transformers (ViT) [15] introduced a fully transformer-based architecture for images, showing that with enough data, attention models can match or even outperform convolutional networks. Masked autoencoders (MAE) [16] extended ViT by masking and reconstructing random patches which enabled scalable self-supervised pre-training. DINOv2 [17] further refined self-distillation for unsupervised visual feature learning and produced more robust representations.

Classical convolutional networks such as ResNet [18], EfficientNet [19], EfficientNetV2 [20], and ConvNeXt [21] are also performant, especially when computational efficiency is important. ImageGPT [22] uses generative pretraining from pixels, treats images as sequences of tokens, and trains auto-regressive transformers in order to predict pixels. In this way, ImageGPT implicitly learns visual structures without explicit supervision.

Although these encoders are typically evaluated on non-glyph image tasks such as ImageNet image classification, their architectures and learned features can also be used to learn glyph structures. Prior work on glyph-based natural language processing has shown that rendering characters as images and processing them with CNNs or transformers can capture sub-character visual structure. This was done in Glyce [23] for Chinese glyph embeddings and in [24] for multi-script deep-learning-based character recognition systems.

2.3 Optical Character Recognition Systems

OCR systems convert documents and images with text into token sequences and are widely used in various industries. Tesseract [25] is an open-source OCR system that combines LSTM-based sequence models with a language model to recognize and read text line-by-line. Tesseract has primarily been used for printed documents, and its performance decreases when there is noise, blur, or non-traditional fonts. Another industry-level OCR system is Meta's Rosetta [5] which uses a CNN-based text detector to propose candidate text regions and then a sequence recognizer that outputs character sequences. Rosetta is designed for large-scale deployment such as understanding text in billions of social media images and optimizes for throughput and robustness.

Beyond these systems, there has been extensive research on scene text recognition and end-to-end text recognition. Baek et al. [26] provided a comprehensive analysis of datasets and model architectures, highlighting inconsistencies in evaluation and benchmarking. Shi et al. [27] proposed an end-to-end trainable pipeline based on convolutional and recurrent networks for image-based sequence recognition. Jaderberg et al. [28] introduced powerful CNN-based models for reading text "in the wild" and performed well on signage and natural scenes. More recently, TrOCR [29] replaced recurrent decoders with transformers and used pre-trained language models, achieving state-of-the-art results on a variety of OCR

benchmarks. Work on text detection, such as border semantics-aware detectors [30] has also further improved recall and precision in challenging image layouts and improved OCR performance.

One challenge of using OCRs in homoglyph attack detection is that while OCR systems are effective at mapping pixels to discrete characters and strings, once text has been recognized and transcribed, any subsequent systems are limited to using only text-based solutions for spoof detection such as string similarity metrics [31]. OCRs are also computationally costly since they are analyzing and processing images pixel-by-pixel.

2.4 Homoglyph and Visual Spoofing Attacks

Homoglyph attacks exploit visually similar characters to construct spoofed entity names that look almost identical visually to legitimate strings. Early work on Unicode-based spoofing relied on manually identifying visually similar characters across scripts and detecting attacks using pattern matching [32], but these approaches weren't scalable due to how large the Unicode character space is.

Woodbridge et al. [1] proposed detecting homoglyph attacks using a Siamese neural network trained on rendered domain names. This model embeds pairs of strings using a shared CNN and predicts whether they are visually similar/spoofs of another, and the model showed improved accuracy over edit-distance and heuristic baselines. Vinayakumar and Soman [33] introduced a Siamese architecture with recurrent structures and character-level embeddings and treated homoglyph detection as a similarity learning problem on raw strings. Lu et al. [34] developed an unpaired homoglyph detection system using convolutional networks, illustrating that spoof detection can be solved without paired data. Other work has explored rapid prediction and detection frameworks [35], data augmentation using GANs (PhishGAN) to generate spoofed domains [36], and hybrid machine-learning models using hash functions [2] to improve the robustness of spoof detection. More recent research such as Gomathy [37] extend neural network spoof detection methods and emphasize their use in cybersecurity contexts.

These methods all attempt to learn a model that distinguishes between benign strings and visually similar spoofs. However, they are typically trained from scratch on homoglyph-specific datasets and do not use large-scale visual pre-training (e.g. ViT, DINOv2, SigLIP) or explicitly utilize text representations. Many approaches rely either solely on image-based CNNs over rendered text [1,34] or on character-level sequence models [33,37], limiting their ability to generalize across different fonts or more complex, unseen spoofing patterns.

2.5 Perceptual Image Similarity

Perceptual similarity metrics aim to better approximate how visually similar two images are to the human eye compared to pixel-level comparisons. Zhang et al. [38] introduced Learned Perceptual Image Patch Similarity (LPIPS), which measures the distance between deep feature activations of images and showed that perceptual similarity metrics correlate significantly better with human perceptual judgments than Peak Signal-to-Noise Ratio (PSNR)

or Structured Similarity Image Metric (SSIM). Classical feature descriptors like SIFT [39] and ORB [40] work by finding distinctive points in an image and describing the small regions around them. This allows images to be matched reliably even if they are shifted, resized, or viewed from different angles. SSIM [41] compares local luminance, contrast, and structure in small windows, providing a widely used perceptual image quality metric for full image comparisons.

In the context of homoglyph detection, we can view pairs of rendered strings as images and apply these perceptual image similarity metrics to detect visual similarity. However, these metrics are not trained specifically for text, so they may be overly sensitive to more complex transformations such as font changes or too insensitive to subtle character substitutions that change semantics (e.g. "rn" vs. "m"). Moreover, they work in the image domain and don't utilize any representation of the underlying text.

2.6 String Similarity Metrics

Before neural networks, homoglyph detection relied heavily on string similarity metrics. Levenshtein distance [3] measures the minimum number of edits (insertions, deletions, substitutions) needed to transform one string into another and is a standard baseline in string matching. Jaro and Jaro-Winkler similarity [4,42] are other widely used string similarity metrics and perform best for shorter strings since they weigh characters agreements at the beginning of a string more heavily. Cohen et al. [43] compared a variety of string metrics for matching names and records in data cleaning tasks and highlighted the trade-offs between precision, recall, and computational cost of each.

String similarity metrics are widely used in homoglyph detection and deduplication tasks since they are relatively simple and easy to interpret. However, they operate on character sequences and may be insensitive to edits that lead to visual similarity. At the same time, minor edits that drastically change the visual appearance of strings may have a low edit distance and be categorized as a spoof.

2.7 Contrastive Learning

Contrastive learning is used to pull matching pairs of strings close together in an embedding space while pushing mismatched pairs apart. In spoof detection, this would mean pulling visually similar strings together and separating visually dissimilar strings. Hadsell et al. [44] introduced contrastive loss for learning invariant mappings by training the model to output small distances for matched pairs and large distances for mismatched ones. Weinberger et al. [45] proposed Large Margin Nearest Neighbor (LMNN) for supervised metric learning by learning a Mahalanobis distance for kNN classification. FaceNet [46] introduced triplet loss for face recognition by learning an embedding where images of the same person are close together and images of different people are far apart in the embedding space.

More recently, self-supervised contrastive methods such as SimCLR [47], BYOL [48], and MoCo [49] have shown that optimizing data augmentations and contrastive objectives can teach models useful visual features without needing any labels. These ideas back CLIP-style

vision–language models [7,14] and influence how we design our loss functions to align text and glyph embeddings. Supervised Siamese neural network architectures for homoglyph detection [1,33] were an early application of these ideas in security tasks but were limited by smaller image backbones and the need for task-specific data.

2.8 Curriculum Learning

Curriculum learning refers to training strategies in which models are trained on samples organized in order of difficulty instead of using a uniformly random distribution. Bengio et al. [50] introduced the idea that learning systems benefit from progressing from easier examples to more difficult ones. This research's results established curriculum learning as a general optimization method that helps to stabilize training and improve model generalization, especially in deeper networks.

Wang et al. [51] provide a comprehensive survey, categorizing curricula into data-level approaches that order samples by difficulty, task-level approaches that sequence subtasks, and model-level approaches that adapt to the internal dynamics of the learner. They also further relate curriculum learning to self-paced learning.

Automated curriculum construction has also become a major research direction. Graves et al. [52] propose a stochastic syllabus based on a multi-armed bandit that selects training examples according to the model's learning progress, speeding up convergence without hand-designing difficulty schedules. Furthermore, Narvekar et al. [53] and Portelas et al. [54] survey and document how curriculum learning has been used in reinforcement learning, where structured task sequencing has shown to improve exploration efficiency.

More recent work explores adaptive curricula for supervised learning. Kong et al. [55] dynamically estimate difficulty and update the training schedule throughout optimization, and Liu et al. [56] provide a unified view of curriculum evaluation systems, discussing the roles of difficulty estimators, training schedulers, and loss-based progress measures.

Overall, curriculum learning provides training strategies for designing the learning schedule of deep models. In the context of homoglyph detection, this motivates our easy–medium–hard data construction, where using progressively more difficult glyph pairs encourages the alignment model to better separate spoof pairs from benign pairs than uniform sampling.

2.9 Cross-Modal Retrieval and Text–Image Matching

Visual–semantic embeddings are models that map images and text into a shared embedding space so that semantically related image–text pairs end up close together. Cross-modal retrieval uses these shared embeddings to search across spaces (i.e. searching for an image based on a text query). VSE++ [57] improved image-caption retrieval by focusing on "hard negatives" in contrastive loss training, illustrating how using negative examples in training can improve retrieval performance. Dual-encoding architectures such as video retrieval [58] for example learn joint embedding spaces for visual and textual spaces, allowing for retrieval tasks without explicit supervision on the target dataset. Young et al. [59] studied new similarity metrics for semantic inference over visual event descriptions.

Across these research areas, a common limitation is that existing methods to support homoglyph detection utilize either only text or visual information and therefore fail to capture the joint visual–textual structure of homoglyph attacks. OCR systems miss the visual structure of text once images are transcribed into text, string similarity metrics ignore visual form entirely, and prior homoglyph detection neural models train task-specific CNNs without leveraging large-scale visual pretraining. At the same time, vision language models are not trained on glyph images and may not perform well when applied directly to homoglyph detection. Motivated by these gaps, this thesis proposes a multimodal alignment framework that exploits strong vision encoders while learning a mapping from text embeddings into a visually grounded representation space designed for homoglyph detection. By integrating techniques from contrastive learning, curriculum design, and cross-modal retrieval, our proposed approach trains a performant lightweight multimodal alignment MLP that supports homoglyph detection.

Chapter 3

Methods

In order to study how visual and textual features contribute to homoglyph detection, we establish the models, representations, and training objectives that support the proposed multimodal embedding alignment model. These components provide the foundation for how our proposed approach assesses how glyph images and text embeddings can be aligned to capture the visual similarity structure associated with spoofed strings.

3.1 Multimodal Embedding Alignment Model

3.1.1 Motivation for Multimodal Alignment

Homoglyph attacks exploit the fact that two strings may be semantically unrelated yet visually similar to the human eye. For example, "Amazon" and "Arnazon" may look visually similar but are spoofs of each other since they are semantically unequal. Traditional text encoders such as BERT focus on semantic meaning and operate in the token space, making them insensitive to complex glyph variations or homoglyph attacks. As a result, methods based entirely on string similarity metrics or language-model embeddings struggle to capture visual similarities, even when the model's semantic embeddings are robust.

On the other hand, image encoders, especially large pre-trained ones like ViT [15], DINOv2 [17], or SigLIP's vision tower [14] capture glyph-level differences because they operate directly on pixels. These encoders learn rich visual features that are robust to noise, distortions, and font variations, enabling them to differentiate between visually similar characters and structures.

However, image-based models don't use the semantic or symbolic structure of the text, and comparing two strings using an image-based model would require rendering the strings as images first. This process is computationally expensive and not scalable for large-scale retrieval processes.

To combine the strengths of both text-based and image-based models, our system learns a multimodal alignment between text and glyph images. The goal is to map textual strings into a visually grounded embedding space so that the embedding of a word's text representation lies close to the embedding of its rendered glyph image, spoofs lie close to their legitimate counterparts, and visually distinct strings (non-spoofs) lie further apart.

This method is motivated by contrastive alignment methods in vision-language models such as CLIP [7] and SigLIP [14] which learn joint embedding spaces for images and text. However, instead of aligning images with captions, we align glyph images with their corresponding text strings. This enables an entity name or string to be interpreted in a visual space without using computationally expensive OCR systems or image-rendering processes.

3.1.2 Proposed System

The proposed system consists of an image encoder which creates the visual embedding space and a text encoder that produces semantic representations of strings and defines our text embedding space. Then an MLP projection head maps text embeddings into the visual space so that text and image representations become directly comparable. Afterwards, we can compute the similarity between any two given strings by mapping them into the joint embedding space to determine whether the two strings are likely to be a homoglyph attack by calculating the distance between the two embeddings.

Image Embeddings

Each business name is converted into a glyph image by rendering the string using a standardized font and image size. The image is then passed through a large pre-trained vision encoder to generate pre-computed image embeddings. We evaluate multiple image backbones, including ResNet [18], SigLIP [14], and ViT [15], but found the SigLIP image encoder to provide the best baseline performance. We also extract embeddings from early, medium, and final layers for each image backbone to capture both low-level and mid-level glyph structure and analyze its performance.

Text Embeddings

We evaluate several candidate text encoders including general transformers, visually grounded text encoders such as VATE, and text towers of vision-language models like SigLIP [14]. Each encoder produces an embedding for the entity's name as a string which are not naturally aligned with the visual structure of its corresponding glyph image. Therefore, we apply a learned projection head to map text embeddings into the visual embedding space.

Multimodal Alignment MLP

The multimodal alignment module is a lightweight multilayer perceptron (MLP) that takes a text embedding as the input and outputs a vector in the same embedding space as the image encoder. The MLP is trained using contrastive loss that brings each projected text embedding closer to its corresponding glyph-image embedding. This follows CLIP's [7] alignment strategy but translated for glyph images and data.

Embedding Similarity Score

Once both text and images share a joint embedding space, we compute a spoof similarity score using cosine similarity. At inference time, an input string is embedded by the chosen

text encoder, projected into the aligned visual space, and then compared using cosine similarity against pre-computed embeddings of legitimate entity names. The spoof similarity is calculated as

$$\text{spoof similarity}(x, y) = \cos\big(t(x),\, v(y)\big),$$

where $t(x)$ is the aligned text embedding and $v(y)$ is a pre-computed embedding of a legitimate entity name. Lower cosine similarity indicates greater visual similarities and therefore a higher likelihood of the inputted string to be a homoglyph attack.

Because homoglyphs are visually similar but symbolically different (token-wise), the visually aligned embedding space enables more accurate detection than using text-only similarity metrics or methods and greater efficiency than image-only models. Therefore, this multimodal system more efficiently and accurately detects visually similar entity name spoofs while leveraging the strengths of both text and image encoders.

3.2 Image Embedding Extraction

To produce an image embedding for each entity-name, each entity name is rendered as a glyph image with a fixed image size and font. This image is passed through a pre-trained vision encoder and the outputted vector is used as the image embedding.

3.2.1 Choice of Image Encoder

We want an image encoder that can handle the nuances of glyph structure while remaining robust to noise or font variation. Image backbones such as ResNet [18], EfficientNet [19], Vision Transformers (ViT) [15], masked autoencoders (MAE) [16], and DINOv2 [17] learn generalizable image features such as edges, textures, shapes, and object-like structures that tend to appear in different image domains and can therefore be extended to glyph images.

In addition, vision-language models such as CLIP [7] and SigLIP [14] train image towers with contrastive alignment against text encoders, producing representations based on visual features in an embedding space designed for similarity search. SigLIP in particular replaces the softmax-based contrastive loss of CLIP with a sigmoid loss over pairwise scores, allowing for more stable optimization with large batch sizes [14].

We evaluated a set of candidate image backbones to serve as image encoders including ResNet-style convolutional networks [18], Vision Transformers (ViTs) [15], self-supervised models such as ImageGPT [22], masked-transformer models like ViT-MAE [16], convolutional architectures such as ConvNeXt V2 [21], and vision–language image towers such as SigLIP [14]. For each backbone, we froze the encoder and measured how well cosine distances between glyph embeddings separated spoof vs. non-spoof pairs as a baseline. In these evaluations, the SigLIP vision tower provided the best baseline performance for homoglyph detection.

3.2.2 Rendering Glyph Images

To generate our pre-computed image embeddings, we converted each string into a glyph image and standardized the rendering process. We standardize the rendering process to ensure that the input images for the image encoder are consistent across names, fonts, and rendering

conditions, allowing the model to focus on meaningful glyph differences rather than artifacts introduced by the rendering pipeline.

We render each string onto a fixed-sized canvas of 224×224 pixels. The canvas is initialized with a black background and the string is rendered in white text. These high-contrast, white-on-black image were chosen because many vision models use convolutional filters that are sensitive to edges and light/dark transitions. Every glyph rendered uses the same font, DejaVu Sans, which was chosen to simulate typical web fonts used in phishing and spoofing contexts. The text is normalized with Unicode NFC so that characters with combined accents are rendered consistently. Most entity names fit within the 224×224 canvas, but if the text is too long, we scale the font size down to keep the glyphs readable while staying on a single line.

3.2.3 Image Encoder Fine-Tuning

After extracting glyph-image embeddings, we further refine the visual representation by training a small MLP head on image-to-image pairs (spoof vs. legitimate). This head is trained with contrastive loss and curriculum learning to create an improved image embedding space for text-to-image embedding alignment.

3.3 Text Embedding Extraction

To understand how different text encoders capture character-level structure and how well their representations perform in spoof detection, we evaluated three types of text models: GPT-style transformer encoders, visually aligned text encoders (VATE), and the text tower from the SigLIP vision-language model. Each encoder produces a fixed-dimensional representation of an inputted string.

3.3.1 Choice of Text Encoder

GPT Transformer Text Embeddings

The Generative Pre-trained Transformer (GPT) transformer focuses on effectively learning text representations and predicting the most likely next word in a sequence. GPT can generate high-quality text which makes it suitable for a wide range of natural language understanding tasks such as analyzing semantic similarity or document classification. The GPT model is not visually grounded and was not trained for glyph similarity detection. However, because it operates at the sub-word level and generates strong contextual embeddings, the model provided a good baseline for how a purely semantic transformer behaves when used for homoglyph detection tasks.

We tokenize business names using the model's built-in Byte-Pair Encoding (BPE) tokenizer and extend it by adding an explicit padding token. After tokenization, we feed the padded and truncated sequences into the GPT model to obtain hidden states for each position in the input sequence. We create a single embedding for the entire entity name by averaging the hidden states of all non-padding tokens and applied L2 normalization so that all embeddings

have unit length. Because GPT is optimized for language-based semantics instead of visual structure, its embeddings did not naturally cluster homoglyph variants.

Visually-Aligned Text Models (VATE)

Visually aligned text encoders like VATE incorporate glyph or visual features during training in order to align character embeddings with their rendered glyphs through a contrastive pre-training objective. This makes VATE more sensitive to minor changes that standard text models might not weigh as heavily when generating embeddings such as Unicode look-alikes or spoofs, characters in different scripts that look similar, or added accents.

Unlike GPT, VATE-like models operate at the character level and incorporate character-level visual embeddings into token representations. As a result, VATE embeddings maintain structural information about glyphs and their visual properties. Because VATE already encodes a visually grounded representation space, we expect it to perform well in baseline evaluations and more easily map into an image embedding space.

SigLIP Text Tower

SigLIP's text tower [14] is a large-scale vision-language model trained with a symmetric pairwise sigmoid loss instead of a softmax-based contrastive loss. The SigLIP text encoder is co-trained with its image tower to ensure that both modalities are projected into a shared embedding space optimized for multimodal similarity comparisons.

This training process means that the SigLIP text tower produces embeddings that are already partially aligned with image features. This can be useful for homoglyph detection because the text embeddings are shaped to be aligned with image features, which could make it easier to generate an effective mapping from text embeddings to glyph-based visual embeddings. SigLIP's text tower embeddings also have the same dimensions and normalization as the SigLIP vision tower embeddings.

Evaluation Methods

We evaluate each text encoder by pre-computing embeddings and used a pairwise classification pipeline. For each labeled pair, we retrieve the two embeddings and retrieve their pre-computed embeddings. Then we compute their cosine similarity and measure how reliably this score distinguishes real matches from mismatches. We summarize performance using ROC–AUC and select operating thresholds that balance true positives and false positives and noting the accuracy, recall, and precision. Comparing GPT, VATE, and the SigLIP text tower with this pipeline allows us to assess which text encoder naturally places homoglyph variants closer to their legitimate names and which requires more alignment work by the MLP. Encoders that have higher ROC-AUC suggest they could be stronger textual backbones for multimodal alignment.

3.4 Text-to-Image Embedding Alignment Model

Given pre-computed embeddings for entity names (glyph-based image embeddings and text embeddings), our goal is to learn projections that bring spoofed names close to their legitimate counterparts and push unrelated names apart. To do this, we first train a small MLP on top of SigLIP glyph embeddings to better structure the visual embedding space for spoof detection. Then we train a separate MLP that maps text embeddings into this refined image embedding space. Both MLPs are trained with the same margin-based contrastive loss [44] and use curriculum learning to vary spoof difficulty.

3.4.1 Image Projection Head

In the first stage, we refine the glyph-image embedding space itself. Starting from SigLIP vision embeddings for rendered entity name glyphs, we train a two-layer MLP head on pairs of glyph images corresponding to (spoof, real) names. Positive pairs within the dataset correspond to spoofed vs. legitimate renderings of the same name and are pulled closer together in the projected space. Negative pairs correspond to glyphs of unrelated entity names and are pushed apart up to a margin. We train this head using curriculum learning over easy, medium, and hard spoof pairs.

3.4.2 Text-to-Image Mapping

In the second stage, we align text embeddings to the refined image space. We use pre-computed text embeddings for all entity names and pre-computed glyph-image embeddings using the refined image projection head. Using the pre-computed embeddings for each dataset pair, we train a text-to-image MLP that takes the text embedding as an input and predicts a vector in the image-embedding space. The predicted text vector and the corresponding image embedding are both L2-normalized and compared with the same contrastive loss. This moves the projected text embedding for a given name close to the glyph embedding of its legitimate or spoofed rendering.

3.4.3 MLP Projection Head

We use small multi-layer perceptrons as the alignment (or "head") network in both stages. In the text-to-image stage, let $x \in R^{\text{input}}$ be an input text embedding. The MLP head implements a mapping

$$f_\theta : R^{\text{input}} \to R^{\text{out}},$$

using a fully connected layer that projects the input to a hidden dimension H followed by a ReLU activation. Then a second fully connected layer maps the hidden representation into the output embedding space of the image encoder, creating a projected text embedding aligned to the visual embedding space.

After the forward pass, the projected vectors are L2-normalized. This normalization makes training more stable and prevents the model from using minor changes in vector scale to satisfy the loss. Instead, it forces the model to encode similarity in the direction of the

vectors. In our experiments, we set $H = 512$ and $\text{ out} = 256$, but these dimensions can be tuned based on the model capacity.

In the image projection head, we train a similar two-layer MLP head with image embeddings as the input instead.

3.4.4 Contrastive Training Objective

We train both MLP heads using a margin-based contrastive loss [44]. Each training example consists of a pair of embeddings (x_1, x_2) and a binary label $y \in \{0, 1\}$, where $y = 1$ represents a positive pair (legitimate and homoglyph name) and $y = 0$ a negative pair (spoof and unrelated name). In the image projection head, x_1 and x_2 are both image embeddings. In the text-to-image embedding alignment, x_1 is a text embedding and x_2 is the corresponding image embedding.

Both x_1 and x_2 are passed through their corresponding heads to obtain projected embeddings $z_1 = f_\theta(x_1)$ and $z_2 = f_\theta(x_2)$, which are both L2-normalized. We compute the Euclidean distance $d = \|z_1 - z_2\|_2$ and optimize based on the contrastive loss:

$$\mathcal{L}_{\text{contrastive}} = y \cdot d^2 + (1 - y) \cdot \max(0, m - d)^2,$$

where $m > 0$ is a margin hyperparameter. Positive pairs are pulled closer together in the joint embedding space ($d \approx 0$), while negative pairs are penalized if they fall within the margin ($d < m$). This trains the model to cluster embeddings of visually or semantically related names while pushing apart unrelated ones.

During validation, we convert distances into similarity scores (higher scores correspond to more likely spoof attempts), compute ROC-AUC, and choose a threshold using the Youden index on the ROC curve.

3.4.5 Curriculum Learning over Spoof Difficulty

We use curriculum learning to stabilize training and incorporate a range of difficulty levels in the spoof pairs. The training data is split into three subsets of easy, medium, and hard pairs with varying levels of spoof complexity and ambiguity. During the first third of training, we only use the easy subset. In the second third, we train on both easy and medium pairs, and in the final third, we train on the full dataset of easy, medium, and hard pairs.

We apply curriculum learning in training both MLPs. In each case, the curriculum allows the model to first learn a decision boundary on simpler examples and then gradually refine it on increasingly challenging spoof pairs.

3.4.6 Optimization, Early Stopping, and Hyperparameter Search

We trained both MLPs using the Adam optimizer with weight decay, an initial learning rate of 10^{-3}, and batch size of 512. At the end of each epoch, we evaluate the model on a held-out validation set.

Learning Rate Scheduling

In both MLP training processes, we use a ReduceLROnPlateau scheduler that halves the learning rate whenever the validation ROC-AUC fails to improve for a set number of epochs. This encourages fast progress early on and prevents overfitting in training.

Early Stopping

We implemented early stopping using validation ROC-AUC as the stopping criterion. If ROC-AUC does not improve for a fixed number of epochs, training stops and we revert to the best-performing set of parameters/model seen so far.

Hyperparameter Tuning

We perform an Optuna study, performing a hyperparameter sweep over the margin in contrastive loss, learning rate, weight decay, hidden dimension, output dimension, and batch size.

Together, these optimization procedures produce a robust joint embedding space in which spoofed and legitimate entity names lie close together when they are visually or homoglyphically related and remain separated from unrelated names. This aligned join embedding space allows for accurate and efficient spoof detection and retrieval.

Chapter 4

Experiments

To evaluate the behavior of different model components, we describe the dataset, sampling strategies, and evaluation metrics used. This experimental set-up establishes a consistent environment for comparing text-only, image-only, and multimodal models.

4.1 Experiment Goals

The goal of our experimental evaluation is to determine whether a text-to-image alignment-based embedding model can reliably capture visual similarity between two strings, especially when characters are swapped with close lookalikes or other complex homoglyph attack strategies. In particular, through our experiments, we want to measure how well our learned embedding space distinguishes between spoofed names and non-spoof pairs that may be textually similar but not visually similar.

First, to determine if alignment improves distinction between visually similar and dissimilar strings, we test whether fine-tuning a lightweight projection head on top of pretrained embeddings (either glyph-image embeddings or text embeddings) produces a space in which homoglyph variants cluster closer together with their legitimate counterparts while unrelated names remain well-separated.

Next, we want to analyze whether the model is generalizable across spoof difficulty. Spoof pairs vary in how challenging they are to detect since some might just differ by a single character such as an added accent or punctuation mark, while others involve more subtle multi-script changes. By training the model across easy, medium, and hard pairs, we assess whether the model is able to capture progressively more and more fine-grained visual changes.

Finally, we want to assess whether the aligned text-to-image embedding space allows for performant spoof detection. We evaluate our model's performance by measuring ROC-AUC, threshold-based accuracy, and confusion statistics.

Together, these goals and questions allow us to evaluate not just raw performance but the behavior and usefulness of the learned embedding space for real-world homoglyph attack detection.

4.2 Dataset

We evaluate our models on the domain name dataset introduced by Woodbridge et al [1], which contains large-scale curated pairs of legitimate domain names, visually spoofed variants, and unrelated negatives. Each pair is annotated with a binary label indicating whether the two strings are homoglyph attacks or not. The dataset spans a wide range of string transformations representative of spoof techniques such as:

- Using cross-script characters (Ex: mixing Latin and Greek characters in `Meta` → `Meta`, where characters can come from other alphabets)
- Accent insertion or removal (Ex: `Cafe` → `Café`).
- ASCII look-alikes (Ex: swapping "O" and "0" in `Zoom` → `Z00m`)
- Visually similar multi-token name variants (Ex: inserting subtle spacing or punctuation changes such as `Pay Pal` or `Pay-Pal` to spoof `PayPal`)

The data is also partitioned into three difficulty categories. Easy pairs are obvious spoofs where substitutions are visually easy to spot with the human eye. Medium pairs are more subtle name variants or multiple small modifications, and hard pairs are cases where spoofed and legitimate names are challenging to differentiate because they may, for example, differ by only a small number of cross-script changes.

Each subset contains hundreds of thousands of labeled examples. The training set is used twice - first for learning the image-space MLP head on image embeddings and again for training the text-to-image alignment model using the same easy/medium/hard curriculum learning structure.

For our glyph image dataset, each domain name in the Woodbridge dataset is rendered into a glyph image using a standardized font and image size. The resulting text and image datasets therefore contain text and image representations for the same string pairs, allowing us to measure and evaluate cross-modal alignment performance.

4.3 Evaluation Metrics

To assess the quality of the learned embedding spaces and its ability to detect homoglyph attacks, we evaluate all models using metrics that capture both how well positive pairs score above negative ones and thresholded classification behavior (how well a decision boundary separates the two classes).

ROC–AUC

The Receiver Operating Characteristic Area Under the Curve (ROC-AUC) was our primary evaluation metric. ROC-AUC is the probability that a randomly chosen positive pair receives a higher similarity score than a randomly chosen negative pair. Therefore, we used the ROC-AUC to tell us how well the model ranks positive pairs (legitimate name vs. spoofed variant) above negative pairs (legitimate vs. unrelated). A value of 0.5 corresponds to random

guessing, while a value of 1.0 would indicate perfect classification. Because it does not rely on any particular threshold, ROC-AUC serves as a stable and comparable measurement for spoof detection performance.

Similarity Scores

For each pair, the model produces a similarity score calculated as the negative Euclidean distance between the projected embeddings:

$$s = -\|z_1 - z_2\|_2.$$

Larger values of s mean that the model considers the two strings visually or semantically similar. These scores are used in constructing the ROC curve and for threshold-based classification.

Youden-Optimal Threshold

Although ROC-AUC is threshold-independent, real-world applications of spoof detection models would require a concrete threshold to build a decision rule around. We determine the classification threshold using the Youden index,

$$J = \text{TPR} - \text{FPR},$$

evaluated over all points on the ROC curve. The threshold that maximizes J produces the best trade-off between true positive rate (TPR) and false positive rate (FPR). This means it picks the point on the ROC curve where the model detects as many spoofed names as possible (high TPR) while keeping the number of benign pairs incorrectly flagged as spoofs as low as possible (low FPR). We optimize for this because in practice both failing to catch a spoof and over-flagging legitimate names carry financial costs or repercussions. We apply this threshold to convert similarity scores into binary predictions (spoof or non-spoof).

Accuracy

Accuracy measures the fraction of correctly classified pairs after applying the threshold calculated using the Youden index. While accuracy is sensitive to class imbalance, it provides a simple measurement of how often the model's predictions agree with the ground truth labels.

We calculate accuracy as:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

Confusion Matrix

We also use the full confusion matrix to evaluate the model's performance. The metrics include number of true positive, false positive, true negative, and false negative pairs. True positives are spoofs correctly detected, false positives are benign pairs incorrectly flagged as spoofs, true negatives are benign pairs correctly left unflagged, and false negatives are spoofs missed by the model. These metrics allow us to gain better insight into how the model performs beyond aggregate metrics like accuracy.

Chapter 5

Results

The following results demonstrate how the proposed and existing models perform on spoof detection tasks. Our results allow us to evaluate the relative strengths and limitations of text and image encoders, the effect of fine-tuning on the image embedding space, and the extent to which multimodal alignment improves robustness against homoglyph attacks.

5.1 Baseline Results

Before training our alignment model, we conducted a wide range of baseline experiments to measure how well different modalities perform on homoglyph detection. These experiments allowed us to measure what information was already present in raw text or image similarity models and how much additional performance can be gained by multimodal alignment.

5.1.1 Text-Based Baselines

We first evaluated traditional string-similarity metrics that operate completely in the textual space. These methods don't utilize glyph images, and instead, they compare strings directly. Before computing any similarities, all names are normalized using Unicode NFKC normalization, casefolding, whitespace collapsing, and trimming to best ensure that the string-similarity metrics are measuring differences that don't come from formatting.

We evaluated the following text-based baselines:

Levenshtein distance

The Levenshtein edit distance between two strings a and b is defined as the minimum number of insertions, deletions, and substitutions required to transform one into the other. It is computed as:

$$d_{i,j} = \min \begin{cases} d_{i-1,j} + 1, & \text{(deletion)} \\ d_{i,j-1} + 1, & \text{(insertion)} \\ d_{i-1,j-1} + 1[a_i \neq b_j], & \text{(substitution)} \end{cases}$$

Because our evaluation requires a similarity score in $[0, 1]$, we convert the raw distance into a normalized similarity:

$$\text{Levenshtein Similarity}(a, b) = 1 - \frac{d(a, b)}{\max(|a|, |b|)}.$$

We expected Levenshtein similarity to perform well for detecting spoof pairs that differ by a single character or accent mark.

Token Sort Ratio (TSR)

Token Sort Ratio applies Levenshtein similarity to the alphabetically sorted token lists of each string. After Unicode normalization and casefolding, each name is tokenized into alphanumeric words, sorted, and concatenated:

$$\text{TSR}(a, b) = \text{Levenshtein Similarity}(\,\text{sort}(\text{tokens}(a)),\ \text{sort}(\text{tokens}(b))\,).$$

Sorting tokens makes TSR insensitive to token order (e.g., "Bank of America" vs. "America Bank of"), which we expect to provide more robustness for multiword entity names.

Jaro similarity

The Jaro similarity [4] between strings a and b is defined as:

$$\text{Jaro}(a, b) = \frac{1}{3}\left(\frac{m}{|a|} + \frac{m}{|b|} + \frac{m - t}{m}\right),$$

where:

- m is the number of matching characters (within the window of size $\lfloor \max(|a|, |b|)/2 \rfloor - 1$)
- t is half the number of transpositions within the matched characters

Jaro similarity is especially sensitive to swapped or nearby characters, performing well on shorter strings where homoglyph spoofs are often generated using close rearrangements.

For each method, we compute a similarity score for every pair in the evaluation set. We then compute an ROC curve and select the best classification threshold using the Youden index (TPR − FPR). The selected threshold is used to produce binary predictions, from which we compute accuracy and the full confusion matrix.

Despite the lack of visual input in these metrics (textual-based only), these baselines achieve strong performance with Levenshtein distance achieving an ROC-AUC of 0.931. This reflects the fact that many spoof pairs differ only by single-character substitutions which edit-distance and token-level metrics are able to capture well.

However, these methods' performance decrease for visually subtle homoglyphs such as cross-script substitutions, illustrating the need for visually grounded approaches.

Table 5.1: Text-only baseline performance on homoglphy detection

baseline	roc_auc	threshold	tpr_at_best	fpr_at_best
Levenshtein	**0.931**	0.6	0.868	0.088
Jaro	**0.907**	0.775	0.841	0.132
TSR	**0.904**	0.6	0.834	0.085

Table 5.2: Confusion Matrix Metrics of Text-Only Baselines

Baseline	**Accuracy**	**Precision**	**Recall**	**TP**	**FP**	**TN**	**FN**
Levenshtein	0.884	0.948	0.868	3379	185	1923	512
Jaro	0.851	0.921	0.841	3274	279	1829	617
Token Sort Ratio	0.862	0.948	0.834	3245	179	1929	646

5.1.2 Image-Based Baselines

Next, we evaluate methods that operate entirely on the rendered glyph images of entity names. For each pair, we load the corresponding grayscale glyphs from disk, resize both images to a common resolution (the maximum height and width across the pair), and then compute a similarity score. As in the text-based baselines, we then compute the ROC-AUC, select a decision threshold via the Youden index, and report accuracy and the full confusion matrix.

We evaluated the following image-based baselines:

Pixel MSE

The first baseline calculates the normalized pixel-wise mean squared error (MSE) as a similarity score in $[0, 1]$. Given two aligned glyph images A and B with pixel intensities in $[0, 255]$, we compute

$$\mathrm{NMSE}(A, B) \;=\; \frac{1}{255^2} \cdot \frac{1}{HW} \sum_{i,j} (A_{ij} - B_{ij})^2,$$

where H = image height and W = image width.

We then define the Pixel MSE similarity

$$sim_{\text{Pixel MSE}}(A, B) \;=\; \max\big(0,\, \min(1,\, 1 - \mathrm{NMSE}(A, B))\big).$$

This baseline is expected to be able to capture rough structural similarity between two rendered images but is highly sensitive to small image shifts, font changes, and interpolation artifacts when resizing.

Oriented FAST and Rotated BRIEF (ORB) Keypoint Matching

ORB keypoint matching is a feature-based image similarity metric. For each glyph image, we detect ORB keypoints and binary descriptors. We then match descriptors between the two

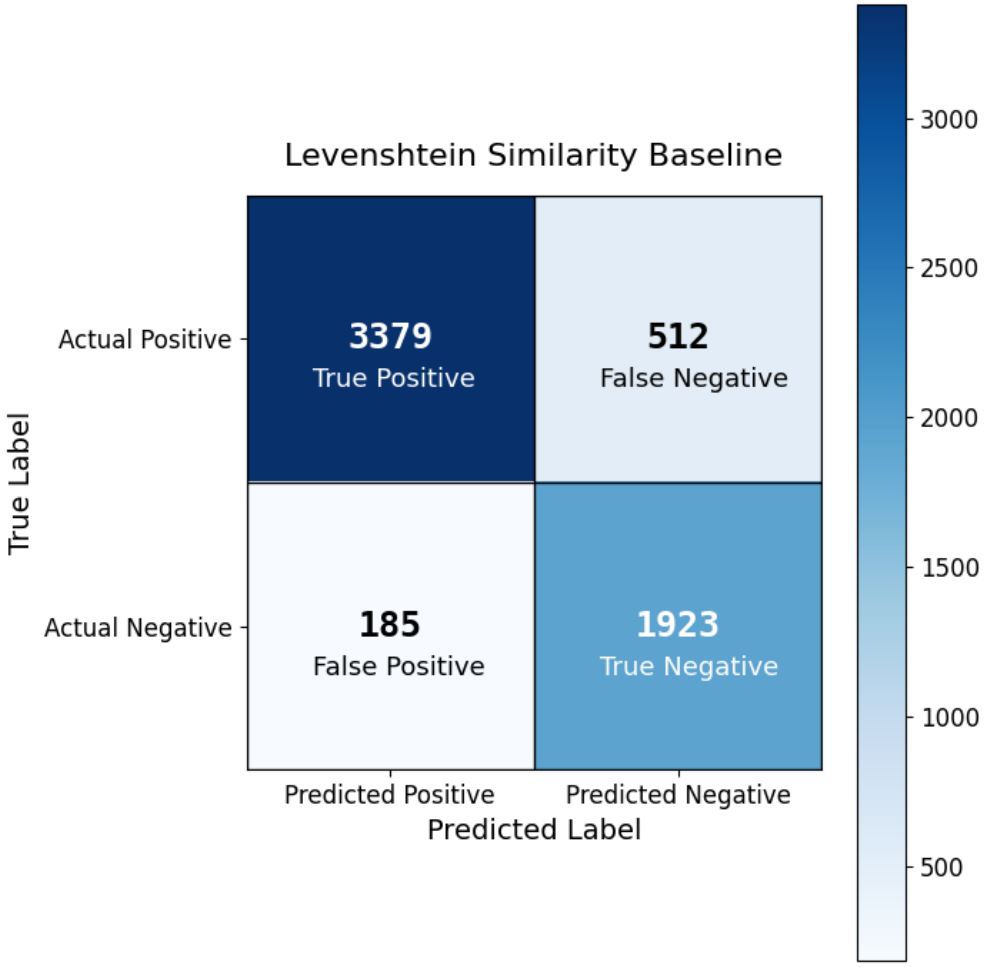

Figure 5.1: Confusion matrix for the Levenshtein Distance baseline

images using a Hamming matcher with cross-checking. ORB similarity is calculated as:

$$sim_{\text{ORB}}(A, B) \;=\; \frac{\text{matches}}{\max\big(\min(\text{descA}, \text{descB}),\, 1\big)},$$

where matches is the number of mutual best matches and descA and descB are the number of ORB descriptors for each image.

Because ORB focuses on local corners and edges, this baseline is expected to be more robust than pixel MSE when analyzing minor shifts or scaling between two images and can better capture structural similarities in the glyph shapes.

As shown in Table 5.3, image-only baselines achieved moderate performance: Pixel MSE achieved ROC-AUC of approximately 0.63 and ORB keypoint matching achieved ROC-AUC of around 0.82. They correctly flagged pairs whose glyphs were nearly identical but struggled when spoofing attacks used cross-script substitutions or multi-character differences that preserved the rough structure of the strings. This motivates moving beyond image-only heuristics towards a multimodal representation.

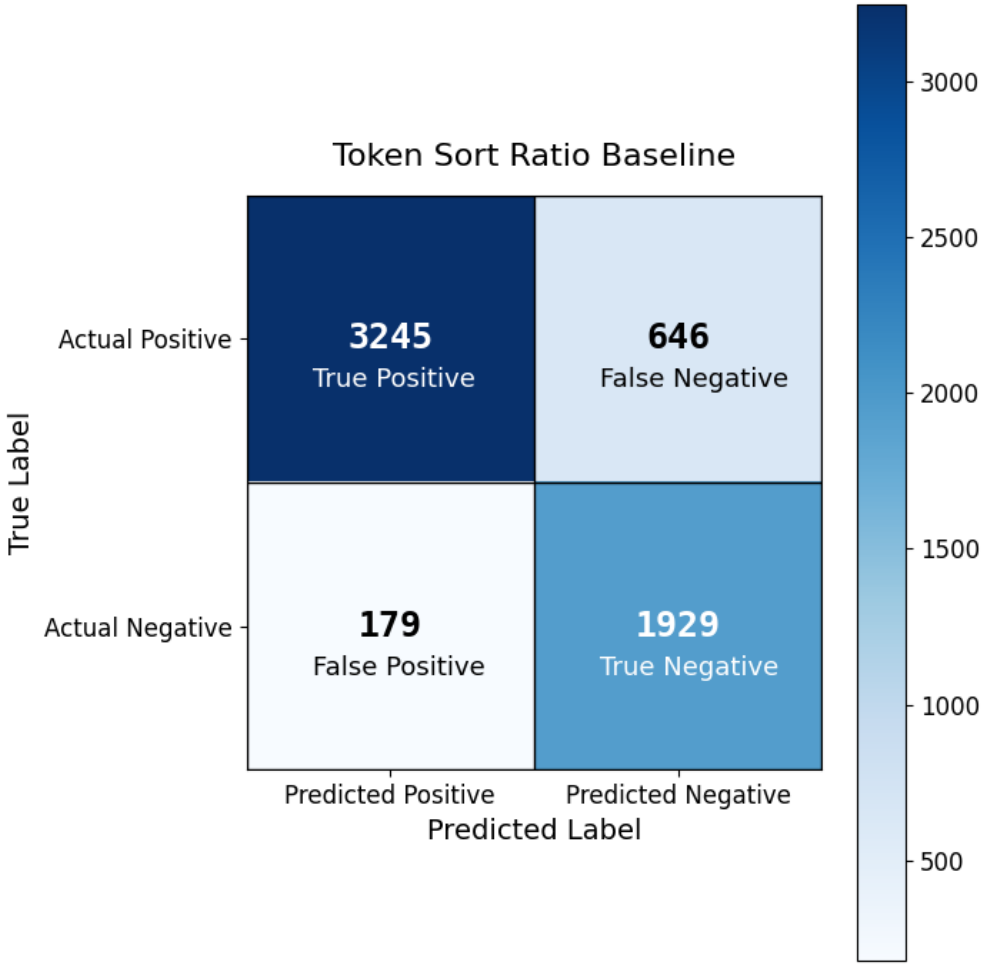

Figure 5.2: Confusion matrix for the Token Set Ratio baseline

5.1.3 Fusion Baselines

To test whether textual and visual similarity metrics are complementary and could provide gains in performance if paired together, we evaluate three early-fusion baselines built on top of the text-only and image-only baseline methods. In all baselines, we treat the similarity metrics as fixed features and train a shallow classifier on top.

Text-Only Fusion

For each pair, we compute three scalar text similarity features: Levenshtein similarity, Token Sort Ratio (TSR), and Jaro similarity (all normalized to $[0, 1]$) and store them in a vector:

$$\mathbf{x}_{\text{text-fusion}}(a, b) = \big[\text{Lev}(a, b),\ \text{TSR}(a, b),\ \text{Jaro}(a, b)\big]$$

We split the labeled pairs into a training set (4,000 pairs) and an evaluation set (5,999 pairs), fit a logistic regression classifier on $\mathbf{x}_{\text{text-fusion}}$, and then compute a spoof probability on the evaluation set.

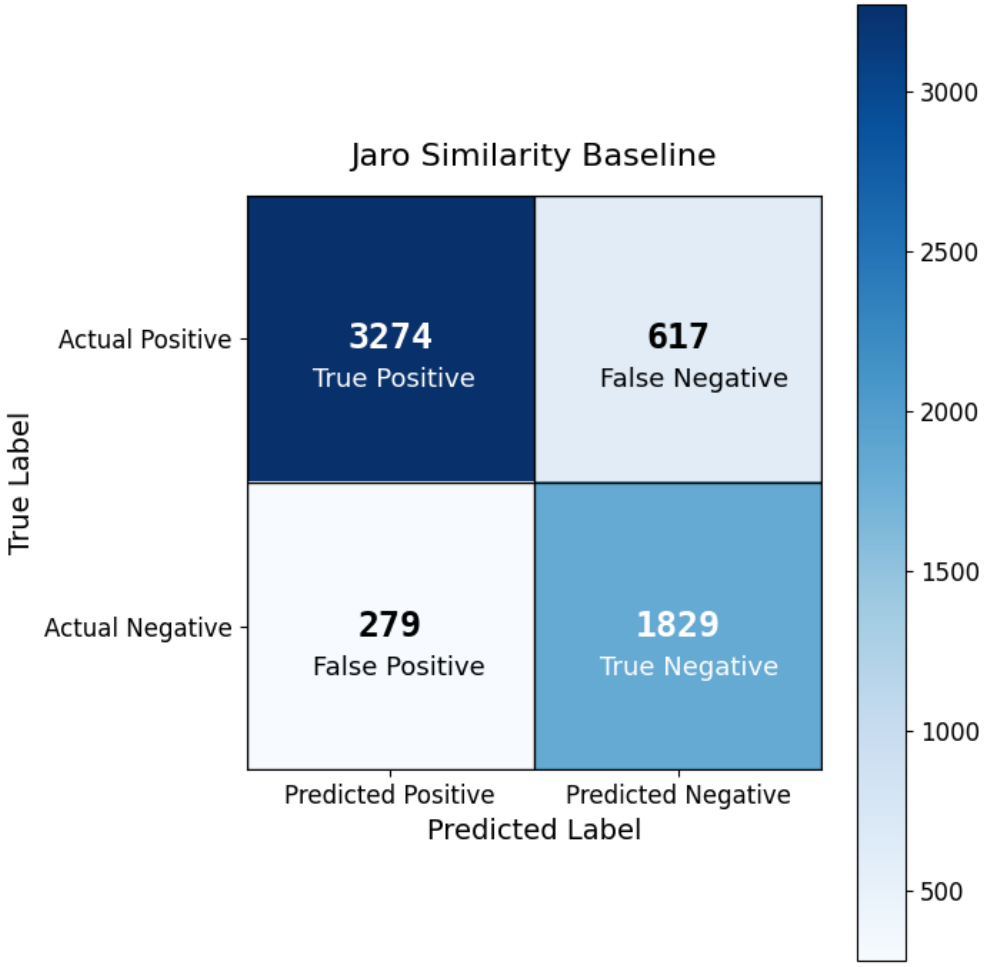

Figure 5.3: Confusion matrix for the Jaro Similarity baseline

Image-Only Fusion

For each pair of strings and their respective glyph images (a, b) we compute two visual similarity features: pixel MSE similarity $s_{\text{MSE}}(a, b)$ and ORB keypoint similarity $s_{\text{ORB}}(a, b)$, both normalized to $[0, 1]$ and store them in a vector:

$$\mathbf{x}_{\text{image-fusion}}(a, b) = \left[sim_{\text{MSE}}(a, b),\ sim_{\text{ORB}}(a, b)\right]$$

We then train a logistic regression model on $\mathbf{x}_{\text{image-fusion}}$ using the same train-eval split as the text-only fusion. At evaluation time, we again threshold the predicted spoof probability using the Youden index.

Hybrid Fusion (All Methods)

Finally, we combined all baseline methods into a single feature vector and incorporate an OCR-based similarity metric. For each pair, we compute:

$$\mathbf{x}_{\text{hybrid}}(a, b) = \left[\text{Lev}(a, b),\ \text{TSR}(a, b),\ \text{Jaro}(a, b),\ s_{\text{MSE}}(a, b),\ s_{\text{ORB}}(a, b),\ \text{Lev}\big(\text{OCR}(a), \text{OCR}(b)\big)\right],$$

Table 5.3: Image-only baseline performance on homoglyph detection

baseline	roc_auc	threshold	tpr_at_best	fpr_at_best	accuracy
Pixel MSE	0.634	0.425	0.587	0.359	0.614
ORB	0.820	0.45	0.763	0.214	0.793

where OCR is a transcription of the glyph image using the Tesseract OCR system which we then take the Levenshtein distance between. We train an XGBoost classifier on these six features using the same train-eval split as the text-only and image-only fusion baselines.

Table 5.4: Fusion baseline performance on the homoglyph detection task.

baseline	roc_auc	threshold	tpr_at_best	fpr_at_best	accuracy
Text Fusion (LR)	0.930	0.642	0.864	0.082	0.883
Image Fusion (LR)	0.822	0.619	0.700	0.196	0.736
Hybrid Fusion (XGB)	0.986	0.619	0.933	0.048	0.940

The hybrid fusion model significantly outperforms all earlier baselines: it achieves ROC-AUC of 0.986 and accuracy around 0.94. This indicates that text-based edit-distance features, image-based similarity scores, and OCR-derived features capture partially complementary aspects of homoglyph spoofing.

However, this hybrid-fusion baseline has a key limitation that it requires access to rendered glyph images and an OCR system at inference time, which adds latency and complexity to the spoof detection process. This further motivates our approach of learning a text-to-image alignment model that transforms visual similarity into a purely textual embedding space, so that homoglyph detection can be performed directly on entity names without rendering images or using an OCR system.

5.2 Image Tower Results

To evaluate how well vision encoders capture fine-grained glyph similarity and determine which image tower to use for the multimodal alignment model, we tested a set of image encoders.

For every image encoder, we precomputed glyph embeddings at three distinct layer depths: Early (low-level edges, strokes, contour features), Medium (mid-level structure), and Deep (high-level semantic abstractions). Homoglyph similarity was then evaluated using cosine similarity of the left vs. right glyph embeddings. We then reported the ROC-AUC, Youden Index threshold, and confusion-matrix metrics.

Based on our results in Table 5.5, early layers consistently outperformed deeper layers for most architectures. This could be due to the fact that early convolutional or attention layers preserve stroke and text structure which are useful for distinguishing between visually similar characters. In contrast, deep layers include more semantic features that are less relevant for homoglyph detection.

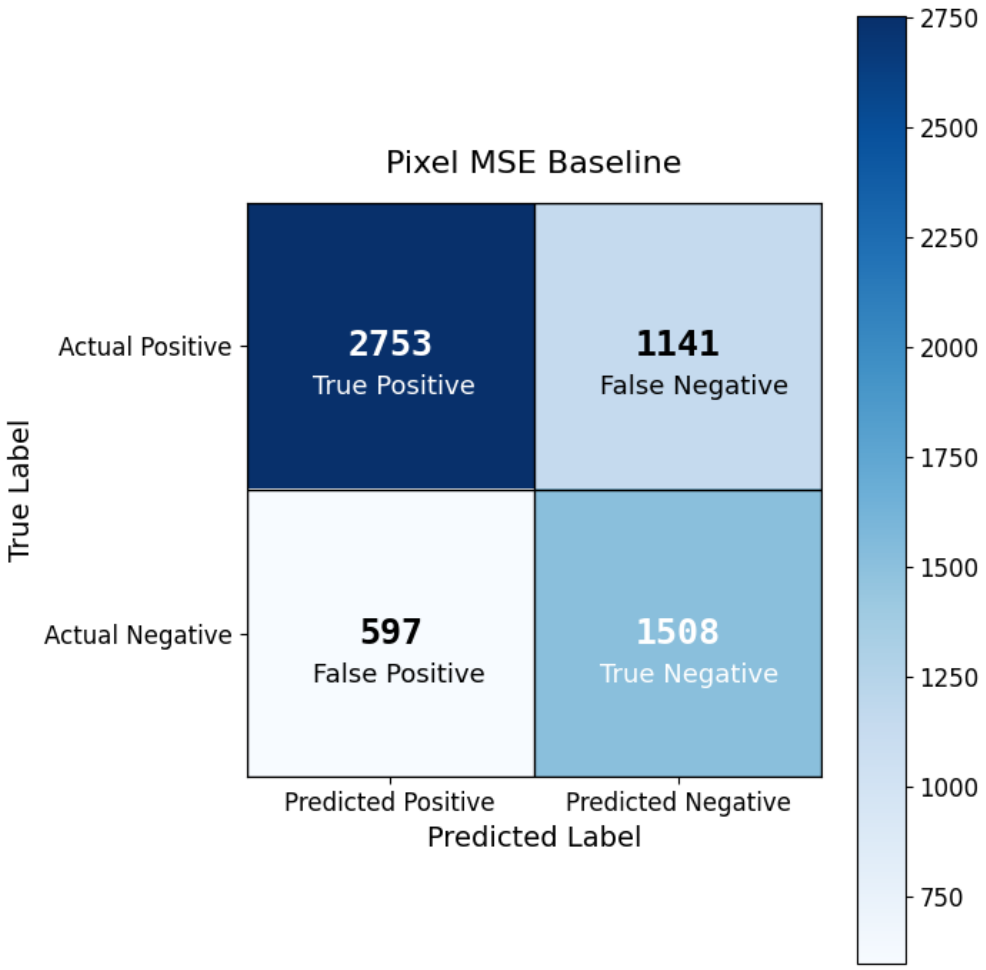

Figure 5.4: Confusion matrix for the Pixel MSE baseline

We also observed that self-supervised generative models such as ViTMAE and ImageGPT struggled at deep layers. ViTMAE is trained to reconstruct missing patches of an image and ImageGPT is trained to predict each pixel in a sequence. Their objectives both bias their representations toward smooth, consistent outputs instead of precise, local pixel patterns in deep layers.

SigLIP, for which results are shown in Figure 5.9, was the strongest encoder which we believe is due to SigLIP's contrastive vision-text training objective which preserves fine-grained visual structure. It achieves an ROC-AUC of around 0.938 and is therefore selected as the image backbone for the multimodal alignment model.

5.3 Fine-Tuning the SigLIP Image Encoder

While zero-shot SigLIP already outperforms all other image encoders on the homoglyph detection task, the backbone has not been trained and optimized for homoglyph detection tasks. Therefore, we fine-tune a lightweight two-layer MLP head on top of the SigLIP glyph

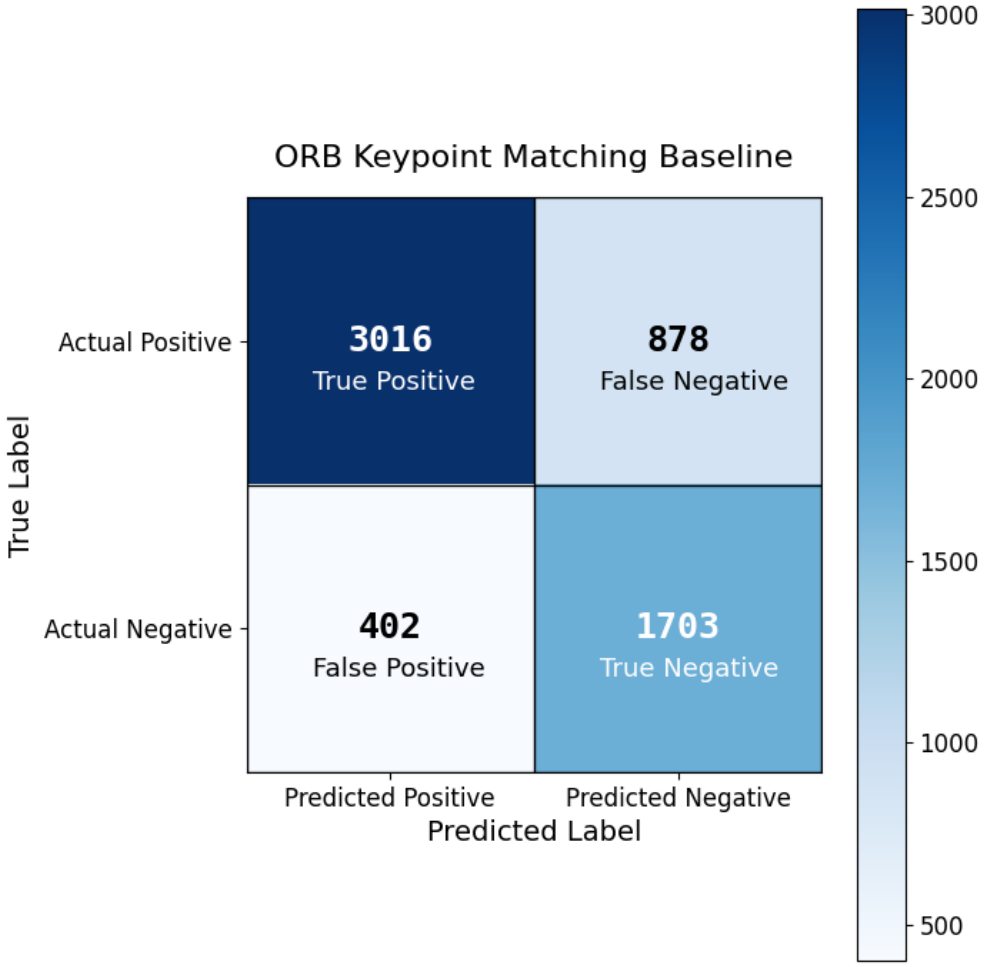

Figure 5.5: Confusion matrix for the ORB Keypoint Matching baseline

embeddings using a supervised contrastive objective.

Training converged after 27 epochs and the best validation model achieved the results in Table 5.6. These results represent an improvement over all zero-shot image encoders, reducing both false positives (benign names incorrectly flagged as spoofs) and false negatives (missed homoglyph attacks).

The hyperparameters selected by Optuna for the final full-length training run are also listed in Table 5.7. These parameters were chosen to maximize validation ROC-AUC under the contrastive learning objective.

We also visualized the training and validation accuracy and loss curves in Figures 5.11 and 5.12 to understand the optimization behavior of the fine-tuned SigLIP MLP head during the 27-epoch fine-tuning process.

Specifically, in Figure 5.11, training accuracy increases quickly during the early epochs showing how the MLP was able to learn spoof detection boundaries in the SigLIP embedding space. There is then a drop in accuracy at epoch 11 that corresponds to a curriculum transition where harder pairs are introduced into training. However, the training accuracy eventually stabilizes and gradually rises again, indicating that the model successfully adapts

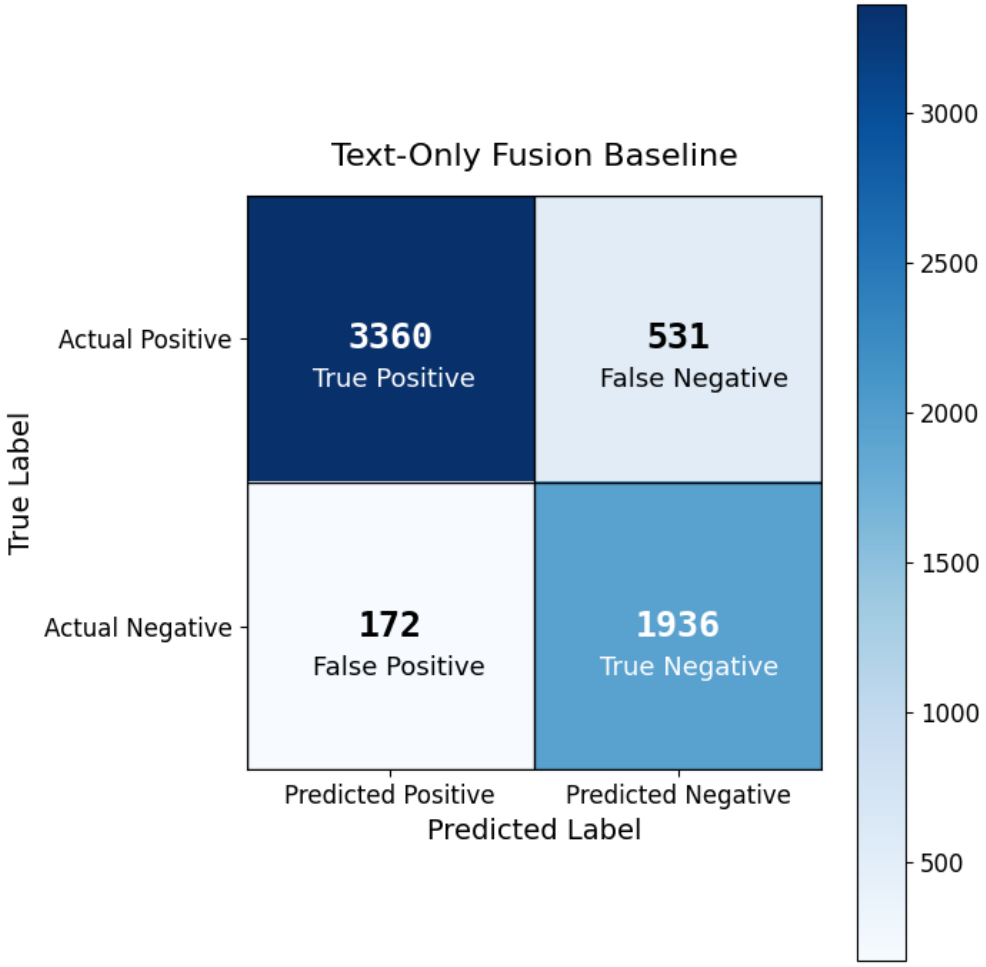

Figure 5.6: Confusion matrix for the Text-Only Fusion baseline

to the more challenging examples and refines its spoof detection boundaries. The validation accuracy consistently shows a steady increase during training.

In Figure 5.12, the training loss dropping sharply during the initial epochs but sharply increases at epoch 11 due to the curriculum transition, similar to the training accuracy curve. The validation loss decreases throughout training and follows the training loss curve.

5.4 Text Encoder Results

To evaluate how well purely textual representations perform at homoglyph detection tasks, we evaluate three text embedding models under the same pairwise cosine similarity framework used for the image encoders.

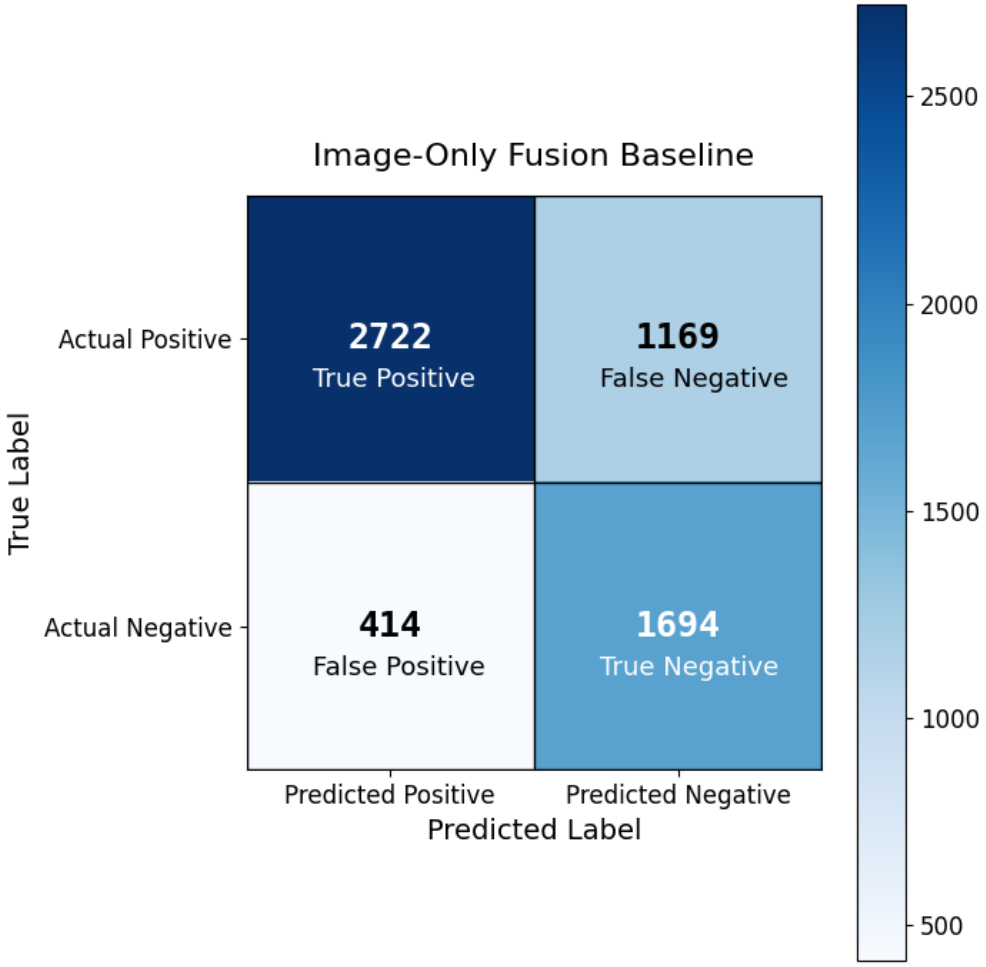

Figure 5.7: Confusion matrix for the Image-Only Fusion baseline

SigLIP Text Tower

The SigLIP text encoder performs poorly on homoglyph detection, achieving an ROC-AUC of only 0.471. This is expected since SigLIP's text tower is optimized for semantic language understanding and contrastive alignment with images not for predicting minor character changes within spoof detection.

GPT-based Encoder

A GPT-style embedding model performs better, achieving an ROC-AUC of 0.758, than the SigLIP text tower, suggesting that the GPT model, which is trained to predict the next token in a sequence, partially preserves character structure in its embeddings. However, GPT embeddings still underperform compared to image-based models, illustrating how detecting homoglyphic similarity is a visual task.

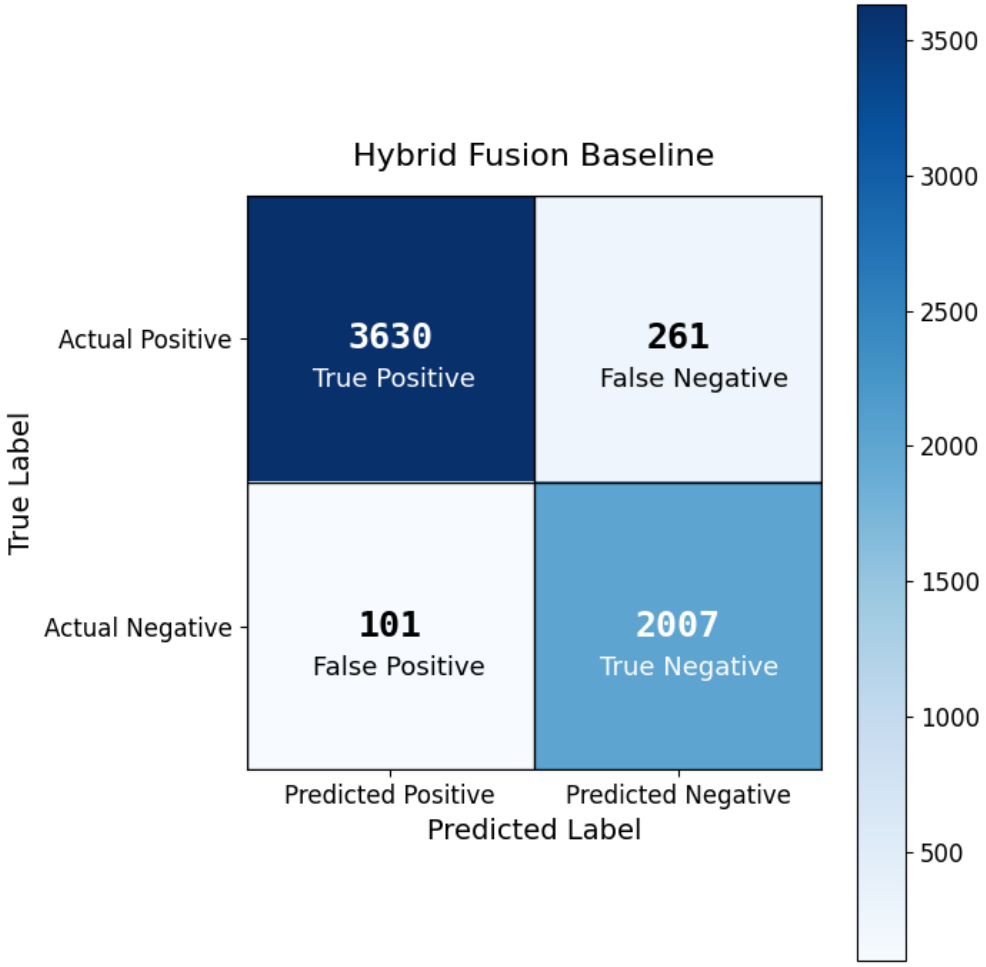

Figure 5.8: Confusion matrix for the Hybrid Fusion (All Methods) baseline

VA-TE (Visual–Aligned Text Encoder)

The VA-TE model, which fine-tunes a text encoder to align with SigLIP's glyph-image embedding space, outperforms both the SigLIP text tower and GPT model, achieving an ROC-AUC of 0.968 and accuracy of 0.903. By training directly on glyph-based similarity using contrastive loss training, VA-TE learns to reproduce the visual structure of homoglyph attacks using text embeddings.

Table 5.9 summarizes the results of each text encoder. Figure 5.13 provides the confusion matrix metrics for the VA-TE model.

5.5 Learning a Text–to–Image Alignment Model

The strong performance of VA-TE suggests that text embeddings can distill the visual similarity structure of homoglyphic attacks when a model is explicitly trained to do so.

Table 5.5: Image Encoder Baselines

Baseline	ROC-AUC	threshold	accuracy	precision	recall
ResNet Early Layer	0.782	0.999	0.699	0.807	0.699
ResNet Medium Layer	0.832	0.999	0.742	0.829	0.753
ResNet Deep Layer	0.821	0.987	0.717	0.876	0.652
ViT Early Layer	0.708	0.998	0.604	0.858	0.460
ViT Medium Layer	0.738	0.997	0.646	0.844	0.552
ViT Deep Layer	0.731	0.990	0.636	0.850	0.528
ImageGPT Early Layer	0.748	0.999	0.728	0.763	0.838
ImageGPT Medium Layer	0.779	0.999	0.684	0.810	0.665
ImageGPT Deep Layer	0.636	0.999	0.530	0.818	0.346
ConvNeXt V2 Early Layer	0.660	0.999	0.584	0.799	0.471
ConvNeXt V2 Medium Layer	0.764	0.999	0.684	0.801	0.676
ConvNeXt V2 Deep Layer	0.790	0.998	0.690	0.856	0.621
VitMAE Early Layer	0.484	0.999	0.366	0.871	0.017
VitMAE Medium Layer	0.539	0.997	0.529	0.671	0.526
VitMAE Deep Layer	0.570	0.999	0.541	0.695	0.510
SigLIP	**0.938**	**0.831**	**0.867**	**0.864**	**0.835**

Table 5.6: Final validation metrics for the fine-tuned SigLIP model.

Metric	Value
ROC-AUC	0.9755
Accuracy	0.9189
Validation Loss	0.0113
Best Threshold (Youden)	−0.1676

Motivated by this, we train a lightweight two-layer MLP with contrastive loss to map text embeddings into the fine-tuned SigLIP image-embedding space. The goal is to learn the visual structure of the glyph image space directly from text, eliminating the need for image rendering during inference and therefore making spoof detection a much more efficient process.

We evaluate spoof detection on validation and test sets using cosine similarity in the aligned space. Metrics include ROC–AUC, accuracy, contrastive loss, and the optimal Youden-index threshold. The threshold defines a similarity cutoff above which a pair is classified as a homoglyphic spoof.

The Optuna-selected hyperparameters in Table 5.11 were used for a final 40-epoch training run. The model achieved a validation ROC–AUC of 0.9774, with 6,024 true positives and only 403 false negatives. Performance remained strong on the previously unseen test set, with a ROC–AUC of 0.9668 and an accuracy of 0.9126, demonstrating strong generalization by the model.

Figures 5.15 and 5.16 show the accuracy and loss curves throughout training and provide insight into the model's behavior during training.

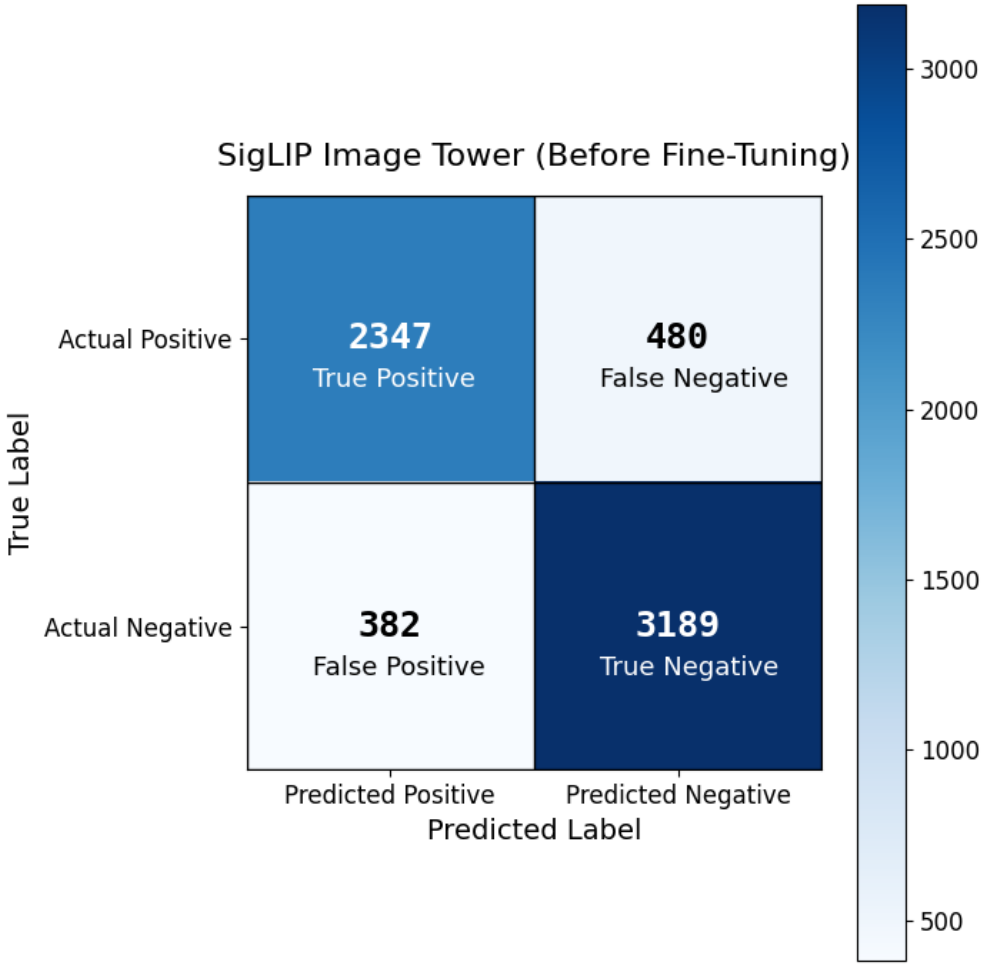

Figure 5.9: Confusion matrix for the SigLIP Image Tower before fine-tuning

Both training and validation accuracy rise quickly during the first 10 epochs as the model learns an alignment between text and glyph embeddings. After this initial phase, accuracy increases more gradually, stabilizing by epoch 20 and remaining stable through the end of training. The validation curve closely follows the training curve, suggesting that the model does not overfit.

The loss curves have an early decrease for the first 10 epochs and stabilize for the last 30 epochs. We again see a small increase in training loss during the transition from easy to medium pairs as a part of our curriculum design.

The high ROC-AUC values indicate that the aligned text embeddings recover nearly all of the visual spoof structure captured by the SigLIP image tower. The model approaches the image-only detection performance without requiring glyph rendering at inference time and out performs VA-TE, a text-only detection method. This makes the alignment model an effective and computationally efficient approach to spoof detection across a wide-range of applications.

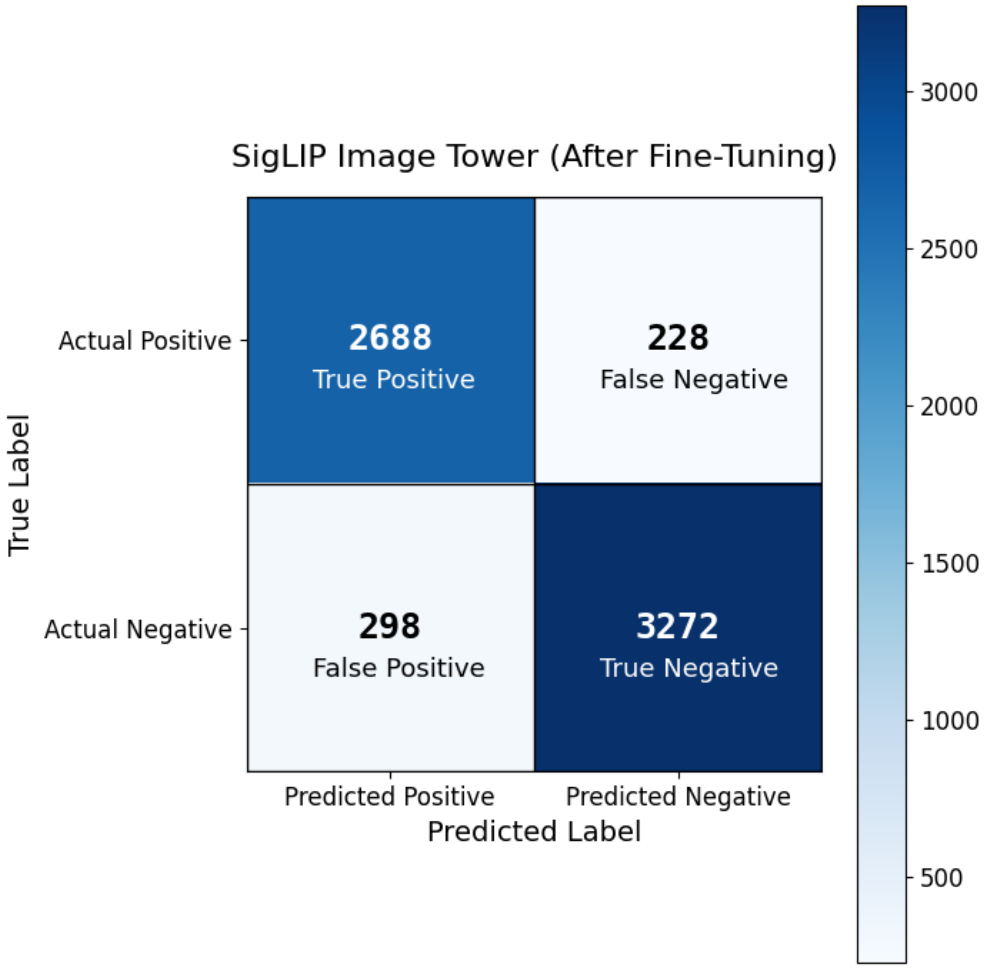

Figure 5.10: Confusion matrix for the SigLIP Image Tower after fine-tuning

Table 5.7: Hyperparameters used for the final SigLIP fine-tuning run

Hyperparameter	Value
Epochs	27
Batch Size	512
Learning Rate	4.14×10^{-5}
Weight Decay	3.26×10^{-4}
Hidden Dimension H	768
Output Dimension D_{out}	128
Contrastive Margin (m)	0.334

Table 5.8: Text Encoder Baselines

Text Encoder	ROC-AUC	threshold	accuracy	precision	recall
SigLIP Text Tower	0.471	0.938	0.377	0.809	0.409
GPT	0.758	0.741	0.672	0.845	0.600
VA-TE	**0.968**	**0.939**	**0.903**	**0.953**	**0.894**

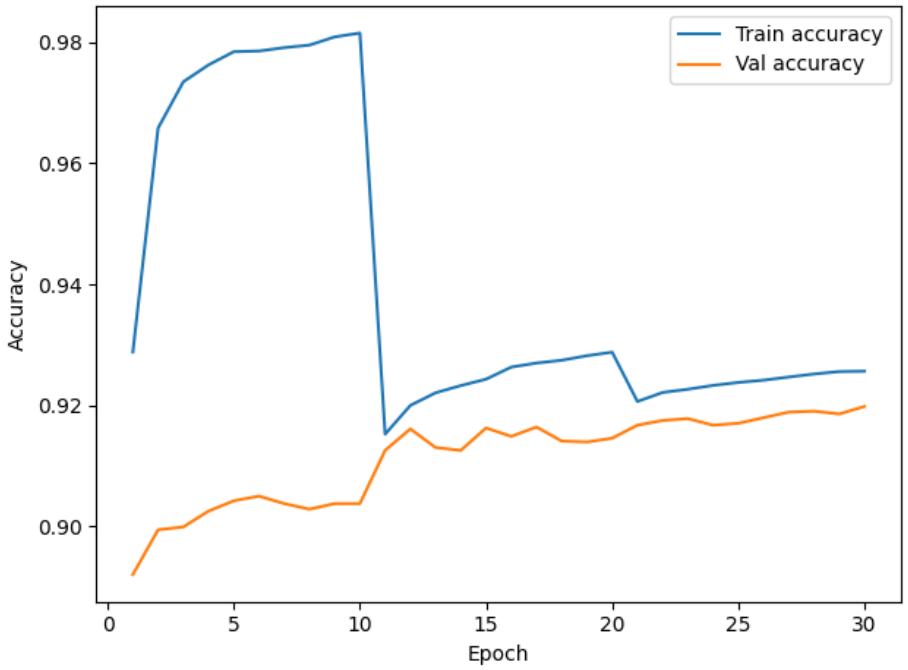

Figure 5.11: Training and validation accuracy curves for the SigLIP Image Tower fine-tuning

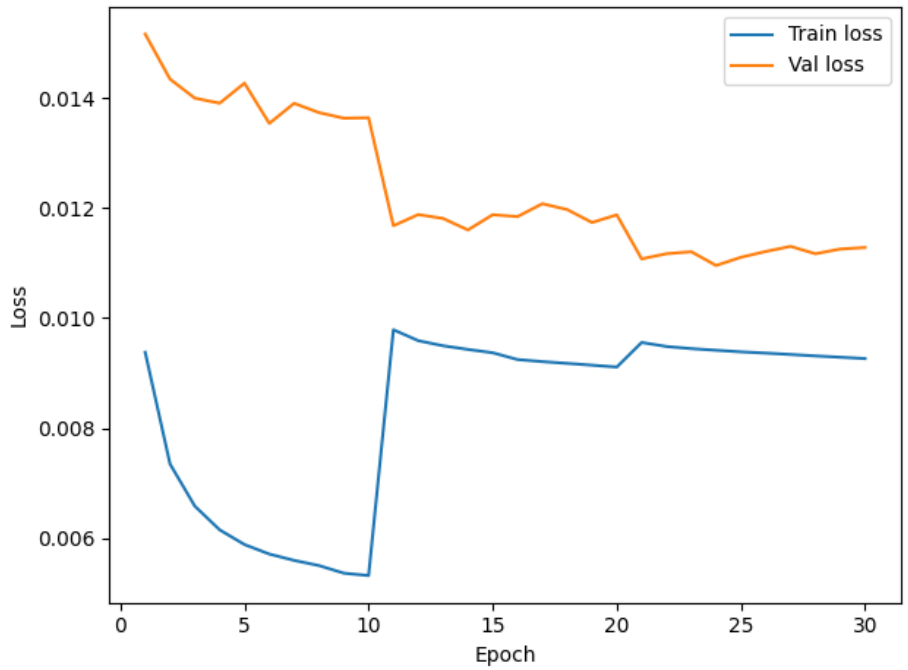

Figure 5.12: Training and validation loss curves for the SigLIP Image Tower fine-tuning

Table 5.9: Text Encoder Baselines

Text Encoder	ROC-AUC	threshold	accuracy	precision	recall
SigLIP Text Tower	0.471	0.938	0.377	0.809	0.409
GPT	0.758	0.741	0.672	0.845	0.600
VA-TE	**0.968**	**0.939**	**0.903**	**0.953**	**0.894**

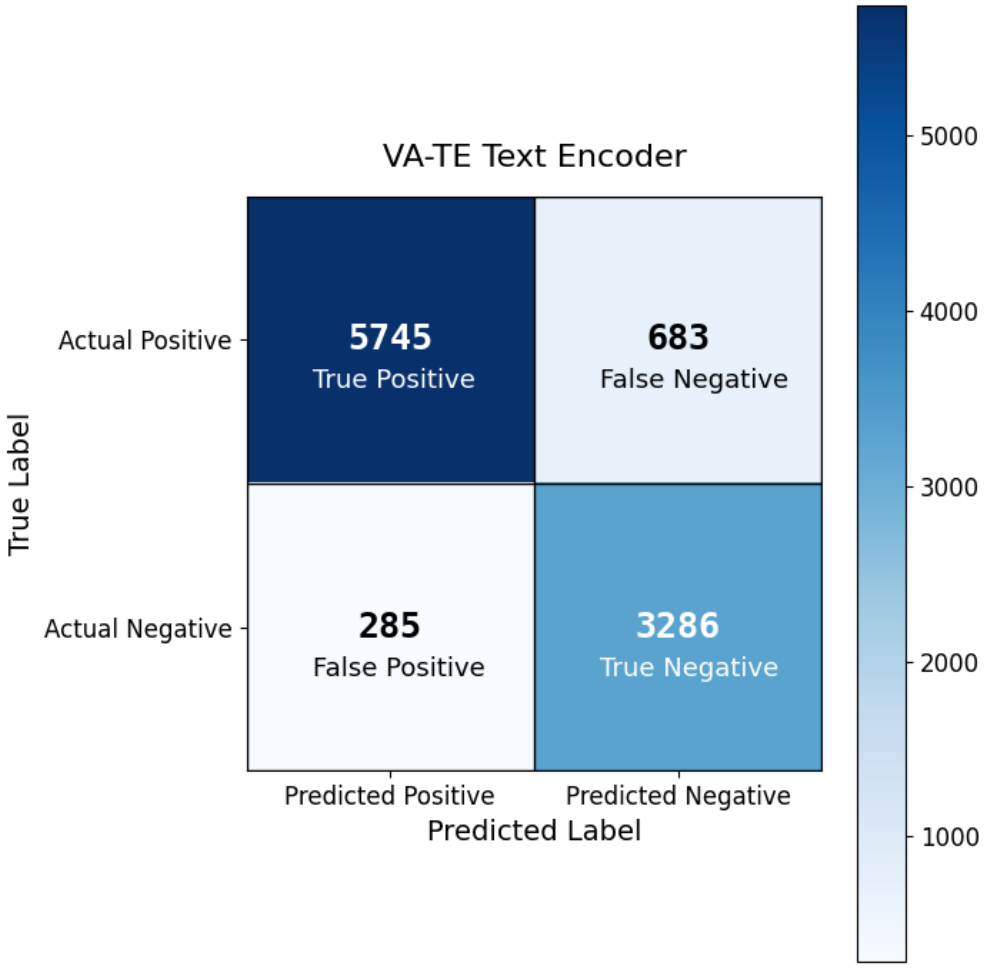

Figure 5.13: Confusion matrix for VA-TE

Table 5.10: Alignment Model Performance (Text → Image Space)

Model	ROC-AUC	threshold	accuracy	precision	recall	loss
Validation	**0.9774**	**-0.2525**	**0.9314**	**0.9551**	**0.9373**	**0.0230**

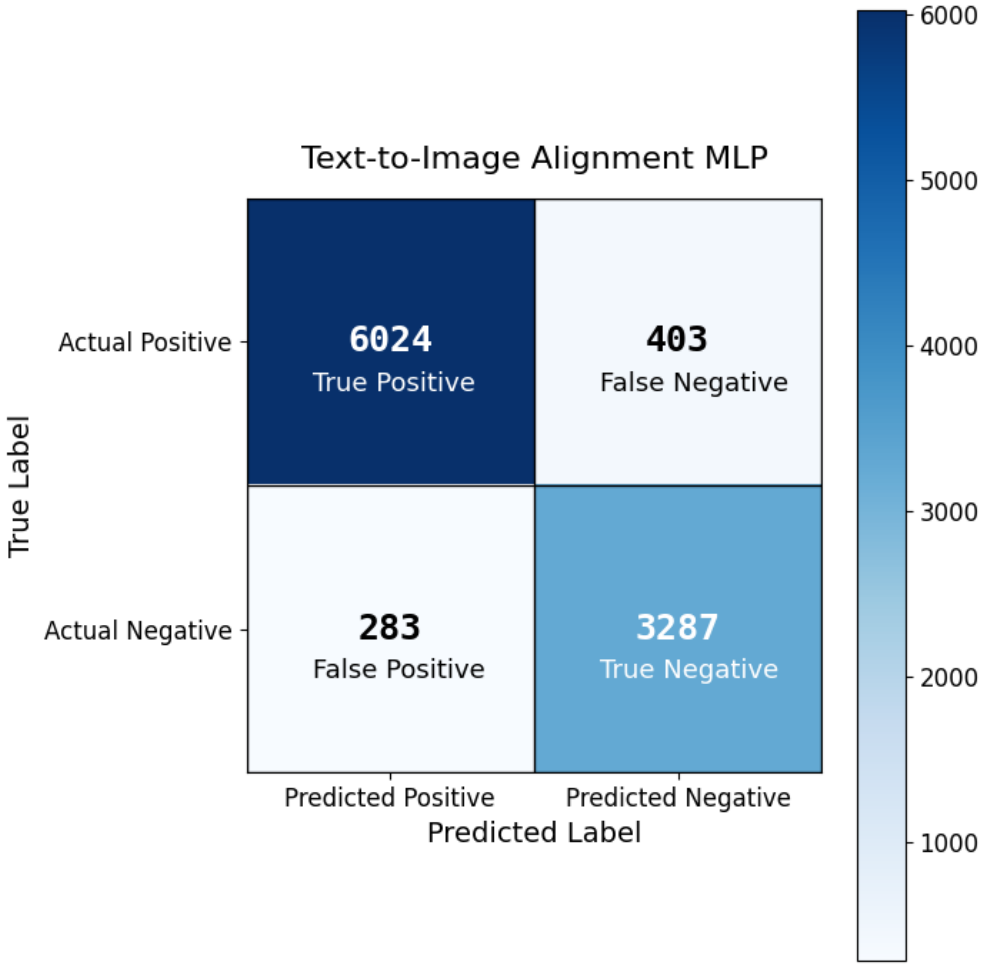

Figure 5.14: Confusion matrix for Text-to-Image Alignment MLP

Table 5.11: Hyperparameters used for the final Text→Image MLP alignment run

Hyperparameter	Value
Epochs	40
Batch Size	1024
Learning Rate	2.57×10^{-3}
Weight Decay	2.43×10^{-5}
Hidden Dimension H	512
Output Dimension D_{out}	256
Contrastive Margin (m)	0.441

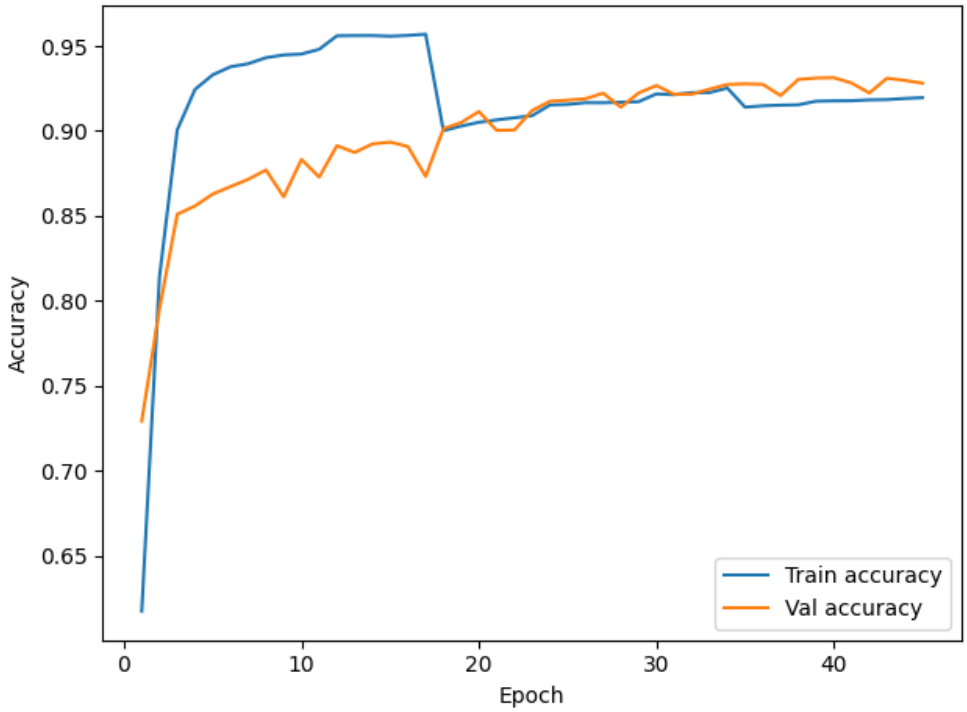

Figure 5.15: Training and validation accuracy curves for the Text-to-Image Alignment MLP

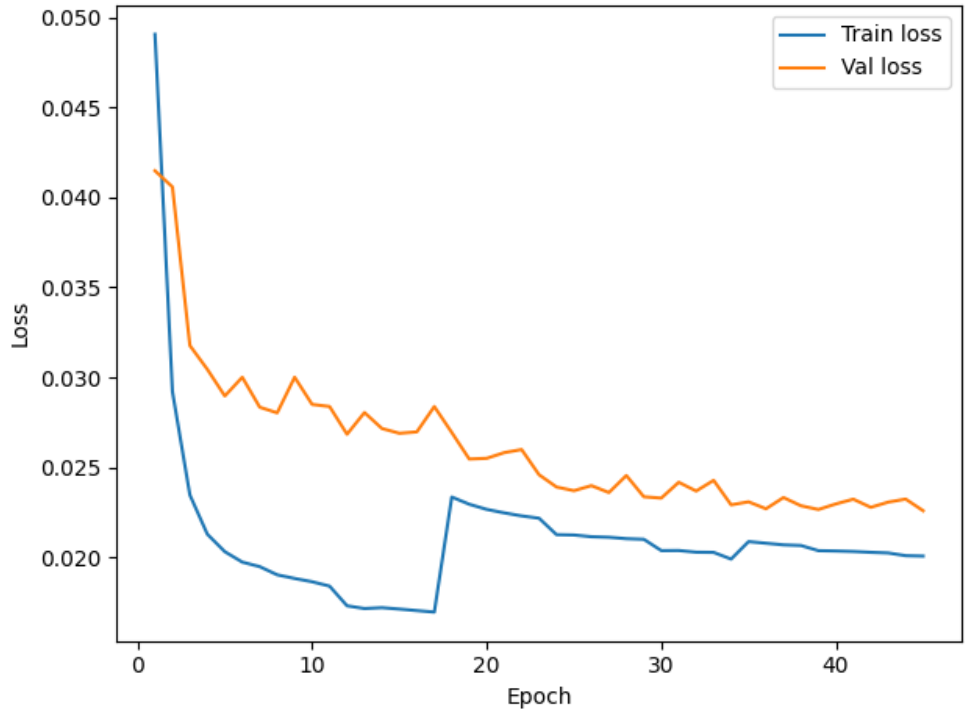

Figure 5.16: Training and validation loss curves for the Text-to-Image Alignment MLP

Chapter 6

Discussion

6.1 Conclusion

With the rise of digital platforms, being able to detect homoglyphic attacks has become an important and consequential problem to solve. Current solutions have large limitations in both performance and/or efficiency. OCR systems, which operate entirely in the image space, require large computational costs since rendering and analyzing images pixel-by-pixel is an expensive and high latency task. Text-only similarity metrics are computationally efficient since they operate on the token-level but they fall short because they are insensitive to the visual structure of strings. This visual structure is what is exploited in homoglyphic attacks since the spoof strings themselves are different. This thesis proposes a text-to-image embedding alignment model to allow for spoof detection that has the computational efficiency of text-based methods with the performance of image-based methods.

Our experiments demonstrate that vision-language models, when used solely as image encoders, already capture substantial visual information relevant to homoglyph similarity. Across six architectures evaluated at three layer depths (ResNet, ViT, ConvNeXt, MAE, ImageGPT, and SigLIP), SigLIP consistently achieved the strongest performance, with an ROC-AUC value of 0.938 using only cosine similarity in its pre-trained image-embedding space. This suggests that the SigLIP tower encodes robust shape, stroke, and style features within images, including glyph images, making it a strong foundation for our image backbone in a text-to-image embedding alignment model.

We further fine-tuned SigLIP's image encoder to achieve better separability between real and spoofed glyphs of entity names, improving SigLIP's ROC–AUC from 0.938 in the pre-trained state to around 0.9755 after contrastive training and a hyperparameter sweep. Fine-tuning sharpened the embedding space by tightening clusters of visually identical characters and widening margins between homoglyph pairs, providing a stronger and more visually grounded target space for our text-to-image alignment model.

Since SigLIP's image tower requires rendering each string into a glyph image and performing pixel-level inference which is not scalable for real-time spoof detection systems, we trained a two-layer MLP to project text embeddings directly into the fine-tuned SigLIP image space again using contrastive learning. The resulting alignment model achieved a validation ROC-AUC of 0.9774 and a test ROC-AUC of 0.9668, surpassing the performance of

text-only and image-based baselines. This model also eliminates the need for glyph rendering at inference and only requires a pre-computed embeddings store. The model also achieved an accuracy of 0.9314 on the validation set and 0.9126 on the test set, and the contrastive loss converged stably which indicates that the model successfully learned a stable mapping from the text embedding space into the glyph-image embedding space.

These results support the idea that visual similarity can be learned directly into a text embedding space when aligned to a visually grounded target space, even without pixel information during inference. The proposed model is an effective and efficient alternative to OCR or image-based spoof detection processes since the method is fast, scalable, and well-suited for integration into systems that process many strings at a time.

Despite these results, there are limitations to the model. First, the alignment model relies on the SigLIP image tower as its image backbone and therefore inherits any biases or failure points that the SigLIP image tower has. In addition, although contrastive loss training encourages robustness to font variation, the current dataset is limited to domain name strings with standardized fonts, resolutions, and styles. The model's performance in a glyph domain with greater text-style variation needs to be further studied.

All in all, this thesis demonstrates that visually grounded spoof detection can be achieved without rendering images at inference time. By leveraging SigLIP's image backbone and embedding space and learning an efficient text-to-image alignment model, we are able to distill nearly all of the visual structure information of image-based homoglyph detection at a much lower computational cost since we are using text embeddings at inference. This thesis represents a step towards more scalable homoglyph detection methods grounded in multimodal alignment techniques.

6.2 Future Work

Although this thesis demonstrates that aligning text embeddings to glyph image representations can be an effective approach to detecting homoglyph attacks, there are many opportunities for future work in extending the approach by improving its robustness and efficiency. First, future work may explore alternative text encoders, such as CANINE, ByT5, or byte-based transformers that might be able to capture text structure at a more granular level than token-based encoders.

Another area of future work is developing more complex text-to-image alignment architectures. This thesis proposes a lightweight two-layer MLP, but richer projectors involving residual MLP blocks and lightweight attention layers may be able to better capture non-linear relationships between textual and visual embeddings. More fine-tuning of the SigLIP vision backbone is another direction for future work. Since the performance of the SigLIP vision backbone represents an upper bound in performance that the alignment MLP can achieve, we can consider different or more advanced fine-tuning approaches such as using LoRA, a parameter-efficient fine-tuning method that adjusts a pre-trained model by training small, low-rank matrices while keeping the original model weights static.

Improvements in data robustness can also strengthen this thesis' approach to spoof detection. The current data is large and contains many spoof variations but as homoglyph attacks become more prevalent, new techniques will emerge that should be considered when

expanding the dataset. Future work can also include exploring synthetic data generation approachs for spoof detection by introducing font-variations, cross-script modifications, and rendering inconsistencies across devices or platforms.

Finally, there is future work to be done in the deployment of the proposed model in industry-level systems. Real-world screening of entity names in different industries/applications such as business application names for bank accounts in attempts to perform brand impersonation require low-latency inference. More analysis on the efficiency of the model must be done including how feasible maintaining an embeddings store of pre-computed image embeddings would be.

References

[1] J. Woodbridge, H. S. Anderson, A. Ahuja, and D. Grant. *Detecting Homoglyph Attacks with a Siamese Neural Network*. 2018. arXiv: 1805.09738 [cs.CR]. URL: https://arxiv.org/abs/1805.09738.

[2] A. M. Almuhaideb, N. Aslam, A. Alabdullatif, S. Altamimi, S. Alothman, A. Alhussain, W. Aldosari, S. J. Alsunaidi, and K. A. Alissa. "Homoglyph Attack Detection Model Using Machine Learning and Hash Function." *Journal of Sensor and Actuator Networks*, **11**(3), 2022, p. 54. DOI: 10.3390/jsan11030054. URL: https://www.mdpi.com/2224-2708/11/3/54.

[3] V. I. Levenshtein. "Binary Codes Capable of Correcting Deletions, Insertions and Reversals." *Soviet Physics Doklady*, **10**(), Feb. 1966, p. 707.

[4] M. A. Jaro. "Advances in Record-Linkage Methodology as Applied to Matching the 1985 Census of Tampa, Florida." *Journal of the American Statistical Association*, **84**(406), 1989, pp. 414–420. ISSN: 01621459, 1537274X. URL: http://www.jstor.org/stable/2289924 (visited on 11/24/2025).

[5] F. Borisyuk, A. Gordo, and V. Sivakumar. "Rosetta: Large Scale System for Text Detection and Recognition in Images." In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2018, pp. 71–79. DOI: 10.1145/3219819.3219861. URL: https://doi.org/10.1145/3219819.3219861.

[6] A. Jovanovic-Hacon, S.-W. Luo, H. Kim, R. Palacios, M. Wanderley, F. S. Beserra, and A. Gupta. "Visually-Aligned Text Embeddings via Vision-Language Models for Homoglyph Detection." 2025 (Paper under review).

[7] A. Radford et al. *Learning Transferable Visual Models From Natural Language Supervision*. 2021. arXiv: 2103.00020 [cs.CV]. URL: https://arxiv.org/abs/2103.00020.

[8] C. Jia, Y. Yang, Y. Xia, Y.-T. Chen, Z. Parekh, H. Pham, Q. V. Le, Y. Sung, Z. Li, and T. Duerig. *Scaling Up Visual and Vision-Language Representation Learning With Noisy Text Supervision*. 2021. arXiv: 2102.05918 [cs.CV]. URL: https://arxiv.org/abs/2102.05918.

[9] J. Li, R. Selvaraju, A. Gotmare, S. Joty, C. Xiong, and S. C. H. Hoi. "Align before Fuse: Vision and Language Representation Learning with Momentum Distillation." In: *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan. Vol. 34. Curran Associates, Inc., 2021, pp. 9694–9705. URL: https://proceedings.neurips.cc/paper_files/paper/2021/file/505259756244493872b7709a8a01b536-Paper.pdf.

[10] X. Zhai, X. Wang, B. Mustafa, A. Steiner, D. Keysers, A. Kolesnikov, and L. Beyer. *LiT: Zero-Shot Transfer with Locked-image text Tuning*. 2022. arXiv: 2111.07991 [cs.CV]. URL: https://arxiv.org/abs/2111.07991.

[11] J.-B. Alayrac et al. "Flamingo: a Visual Language Model for Few-Shot Learning." In: *Advances in Neural Information Processing Systems*. Ed. by S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh. Vol. 35. Curran Associates, Inc., 2022, pp. 23716–23736. URL: https://proceedings.neurips.cc/paper_files/paper/2022/file/960a172bc7fbf0177ccccbb411a7d800-Paper-Conference.pdf.

[12] X. Chen et al. *PaLI-X: On Scaling up a Multilingual Vision and Language Model*. 2023. arXiv: 2305.18565 [cs.CV]. URL: https://arxiv.org/abs/2305.18565.

[13] H. Liu, C. Li, Q. Wu, and Y. J. Lee. *Visual Instruction Tuning*. 2023. arXiv: 2304.08485 [cs.CV]. URL: https://arxiv.org/abs/2304.08485.

[14] X. Zhai, B. Mustafa, A. Kolesnikov, and L. Beyer. *Sigmoid Loss for Language Image Pre-Training*. 2023. arXiv: 2303.15343 [cs.CV]. URL: https://arxiv.org/abs/2303.15343.

[15] A. Dosovitskiy et al. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. 2021. arXiv: 2010.11929 [cs.CV]. URL: https://arxiv.org/abs/2010.11929.

[16] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick. *Masked Autoencoders Are Scalable Vision Learners*. 2021. arXiv: 2111.06377 [cs.CV]. URL: https://arxiv.org/abs/2111.06377.

[17] M. Oquab et al. *DINOv2: Learning Robust Visual Features without Supervision*. 2024. arXiv: 2304.07193 [cs.CV]. URL: https://arxiv.org/abs/2304.07193.

[18] K. He, X. Zhang, S. Ren, and J. Sun. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV]. URL: https://arxiv.org/abs/1512.03385.

[19] M. Tan and Q. V. Le. *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks*. 2020. arXiv: 1905.11946 [cs.LG]. URL: https://arxiv.org/abs/1905.11946.

[20] M. Tan and Q. V. Le. "EfficientNetV2: Smaller Models and Faster Training." In: *Proceedings of the 38th International Conference on Machine Learning*. PMLR, 2021, pp. 10096–10106. URL: https://proceedings.mlr.press/v139/tan21a.html.

[21] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie. *A ConvNet for the 2020s*. 2022. arXiv: 2201.03545 [cs.CV]. URL: https://arxiv.org/abs/2201.03545.

[22] M. Chen, A. Radford, R. Child, J. Wu, H. Jun, D. Luan, and I. Sutskever. "Generative Pretraining From Pixels." In: *Proceedings of the 37th International Conference on Machine Learning*. PMLR, 2020, pp. 1691–1703. URL: https://proceedings.mlr.press/v119/chen20s.html.

[23] Y. Meng, W. Wu, F. Wang, X. Li, P. Nie, F. Yin, M. Li, Q. Han, X. Sun, and J. Li. *Glyce: Glyph-vectors for Chinese Character Representations*. 2020. arXiv: 1901.10125 [cs.CL]. URL: https://arxiv.org/abs/1901.10125.

[24] A. Jain and M. Ingle. "A Comparative Study of Deep Learning Techniques in Character Recognition Systems." *Journal of the Institute of Oriental Studies RAS*, **78**(), May 2024.

[25] R. Smith. "An Overview of the Tesseract OCR Engine." In: *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*. Vol. 2. 2007, pp. 629–633. DOI: 10.1109/ICDAR.2007.4376991.

[26] J. Baek, G. Kim, J. Lee, S. Park, D. Han, S. Yun, S. J. Oh, and H. Lee. *What Is Wrong With Scene Text Recognition Model Comparisons? Dataset and Model Analysis*. 2019. arXiv: 1904.01906 [cs.CV]. URL: https://arxiv.org/abs/1904.01906.

[27] B. Shi, X. Bai, and C. Yao. *An End-to-End Trainable Neural Network for Image-based Sequence Recognition and Its Application to Scene Text Recognition*. 2015. arXiv: 1507.05717 [cs.CV]. URL: https://arxiv.org/abs/1507.05717.

[28] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman. *Reading Text in the Wild with Convolutional Neural Networks*. 2014. arXiv: 1412.1842 [cs.CV]. URL: https://arxiv.org/abs/1412.1842.

[29] M. Li, T. Lv, J. Chen, L. Cui, Y. Lu, D. Florencio, C. Zhang, Z. Li, and F. Wei. *TrOCR: Transformer-based Optical Character Recognition with Pre-trained Models*. 2022. arXiv: 2109.10282 [cs.CL]. URL: https://arxiv.org/abs/2109.10282.

[30] C. Xue, S. Lu, and F. Zhan. *Accurate Scene Text Detection through Border Semantics Awareness and Bootstrapping*. 2018. arXiv: 1807.03547 [cs.CV]. URL: https://arxiv.org/abs/1807.03547.

[31] S. Faizullah, M. S. Ayub, S. Hussain, and M. A. Khan. "A Survey of OCR in Arabic Language: Applications, Techniques, and Challenges." *Applied Sciences*, **13**(7), 2023. ISSN: 2076-3417. DOI: 10.3390/app13074584. URL: https://www.mdpi.com/2076-3417/13/7/4584.

[32] N. Roshanbin and J. Miller. "Finding homoglyphs-a step towards detecting unicode-based visual spoofing attacks." In: *International Conference on Web Information Systems Engineering*. Springer. 2011, pp. 1–14.

[33] V. R. and S. K.P. "Siamese neural network architecture for homoglyph attacks detection." *ICT Express*, **6**(1), 2020, pp. 16–19. ISSN: 2405-9595. DOI: https://doi.org/10.1016/j.icte.2019.05.002. URL: https://www.sciencedirect.com/science/article/pii/S2405959519300025.

[34] Y. Lu, M. K. K, N. Mohammed, and Y. Wang. "Homoglyph attack detection with unpaired data." In: *Proceedings of the 4th ACM/IEEE Symposium on Edge Computing*. SEC '19. Arlington, Virginia: Association for Computing Machinery, 2019, pp. 377–382. ISBN: 9781450367332. DOI: 10.1145/3318216.3363337. URL: https://doi.org/10.1145/3318216.3363337.

[35] A. Ginsberg and C. Yu. "Rapid Homoglyph Prediction and Detection." In: *2018 1st International Conference on Data Intelligence and Security (ICDIS)*. 2018, pp. 17–23. DOI: 10.1109/ICDIS.2018.00010.

[36] L. J. Sern, Y. G. Peng David, and C. J. Hao. "PhishGAN: Data Augmentation and Identification of Homoglyph Attacks." In: *2020 International Conference on Communications, Computing, Cybersecurity, and Informatics (CCCI)*. 2020, pp. 1–6. DOI: 10.1109/CCCI49893.2020.9256804.

[37] M. Gomathy. "Detecting Homoglyph Attacks with Neural Networks: A Robust Approach to Cybersecurity." *Journal of Innovations in Computer Science and Trends in IT*, **1**(2), 2024, pp. 3048–4707.

[38] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang. *The Unreasonable Effectiveness of Deep Features as a Perceptual Metric*. 2018. arXiv: 1801.03924 [cs.CV]. URL: https://arxiv.org/abs/1801.03924.

[39] D. G. Lowe. "Distinctive Image Features from Scale-Invariant Keypoints." *International Journal of Computer Vision*, **60**(2), Nov. 2004, pp. 91–110. ISSN: 1573-1405. DOI: 10.1023/B:VISI.0000029664.99615.94. URL: https://doi.org/10.1023/B:VISI.0000029664.99615.94.

[40] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. "ORB: An efficient alternative to SIFT or SURF." In: *2011 International Conference on Computer Vision*. 2011, pp. 2564–2571. DOI: 10.1109/ICCV.2011.6126544.

[41] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli. "Image quality assessment: from error visibility to structural similarity." *IEEE Transactions on Image Processing*, **13**(4), 2004, pp. 600–612. DOI: 10.1109/TIP.2003.819861.

[42] W. Winkler. "The State of Record Linkage and Current Research Problems." *Statist. Med.*, **14**(), Oct. 1999.

[43] W. Cohen, P. Ravikumar, and S. Fienberg. "A Comparison of String Metrics for Matching Names and Records." *Proc of the KDD Workshop on Data Cleaning and Object Consolidation*, (), Oct. 2003.

[44] R. Hadsell, S. Chopra, and Y. LeCun. "Dimensionality Reduction by Learning an Invariant Mapping." In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*. Vol. 2. 2006, pp. 1735–1742. DOI: 10.1109/CVPR.2006.100.

[45] K. Q. Weinberger, J. Blitzer, and L. Saul. "Distance Metric Learning for Large Margin Nearest Neighbor Classification." In: *Advances in Neural Information Processing Systems*. Ed. by Y. Weiss, B. Schölkopf, and J. Platt. Vol. 18. MIT Press, 2005. URL: https://proceedings.neurips.cc/paper_files/paper/2005/file/a7f592cef8b130a6967a90617db5681b-Paper.pdf.

[46] F. Schroff, D. Kalenichenko, and J. Philbin. "FaceNet: A unified embedding for face recognition and clustering." In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2015, pp. 815–823. DOI: 10.1109/cvpr.2015.7298682. URL: http://dx.doi.org/10.1109/CVPR.2015.7298682.

[47] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. *A Simple Framework for Contrastive Learning of Visual Representations*. 2020. arXiv: 2002.05709 [cs.LG]. URL: https://arxiv.org/abs/2002.05709.

[48] J.-B. Grill et al. "Bootstrap Your Own Latent - A New Approach to Self-Supervised Learning." In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., 2020, pp. 21271–21284. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/f3ada80d5c4ee70142b17b8192b2958e-Paper.pdf.

[49] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick. *Momentum Contrast for Unsupervised Visual Representation Learning*. 2020. arXiv: 1911.05722 [cs.CV]. URL: https://arxiv.org/abs/1911.05722.

[50] Y. Bengio, J. Louradour, R. Collobert, and J. Weston. "Curriculum learning." In: *Proceedings of the 26th Annual International Conference on Machine Learning*. ICML '09. Montreal, Quebec, Canada: Association for Computing Machinery, 2009, pp. 41–48. ISBN: 9781605585161. DOI: 10.1145/1553374.1553380. URL: https://doi.org/10.1145/1553374.1553380.

[51] X. Wang, Y. Chen, and W. Zhu. "A Survey on Curriculum Learning." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **44**(9), 2022, pp. 4555–4576. DOI: 10.1109/TPAMI.2021.3069908.

[52] A. Graves, M. G. Bellemare, J. Menick, R. Munos, and K. Kavukcuoglu. "Automated Curriculum Learning for Neural Networks." In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by D. Precup and Y. W. Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, June 2017, pp. 1311–1320. URL: https://proceedings.mlr.press/v70/graves17a.html.

[53] S. Narvekar, B. Peng, M. Leonetti, J. Sinapov, M. E. Taylor, and P. Stone. "Curriculum Learning for Reinforcement Learning Domains: A Framework and Survey." *Journal of Machine Learning Research*, **21**(181), 2020, pp. 1–50. URL: http://jmlr.org/papers/v21/20-212.html.

[54] R. Portelas, C. Colas, L. Weng, K. Hofmann, and P.-Y. Oudeyer. *Automatic Curriculum Learning For Deep RL: A Short Survey*. 2020. arXiv: 2003.04664 [cs.LG]. URL: https://arxiv.org/abs/2003.04664.

[55] Y. Kong, L. Liu, J. Wang, and D. Tao. "Adaptive Curriculum Learning." In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. 2021, pp. 5067–5076.

[56] F. Liu, T. Zhang, C. Zhang, L. Liu, L. Wang, and B. Liu. "A Review of the Evaluation System for Curriculum Learning." *Electronics*, **12**(7), 2023. ISSN: 2079-9292. DOI: 10.3390/electronics12071676. URL: https://www.mdpi.com/2079-9292/12/7/1676.

[57] F. Faghri, D. J. Fleet, J. R. Kiros, and S. Fidler. *VSE++: Improving Visual-Semantic Embeddings with Hard Negatives*. 2018. arXiv: 1707.05612 [cs.LG]. URL: https://arxiv.org/abs/1707.05612.

[58] J. Dong, X. Li, C. Xu, S. Ji, Y. He, G. Yang, and X. Wang. *Dual Encoding for Zero-Example Video Retrieval*. 2019. arXiv: 1809.06181 [cs.CV]. URL: https://arxiv.org/abs/1809.06181.

[59] P. Young, A. Lai, M. Hodosh, and J. Hockenmaier. "From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions." *Transactions of the Association for Computational Linguistics*, **2**(), 2014, pp. 67–78. URL: https://api.semanticscholar.org/CorpusID:3104920.