Smile Detection: Comparative Study of CNNs, Autoencoders, and SVM

Mariana L. Maldonado, Ximena C. Ayllón, Nubia S. G. Barajas, Marissa L. Landa

Abstract—**This work evaluates three fundamentally different approaches to smile detection on a curated subset of the LFW dataset: a Convolutional Neural Network (CNN), a Convolutional Autoencoder trained as a one-class anomaly detector, and a Support Vector Machine (SVM) with PCA. All experiments used identical stratified splits, and we report predictive performance, training time, inference time, number of parameters, and FLOPs. Our results show clear differences in performance and computational efficiency, outlining the trade-offs of deep learning, anomaly detection, and classical machine learning for subtle facial expression classification.la**

I. Introduction and Motivation

Smile detection is an essential component of facial-expression analysis, with broad applications in affective computing, user-experience design, and human–behavior studies. Understanding how different learning approaches interpret subtle facial cues helps us build systems that are both accurate and computationally efficient. In this project, we compare three modeling paradigms:

- **CNN:** a supervised deep-learning classifier that learns features directly from images.
- **Autoencoder:** an unsupervised, one-class model that detects deviations from a learned "normal" representation.
- **SVM:** a classical machine-learning pipeline that relies on dimensionality reduction and a margin-based classifier.

The goal is not only to determine which model achieves the highest accuracy, but also to analyze how they differ in terms of learned representations, computational cost, and robustness when generalizing to new data

II. Methodology

A. *Dataset Description*

The dataset consists of 1,203 color face images extracted from the LFW "lfwcrop_color" archive, with 600 labeled as "Smile" and 603 as "Non-smile". Each image was converted to RGB, resized to 64×64, and normalized to $[0, 1]$. These preprocessing steps were implemented directly inside the PyTorch `LFWDataset` class.

B. *Preprocessing and Stratified Splitting*

All models were trained using the same data pipeline and identical train/validation/test partitions to guarantee a fair comparison. The preprocessing begins with a custom PyTorch `Dataset` that loads the original LFW images, matches each filename with the smile and non-smile label lists, converts every image to RGB, and applies a basic transform that resizes all samples to **64×64** and converts them into tensors.

To ensure reproducibility, all random generators from Python, NumPy, and PyTorch were fixed with **seed = 42**, and the computation device (CPU or GPU) was selected automatically.

Because the dataset contains two classes with slightly different frequencies, we applied a **stratified splitting procedure** to preserve class balance across partitions. The final proportions were:

- **70%** training
- **10%** validation
- **20%** test

The splitting was performed in two steps using scikit-learn's `train_test_split`: first, separating 70% for training and 30% for validation/test; second, dividing that 30% into validation and test using their relative proportions, while maintaining stratification.

The final indices were wrapped into PyTorch `Subset` objects, and three dataloaders were created using a **batch size of 32**, with shuffling enabled only for the training loader. This procedure provides consistent preprocessing, balanced subsets, and a controlled experimental setup for evaluating CNN, Autoencoder, and SVM on exactly the same data.

Stratification was performed using `train_test_split` (scikit-learn) with seed = 42. The resulting indices were wrapped into PyTorch `Subset` objects, and dataloaders were created with batch size 32.

C. *CNN Architecture*

The convolutional neural network used in this project is a compact but expressive model designed to learn discriminative spatial patterns relevant for smile detection [**?**]. The architecture is organized into two main components: a feature extractor and a classification head.

a) Feature Extractor.: The network begins with two convolutional blocks. Each block consists of:

- a 3×3 convolution with padding to preserve spatial resolution,
- batch normalization to stabilize activation dynamics,
- a ReLU non-linearity,
- a 2×2 max-pooling layer that halves the spatial dimensions (from $64 \to 32 \to 16$).

These layers progressively capture edges, textures, and mid-level facial structures that differentiate smiles from neutral expressions.

b) Classification Head.: After feature extraction, the spatial tensor is flattened and passed through:

- a fully connected layer with 64 hidden units and ReLU,
- a dropout layer with probability 0.65 to reduce overfitting observed in initial experiments,
- a final linear layer that outputs the logits for the two target classes (smile / non-smile).

c) Optimization.: Training was performed using the Adam optimizer with a learning rate of 10^{-3} and weight decay 10^{-3}, which acts as L2-regularization. These hyperparameters were selected based on preliminary training behavior; both dropout and weight decay were necessary to control overfitting, while more extensive hyperparameter search was not conducted due to the model's small size and stable convergence.

This architecture provides a balance between expressive power and computational efficiency, making it suitable for a small-scale supervised smile detection task.

D. Autoencoder Architecture

The autoencoder (AE) employed in this project is a convolutional encoder–decoder model trained exclusively on the "non-smile" class, which is treated as the normal class. Its objective is to learn a compact latent representation of normal facial structures and reconstruct them with minimal error; deviations in reconstruction error are later used for anomaly detection (i.e., detecting smiles).

a) Encoder.: The encoder compresses the input image (64×64) into a low-dimensional latent representation. It consists of:

- a first convolutional layer with stride 2 that reduces spatial resolution from 64×64 to 32×32,
- a ReLU activation,
- a second strided convolution that further compresses the feature maps from 32×32 to 16×16,
- a ReLU activation.

This structure allows the network to encode edges, textures, and face geometry into a compact latent space while discarding high-frequency details.

b) Decoder.: The decoder mirrors the encoder by progressively expanding the spatial resolution back to the original size. It is composed of:

- a transposed convolution layer that upsamples the latent representation from 16×16 to 32×32,
- a ReLU activation,
- a second transposed convolution that restores the original 64×64 resolution and reconstructs the three RGB channels,
- a sigmoid activation to constrain pixel intensities to the $[0, 1]$ interval.

E. SVM with PCA

The SVM model represents a classical machine learning baseline built on vectorized image representations. Because Support Vector Machines cannot operate directly on high-dimensional raw images, each 64×64 RGB face was first flattened into a vector of $12{,}288$ features. These vectors were extracted from the PyTorch datasets using batched loaders to ensure memory-efficient conversion.

a) Dimensionality Reduction.: Before classification, a Principal Component Analysis (PCA) stage was applied to reduce the dimensionality of the input space. In all experiments, PCA was configured to retain 50 components. This step serves two purposes:

- it mitigates overfitting by discarding noisy or redundant pixel-level information,
- it accelerates training and inference by reducing the feature dimensionality by more than two orders of magnitude.

b) Classifier.: After PCA, a linear Support Vector Machine (LinearSVC) is trained to separate the two classes (smile / non-smile). The SVM uses:

- L_2 regularization with $C = 1.0$,
- a maximum of 5000 optimization iterations,
- standardized features (zero mean, unit variance) applied through a preprocessing pipeline.

This configuration was chosen because linear SVMs tend to perform well when PCA has already extracted the dominant directions of variance.

Overall, the SVM serves as a lightweight, interpretable baseline. Its performance reflects how well a linear boundary can separate the PCA-projected features, providing a useful contrast to the feature-learning capabilities of the CNN.

F. Model Evaluation Metrics

To enable a consistent comparison across the three learning approaches (CNN, Autoencoder, and SVM), a common set of quantitative metrics was computed. These metrics capture model complexity, computational cost, and predictive performance.

a) Trainable Parameters.: For the CNN and the Autoencoder, the number of trainable parameters was computed by summing all elements in tensors that require gradients. This metric reflects model capacity and helps explain differences in generalization and overfitting behavior. The SVM, being a classical ML model implemented through scikit-learn, does not expose trainable parameters under the same convention, and is therefore reported as having zero learnable parameters.

b) Floating-Point Operations (FLOPs).: The computational cost of a single forward pass was estimated using the `thop` profiler. FLOPs quantify how many arithmetic operations the model performs on one input image of size $3 \times 64 \times 64$. Only the CNN and Autoencoder were profiled, since the SVM pipeline (PCA + linear classifier) does not follow the convolutional computation paradigm and reports FLOPs as zero for consistency with deep learning metrics.

c) Inference Time.: Inference latency was measured by repeatedly evaluating each model on a batch of images. After a warm-up phase, the mean and standard deviation of the forward-pass duration were computed across 100 runs. All measurements were performed with models in evaluation mode and without gradient tracking.

d) Training Time.: The training time for each model was tracked by recording the wall-clock time between the start and end of the training loop. For the Autoencoder, only the normal-class subset was used during training. The SVM training time corresponds to the full scikit-learn pipeline: standardization, PCA fitting, and linear SVM optimization.

e) Accuracy.: For the CNN and SVM, predictive performance was measured on the training, validation, and test splits. Accuracy (%) was reported for each partition to evaluate fit quality and generalization. The Autoencoder, being a one-class model trained only on non-smiling faces, is not evaluated using accuracy during training. Instead, once training was completed, a reconstruction-error threshold was established, and accuracy was computed on all splits under the anomaly-detection formulation (smile = anomaly).

f) Summary.: Together, these metrics allow a multi-dimensional comparison:

- **Capacity** (parameters),
- **Computational cost** (FLOPs and inference time),
- **Training efficiency** (wall-clock duration),
- **Generalization** (accuracy on validation and test).

This provides a unified evaluation framework across deep, unsupervised, and classical models.

III. Experiments and Results

This section details the experimental setup, data preprocessing, training procedures, and quantitative results obtained for the three evaluated models: a Convolutional Neural Network (CNN), an Autoencoder, and a Support Vector Machine (SVM) with PCA.

A. *Dataset Preparation*

We used a curated subset of the *Labeled Faces in the Wild (LFW)* dataset, consisting of 1,203 RGB facial images evenly split between the "Smile" (600) and "Non-Smile" (603) classes. All images were resized to 64×64 pixels and normalized to $[0, 1]$.

A stratified data split was performed to preserve class balance:

- **70%** training (842 images)
- **10%** validation (120 images)
- **20%** testing (241 images)

A fixed random seed (42) ensured reproducibility of all experiments.

B. *Training Configuration*

All neural models were implemented in PyTorch and trained on CPU. For each architecture, we tracked training accuracy, validation accuracy, test accuracy, training time, inference latency, parameter count, and FLOPs.

C. *Convolutional Neural Network (CNN)*

Architecture: The CNN consisted of two convolutional blocks with Batch Normalization and ReLU activations, each followed by MaxPooling. A fully connected classifier mapped the extracted features to two output classes. Dropout ($p = 0.65$) and L2 weight decay (10^{-3}) were applied to mitigate overfitting.

Training Procedure:

- Loss function: **CrossEntropyLoss**
- Optimizer: **Adam** with learning rate 1×10^{-3}
- Epochs: **13**
- Batch size: **32**

Training and validation accuracy were computed at every epoch.

CNN Results: The CNN achieved the best performance across all models:

- **Training accuracy:** 0.98
- **Validation accuracy:** 0.95
- **Test accuracy:** 0.9419
- **Training time:** 10.85 s
- **Inference latency:** 8.60 ms
- **Parameters:** 529,666
- **FLOPs:** 7.40M

These results show strong generalization and no evidence of overfitting.

D. *Autoencoder for Anomaly Detection*

Architecture: The autoencoder consisted of a two-layer convolutional encoder that reduced spatial resolution, followed by a symmetric transposed-convolution decoder restoring it back to 64×64.

Training Strategy: Unlike the CNN, the Autoencoder was trained as a **one-class model**:

- Training was performed **only on Non-Smile images**.
- The objective was to detect "Smile" images as anomalies via reconstruction error.

Training Details:

- Loss: **Mean Squared Error (MSE)**
- Optimizer: **SGD** with learning rate 0.01
- Epochs: **20**
- Threshold: mean + 2 standard deviations of training reconstruction error

Autoencoder Results: The autoencoder failed to learn a discriminative reconstruction pattern for smile-related features:

- **Training accuracy:** 0.496
- **Validation accuracy:** 0.508
- **Test accuracy:** 0.485

- **Training time:** 9.16 s
- **Inference latency:** 7.43 ms
- **Parameters:** 38,723
- **FLOPs:** 28.0M

Its performance was close to random guessing, indicating that global reconstruction error is insufficient to capture subtle local features such as mouth curvature.

E. Support Vector Machine with PCA

Feature Extraction: Each 64×64 RGB image (12,288 features) was flattened and processed using PCA:

- **50 principal components** retained
- Standardization applied before PCA

Training Details:

- Classifier: **Linear SVM (LinearSVC)**
- Regularization parameter: $C = 1.0$
- Maximum iterations: 5,000

SVM Results: The SVM provided an excellent accuracy–efficiency trade-off:

- **Training accuracy:** 0.9157
- **Validation accuracy:** 0.8583
- **Test accuracy:** 0.8714
- **Training time:** 0.19 s
- **Inference latency:** 1.69 ms

The model trained extremely fast and required negligible computation at inference time.

F. Comparative Summary

Table I summarizes the performance and efficiency of all models.

Model	Test Acc.	Train Time (s)	Latency (ms)	Params
CNN	94.1%	10.85	8.59	529,666
SVM + PCA	87.1%	0.19	1.69	0
Autoencoder	48.5%	9.16	7.14	38,723

TABLE I: Performance comparison across models.

G. Brief Discussion

The CNN achieved the highest accuracy by effectively learning spatial facial features. The SVM offered the best efficiency-to-accuracy ratio, making it suitable for low- resource applications. The Autoencoder underperformed due to the subtlety of smile features, which are not well captured by reconstruction-based anomaly detection.

IV. Discussion

The three evaluated models exhibit distinct strengths and weaknesses across accuracy, computational efficiency, and robustness. Their behaviors reflect the fundamental differences between deep learning, classical machine learning, and anomaly detection approaches.

A. CNN: Highest Predictive Power at Moderate Cost

The CNN consistently delivered the best test accuracy (94.1%), confirming the effectiveness of convolutional architectures for capturing spatial patterns in facial expressions. Its hierarchical feature extraction enables the model to detect subtle geometric variations around the mouth and cheeks that distinguish smiles from neutral expressions.

In terms of tradeoffs, the CNN required considerably longer training time than the SVM, and its inference latency was several milliseconds slower. Nevertheless, its parameter count (529k) and FLOPs (7.40M) remain relatively lightweight for a neural model, making it suitable for deployment on resource-limited devices if accuracy is the primary objective.

B. SVM with PCA: Best Efficiency–Accuracy Tradeoff

The SVM achieved strong performance (87.1% test accuracy) at a fraction of the computational cost of the CNN. Both training and inference were extremely fast, and the PCA step reduced the dimensionality from 12,288 raw pixels to 50 features, effectively mitigating overfitting while maintaining discriminability.

The main tradeoff is that SVMs lack the expressive power to model complex spatial relationships. Unlike convolutional filters, PCA components do not explicitly encode local patterns, limiting the classifier's ability to capture fine-grained facial cues. As a result, the method cannot reach CNN-level accuracy. However, when computational efficiency or low-latency inference is the main constraint, the SVM offers the most balanced solution.

C. Autoencoder: Limitations of Reconstruction-Based Detection

The Autoencoder severely underperformed, achieving approximately 48.5% test accuracy—essentially random guessing. Since it was trained exclusively on "Non-Smile" images, successful classification relied on the assumption that "Smile" images exhibit significantly higher reconstruction error. However, smile-related features (such as lip curvature) occupy a small and localized portion of the face. Reconstruction-based anomaly detection tends to capture global structure rather than subtle local variations, causing both classes to produce similar reconstruction errors.

Although the Autoencoder had the smallest parameter count after PCA+SVM, its FLOPs were significantly higher due to the decoder's upsampling operations. This makes it both less accurate and less efficient than the alternatives for this task.

D. Overall Comparison

If accuracy is prioritized, the CNN is clearly the best model. If computational resources or inference speed are the main concern, the SVM with PCA provides the optimal tradeoff. The Autoencoder, while conceptually attractive for unsupervised detection, is not suitable for distinguishing high-level semantic attributes like “smile” vs. “non-smile” in this dataset.

Overall, the three paradigms highlight a common tradeoff pattern: *deep models achieve superior accuracy at moderate computational costs, classical models achieve reasonable accuracy with excellent efficiency, and reconstruction-based anomaly detection is ineffective for tasks requiring localized discriminative features.*

V. Conclusions

This study compared three modeling paradigms—Convolutional Neural Networks, Support Vector Machines with PCA, and Autoencoder-based anomaly detection—for the task of smile classification using a curated subset of the LFW dataset. The results demonstrate that each method offers distinct advantages depending on the evaluation criteria.

The CNN achieved the highest predictive accuracy, confirming its ability to learn spatially meaningful facial representations. Its performance indicates that deep feature extraction is highly effective for distinguishing subtle differences in facial expressions. Although its training time and inference latency are higher than those of the SVM, these costs remain moderate and well within practical deployment limits.

The SVM with PCA achieved lower accuracy but excelled in computational efficiency. It required minimal training time, negligible inference cost, and no learnable parameters. This makes it a strong candidate for scenarios where computational resources, power consumption, or real-time constraints dominate the design requirements.

The Autoencoder approach underperformed, highlighting the limitations of reconstruction-based anomaly detection for localized facial attributes. Its inability to discriminate smiles stems from the fact that subtle mouth-region changes contribute only marginally to global reconstruction error, making the method unsuitable for this classification problem.

Overall, the CNN emerges as the best-performing model in terms of accuracy, while the SVM offers the best tradeoff between speed and performance. The Autoencoder, despite its conceptual appeal, is not recommended for tasks involving fine-grained semantic distinctions such as smile detection.

References

[1] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller, “Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments,” University of Massachusetts, Amherst, Technical Report 07-49, 2007.

[2] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-Based Learning Applied to Document Recognition,” *Proceedings of the IEEE*, 1998.

[3] C. Cortes and V. Vapnik, “Support-Vector Networks,” *Machine Learning*, vol. 20, pp. 273–297, 1995.