Using Siamese Neural Networks to Distinguish Faces from Face-Like Patterns

William Locklier
University of South Alabama
School of Computing
Mobile, United States of America
williamlocklier27@gmail.com

Kylie Arnett
University of South Alabama
School of Computing
Mobile, United States of America
Kma2022@jagmail.southalabama.edu

Abstract—**Pareidolia, seeing faces in inanimate objects, remains a persistent challenge for facial recognition systems, often leading to false positives and reduced reliability in security and biometric applications. This work investigates the use of a Siamese Neural Network (SNN) to distinguish genuine human faces from pareidolic face-like patterns by learning a similarity function rather than performing direct classification. Using paired inputs constructed from the Labeled Faces in the Wild (LFW) dataset and the Faces in Things dataset, the model learns discriminative embeddings through a shared-weight convolutional architecture optimized with contrastive loss and Euclidean distance. Across 42 controlled experiments varying data fraction, batch size, image resolution, margin, and hold-out pareidolia categories, the SNN consistently demonstrated strong separability between face–face, nonface–nonface, and face–nonface pairs. The best configuration achieved 99.6% accuracy, 0.999 AUC, and average F1-scores above 0.995, with robust generalization to previously unseen pareidolia subcategories. Notably, the model maintained meaningful discriminatory performance using as little as 0.5% of the available non-face training data, highlighting significant data efficiency. These results show that similarity-based learning offers an effective strategy for reducing pareidolia-induced false positives in facial recognition systems, providing a lightweight and interpretable framework for improving reliability in artificial vision systems.**

Index Terms—**Siamese Neural Networks, Euclidean Distance, Machine Learning, Pareidolia, Keras**

I. Introduction and Motivation

In an era increasingly shaped by digital security and biometric authentication, facial recognition technology has become ubiquitous, powering applications ranging from smartphone unlocking to large-scale surveillance systems. However, a subtle yet critical flaw persists: many models incorrectly classify non-human objects containing face-like patterns, known as pareidolia, as genuine human faces [1]. These misclassifications lead to false positives that degrade system reliability and efficiency. For example, a security camera might mistakenly flag an object such as a cloud formation or a patterned household item as a human face, diverting resources and introducing unnecessary alerts. Such errors not only waste operational effort but also expose vulnerabilities in high-stakes environments such as airport screening and fraud detection.

The core challenge lies in the inability of many facial recognition systems to reliably distinguish true human facial structures from pareidolic illusions [2]. This limitation stems from an over-reliance on superficial pattern matching, where features such as symmetry, eye-like shapes, or simple geometric arrangements trigger false detections. Although current solutions, including convolutional neural networks (CNNs) and embedding-based models such as FaceNet, excel at identity verification, they often fail to filter out non-faces. Prior research indicates that neural networks can develop human-like pareidolia tendencies, misclassifying objects with minimal resemblance to faces [2].

Existing approaches largely rely on supervised classification models trained on labeled face datasets, achieving high accuracy in controlled environments but generalizing poorly to novel pareidolic stimuli. For example, FaceNet [3] optimizes embeddings for identity clustering but does not explicitly address the challenge of distinguishing pareidolic non-faces. Other studies examine pareidolia in deep neural networks through representational similarity analysis, revealing that models trained for object categorization can inadvertently develop face-like responses to non-face images [1], [2]. However, these efforts focus primarily on characterizing the phenomenon rather than developing practical mitigation strategies.

To address this gap, we propose the use of Siamese Neural Networks (SNNs) as a principled approach for distinguishing genuine human faces from face-like objects. Unlike traditional classification models, SNNs learn a similarity function by comparing pairs of inputs, making them well-suited for capturing the nuanced differences between real faces and pareidolic patterns. This approach offers several advantages, including reduced reliance on extensive labeled datasets, improved generalization to unseen categories, and the ability to learn discriminative features specific to human versus non-human facial structures.

A. Contributions

Our contributions are as follows:

1) A specialized SNN-based similarity-learning framework for distinguishing human faces from pareidolic face-like patterns.
2) A paired-image dataset construction method capable of pairing images from any two datasets.
3) A comprehensive preprocessing and evaluation pipeline, including ROC analysis, similarity score distributions,

category difficulty assessment, and hold-out category testing.
4) An extensive ablation study examining the effects of data availability, batch size, and image resolution on model performance.
5) Insights into how dataset diversity influences generalization across unseen pareidolic categories.
6) To the broader understanding of human-like perception in artificial vision systems through a similarity-based learning perspective.

II. Background

A. Siamese Neural Networks

Siamese Neural Networks (SNNs) are specialized neural architectures composed of two identical subnetworks that share the same parameters and weights, enabling them to learn a comparative representation for determining similarity or dissimilarity between paired inputs. First introduced by Bromley et al. [4] in the 1990s for signature verification, SNNs have since evolved into powerful models for a wide range of similarity-based learning tasks.

Initially referred to as *twin neural networks*, SNNs were designed to determine whether two inputs match by learning and extracting distinctive feature representations [5]. Throughout the 1990s, they gained traction due to their scalability and efficiency in similarity-based tasks, including object comparison, facial recognition and recommendation systems [5]. Although advances in artificial neural networks have introduced alternative comparative architectures, SNNs remain highly effective for tasks that require precise and interpretable similarity measurements [5].

An SNN takes two inputs that share some underlying relationship, such as images, audio signals, or Field-Programmable Gate Array (FPGA) power traces, and processes them through identical subnetworks that extract key features and generate vector embeddings [5]. These embeddings are then compared using a distance-based loss function to quantify the similarity between the two inputs.

The architecture processes two inputs through identical feature extractors, generating embeddings that are compared using a distance metric. The network is trained using the contrastive loss function, which minimizes the distance between similar pairs while maximizing the distance between dissimilar ones [5]. The loss is defined as:

$$Loss = Y * D^2 + (1 - Y) * max(X - D, 0)^2$$

Where Y is the pair label (1 for similar, 0 for dissimilar), D is the Euclidean distance between embeddings, and X is the enforced margin. This formula ensures that similar inputs yield small distance values, while dissimilar inputs are pushed apart beyond the margin, enabling the model to learn discriminative features [5]. This overall process is illustrated conceptually in Figure 1.

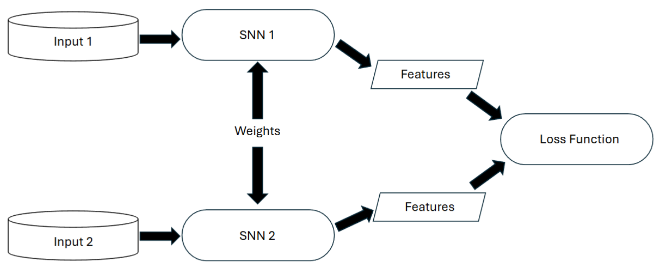


Fig. 1. A simple diagram of how SNNs take two inputs, process them through identical feature extractors with shared weights, produce embeddings, and use a loss function to train the network to measure similarity or difference between the inputs

Serrano and Bellogín [5] provide a comprehensive review of SNN applications in recommender systems, classifying existing approaches by task type, architectural design, and evaluation methodology. Their work highlights how SNNs leverage similarity learning to mitigate information overload by generating recommendations based on user-item or item-item similarity patterns. They further demonstrate the scalability and efficiency of SNN-based models across domains that extend well beyond traditional classification.

Koch et al. [6] demonstrated the effectiveness of SNNs in one-shot image recognition, achieving state-of-the-art performance on the Omniglot dataset by learning a similarity function rather than performing direct classification. Building on this paradigm, Solomon et al. [7] introduced an unsupervised face recognition method that uses SNNs to generate positive and negative training pairs without labeled data, achieving 99.40% accuracy on the Labeled Faces in the Wild (LFW) dataset.

B. Facial Recognition

Facial recognition research has advanced significantly, with numerous approaches addressing the challenges of face detection, feature extraction, and identity verification. As a core area within computer vision, facial recognition continues to drive progress in security, healthcare, and human-computer interaction, enabling increasingly accurate and efficient biometric systems.

Schroff et al. [3] introduced FaceNet, a system that learns 128-dimensional face embeddings using triplet loss, achieving 99.63% accuracy on the LFW dataset and 95.12% on YouTube Faces. The model clusters embeddings such that images of the same identity lie close together in features space while those of different identities are pushed apart. A key contribution of FaceNet was the introduction of online semi-hard negative mining, which enabled efficient and stable training on large-scale datasets. This work established a unified embedding-based framework that has since become a foundational standard for modern, scalable face recognition systems.

Recent research has also begun addressing the pareidolia phenomenon in computer vision. Gupta and Dobs [2] investigated face pareidolia in deep neural networks using five VGG16-based CNN models, finding that networks trained for both face identification and object categorization develop human-like pareidolia responses.

Hamilton et al. [1] created the *Faces in Things* dataset, containing around 5,000 annotated pareidolic face images, and identified substantial performance gaps between human and machine detection of pareidolic stimuli. Their experiments showed that fine-tuning detectors such as RetinaFace on animal face datasets improves the detection of pareidolic patterns. This work establishes a benchmark dataset and a set of computational models that provide a foundation for systematically studying pareidolia within computer vision research.

C. Synthesis

Although existing facial recognition methods achieve high accuracy in human face identification and verification, a notable gap remains in distinguishing genuine human faces from pareidolic face-like patterns using similarity-learning frameworks. Current approaches primarily address human-to-human face discrimination or rely on general object detection pipelines, but none explicitly leverage the unique strengths of SNNs for pareidolia-specific classification. This gap highlights the need for more robust facial recognition systems capable of filtering out false positives caused by face-like objects, thereby improving the reliability of real-world biometric and computer vision applications.

III. Methodology

A. Datasets

For training and evaluation, we used two curated datasets. Real human face images were sourced from the Labeled Faces in the Wild (LFW) dataset, a widely used benchmark for face verification. LFW contains 13,233 images of 5,749 individuals, captured under varying illumination, poses, and background conditions. Non-human face-like objects (pareidolic patterns) were sourced from the Faces in Things dataset [1], which provides 4,537 annotated images in which ordinary objects evoke human-like facial structures. This dataset also includes metadata describing several visual categories (e.g., animal, cartoon, accidental, design), along with additional contextual information. Combined, the two datasets provide 17,770 images, which are used to construct the following pair types:

1) Face–Face (similar, label = 1)
2) Non-Face–Non-Face (similar, label = 1)
3) Face–Non-Face (dissimilar, label = 0)

A fixed set of 9,000 training pairs and 3,000 testing pairs is generated using NumPy [8] with balanced sampling and stratification, with 20% of the training pairs held out for validation according to the methodology outlined by Bengio et al. [9]. This pipeline ensures both reproducibility and balanced class distributions across all experiments.

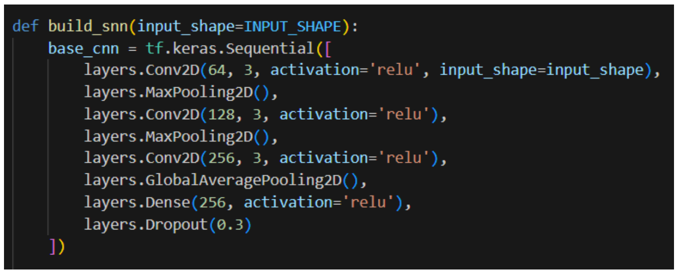
B. Preprocessing

Prior to pairing, all images from both datasets are resized to 150×150 pixels to reduce computational overhead and accommodate GPU memory constraints. Each image is then converted to Red, Green, Blue (RGB) format and normalized to values between 0 and 1. OpenCV is used for this preprocessing stage. The images are subsequently indexed and filtered using metadata from the Faces in Things dataset and processed with Pandas [10] to support category-based evaluations. We additionally enforce a constraint that only images containing a single annotated bounding box are used, excluding those with multiple embedded sub-images. This filtering step reduces the available pareidolia images from 4,537 to 3,887 prior to pairing.

Sampling is randomized and split into training and testing sets using fixed seeds (0–2) for reproducibility. No data augmentation techniques are applied in order to preserve the natural structural cues that differentiate real human faces from pareidolic illusions. Altering these cues through augmentation could unintentionally distort or obscure the subtle features that are essential for accurate discrimination.

C. Siamese Network Architecture

The Siamese Convolutional Neural Network architecture adopts a lightweight design implemented using Keras and TensorFlow [11]. The model consists of two identical CNN subnetworks with shared weights, inspired by the one-shot image recognition architecture proposed by Koch et al. [6]. Each branch extracts deep visual embeddings from its input image through three progressively larger convolutional layers, each followed by a pooling layer for spatial downsampling, culminating in an embedding layer and a dropout layer. This recognition architecture is illustrated in Figure 2.



```
def build_snn(input_shape=INPUT_SHAPE):
    base_cnn = tf.keras.Sequential([
        layers.Conv2D(64, 3, activation='relu', input_shape=input_shape),
        layers.MaxPooling2D(),
        layers.Conv2D(128, 3, activation='relu'),
        layers.MaxPooling2D(),
        layers.Conv2D(256, 3, activation='relu'),
        layers.GlobalAveragePooling2D(),
        layers.Dense(256, activation='relu'),
        layers.Dropout(0.3)
    ])
```

Fig. 2. Code snippet showing the layers used in each subnetwork totaling 3 convolutional layers, 3 pooling layers, an embedding layer, and a dropout regulation layer

Embedded vectors from both branches are compared using Euclidean distance computed through a Lambda layer. The network is trained using contrastive loss, with Adam serving as the optimizer for adjusting learning rates, weights, and biases. Training proceeds until the validation loss no longer improves, using an early stopping callback [12] to revert to the best-performing epoch and prevent overfitting.

Training continues until the contrastive loss is minimized, after which the model outputs the Euclidean distance between the two embedding vectors. Lower distances indicate higher similarity. These raw distance values are later converted into similarity scores ranging from 1 to 0, where values approaching 1 denote similar pairs (both images are faces or both are non-faces), and values approaching 0 denote dissimilar pairs (one face and one non-face). Classification is performed using an optimal threshold τ identified during training validation. The full architecture and processing pipeline are shown in Figure 3.

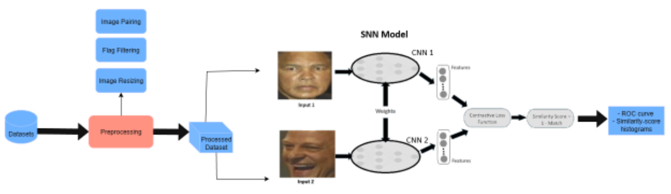


Fig. 3. Diagram showcasing the entire pipeline for dataset manipulation, preprocessing and model training and testing

To thoroughly evaluate the model's robustness and identify the most effective training configuration, we varied several parameters across 42 experimental runs. Each parameter was modified independently and tested across three random seeds to obtain a more complete understanding of optimal training, testing, and validation behaviors. A summary of these experimental groups and their corresponding parameter ranges is provided in Table I.

TABLE I
PARAMETERS VARIED ACROSS FIVE EXPERIMENTAL GROUPS.

Experiment group	**Parameters varied**
Data Fraction	Data fraction = {1, 0.5, 0.25, 0.1, 0.005}
Batch Size	Batch size = {16, 32, 64, 128, 256}
Margin	Margin = {0.5, 1.0, 1.5}
Image Size	Image size = {64x64, 128x128, 150x150} pixels
Holdout	Hold-out category = Design, Accident

Data fraction represents the percentage of available pareidolia images that are available for training image pair generation. Batch size details how many pairs are evaluated per epoch. Margin is the hyperparameter for the contrastive loss function, as discussed in Section II. Image size is only reduced for computational constraints. The two hold-out categories are chosen due to their binary relationship in the pareidolia dataset (split between intentional and unintentional pareidolia).

D. Evaluation

To measure model performance, we evaluate the SNN on unseen test pairs using the following metrics:

- **Accuracy:** Proportion of correctly classified pairs [13]
- **True Positive Rate (TPR):** Sensitivity for detecting pairs belonging to the similar category pairs [14].
- **False Positive Rate (FPR):** Rate at which dissimilar pairs are incorrectly classified as similar faces [14].
- **Receiver Operating Characteristic (ROC) Curve:** Visualization of the trade-off between TPR and FPR [15].
- **Area Under the Curve (AUC):** Scalar value summarizing the ROC curve.
- **F1-Score:** Harmonic mean of precision and recall.

To evaluate generalization, we perform hold-out category testing by withholding the "Design" images from the *Faces in Things* dataset and using them exclusively for testing. We additionally conduct per-subcategory difficulty analysis (e.g., "Animal," "Cartoon," "Human–Young," etc.) by mapping each test pair to its corresponding category labels, enabling fine-grained assessment of accuracy and AUC.

Visualization is performed using Matplotlib [16], including similarity score distributions and ROC curve plots. Metrics and confusion matrices are computed using Scikit-learn [17]. Because we use 2,000 similar pairs and 1,000 dissimilar pairs for each evaluation configuration, the overall accuracy metric is biased toward true positives and false negatives. However, our primary research objective is to evaluate the model's ability to distinguish non-faces from real faces; therefore, greater emphasis is placed on true negatives and false positives. As such, we define success as achieving greater than 0.80 F1-score and less than 10% FPR on novel pareidolic categories, which would outperform baseline CNN classifiers [18].

IV. Results

For each input pair of images, the SNN produces an output score between 0 and 1, where scores closer to 1 indicate that the two images belong to the same category (similar), and scores closer to 0 indicate that they come from different categories (dissimilar), however these are flipped later in the distribution charts. Figure 4 illustrates an example of the labeling behavior, where the SNN correctly classifies both face–face and nonface–nonface pairs with a label of 1, while assigning a label of 0 to dissimilar face–nonface pairs. This demonstrates how the model learns to map semantically similar images closer together in embedding space while pushing apart those drawn from different distributions. For each configuration, the similarity threshold used to convert scores into binary decisions was chosen on the validation set and then applied to the held-out test pairs.

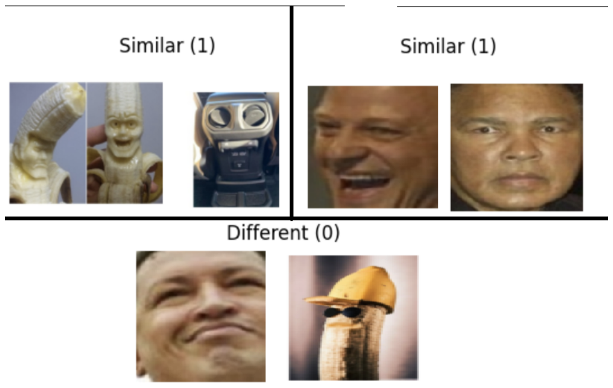


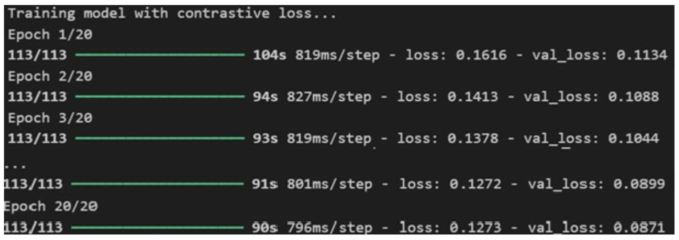
Fig. 4. Visualization of labeling behavior. The SNN correctly classified a face-face pair and a nonface-nonface pair with a label of 1, and a face-nonface pair with a label of 0

A. Model Training and Convergence

The training dataset consisted of 9,000 image pairs, with 20% reserved for validation loss to monitor generalization performance during training. The training objective was the contrastive loss function, which minimizes the distance between embeddings of similar pairs while maximizing the distance between embeddings of dissimilar pairs.

In our experiments, we employed early stopping based on validation loss to prevent overfitting and select the best-performing model. For each run, we monitored validation loss and recorded the epoch with the minimum value as the best epoch. Across all 42 runs, the best epoch varied substantially, with a mean of 15.8 epochs and a median of 12.5 epochs. This variability reflects how parameter changes influence convergence behavior. To compare training efficiency across runs, we also computed the normalized training time per epoch (total training time divided by best epoch index), which averaged 135 ± 53 seconds. Overall, these results indicate that the network typically converges well before 20 epochs under most configurations.

Throughout training, both the training and validation losses steadily decreased, indicating effective learning and minimal overfitting. As shown in Figure 5, the loss curves exhibit consistent downward trends across epochs, demonstrating that the model successfully converges and learns discriminative representations capable of distinguishing between face and non-face images.



```
Training model with contrastive loss...
Epoch 1/20
113/113 ━━━━━━━━━━━━━━━━━━━━ 104s 819ms/step - loss: 0.1616 - val_loss: 0.1134
Epoch 2/20
113/113 ━━━━━━━━━━━━━━━━━━━━ 94s 827ms/step - loss: 0.1413 - val_loss: 0.1088
Epoch 3/20
113/113 ━━━━━━━━━━━━━━━━━━━━ 93s 819ms/step - loss: 0.1378 - val_loss: 0.1044
...
113/113 ━━━━━━━━━━━━━━━━━━━━ 91s 801ms/step - loss: 0.1272 - val_loss: 0.0899
Epoch 20/20
113/113 ━━━━━━━━━━━━━━━━━━━━ 90s 796ms/step - loss: 0.1273 - val_loss: 0.0871
```

Fig. 5. A snippet of training and validation loss across epochs. The steady decrease in both metrics indicates convergence and absence of overfitting.

B. Accuracy and Performance Metrics

Data fraction was the first parameter evaluated for its impact on performance. Data fraction represents the proportion of non-face images available for constructing training pairs with other non-face or face images. Reducing the data fraction limits the pool of non-face images; for example, a fraction of 0.5 retains only half of the available non-face samples (1,944 images in this case).

For each data fraction configuration, the model was trained on 9,000 training pairs and evaluated on 3,000 test pairs while sub-sampling the non-face dataset according to fractions $f \in \{0.005, 0.10, 0.25, 0.50, 1.00\}$. Each setting was run with three seeds (0–2) under the standard configuration (batch size = 64, margin = 1.0, image size = 150, hold-out category = Design).

Table II summarizes the results. Even with only 10% of the training pairs, the SNN achieved an average test accuracy of 99.30 ± 0.22% and false positive rate of 0.37%, indicating that the model saturates quickly. Only at the lowest fraction (0.005) did performance degrade drastically to 85.36 ± 7.50% and false positive rate of 2.80%, with the worst average F1

TABLE II
DATA FRACTION EFFECT ON MODEL PERFORMANCE.

Data frac.	Best epoch	Acc. (%)	TPR (%)	FPR (%)	Avg. train time (s)
0.005	11.3 ± 11.4	85.36 ± 7.50	79.43 ± 12.33	2.80 ± 2.33	1633 ± 1091
0.100	19.3 ± 14.5	99.30 ± 0.22	99.13 ± 0.38	0.37 ± 0.38	2443 ± 1323
0.250	16.0 ± 5.3	99.27 ± 0.23	99.37 ± 0.36	0.93 ± 0.15	2058 ± 462
0.500	16.0 ± 9.8	99.30 ± 0.38	99.35 ± 0.52	0.80 ± 0.36	2138 ± 979
1.000	13.7 ± 6.4	99.04 ± 0.38	98.83 ± 0.52	0.90 ± 0.36	1848 ± 665

score recorded being 0.7645. Training times per epoch remain relatively consistent (130–140 seconds), suggesting that efficiency scales well with data reduction. The second parameter evaluated for its impact on performance and training efficiency was batch size, tested across values $b \in \{16, 32, 64, 128, 256\}$ using the full data fraction. Table III shows that smaller batch sizes (e.g., 16) yielded higher accuracy (99.39 ± 0.08%) but required longer overall training times but similar per epoch (122 - 159 seconds), while larger batch sizes produced slightly lower accuracy with mixed effects on efficiency.

TABLE III
BATCH SIZE EFFECT ON MODEL PERFORMANCE.

Batch size	Best epoch	Acc. (%)	TPR (%)	FPR (%)	Avg. train time (s)
16	19.3 ± 9.0	99.39 ± 0.08	99.33 ± 0.12	0.60 ± 0.20	2273 ± 804
32	18.7 ± 15.0	98.58 ± 1.35	98.17 ± 1.89	1.07 ± 0.67	2356 ± 1487
64	13.7 ± 6.4	99.04 ± 0.38	98.83 ± 0.52	0.90 ± 0.36	1848 ± 665
128	17.7 ± 9.5	98.77 ± 0.62	98.43 ± 0.90	1.07 ± 0.67	2465 ± 1063
256	15.3 ± 13.3	97.90 ± 0.55	97.65 ± 0.78	1.93 ± 0.95	2121 ± 1566

Although batch size did not drastically affect performance within the optimal range (16–64), increasing the batch size beyond 128 resulted in a consistent 1–2% reduction in accuracy. This reduction could be consequential in high-precision applications. These findings align with expectations for contrastive learning frameworks, where smaller batch sizes often promote better generalization due to more frequent weight updates and finer gradient signals

For the margin parameter, we varied the default value of 1.0 to 0.5 and 1.5, again evaluating each configuration across three seeds. As shown in Table IV, a higher margin (1.5) improved accuracy (99.30 ± 0.22%), increased the true positive rate, and reduced the false positive rate. However, this improvement came at the cost of approximately 50% longer overall training time. Time per epoch remained relatively constant (around 140 seconds), suggesting that the additional training time resulted from slower convergence rather than increased per-epoch computation.

TABLE IV
CONTRAST MARGIN EFFECT ON MODEL PERFORMANCE.

Margin	Best epoch	Acc. (%)	TPR (%)	FPR (%)	Avg. train time (s)
0.5	13.7 ± 6.1	99.07 ± 0.34	99.17 ± 0.38	1.07 ± 0.58	1824 ± 560
1.0	13.7 ± 6.4	99.04 ± 0.38	98.83 ± 0.52	0.90 ± 0.36	1848 ± 665
1.5	20.0 ± 8.7	99.30 ± 0.22	99.17 ± 0.52	0.73 ± 0.31	2786 ± 764

Among all parameters tested, margin had the second-lowest impact on overall accuracy. Higher margins produced only modest improvements while slowing convergence, resulting in longer training times. Nevertheless, these gains were consistent and reliable across seeds, indicating that a slightly larger margin can enhance separability without significantly altering the model's behavior.

The next parameter evaluated was image size. To assess the effect of spatial resolution, we reduced the default input size of 150×150 pixels to 128×128 and 64×64 pixels, again testing each configuration across three seeds with default settings. As shown in Table V, the smallest size (64×64) reduced total training time by nearly 90% while causing only a minor accuracy drop and increase of false positives by 0.17%. Time per epoch decreased sharply with smaller images (approximately 20 seconds at 64×64 compared to 140 seconds at 150×150), reflecting the reduced computational cost. A slight reduction to 128×128 pixels unexpectedly improved accuracy and reduced false positives.

TABLE V
IMAGE SIZE EFFECT ON MODEL PERFORMANCE.

Img size	Best epoch	Acc. (%)	TPR (%)	FPR (%)	Avg. train time (s)
64	10.3 ± 4.5	98.87 ± 0.64	98.67 ± 0.90	1.07 ± 0.58	203 ± 61
128	19.3 ± 13.6	99.17 ± 0.38	99.17 ± 0.38	0.67 ± 0.58	1580 ± 807
150	13.7 ± 6.4	99.04 ± 0.38	98.83 ± 0.52	0.90 ± 0.36	1848 ± 665

In terms of accuracy, image size had the third-largest impact, following data fraction and batch size. Despite this, the performance trade-off favored smaller images: the dramatic reduction in training time at 64×64 pixels generally outweighed the slight loss in accuracy, which remained within the standard deviation of higher-resolution runs. These results suggest that significantly downscaled inputs are viable for resource-constrained environments without substantially compromising performance.

Lastly, we evaluated the model's generalization ability using the alternate hold-out category "Accident," training on the full dataset and testing exclusively on this split under default settings. Across three seeded runs, the SNN achieved an average hold-out accuracy of 99.19%, average training times (143 seconds per epoch), and a false positive rate of 0.70%. These results indicate that the learned similarity function transfers effectively to unseen categories, reinforcing the suitability of SNNs as category-agnostic discriminators. Among all parameters tested, the hold-out category had the smallest impact on accuracy, further confirming the model's strong cross-category generalization.

Across all experiments, the corresponding minimum and maximum F1 and AUC scores are shown in Table VI, highlighting which parameters had the greatest impact on testing performance. Data fraction exerted the largest influence, with the worst-case configuration yielding an F1-score of 0.7646 and an AUC of 0.9678 at the lowest data fraction. Batch size had the second-largest effect, with extreme batch values producing a minimum F1-score of 0.9672 and a minimum AUC of 0.9951. Image size, margin, and hold-out category had comparatively minor effects on accuracy; however, image size provided substantial efficiency benefits, reducing training time by roughly 90% at 64×64 pixels and 128x128 producing one of the highest recorded accuracies (99.6%). A higher margin produced slight accuracy improvements but slowed convergence, resulting in longer training times.

TABLE VI
SUMMARY OF MAXIMUM AND MINIMUM AVERAGE F1 AND AUC SCORES ACROSS PARAMETER GROUPS.

Parameter Group	**Max Avg F1**	**Min Avg F1**	**Max AUC**	**Min AUC**
Data Fraction	0.9955	0.7645	0.9999	0.9678
Batch Size	0.9948	0.9672	0.9999	0.9951
Margin	0.9944	0.9858	0.9999	0.9984
Image Size	0.9955	0.9832	0.9995	0.9977
Holdout Category	0.9940	0.9858	0.9998	0.9986

C. Best Case Outcome

Across the 42 experimental runs, including both training and testing, many configurations performed well, with results often falling within overlapping standard deviations. For deeper analysis, we focused on one of the three top-performing parameter sets (DF = 1.0, BS = 64, M = 1.0, IS = 128, HC = "Design"), which achieved the highest recorded accuracy (99.6%) and correspondingly high true positive and true negative rates.

In this configuration, the model exhibited near-perfect separation between similar and dissimilar pairs, as shown in the similarity score distribution in Figure 6 and the corresponding confusion matrix in Figure 7. The distribution displays two clearly defined clusters with minimal overlap near the decision threshold, indicating strong discriminative ability. These results highlight how the embeddings learned by the SNN occupy distinct and well-separated regions of the feature space.

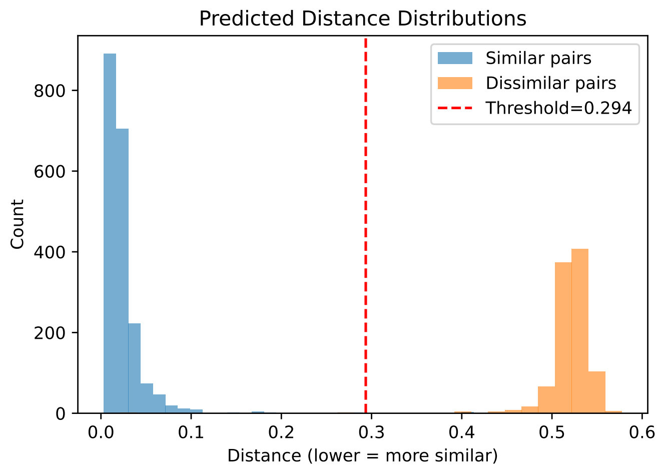
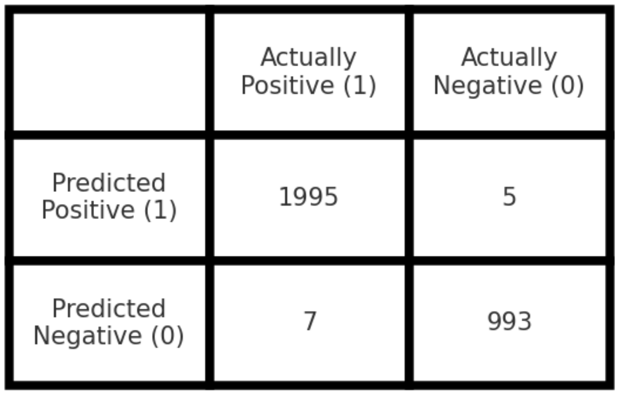


Fig. 6. Predicted similarity score distributions for the best parameter set (DF=1.0, BS=64, M=1.0, IS=128, HC='Design')



	Actually Positive (1)	Actually Negative (0)
Predicted Positive (1)	1995	5
Predicted Negative (0)	7	993

Fig. 7. Confusion matrix summarizing classification results across 3,000 test pairs for the best parameter set (DF=1.0, BS=64, M=1.0, IS=128, HC='Design')

The ROC curve in Figure 8 corroborates these findings, showing a steep rise toward a true positive rate of 1.0 with an accompanying false positive rate approaching 0. This pattern reflects exceptionally strong classification performance and confirms that the SNN achieves robust separability between similar and dissimilar image pairs under its best-performing configuration.

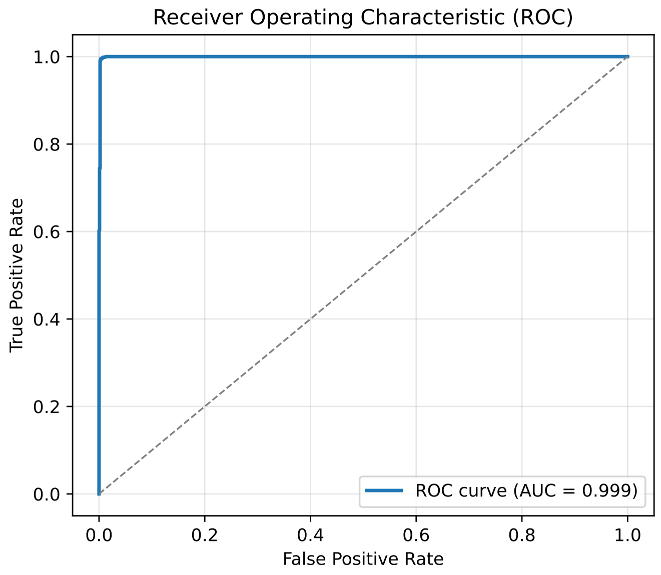


Fig. 8. Receiver Operating Characteristic curve for the SNN trained with its best parameter set (DF=1.0, BS=64, M=1.0, IS=128, HC='Design')

Subcategory performance for the best run of this configuration, shown in Table VII, revealed consistently high accuracy across all groups, including "Human-Adult," "Human-Young," "Animal," "Cartoon," and "Robot." Each subcategory achieved an AUC greater than 0.999, with accuracies ranging from 97.7% in the Human-Young subgroup to a perfect 100% in the Cartoon, Animal, and Robot categories. These results demonstrate the model's strong generalization across diverse visual domains and its particular effectiveness in distinguishing highly stylized or non-human facial structures.

TABLE VII
PER-CATEGORY ACCURACY AND AUC SCORES FOR THE SNN TRAINED WITH ITS BEST PARAMETER SET.

Category	**Samples**	**AUC**	**Accuracy**
Human-Adult	247	0.9998	0.9939
Human-Young	111	0.9995	0.9775
Cartoon	280	1.0	1.0
Animal	370	1.0	1.0
Robot	93	1.0	1.0
Other	348	0.9998	0.9943
Overall	**1449**	–	**0.9943 (99.43%)**

These results indicate that as subcategories deviate further from realistic human appearances, the model's ability to distinguish them improves, yielding perfect accuracy in several cases. This confirms that the SNN generalizes effectively across a wide range of facial and face-like structures. Additionally, two parameter configurations achieved perfect true negative detection (DF = 1.0, BS = 16, M = 1.0, IS = 150, HC = "Design"), although their overall accuracies were lower due to increased false positives and false negatives, resulting in F1-scores of 0.9922 and 0.9800, respectively.

D. *Worst Case Outcome*

Given the previous findings, it is unsurprising that the worst-performing configuration corresponded to the lowest data fraction setting (DF = 0.005, BS = 64, M = 1.0, IS = 150, HC = "Design"), since data fraction had the strongest influence on model accuracy. With this configuration, the model had access to approximately 20 non-face training images and achieved the worst recorded accuracy result (76.7%). As shown in Figures 9, 10, and 11, performance degraded substantially under these constraints.

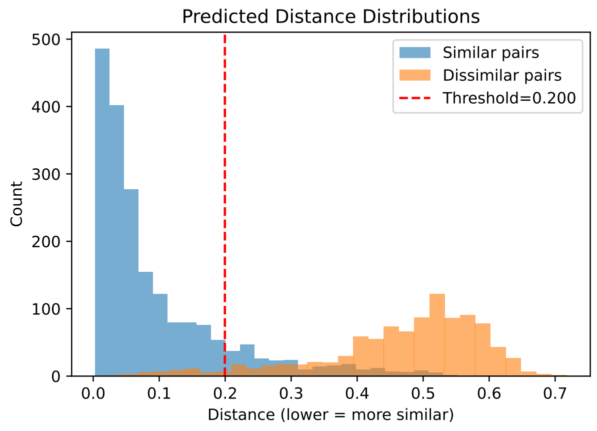
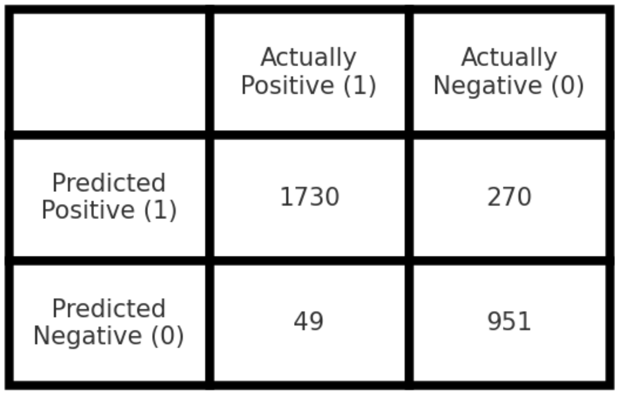


Fig. 9. Predicted similarity score distributions for the worst parameter set (DF=0.005, BS=64, M=1.0, IS=150, HC='Design')

Nonetheless, despite the extremely limited training data, the model was able to achieve an average accuracy above 80%

and maintain a false positive rate below 10%, demonstrating that the SNN retains some separability even under severe data scarcity. Given the previous example of SNNs being used for one-shot learning [6], these results make sense.



	Actually Positive (1)	Actually Negative (0)
Predicted Positive (1)	1730	270
Predicted Negative (0)	49	951

Fig. 10. Confusion matrix summarizing classification results across 3,000 test pairs for the worst parameter set (DF=0.005, BS=64, M=1.0, IS=150, HC='Design')

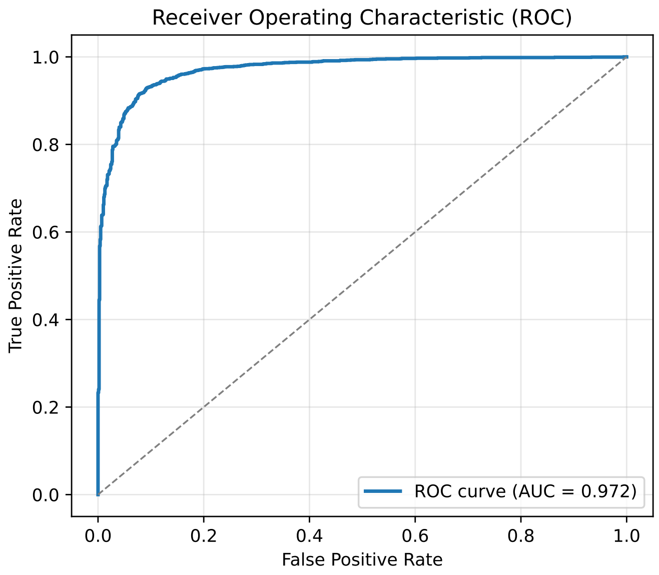


Fig. 11. Receiver Operating Characteristic curve for the SNN trained with its worst parameter set (DF=0.005, BS=64, M=1.0, IS=150, HC='Design')

Subcategory performance for the worst run of this configuration, shown in Table VIII, revealed substantially reduced accuracy across all groups. No subcategory achieved an AUC higher than 0.9705, and accuracies ranged narrowly between 73.12% and 76.13% across sampled categories. These results illustrate the model's diminished but still reliable ability to discriminate between real and pareidolic images when trained with severely limited non-face data.

TABLE VIII
PER-CATEGORY ACCURACY AND AUC SCORES FOR THE SNN TRAINED WITH ITS WORST PARAMETER SET.

Category	**Samples**	**AUC**	**Accuracy**
Human-Adult	247	0.9532	0.7337
Human-Young	111	0.9540	0.7613
Cartoon	280	0.9465	0.7419
Animal	370	0.9705	0.7398
Robot	93	0.9702	0.7312
Other	348	0.9615	0.7356
Overall	**1449**	–	**0.7406 (74.06%)**

V. CONCLUSION

This study demonstrates the effectiveness of SNNs in distinguishing genuine human faces from pareidolic patterns, addressing a critical source of false positives in facial recognition systems. By leveraging contrastive loss and Euclidean distance on paired embeddings, the proposed SNN achieved up to 99.6% test accuracy in its best performing configuration. Key evaluation metrics, including macro F1-scores exceeding 0.99 and AUC values approaching 1.0, confirm strong separability between true faces and face-like non-faces. Ablation experiments further showed that the model saturates quickly, maintaining over 99% accuracy with data fractions as low as 10%, while smaller batch sizes and higher margins yielded additional performance gains at the cost of longer training times, especially for larger margins.

Across 42 experimental runs varying data fraction, batch size, image resolution, margin, and a hold-out category, the model consistently achieved high accuracy, strong separation between similar and dissimilar pairs, and near-perfect AUC scores in most configurations. Even under constrained settings, the SNN retained meaningful discriminative capability. Among the evaluated parameters, data fraction and batch size exerted the largest influence on performance, while image resolution and margin contributed smaller but measurable effects. Hold-out testing further confirmed that the learned similarity metric generalizes well to unseen pareidolic subgroups, supporting the SNN's effectiveness as a category-agnostic discriminator.

Overall, the findings provide strong evidence that similarity-based learning is a highly effective strategy for mitigating pareidolia related false positives in facial recognition systems. With accuracies exceeding 99%, low false positive rates, strong generalization, and clear separation in embedding space, this SNN architecture demonstrates both practical utility and scientific value. These results position paired-input learning as a promising direction for developing more reliable, robust, and interpretable human vision models.

A. Limitations

While the Siamese Network demonstrated exceptional performance, several limitations remain. Hardware constraints prevented experimentation with larger, more complex image datasets, which may influence the impact of our resolution scheme in testing performance. Additionally, this study relied only on two datasets, which despite providing good coverage of real and pareidolic faces, may introduce bias based on their collection methods. Incorporating broader and more diverse datasets would likely improve the model's generalization capabilities.

Pair generation relied on controlled sampling with a balanced distribution of face-face, nonface-nonface, and face-nonface pairs. This distribution does not fully reflect real-world conditions, where pair types occur with highly uneven frequencies and ambiguous visual cues. As a result, model performance may differ when deployed in unconstrained environments. Additionally, the network architecture was intentionally lightweight, using a shallow CNN for efficiency; while effective in this study, it may lack the depth needed to capture higher-level semantic features present in larger or more complex images.

Lastly, all experiments were conducted offline using curated test pairs. Real-world deployment, such as integration with live camera systems, would introduce additional challenges, including noise, compression artifacts, occlusions, and motion blur—that were not evaluated in this study. These factors may impact similarity measurements and should be considered in future work.

B. Future Work

Future research should focus on scaling this approach by utilizing systems with greater memory capacity and computational resources for training. The current model was trained on 150×150, 128×128, and 64×64 pixel images to accommodate hardware constraints, which may limit the representation of fine-grained facial details and textural nuances. Increasing image resolution would enable the model to capture richer visual information, potentially improving embedding quality and overall performance on larger or noisier images.

With improved memory capacity, the system could also support larger batch sizes and expanded datasets, enabling more comprehensive training and evaluation across diverse image domains. These enhancements would allow the SNN to process higher-resolution data efficiently while maintaining stable convergence, ultimately improving model robustness and generalization.

In addition, deploying this model in real-time or embedded environments could provide valuable insights into its practical performance in applications such as biometric authentication, content moderation, and synthetic image detection. Further testing should be done with other models in order to draw comparisons between this binary classification strategy and others.

VI. Acknowledgment

The authors would like to thank Chris Mijangos Xicara for generously sharing his code, references, and detailed posts on SNNs. Although the code was not directly used in this research, reviewing it provided valuable context and direction at the outset of this work. His contributions helped us better understand the foundations of SNNs and guided the development of our study.

References

[1] M. Hamilton, S. Stent, V. DuTell, A. Harrington, J. Corbett, R. Rosenholtz, and W. T. Freeman, "Seeing faces in things: A model and dataset for pareidolia," in *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer, 2023, pp. 1–19. [Online]. Available: https://aka.ms/faces-in-things

[2] P. Gupta and K. Dobs, "Human-like face pareidolia emerges in deep neural networks optimized for face and object recognition," *PLoS Computational Biology*, vol. 21, no. 1, p. e1012751, 2025. [Online]. Available: https://doi.org/10.1371/journal.pcbi.1012751

[3] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 815–823. [Online]. Available: https://arxiv.org/abs/1503.03832

[4] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, "Signature verification using a Siamese time delay neural network," in *Advances in Neural Information Processing Systems*, vol. 6, 1993, pp. 737–744. [Online]. Available: https://proceedings.neurips.cc/paper/1993/file/288cc0ff022877bd3df94bc9360b9c5d-Paper.pdf

[5] N. Serrano and A. Bellogín, "Siamese neural networks in recommendation," *Applied Intelligence*, 2023. [Online]. Available: https://doi.org/10.1007/s10489-023-05097-7

[6] G. Koch, R. Zemel, and R. Salakhutdinov, "Siamese neural networks for one-shot image recognition," in *Proceedings of the 32nd International Conference on Machine Learning (ICML) Deep Learning Workshop*, 2015. [Online]. Available: https://www.cs.cmu.edu/~rsalakhu/papers/oneshot1.pdf

[7] E. Solomon, A. Woubie, and E. S. Emiru, "Deep learning based face recognition method using siamese network," *arXiv preprint arXiv:2312.14001*, 2023. [Online]. Available: https://arxiv.org/abs/2312.14001

[8] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. Fernández del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," *arXiv preprint arXiv:2006.10256*, Jun. 2020, [Preprint].

[9] Y. Bengio, "Practical recommendations for gradient-based training of deep architectures," in *Neural Networks: Tricks of the Trade*. Springer, 2012, pp. 437–478. [Online]. Available: https://arxiv.org/abs/1206.5533

[10] W. McKinney, "Data structures for statistical computing in python," in *Proceedings of the 9th Python in Science Conference*, S. van der Walt and J. Millman, Eds., 2010, pp. 56–61.

[11] Keras Team, "Keras guides," https://keras.io/guides/, 2024. [Online]. Available: https://keras.io/guides/

[12] L. Prechelt, "Early stopping – but when?" in *Neural Networks: Tricks of the Trade*. Springer, 1998, pp. 55–69. [Online]. Available: https://page.mi.fu-berlin.de/prechelt/Biblio/stop_tricks1997.pdf

[13] D. Chicco, "Ten quick tips for machine learning in computational biology," *BioData Mining*, vol. 10, no. 1, p. 35, 2017.

[14] T. Fawcett, "An introduction to roc analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, 2006.

[15] J. A. Hanley and B. J. McNeil, "The meaning and use of the area under a receiver operating characteristic (roc) curve," *Radiology*, vol. 143, no. 1, pp. 29–36, 1982.

[16] J. D. Hunter, "Matplotlib: A 2D graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[17] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[18] V. D. Nguyen, N. Van Toan, D. Van Hieu *et al.*, "Real-time deep learning-based face recognition system," in *Proceedings of the 2018 International Conference on System Science and Engineering (ICSSE)*. IEEE, 2018, pp. 62–67.