Two approaches for Car Make Verification

Annalisa Gallina, Enrico Gottardis, Elisa Silene Panozzo

***Abstract*—This study addresses the challenging task of Car Make Verification (CMV), which involves determining whether two vehicles belong to the same manufacturer. The problem is approached using a Siamese network, designed specifically for verification tasks, and exploiting a Support Vector Machine (SVM) technique, operating on feature differences extracted by a pre-trained classification network.The dataset used is CompCars that consists of cars' images. It was preprocessed and adapted for training and testing the verification models. Experimental results show that the Siamese network is not able to achieve excellent performances. Meanwhile, the SVM approach demonstrates that exploiting features from a pre-trained classification network is a great alternative, leading to better results. This study underscores the importance of verification-specific architectures and provides valuable insights for future research into efficient and accurate methods for CMV.**

I. Introduction

The breakthroughs in the field of Computer Vision have paved the way for numerous innovative applications across various domains, enabling tasks that were once considered complex or infeasible. Among these, applications in the automotive sector are of particular significance, given the fundamental role of cars in modern society. In this context, car make verification emerges as a powerful tool, addressing a range of challenges with practical and impactful solutions. Cars' make verification (CMV) is the process of recognizing whether two cars have the same make, i.e. they belong to the same class. In order to do so, the images of two cars are compared to determine if they have similar features that allow to conclude they are produced by the same manufacturer.

In general, when dealing with a dataset of images, there are two main tasks that can be faced: classification and verification. Classification deals with the problem of identifying to which class a certain image belongs from a predefined set of categories: that is, one-to-many matching. On the other hand, verification analyzes a pair of images to check whether they belong to the same class, so it is a one-to-one matching, that is whether the pair is recognized or not. Usually, the verification task is more difficult than the identification because a global predefined threshold is required to make a decision that discriminates between the subjects. Even though the image verification is a harder task, it is becoming increasingly important due to the widespread use of images: the rise of image manipulation tools (e.g., *deepfake* generation), concerns about misinformation, and the growing importance of secure identity verification are just some factors that remark the need for image verification.

Among the image verification applications, it is possible to find the CMV: it plays a crucial role in sorting and categorizing large car images datasets, which may be useful for statistical analysis and systematic management of data. The automotive sector can significantly benefit from advancements in CMV, obtained through the use of Machine Learning (ML) and Deep Learning (DL) tools, such as improved operational efficiency and enhanced automation capabilities.

Despite its importance, CMV is still a difficult task to address and this is caused by various issues, but mainly due to the presence of a wide range of car manufacturers: this requires that the designed model learns a greater variety of features, as the larger number of car brands increases the likelihood that distinct vehicles share similar characteristics and makes it more challenging for the model to accurately determine whether two cars are produced by the same manufacturer. Furthermore, the scarcity of prior research on CMV adds to the complexity of the task, as there are no existing DL models already trained specifically for this purpose. Although the specific problem of CMV has not been extensively explored in the literature, the broader field of image verification has seen significant development, particularly due to its applications in security. Verifying a user's identity before granting them permissions is critical and neural networks are powerful tools to handle the problem, having proven to be suitable for these tasks due to their ability to learn complex patterns from large datasets. This leads to considerable progress in face recognition technologies [1], [2] and, similarly, in other areas like signature verification [3] and fingerprint identification [4].

The lack of work on the CMV task led us to investigate the problem and possible solutions. In order to do so, we opted for two different methods: combining image classification, using a pretrained model, with a Support Vector Machine (SVM) classifier for the first approach and a Siamese Neural Network for the second approach. These approaches will be investigated in the next sections of the paper, remarking the difference between the two solutions and suggesting improvements.

The outcome of this research has the potential to advance the field of the CMV, an area that will likely develop in the near future. Furthermore, the results obtained can be applied for future research in similar topics, as Car Model Verification, which not only checks the make but also the model.

II. Related Work

Despite the fact that the CMV problem has not been extensively explored in the literature, there are a lot of research papers in the field of image classification and verification. We mainly cover related works exploiting Support Vector Machines and Siamese Networks.

The combination of Convolutional Neural Network (CNN) with the SVM model appears to be an efficient approach as far as classification problems are concerned. In [3] to achieve an

efficient signature verification method, the extracted features from a signatures' dataset are supplied to the SVM classifier, which finds the best decision boundary, a straight line or a hyperplane depending on the number of parameters involved, that can separate the data points into classes. This type of approach leads to the best accuracy of 93.63% and further studies concerning this promising state of the art approach are suggested.

To demonstrate the effectiveness of this hybrid model, [5] proposes to use it for recognition of handwritten digits taken from MNIST dataset and achieving a recognition accuracy of 99.28%. The proposed approach consists on passing the automatically generated features to the SVM module for training and testing the handwritten digit dataset.

In a similar way, in [6], the classification model, obtained training the network on a portion of the dataset, is adopted as a feature extractor of the car images, and then either a SVM or a Joint Bayesian model is applied to train a verification model: this combination has been proven successful for CMV.

Although this is a very efficient approach, the most common strategy for solving verification tasks is to use Siamese Networks. There are several examples of applications concerning the use of this type of network. For instance in [1] and in [2] it is used for face recognition demonstrating very good performances compared to other methods while in [7], Siamese Network is proposed as a novel method to achieve end-to-end palmprint recognition. Another example is [8], where the authors propose a "*Siamese Network to predict the authenticity of test signatures discerning genuine ones from fraudulent ones*". The paper shows that the network coupled with contrastive loss provides a powerful framework for learning image similarity in a data-driven manner, enabling them to learn meaningful representations that can be used for various tasks like image or text similarity and face verification.

Moreover, Siamese Networks are effective for "few-shot learning," particularly "one-shot learning," where only one example is available [9]. Traditional CNN training in such cases often produces suboptimal results, whereas Siamese Networks excel when labeled data is limited or expensive to obtain. These networks extract meaningful representations from minimal examples, enabling accurate predictions for new, unseen samples. This makes them valuable for applications like face recognition [10] and signature verification [11], where even a single example can suffice for reliable classification.

III. Processing Pipeline

Our goal is to present various approaches to solve the CMV problem. We focused mainly on two different alternatives:

- A model trained for classification followed by a SVM.
- A Siamese network with a threshold on the euclidean distance between the two feature representations.

A. Classification + SVM

Inspired by the method outlined in the reference paper [6] we consider a classification model trained on a portion of the same dataset as a feature extractor for car images. Then apply a SVM to the difference between the two feature vectors to determine whether the two images have the same make or not. A summary scheme of the process is shown in Figure 1.

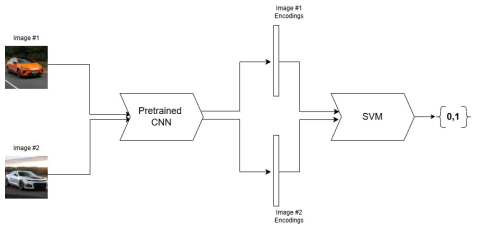


Fig. 1: Classification Network + SVM diagram

The rationale is that a well-trained classification model typically learns relevant features characterizing the input image. Thus, two images of the same car make are typically characterized by similar features while those belonging to different classes should exhibit different representations. This suggests that it should be possible to perform verification by analyzing the difference between their feature representations.

A straightforward method might involve setting a threshold on the Euclidean distance between the two feature vectors. However, this approach may not be enough: since the classification network is not explicitly trained to maximize inter-class distances, a single threshold might not work in all cases. For this reason, we opt for a SVM, as it is better suited to handle the complexity of the feature space. Unlike a simple threshold-based approach, a SVM can learn a decision boundary that separates pairs of feature differences corresponding to images of the same make from those of different makes.

To tackle classification, we consider two alternatives as well: we first define a new model starting from scratch and then we try with the fine-tuning of a pretrained model. The two approaches are implemented within a shared framework that can be broken into three main stages: data pre-processing, feature extraction, and classification.

The first stage is similar for both approaches and it will be further investigated in section IV. The main difference is that to tailor the increasing presence of overfitting in the custom network stronger transformations are applied to the data.

For the feature extraction stage, in the first case a neural network designed specifically for this task is trained from the ground up. Its architecture, detailed in Section V, follows common practices in the literature for image classification tasks. In the second approach, a pre-trained network is used to leverage features learned from a large, generic dataset . This method builds on the assumption that many features learned from such a dataset are transferable to any classification task. Then the network is fine-tuned using the target dataset to learn task-specific features. Since the model has already learnt useful features from the pre-trained dataset, it typically converges faster and allows to reach better performance.

B. Siamese Network

The second approach involves the usage of a Siamese Network, which is a pair-based neural network consisting

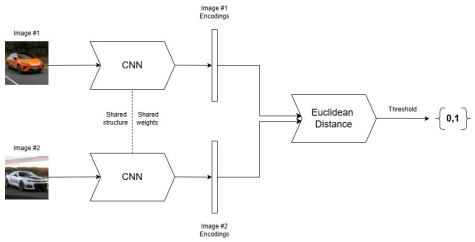


Fig. 2: Siamese network diagram

of two identical branches that share the same structure. A summary description of the process is outlined in Figure 2.

The goal of this structure is to learn optimal feature representations of the input pairs where matched images are pulled closer and unmatched images are pushed far away. Unlike traditional methods focusing on deciphering image content, it concentrates on variations and resemblances among images.

While the Siamese network outputs feature representations tailored to the verification task, it does not directly perform the verification itself. Instead, a method must be defined to determine whether the two images belong to the same make based on their feature representations. A common approach in the literature involves setting a threshold on the Euclidean distance between the two feature vectors. If the distance is below the threshold, the pair is classified as belonging to the same make; otherwise, they are classified as different. Although this approach aligns with the training process of the network, it may become less effective when dealing with high-dimensional feature vectors, where the Euclidean distance may not be able to identify similar pairs. To address this, two alternative strategies are considered. First, we enforce the feature vectors to have unit norm, thereby making the distance influenced only by the angular separation between vectors rather than their magnitudes. We also tried to consider more advanced separation methods that go beyond relying solely on the Euclidean distance and instead leverage the full feature representations of the vectors: an SVM and a decision tree were trained using as input the difference between the feature representations of the two images, similarly to what is previously done with the outputs of the classification model. However, the results obtained were not satisfying and we decided to concentrate on the other methods.

IV. Dataset

To achieve our goal, we use the CompCars dataset that has been introduced in [6]. This dataset comprises two distinct scenarios: we focus solely on the web-nature subset, as it offers higher-quality images. It consists of 136,727 images spanning 163 car makes and 1,716 car models.

The dataset is partitioned into three subsets:

1) A set for classification tasks.
2) A set for verification model training.
3) A set for creating image pairs (categorized as easy, medium, and hard) for testing verification models.

We keep the provided partitions for the classification task, using the training and test sets as-is, while, the original training set for the verification task proved insufficient for training a Siamese Network. To address this, we repurpose the third set by considering two subsets of the same size, for training and testing the verification model.

For both the tasks, we further subdivide the test set into a validation set (60%) and a final test set (40%).

To prepare the data for use, all images are resized to 256x256 pixels, and only the central crop is retained, resulting in input images of size 224x224 pixels. Moreover, various random transformations are applied to avoid overfitting, with the specific transformations tailored to each task and model. These may include operations such as random rotations or horizontal flips. This preprocessing ensures consistency and robustness in training and evaluation.

V. Learning Framework

In this section we will describe more in detail the models and the parameters for the two presented approaches.

A. Classification + SVM

For the classification task we exploit two networks: first, we employ a ResNet18 convolutional neural network as the backbone for feature extraction and classification. ResNet18, a residual network with 18 layers, leverages skip connections to mitigate the vanishing gradient problem and facilitate the training of deeper networks (Figure 3). To accelerate convergence and benefit from robust feature representations, we initialize the model with weights pre-trained on the ImageNet dataset [12]. The pre-trained network serves as a feature extractor, leveraging its learned hierarchical representations to capture image features. We fine-tune the model on our domain-specific dataset, enabling it to adapt to the task while retaining general features learned from ImageNet.

We considered the model that is provided by the Pytorch library. Then for the fine-tuning process we choose SGD as optimizer with a learning rate of 10^{-4}. We prefer not to consider a higher rate even if it may speed up the updates because fine tuning requires a careful balance between updating the weights and preserving the learned representations.

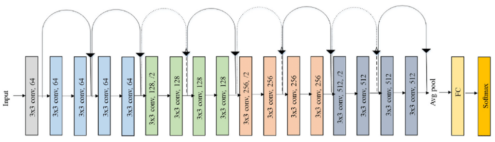


Fig. 3: Building blocks of the ResNet18 provided by PyTorch

For the loss function we chose the Cross Entropy Loss, one of the most common alternatives in the literature which, in case of discrete probability distributions is defined as:

$$H(p, q) = -\sum_{x} p(x) \log q(x)$$

$p(x)$ is the true probability distribution (1 for the true class) and $q(x)$ represents the output of the network, i.e. the estimated probability of the input belonging to each class. In

practice, a higher predicted probability results in a smaller loss, so greater confidence in the correct prediction.

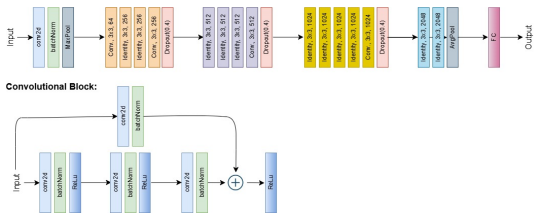


Fig. 4: Building blocks of the model designed from scratch

As an alternative, we tried to build a neural network from scratch: the designed network incorporates key components of ResNet, including identity and convolutional blocks that facilitate efficient training of deep models by mitigating the vanishing gradient problem. Figure 4 shows the building blocks of the designed model.

After an initial step, which mainly includes a convolutional layer of 64 filters, the network architecture is divided into four stages, each comprising multiple residual blocks. At the beginning of each stage, convolutional blocks downsample the feature maps using a stride of 2, while identity blocks maintain spatial resolution within the stages. This hierarchical structure progressively expands the feature map size from 64 to 2048. To improve generalization, a dropout layer with a rate of 0.4 is applied after each stage aiming to address a significant overfitting issue. The drawback is that such a high dropout rate introduces a lot of noise into the learning process, as a large number of neurons are "dropped" at each epoch.

Finally, a fully connected layer is then applied to project the feature representation into $N = 75$ output classes, corresponding to the number of target categories.

The model was trained using the same framework as the fine-tuned network, with the only difference being the choice of optimizer: Adam, with a learning rate of 10^{-3}.

As described in Section III, the classification network was exploited as a feature extractor. The difference between the feature vectors of a pair of images is used as input to a SVM: we considered a RBF kernel, which is effective at modeling complex relationships and at generalizing well to new data.

B. Siamese network

The second proposed model is a Siamese Network: the architecture consists of two identical sub-networks, each employing a modified ResNet-18 backbone to extract feature embeddings from input images. We start from the network provided by Pytorch and to focus on feature extraction, the final fully connected layer is removed: this encodes each input image into a 512-dimensional feature vector.

Then, the network includes a fully connected projection head that reduces the feature vector to a lower-dimensional embedding of size 128. It consists of two linear layers, separated by a ReLU activation function, to introduce non-linearity. The final vectors are normalized using ℓ_2-normalization: this facilitates distance-based comparisons by ensuring that magnitudes do not bias similarity measures.

This architecture processes input pairs through the two sub-networks, generating embeddings for each image. These embeddings can be used with similarity-based loss functions, which optimize the model to minimize the distance between vectors of similar pairs and maximize it for dissimilar pairs.

We train the network using Contrastive Loss, defined as:

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

where y is the ground truth label ($y = 0$ for similar pairs, $y = 1$ for dissimilar ones), d represents the Euclidean distance between the two representations and m is a margin that defines how far apart dissimilar image representations should be.

For optimization, we employ the SGD optimizer with a learning rate of 10^{-4}. The margin m is set to 1.5. These hyperparameters were determined through an exploration of the hyperparameter space, considering learning rates in the range $[10^{-4}, 10^{-2}]$ and margins in the range $[1, 4]$. These intervals were chosen based on common practices in the literature. The final configuration was selected based on validation performance, achieving an optimal balance between convergence speed and representation quality.

The threshold used to establish similarity is determined empirically based on the results for the validation set. We began with an initial value of 1.5 and adjusted it iteratively.

VI. Results

In this section we are going to analyze and compare the results obtained with the two proposed methods.

The project was developed using Colab framework, which introduced some limitations that impacted the results. Colab's GPU limit and random disconnections disrupted training, forcing us to save models only when validation performance improved. This caused training to sometimes resume from earlier states, leading to irregularities in the loss curve.

A. Classification with SVM

As described in Section III, we evaluated two distinct models: the fine-tuned model was trained for 1523 epochs, while the custom-designed network was trained for 857 epochs. In the first case, training was ended when accuracy showed negligible improvement, indicating convergence. For the custom network, it stopped after no new model was saved for more than 50 epochs, as the validation performance plateaued. Although extending the training for additional epochs might have yielded better results - particularly for the second model - it was unnecessary for the scope of this study.

Table 1 presents the accuracy achieved by the two models. As anticipated, the fine-tuned model outperforms the other network. This is partly due to its longer training duration and partly because the pretrained model benefits from inherited weightsand so it requires only task-specific fine-tuning.

However, it is worth noticing that the custom-designed network also performs remarkably well. Considering that this

TABLE 1: Accuracy for classification task

	Pretrained Model		Model from Scratch	
Metric	Train	Test	Train	Test
Top 1	0.9917	0.7010	0.4801	0.4824
Top 5	0.9989	0.8910	0.7550	0.7519
Top 10	0.9995	0.9428	0.8671	0.8663

classification task involves 75 classes, achieving an accuracy of 50% is far superior to random guessing.

Another interesting observation is the difference in the training and test performance between the two models. For the custom-designed network, the training and test accuracies are nearly identical, which can be attributed to the high dropout rate employed during training.

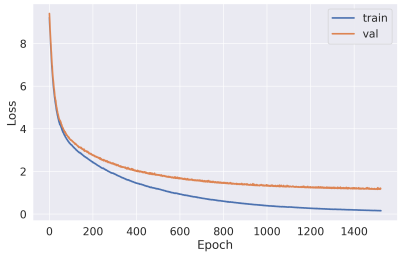


Fig. 5: Loss curves for the pretrained model

Figure 5 illustrates the loss curves for the pretrained network. Initially, the loss curve for the training and validation sets are nearly identical, indicating consistent learning across both sets. However, in the later stages, the validation loss curve flattens while the training loss continues to decrease. This divergence suggests that it is an optimal point to stop training to prevent overfitting. Additionally, the accuracy curve shows signs of saturation during the final stages, further reinforcing the decision to conclude training at this point.

However, much more interesting is Figure 6, which shows the loss curve for the custom network: it is possible to notice that, while the training loss has an almost regular decreasing behaviour, the corresponding curve for the validation set is highly noisy. This irregularity arises from the high dropout rate applied during training, which randomly deactivates a substantial number of neurons at each epoch. This dropout mechanism, while effective for regularization, adds complexity to the training process, making it more challenging for the network to converge to an optimal state.

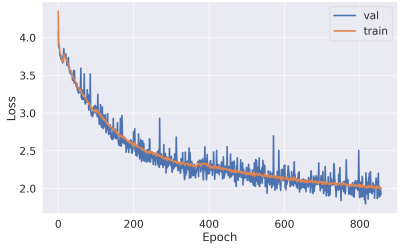


Fig. 6: Loss curve for the custom network

Finally, since the fine-tuned network showed better performance, we exploited it as a feature extractor for the SVM. The resulting system achieved a maximum accuracy of 78.57% on the verification task, with a False Positive Rate of 20% and a False Negative Rate of 22%.

The interesting thing is that the best results for the verification task were not achieved using the optimal weights for the classification model. This suggests that as long as the network performs reasonably well, the results for the verification task remain relatively stable.

B. Siamese network

For the Siamese Network approach, we evaluated two different alternatives: using black and white images or using colored images. In the former, the model was trained for 400 epochs and stopped because the improvement on the loss function was becoming almost negligible. For the latter instead, the model was trained for more than 500 epochs and again training was stopped for the same reason.

TABLE 2: Accuracy for verification task

	B&W images		Colored images	
Metric	Train	Test	Train	Test
Positive acc.	0.6734	0.5941	0.7377	0.6440
Negative acc.	0.7162	0.6939	0.7440	0.7193
Overall acc.	0.6946	0.6389	0.7429	0.6803

Table 2 shows the accuracy achieved by the Siamese network. As expected, the performance for the training set is better than for the test set but, in both cases, we can notice that the values are not astounding: the reason is intrinsic with the difficulty of the task. As a matter of fact, the high number of makes, made it very challenging for the network to learn features and distinguish between pair of images with the same car manufacturer from images with different car makes. However, it is easy to notice that the model performs better with colored images: this means that color plays an important role in the verification task for a pair of images.

Understanding the reason behind such results lead to reviewing the model multiple times. Before coming to the possible reason explained above, many options were examined, leading nowhere. We also decided to plot the predictions made by the model regarding the brands: the idea was that the network not only learns to distinguish pairs of images with the same car make, but it will also learn a representation of the car manufacturer. In order to show this idea, we exploited Uniform Manifold Approximation and Projection (UMAP), which made it possible to represent 128-dimensional vectors in a bidimensional space. Fig 7 efficiently displays the situation: mainly due to dimensionality reduction, the brands are not well clustered. This suggests that the model is not achieving great results in learning the representation of the car makes, which leads to issues also in the image verification.

As regards the loss curves instead, it is possible to notice from Fig 8a there were some issues at approximately epoch 100, whose cause is mainly to be searched in the problems related to Colab. Apart from this, the loss curves continue decreasing up to the last epoch: this means that a higher number of epochs may lead to more refined results.

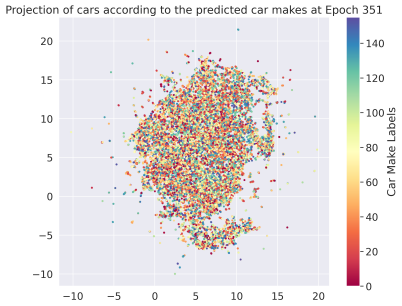
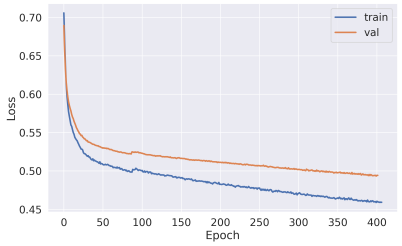
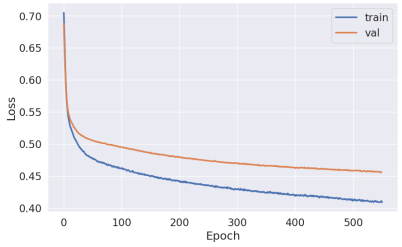


Fig. 7: UMAP representation of predicted cars at epoch 351



(a) Loss curve for B&W images



(b) Loss curve for colored images

Fig. 8: Loss curves for Siamese Network

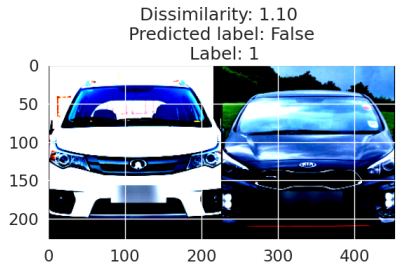


Fig. 9: Example of model prediction for colored images

Finally, Fig 9 shows how the model works: after learning the representation of the image using a 128-dimensional vector, we use the Euclidean distance to determine how much dissimilar the image is with respect to another image. If the dissimilarity value is below a threshold, then the images are considered to contain a car of the same brand. These images are taken from the train set and it is possible to see that the prediction of the model is consistent with the ground truth.

VII. Concluding Remarks

With this paper, we have presented multiple ways to approach the CMV, a task which is still understudied in the literature. As explained above, there are many factors that make the CMV a challenging task and this can be seen by the performances of the model not being exceptional.

The two proposed methods are both valid, however the results show that one is to be preferred: the first approach, which involves performing a classification of the cars and then applying SVM, is able to obtain more than 10% higher accuracy with respect to the Siamese network.

It is challenging to identify the reasons for the poor performance of the Siamese Network, especially considering that the approach employed aligns with methodologies previously established in the literature for addressing similar problems. One possible explanation could be the high dimensionality of the output space. In a 128-dimensional space, the Euclidean distance may struggle to adequately capture the complexity of the task. For future research, it would be valuable to investigate how the results vary when the output is mapped to a lower-dimensional space or with different hyperparameters.

In conclusion, it is worth to remark the complexity of the task. When conidering a simpler task, such as classification, the results differ a lot. Indeed, as reported earlier, the performance obtained on the classification task is highly impressive.

References

[1] H. Wu, Z. Xu, J. Zhang, W. Yan, and X. Ma, "Face recognition based on convolution siamese networks," in *2017 10th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*, pp. 1–5, 2017.

[2] C. R. Kumar, S. N, M. Priyadharshini, D. G. E, and K. R. M, "Face recognition using cnn and siamese network," *Measurement: Sensors*, vol. 27, p. 100800, 2023.

[3] S. B H, W. Abraham, and B. Pilar, "Offline signature verification using cnn and svm classifier," in *Conference: 2022 IEEE 7th International Conference on Recent Advances and Innovations in Engineering (ICRAIE)*, pp. 304–307, 12 2022.

[4] Z. Li, Y. Wang, Z. Yang, X. Tian, L. Zhai, X. Wu, J. Yu, S. Gu, L. Huang, and Y. Zhang, "A novel fingerprint recognition method based on a siamese neural network," *Journal of Intelligent Systems*, vol. 31, no. 1, pp. 690–705, 2022.

[5] S. Ahlawat and A. Choudhary, "Hybrid cnn-svm classifier for handwritten digit recognition," *Procedia Computer Science*, vol. 167, pp. 2554–2560, 2020. International Conference on Computational Intelligence and Data Science.

[6] L. Yang, P. Luo, C. C. Loy, and X. Tang, "A large-scale car dataset for fine-grained categorization and verification," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3973–3981, 2015.

[7] D. Zhong, Y. Yang, and X. Du, "Palmprint recognition using siamese network," in *Biometric Recognition* (J. Zhou, Y. Wang, Z. Sun, Z. Jia, J. Feng, S. Shan, K. Ubul, and Z. Guo, eds.), (Cham), pp. 48–55, Springer International Publishing, 2018.

[8] Analytics Vidhya, "Introduction and implementation of siamese networks," 2023. Accessed: 2025-01-02.

[9] G. Koch, R. Zemel, R. Salakhutdinov, *et al.*, "Siamese neural networks for one-shot image recognition," in *ICML deep learning workshop*, vol. 2, pp. 1–30, Lille, 2015.

[10] K. V. Kumar, K. A. Teja, R. T. Bhargav, V. Satpute, C. Naveen, and V. Kamble, "One-shot face recognition," in *2023 2nd International Conference on Paradigm Shifts in Communications Embedded Systems, Machine Learning and Signal Processing (PCEMS)*, pp. 1–6, 2023.

[11] M. V. Arisoy, "Signature verification using siamese neural network one-shot learning," *International Journal of Engineering and Innovative Research*, vol. 3, pp. 248–260, 2021.

[12] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009.