Real-Time Goat Face Recognition Using Convolutional Neural Networks: A Replication and Comparative Study

Ifeanyi Nwaneri
Michigan State University
nwanerii@msu.edu

Grace Akintan
Michigan State University
akintano@msu.edu

Soumadeep Pal
Michigan State University
palsoum1@msu.edu

Abstract

This project investigates real-time goat face recognition using convolutional neural networks. We replicate the facial detection and identification pipeline proposed by Billah et al. and evaluate its performance on the publicly available Saanen goat dataset. In addition to replicating the baseline results, we implement and compare alternative modern architectures for both face detection and individual identification. Our analysis highlights the strengths and limitations of the original pipeline and examines whether newer models offer improvements in accuracy, speed, and robustness. The outcomes form the foundation for a complete project report and guide future extensions in precision livestock monitoring.

1. Introduction

The global livestock industry is at a critical juncture, where demands for scale, efficiency, food safety, and animal welfare are challenging traditional husbandry practices. A key step in this evolution is addressing the problem of individual animal identification. This is a fundamental requirement for shifting from herd-level management to precise, individual-level care. Conventional livestock identification methods, including those for goats, have traditionally relied on physical marking techniques such as ear tags, tattoos, paint brands, and ear notching. While widely used, these methods suffer from durability issues, risk of misreading, and negative impacts on animal welfare [1].

Non-invasive biometric systems powered by computer vision and deep learning offer a superior alternative. Among these, the facial recognition pipeline proposed by Billah et al. for Saanen goats serves as a key benchmark. The goal of this project is to replicate their detection and identification pipeline for 10 Saanen goats and further conduct a comparative analysis using modern architectures. Through replication and extension, we aim to evaluate whether state-of-the-art models can improve performance and robustness in goat facial detection and recognition.

2. Image Data Curation and Preprocessing

The curated dataset consists of RGB goat-face images paired with YOLO-normalized annotation files using four classes: face (0), eye (1), nose or muzzle (2), and ear (3). Visualization tools were implemented to verify annotation consistency across the entire dataset.

An example annotated image is shown in Figure 1, while Figure 2 illustrates variations in pose and lighting conditions. A Python-based verification pipeline overlays bounding boxes, checks annotation bounds, and ensures a one-to-one mapping between images and their annotation files. Verified samples are stored in an `annotated/` directory to support reproducible data inspection.

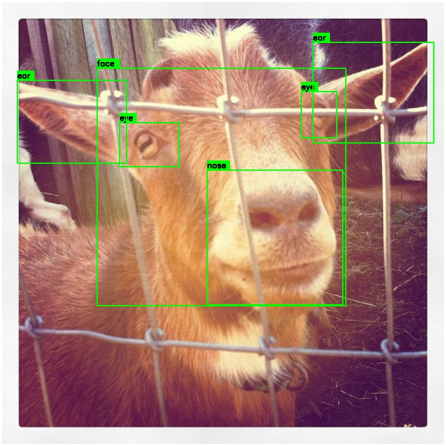


Figure 1. Annotated image showing bounding boxes for face, eye, muzzle, and ear components.

A Python-based verification pipeline overlays bounding boxes, checks annotation bounds, and ensures 1-to-1 mapping between images and their annotation files. Verified samples are stored in an `annotated/` directory to support reproducible data inspection.

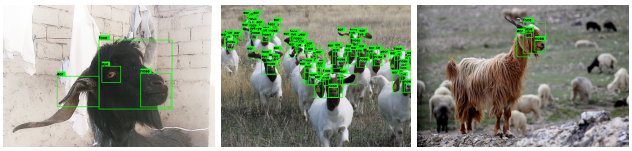


Figure 2. Multiple annotated examples showing consistency across individuals, poses, and illumination.

2.1. Data Augmentation

All externally applied augmentation in this project is used exclusively for the recognition component. YOLOv4 handles its own internal augmentation during detection training, and therefore no manual augmentations were applied to the detection dataset.

For recognition, we employ an alignment-based augmentation pipeline. The largest detected face and the two eye centers are extracted from each image. The eye centers define a rotation angle used to geometrically normalize the face. The rotated image is then cropped around the transformed face bounding box and resized to 64×64. Histogram equalization enhances local contrast, while bilateral filtering reduces noise without destroying structural identity cues.

After alignment, we apply controlled augmentations including small rotations up to $\pm 10°$, mild scaling and translations, photometric variations in brightness and contrast, and the addition of low-magnitude Gaussian noise. These transformations expand the diversity of the dataset while preserving the identity-relevant characteristics of each goat. Figure **??** summarizes the normalization and augmentation workflow.

3. Face Detection

Before recognition, an essential preprocessing step is face detection. We perform detection using a YOLOv4-based pipeline consistent with the original work. Below, we outline the architecture, data preparation procedures, augmentation approach, and loss formulation.

3.1. Model Architecture

The YOLOv4 model consists of a deep CNN backbone, a multiscale feature-fusion neck, and multiple prediction heads. In our implementation:

- **Backbone:** CSPDarknet53 [6]
- **Neck:** SPP [3], PAN [4]
- **Head:** YOLOv3 [5]

We initialize the model with pretrained YOLOv4 weights. The neck fuses hierarchical features, which is particularly effective for detecting small structures such as goat eyes and ears at varying scales.

The original YOLO head outputs predictions for 80 COCO classes. We modify the prediction layers to output 27 channels corresponding to $(4 + 1 + 4 \text{ classes}) \times 3$ detection anchors. Predictions occur at strides of 8, 16, and 32 pixels.

3.2. Bag of Freebies

YOLOv4 uses a combination of different data augmentation strategies called 'Bag of freebies' [2]. From these , we apply a set compositional augmentations. All transformations were applied jointly to images and bounding boxes.

Geometric Transformations Images were perturbed by a randomly sampled jittered crop. Horizontal and vertical offsets were drawn from uniform ranges, producing translations, mild scalings, and partial crops, which were resized to the original dimension using padding. Horizontal flipping was also applied with 50% probability.

Photometric Transformations We also apply a HSV-based color jitter, including random hue shifts and multiplicative saturation and brightness changes.

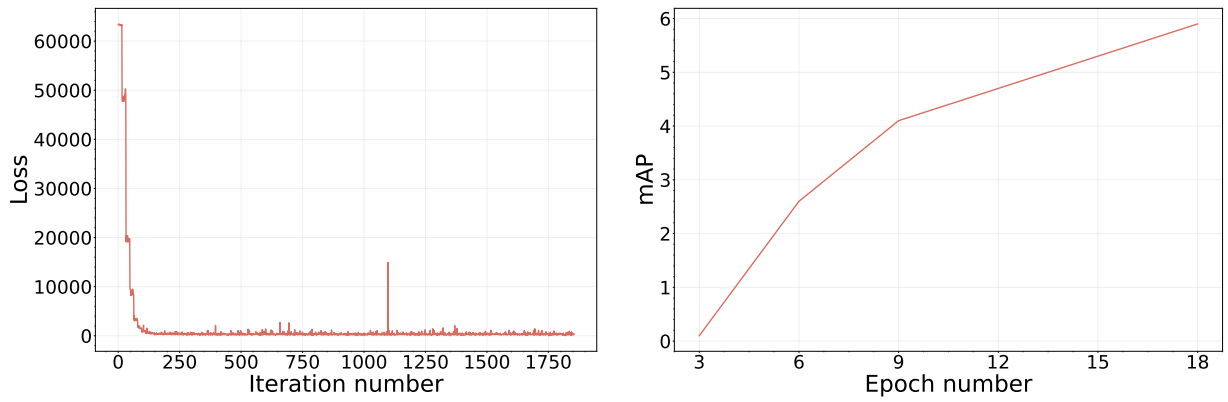
Compositional Augmentations Finally, a combination of Mixup and Mosaic transformations were applied. Mixup consists of blending two augmented images with equal weights and their annotations being merged. While mosaic takes four such images and combine them into a 2×2 layout using a randomly sampled cut point. Bounding boxes are translated to a new coordinate system correspondingly.

3.3. Loss Formulation

We use the YOLOv4 objective with multiple terms. The loss terms are based on the principles of (a) where an object is located, (b) whether an object is present (c) which class the object belongs to, which we explain below.

Bounding box regression. The YOLO model predicts bounding box estimates in terms offsets with respect to a discrete grid and an associated anchor box. The YOLO output is first decoded into the bounding box center coordinates and width and height values of the bounding box. The localization of the center coordinates are considered using a BCE loss , while the width and height predictions are localized using a L2 loss . Instead of using the MSE on the raw coordinates of the bounding box, this YOLO loss provides a more robust regression objective.

Objectness loss. Once a bounding box is predicted, there is a probability that no object resides inside this bounding box. To indicate this, the YOLO model also provides an objectness score as an output, which is used in conjuction with



(a) (b)

Figure 3. Results for YOLOv4 goat facial landmark detection (a) Training loss curves for 1750 iterations (b) mAP (%) for 24 epochs

the anchor box to compute a BCE loss, which indicates if the anchor - cell combination overlaps a ground truth object.

Classification objective. In addition to the bounding box regression objective, we also include a classification loss. For each bounding box output of the YOLO model, the model also provides 4 class probabilities as its output. We use the cross-entropy loss between these outputs and the one-hot encoding of the original class for this objective.

3.4. Training Optimization and Results

Using the above data augmentations and losses, we train the YOLO model for 24 epochs with a batch size of 4. We update model parameters using Adam optimizer with an initial learning rate of 5e-3. After every 16 steps, we use a scheduler to reduce the learning rate by multiplying it by 0.99. Using these training configurations, we show the loss curve for the mean training loss. As we observe in Figure 3 (a), the loss decreases successfully. We also plot the mean Average Precision (mAP) on the test set over epochs in Figure 3 (b). Inspite of successful loss reduction, we find that we are unable to achieve a high AP.

Discussion. The fact that the AP increases tells us that the model is definitely learning something regarding landmark detection. The low value may be attributed to training using a bad initial model. Even though we start with a pretrained YOLOv4 model, it is trained for 80 classes and we are using it from a third-party source. Another source of failure could be a lack of more hyperparameter tuning. However, even after using different optimizers like SGD with a variety of learning rates, we do not find much success in getting a high AP. Based on the observation of the training dynamics, we feel that having a scheduler which *increases* the learning rate later on in training may be a possible option to explore. We leave this part for future exploration.

4. Face Recognition

The goal of the recognition stage is to replicate and validate the Convolutional Neural Network (CNN) approach for identifying 10 individual Saanen goats, as described in Billah et al. [1]. The reference study reported a benchmark accuracy of 96.4%, achieved through a specialized preprocessing pipeline involving geometric alignment based on eye centroids and subsequent noise-reduction filtering. In our implementation, we reproduced this end-to-end pipeline; however, a key requirement for the alignment step is the visibility of both eyes in the input image. As a result, all images lacking full eye visibility were excluded. This constraint reduced the available dataset to approximately 650 images (522 for training and 131 for validation), representing roughly 43% of the 1,500 samples used in the original study. Despite this substantial reduction in data volume, our aim was to assess whether comparable recognition performance could still be achieved.

4.1. Model Implementation and Experimental Phases

To systematically evaluate the recognition pipeline under limited-data conditions, training was conducted in three phases, each targeting different architectural or optimization challenges:

Phase I: Baseline Custom CNN. We first implemented the 5-block CNN architecture described in the original paper, consisting of convolutional layers with 3×3 kernels, 2×2 max-pooling operations, and a final feature map of size $4 \times 4 \times 128$. When trained using the Adam optimizer, the model rapidly overfit the training set, achieving near-perfect accuracy while failing to generalize, with validation accuracy plateauing at approximately 81%. The divergence between training and validation loss suggests the model is

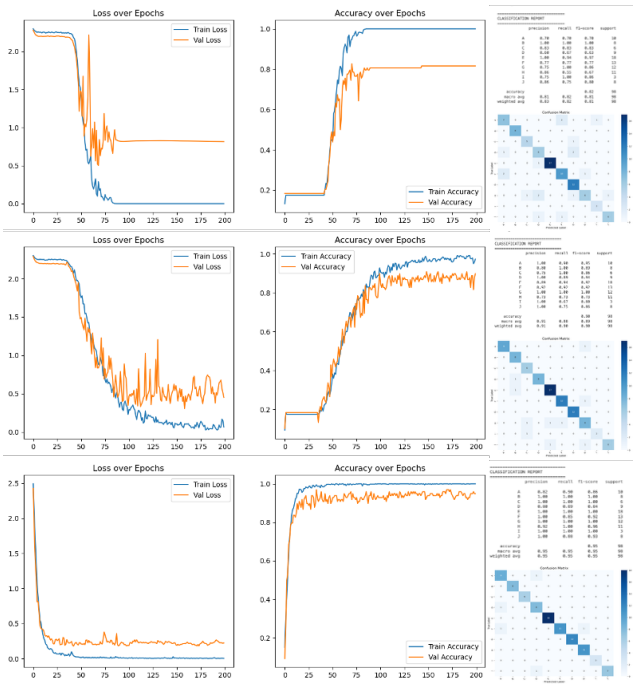


Figure 4. Training and evaluation summary for the three experimental phases...

unable to learn discriminative representations robustly under reduced data conditions.

Phase II: Regularization and Optimization Strategies. To address overfitting, several regularization techniques were introduced. The optimizer was switched from Adam to Stochastic Gradient Descent (SGD), and a 50% dropout rate was applied to the fully connected layers. Additionally, data augmentation—consisting of random rotations and horizontal flips—was incorporated to increase dataset diversity. These modifications substantially improved generalization, leading to a test accuracy of 91%.

Phase III: Transfer Learning Using ResNet18. Given the inherent limitations of training CNNs from scratch on small datasets, we implemented a transfer learning approach using ResNet18 pretrained on ImageNet. The first convolutional layer was adapted to accept single-channel grayscale inputs while retaining the pretrained weights. This configuration achieved the best performance among all experiments, reaching a test accuracy of 95%, demonstrating that feature reuse from large-scale datasets offers a significant advantage when training data is limited.

Results for the 3 phases are shown in Figures 4

5. Members Contribution

Ifeanyi Nwaneri: Project conceptualization and coordination. Implemented the complete preprocessing pipeline (alignment, cropping, and filtering) for facial recognition. Developed and trained both the custom baseline CNN and the pretrained ResNet18 architectures, conducting the comparative analysis between them.

Grace Akintan: Responsible for dataset acquisition and rigorous manual inspection to ensure data integrity. Facilitated the writing process and supported the overall coordination of the project workflow.

Soumadeep Pal : Implemented the full facial landmark detection pipeline. That includes implementing different components of YOLOv4 i.e. architecture, bag of freebies, loss formulation and training. Helped in project coordination.

References

[1] Mahadi Hasan Billah, Mohammad Rashedul Hoque, Ahmed Safa, Rajib Chakraborty, A A Mamun Khan, and Mohammad Shah Alam. Automatic goat face and landmarks detection and identity recognition using deep convolutional neural networks. *Computers and Electronics in Agriculture*, 194: 106761, 2022. 1, 3

[2] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020. 2

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 37(9):1904–1916, 2015. 2

[4] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8759–8768, 2018. 2

[5] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018. 2

[6] Chien-Yao Wang, Hong-Yuan Mark Liao, Yueh-Hua Wu, Ping-Yang Chen, Jun-Wei Hsieh, and I-Hau Yeh. Cspnet: A new backbone that can enhance learning capability of cnn. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 390–391, 2020. 2