04/29/2024 TEAM 10

Deep Learning for advanced facial recognition

Aditya Dhanotia
Anthony Lopez
Christian Ingul
Sharon Lu

DSO 569
PROFESSOR FAN

Abstract

This project explores applying deep learning techniques in facial recognition and bounding box prediction, utilizing a modified MobileNetV2 architecture for enhanced performance in real-time applications. By integrating the MobileNetV2 model with custom dense layers for classification and bounding box predictions, we aimed to develop a robust system capable of accurately identifying and locating faces within images. The dataset comprised diverse facial images divided into training, validation, and test sets to train and evaluate the model. Testing and real-time prediction results demonstrate the model's potential, achieving high accuracy and precision in classification and localization tasks. This report outlines the methods, experiments, and results obtained and comprehensively analyzes the project's relevance, usefulness, soundness, and clarity. The findings highlight the potential of deep learning in transforming facial recognition technologies and suggest methods for future enhancements.

1. Introduction and Project Objective

In DSO 569, we delved into deep learning principles, with a particular focus on convolutional neural networks (CNNs) and their applications in computer vision. CNNs are instrumental in processing image data and capturing hierarchical patterns, which are the backbone of facial recognition technology. Moreover, we learned advanced methodologies such as data augmentation, transfer learning, and fine-tuning pre-trained models like **VGG16**, which served as a foundation for creating the object detection system.

In this project, we extend our classroom learning by applying these concepts to create a facial recognition system to meet the growing demands for such technology in security, retail, user interface interactions, and workplace efficiency.

Our project focuses on introducing a novel facial detection system trained on primary image data with a scalable pipeline enabling real-time facial recognition. The system's feasibility can be extended across multiple use cases in fraud prevention, more responsive marketing using real-time emotion analysis, streamlining shopping with faster checkouts, and improving workplace efficiency with automated check-ins for small to medium-sized enterprises. By tackling these challenges, we aim to create a tool that’s not just technologically sophisticated but genuinely useful for everyday business operations, helping them run smoother and more securely.

This project has also laid the groundwork for future system scaling to meet the needs of large corporations and government entities. By leveraging the efficient and adaptable MobileNetV2 architecture, we've demonstrated that our model can handle more substantial computational demands and larger datasets. This scalability ensures that as organizations grow and their data and security requirements increase, our facial recognition system can be expanded and enhanced to meet these demands, making it a viable solution for high-stakes, large-scale applications.

2. Methodology

2-1. Data Description

The group crafted the dataset to address the requirements of facial recognition and bounding box prediction tasks. 99 original images were captured using OpenCV [1], a popular open-source computer vision library. These images encompass 33 images of Christian Ingul, Aditya Dhanotia, and Sharon Lu. Also, to enhance the robustness of our model, the dataset includes images where no individual is present, allowing the model to learn from both positive (face present) and negative (no face present) scenarios.

Each image was manually annotated using the LabelMe annotation tool, a versatile application that facilitates labeling images with object-bounding boxes and classification labels [2]. In our case, each face in the photos was labeled with the corresponding individual's name and marked with a drawn bounding box that specifies the face's location within the image.

To expand our dataset and introduce variability, we applied data augmentation techniques using the Albumentations library, a powerful image augmentation tool that integrates seamlessly with deep learning models [3].

The following transformations were applied to each image:

- Horizontal Flip (probability = 0.5): Mirrors the image along the vertical axis.
- Random Brightness Contrast (probability = 0.2): Adjusts brightness and contrast to simulate lighting conditions.
- Random Gamma (probability = 0.2): Alters, the gamma levels, affecting the luminance of the images.
- RGB Shift (probability = 0.2): Randomly shifts the RGB color channels, introducing color variations.
- Vertical Flip (probability = 0.5): Flips the image along the horizontal axis.

This augmentation process generated **60 additional images for each original**, resulting in a comprehensive dataset of **5,940 images.**

Subsequently, to align with the input specifications of the MobileNetV2 model, we resized the images from their original dimensions of 1280x780 to 224x224 pixels. This resizing is crucial as it ensures the input layer of the MobileNetV2 model receives data in a format it is conditioned to process efficiently. Additionally, to optimize model training and performance, we normalized the pixel values of each image by dividing by 255, the maximum pixel value (scaling pixel values to a range of 0 to 1). Similarly, we normalized the bounding box coordinates to the same range by dividing it by the maximum pixel size for each coordinate (1280 (x_max), 780 (y_max), 1280 (x_max), 780 (y_max)) in the original image size. These preprocessing steps are best practices in deep learning, enhancing numerical stability and ensuring faster convergence during training.

2-2. Model Architecture

MobileNetV2 was chosen as the foundational architecture in the project due to its efficiency and effectiveness in handling image classification and feature extraction tasks, particularly in environments with limited computational resources. MobileNetV2 comes pre-trained on the ImageNet dataset, allowing transfer learning. It utilizes **depthwise separable convolutions**

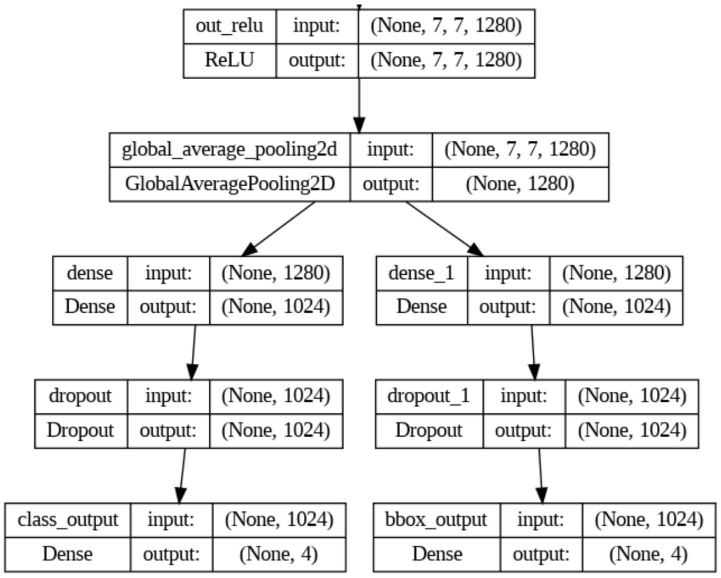
[1] *"Introduction to OpenCV." OpenCV 4.9.0 Documentation, OpenCV, docs.opencv.org/4.x/d1/dfb/intro.html*

[2] *Wada, Kentaro. "labelme." GitHub, repository by labelmeai, GitHub, https://github.com/labelmeai/labelme?tab=readme-ov-file.*

[3] *"Albumentations Documentation." Albumentations, albumentations.ai/docs/.*

that significantly reduce the number of parameters compared to traditional convolutional networks, thus speeding up the processing time without compromising accuracy [4].

To tailor the MobileNetV2 architecture for the dual tasks of facial recognition and bounding box prediction, several modifications were implemented to the **fully connected layers:**


out_relu | input: | (None, 7, 7, 1280)
ReLU | output: | (None, 7, 7, 1280)
global_average_pooling2d | input: | (None, 7, 7, 1280)
GlobalAveragePooling2D | output: | (None, 1280)
dense | input: | (None, 1280)
Dense | output: | (None, 1024)
dense_1 | input: | (None, 1280)
Dense | output: | (None, 1024)
dropout | input: | (None, 1024)
Dropout | output: | (None, 1024)
dropout_1 | input: | (None, 1024)
Dropout | output: | (None, 1024)
class_output | input: | (None, 1024)
Dense | output: | (None, 4)
bbox_output | input: | (None, 1024)
Dense | output: | (None, 4)

New model architecture

Post convolutional feature extraction, a Global Average Pooling layer is applied to reduce the spatial dimensions to a single vector per channel. This step condenses the feature information, preparing it for the final classification and regression layers. A Dense layer with 1024 neurons followed by an L2 Regularization layer and a Dropout layer at 0.5 probability enhances the network's generalization ability by preventing overfitting. This configuration is replicated for the classification output (facial identity) and the bounding box prediction.

In the classification output layer, a Dense layer with a softmax activation function outputs the probabilities of the four classes (three individual faces and 'none'). Softmax is ideal for multi-class classification, where each class probability is required. In the Bounding Box Output**,** a Dense layer with a sigmoid activation function predicts four coordinates defining the bounding box. Sigmoid is chosen to ensure the outputs are in the range [0,1], corresponding to normalized coordinates relative to the image dimensions.

By integrating these modifications, the adapted MobileNetV2 architecture learns to identify facial features accurately and locate them within the image frame, providing us with an efficient facial recognition system.

Training Process

The dataset of non-augmented and augmented images was divided into training, validation, and testing sets, with a distribution of 70%, 16%, and 14%, respectively. This split ensures

[4] *Tragoudaras, Antonios, et al. "Design Space Exploration of a Sparse MobileNetV2 Using High-Level Synthesis and Sparse Matrix Techniques on FPGAs." Sensors, vol. 22, no. 12, 2022, article no. 4318, www.mdpi.com/1424-8220/22/12/4318.*

substantial data for training the model while providing adequate resources for validation and independent testing to assess its performance.

The group set training to 40 epochs, utilizing batches of 10 images each. We employed shuffling to promote model generalization by presenting the training data in a random sequence at each epoch, preventing the model from learning **spurious patterns** from the order of the data. Additionally, data prefetching was implemented to enhance training efficiency by pre-loading data into memory, ensuring that data processing and model training operations co-occur.

Early stopping was integrated with a patience parameter of four epochs to mitigate overfitting and optimize computational resources, halting training if the validation loss does not improve consecutively over four epochs. Lastly, we employed the **Adam optimizer** because of its adaptive learning rate capabilities. By adjusting the learning rate for each parameter, Adam helps converge faster and more effectively. Adam combines the advantages of two other extensions of stochastic gradient descent: the Adaptive Gradient Algorithm (AdaGrad) and Root Mean Square Propagation (RMSProp) [5], making it well-suited for handling sparse gradients on noisy problems.

2-3. Loss Functions and Metrics

Our model architecture involves **dual outputs:** classifying faces and predicting bounding boxes. To train this model, we employed two loss functions tailored to each output's unique requirements.

- Localization Loss: This custom function focuses on the bounding box predictions. It comprises two main components:
  - Delta Coords Loss: It measures the squared Euclidean distance between the predicted and true coordinates of the bounding box's top-left corner (both x and y coordinates). By minimizing this loss, we ensure that the bounding box is correctly positioned from the start, enhancing the overall prediction accuracy.
  - Delta Size Loss: This calculates the squared differences between the bounding box's predicted and actual width and height. This loss component ensures that the model accurately predicts the scale of the bounding box relative to the object it is meant to frame.
- Sparse Categorical Crossentropy: Used for the classification task, this loss function is suitable for multi-class classification problems where each class is mutually exclusive. The loss function suits our model because it simplifies working with integer target classes, enhancing computational efficiency.

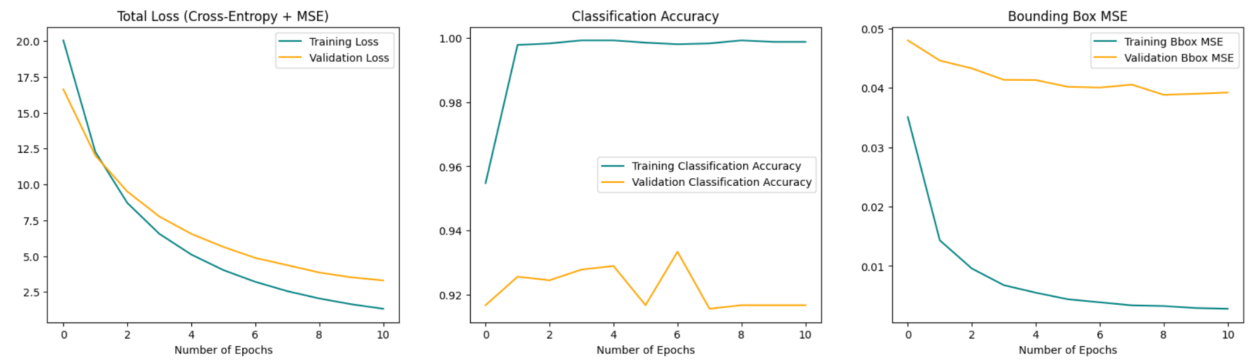
Weighting Loss Components: In compiling our model, we emphasized the localization loss more than the classification loss. The rationale behind this decision is to prioritize precise bounding box predictions in **practical applications of facial recognition systems.** We assigned higher penalties to errors in bounding box coordinates (delta coords) to prioritize spatial accuracy and gave the bounding box loss a 50% higher weight than classification loss.

Metrics: For the classification task, accuracy is a straightforward metric to evaluate how often the model correctly identifies the face. However, in the regression task of predicting bounding box coordinates, we chose MSE because it heavily penalizes more significant errors, which is

[5] *"tf.keras.optimizers.Adam." TensorFlow, version 2.16.1, TensorFlow, www.tensorflow.org/api_docs/python/tf/keras/optimizers/Adam.*

desirable in bounding box prediction. MSE significantly penalizes larger errors in bounding box predictions, prompting the model to prioritize their correction and adjust its weights accordingly.

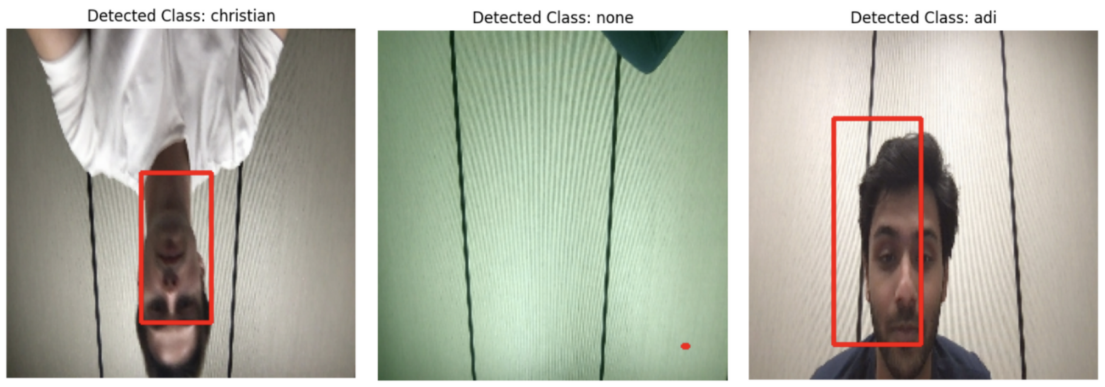
2-4. Training and Validation Result Plots



Total Loss in both training and validation loss decreases smoothly across each epoch and appears to be reaching a convergence point. For **Classification Accuracy**, the validation accuracy jumps up and down over the first 6 epochs and then decreases slightly. This could suggest that the model slightly overfits from the start, or the validation set may contain a few examples not well represented in the training set. The fact that the training accuracy is perfect could also suggest that the model may be too closely fitting to training data, indicating overfitting. In **Bounding Box MSE** the smooth decline in validation MSE indicates that the model is learning the task of bounding box regression without overfitting, as there is no increase in validation MSE.

2-5. Results – Testing

We tested the model on 840 images and achieved the following metrics:
bbox_output_mse: 0.018, **class_output_accuracy:** 0.99, **loss**: 1.88



Above are three predictions on the test set, and the model predicts the correct class for each image, but the bounding box is slightly off in its coordinates. We expect the model to predict the correct class 99% of the time, which is really good. This is consistent with the metrics we are looking at above, where the bounding box MSE has a 0.018 loss. To interpret MSE, we convert it to RMSE of approximately 0.134 (since the sqrt(0.018) ≈ 0.134), which means that, on average, our model's bounding box predictions are about 0.134 units away from the true box coordinates. **Hence, a small deviation from the actual bounding box values is indicated.**

3. Discussion and Conclusion

3-1. Interpretation of Results

The performance of our model during the training and validation phases was strong, achieving high training and validation accuracy in classifying faces across different images. However, while the model excelled in facial recognition, the real-time bounding box prediction showed room for improvement. This discrepancy suggests that while the model is proficient in identifying faces, its spatial localization abilities could be refined.

3-2. Challenges and Resolutions

One significant challenge we encountered was the labor-intensive process of manually labeling images, which proved time-consuming and susceptible to human error. To streamline this process in future iterations, we are considering the adoption of semi-automated labeling tools, such as Roboflow, that can pre-label images and require only human verification. This change could drastically reduce time and increase labeling accuracy [6].

Another area for improvement was the extensive duration of the training process, which initially hindered our progress. To address this, we implemented several strategic measures: introducing early stopping to prevent overfitting and unnecessary computations, leveraging the efficient architecture of MobileNetV2, known for its quick processing times without sacrificing performance, and utilizing the Adam optimizer for its effective learning rate adjustments. Dropout layers and L2 regularization were incorporated to enhance generalization and prevent overfitting.

3-3. Proposed Future Work

Looking forward, there are several avenues to enhance our project potentially:

- Hardware Enhancements: Utilizing Nvidia GPUs would likely significantly decrease our model's training time, allowing for more complex experiments and faster iterations.
- Data Augmentation: Increasing the size and diversity of our training dataset will be critical. More diverse data will better equip the model to handle real-world variability in facial recognition tasks.
- Exploring Advanced Models: We aim to explore cutting-edge architectures such as YOLOv8 and various R-CNN models, which might offer better performance in both classification and localization tasks compared to our current model [7].
- Integration with IoT Devices: Implementing the model in a real-time application on IoT devices could also be explored to enhance its applicability in practical scenarios, such as security systems and personalized user experiences.

These steps will not only aim to enhance the model's performance but also broaden its applicability and efficiency in real-world scenarios.

[6] *"Roboflow Auto Label: Fast, Automatic Vision Labeling." Roboflow, Roboflow, https://roboflow.com/auto-label.*

[7] *"YOLOv8: A New State-of-the-Art Computer Vision Model." YOLOv8, Ultralytics, https://yolov8.com/.*

4. References

1. "tf.keras.optimizers.Adam." *TensorFlow*, version 2.16.1, TensorFlow, www.tensorflow.org/api_docs/python/tf/keras/optimizers/Adam.
2. "Albumentations Documentation." *Albumentations*, albumentations.ai/docs/.
3. "Introduction to OpenCV." *OpenCV 4.9.0 Documentation*, OpenCV, docs.opencv.org/4.x/d1/dfb/intro.html.
4. Wada, Kentaro. "labelme." *GitHub*, repository by labelmeai, GitHub, https://github.com/labelmeai/labelme?tab=readme-ov-file.
5. Tragoudaras, Antonios, et al. "Design Space Exploration of a Sparse MobileNetV2 Using High-Level Synthesis and Sparse Matrix Techniques on FPGAs." *Sensors*, vol. 22, no. 12, 2022, article no. 4318, www.mdpi.com/1424-8220/22/12/4318.
6. "Roboflow Auto Label: Fast, Automatic Vision Labeling." *Roboflow*, Roboflow, https://roboflow.com/auto-label.
7. "YOLOv8: A New State-of-the-Art Computer Vision Model." *YOLOv8*, Ultralytics, https://yolov8.com/.