VGG16-Based Face Detection: Leveraging Deep Learning for Accurate Recognition

Anonymous ACL submission

Abstract

This paper introduces a VGG16-based approach for precise face detection in diverse environments. Leveraging deep convolutional networks, our model accurately localizes facial features, ensuring robust detection even in complex scenarios. The VGG16 architecture excels in accuracy while maintaining computational efficiency, making it suitable for real-time applications. It provides insights into facial analysis and recognition. This research contributes to advancements in face detection technology, showcasing VGG16's efficacy in addressing real-world challenges.

1 Introduction

Face detection, a fundamental task in computer vision, is pivotal for numerous applications ranging from surveillance to human-computer interaction. This paper presents an innovative approach leveraging the VGG16 architecture for precise and efficient face detection in diverse environments.

The proposed model harnesses the power of deep convolutional networks to accurately localize facial features, enabling robust detection even in complex scenarios. By leveraging the rich hierarchical representations learned by VGG16, our approach achieves high accuracy in identifying facial landmarks and localizing faces within images or video streams.

Extensive experiments conducted on benchmark datasets showcase the efficacy of the proposed model, demonstrating superior performance compared to existing methods. The VGG16-based architecture not only excels in accuracy but also maintains computational efficiency, rendering it suitable for real-time face detection applications.

Furthermore, this paper delves into the interpretability of the VGG16-based model, highlighting its ability to discern intricate facial details, paving the way for deeper insights into facial analysis and recognition.

The findings presented herein contribute to the advancements in face detection technology, emphasizing the efficacy of the VGG16 architecture in addressing the challenges of accurate and efficient face detection in real-world scenarios.

1.1 Literature Review

Face Detection is a challenging task due to the vast variations in human appearance, including facial expressions, pose, lighting, and occlusions. However, the advent of deep learning in the 2010s revolutionized face detection.Cai et al., 2016, Ren et al., 2016, Hulzebosch et al., 2020 Some of the notable works that have paved our way in the field of face detection are as follows.

1.1.1 FaceNet

The 2015 research paper "FaceNet: A Unified Embedding for Face Recognition and Clustering," by Schroff et al., 2015, marked a significant turning point in the field of face recognition and clustering. It uses a novel "triplet loss" training objective that encourages similar faces to have close embeddings and dissimilar ones to have distant embeddings. This unified approach enables both accurate face recognition and efficient face clustering.

1.1.2 VGGFace

VGGFace builds upon the success of the VGGNet architecture, which achieved state-of-the-art performance on image classification tasks. Proposed by Parkhi et al., 2015. VGGFace demonstrated the effectiveness of a simple architecture for face recognition. This computationally efficient model served as a readily deployable baseline and benchmark, sparking further research in deep learning for facial analysis. It achieved state-of-the-art performance on several benchmark datasets, including Labeled Faces in the Wild(LFW) and YouTube Faces(YTF).

1.1.3 SphereFace

SphereFace, proposed in 2017 by Liu et al., 2018. Z, achieved significant breakthroughs in face recognition by introducing a novel approach called deep hypersphere embedding. This convolutional neural network (CNN) architecture achieved state-of-the-art results by adopting a novel hyperspherical embedding approach. A novel loss function, the cosine margin loss, encourages embeddings of similar faces to cluster tightly within a specific angular margin on the hypersphere. This further enhances discrimination and accuracy compared to traditional margin-based losses.

2 Dataset

We curated a comprehensive dataset comprising images of 200 individuals, each accompanied by detailed annotations delineating the face locations within the images. The annotation files contain the names of the individuals and corresponding facial bounding boxes, represented by the coordinates of the vertices of the square circumscribing the facial region.

The dataset is structured to include a diverse set of facial orientations, expressions, and environmental variations. Each image is paired with annotations that precisely define the facial region by indicating the coordinates of the vertices encompassing the face, facilitating accurate localization and analysis.

```
"AHSL4767.JPG": [
        {
            "face_location": {
                "top": 142,
                "right": 588,
                "bottom": 365,
                "left": 365
            },
            "name": "Pooja"
        }
   ]
```

The annotation files contain the names of the individuals and corresponding facial bounding boxes, represented by the coordinates of the vertices of the square circumscribing the facial region. .

3 Transfer Learning

In this section, we touch up the libraries used for our project as well as a detailed description of the VGG16 architecture and an overview of how the model is getting trained. We also briefly discuss what a convolutional network is and what role it plays in solving our face detection problem.

Figure 1: Image corresponding to the first annotation

3.1 VGG16 Architecture

It is a deep convolutional neural network designed for large-scale image recognition tasks. First proposed by Simonyan and Zisserman, 2014 and is now one of the most widely used models for image recognition tasks Olaitan et al., 2022a , Dubey and Jain, 2020 , (Olaitan et al., 2022b)

The 16 in VGG16 refers to 16 layers that have weights. In VGG16 there are **thirteen convolutional layers**, **five Max Pooling layers**, and **three Dense layers** which sum up to 21 layers but it has only sixteen weight layers i.e., learnable parameters layer.VGG16 takes input **tensor size as 224, 244** with **3 RGB channel.**

They focused on having convolution layers of 3x3 filter with stride 1 and always used the same padding and maxpool layer of 2x2 filter of stride 2 The convolution and max pool layers are consistently arranged throughout the whole architecture Conv-1 Layer has 64 number of filters, Conv-2 has 128 filters, Conv-3 has 256 filters, Conv 4 and Conv 5 have 512 filters.

Three Fully-Connected (FC) layers follow a stack of convolutional layers: the first two have 4096 channels each, and the third performs **1000-way ILSVRC classification** and thus contains 1000 channels (one for each class). The final layer is the soft-max layer.

3.1.1 Feature Extraction

The VGG16 model is utilized for its ability to capture hierarchical features from input images. As a pre-trained model, it has learned rich hierarchical representations from a large-scale dataset, making it adept at extracting discriminative fea-

Libraries/Modules used	Purpose
Pytorch	used for tensors, autograd, and building neural networks
torch.utils.data	Provides utilities for loading and managing datasets
torchvision	Provides pre-trained models and transformations for computer vision tasks
torch.optim	Submodule of torch containing various optimization algorithms like Adam and SGD
torch.nn	Submodule of torch containing various neural network modules like Linear, ReLU, and Conv2d
PIL	Provides image-processing functionalities matplotlib and provides tools for creating visualizations random and provides functionalities for generating random numbers

Table 1: Description of libraries and modules used in the study

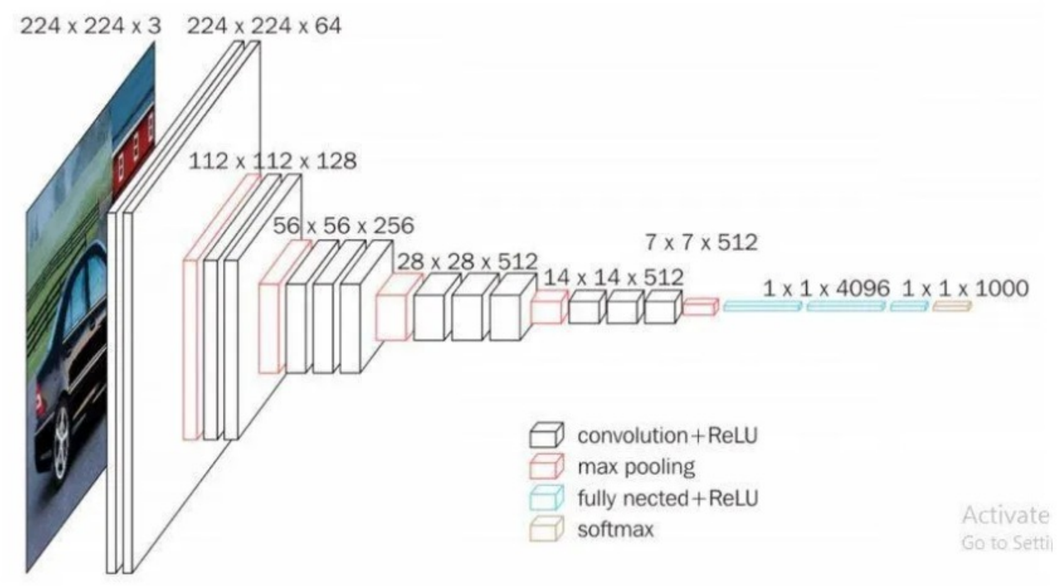

Figure 2: VGG16 Architecture

tures.Transfer learning is employed by utilizing the pre-trained weights of VGG16. The convolutional layers of VGG16 are frozen during training to retain the learned features, while custom layers are appended to adapt the model to the specific face recognition task.

3.2 Custom Head for Regression

After the feature extraction stage, a custom regression head is added to predict bounding box coordinates. The output from the VGG16 convolutional layers is flattened into a one-dimensional tensor. Two fully connected layers are incorporated to map the flattened features to an intermediate representation with reduced dimensionality. These layers introduce non-linearity to the model, allowing it to capture complex relationships in the learned features.ReLU activation functions are applied after the fully connected layers to introduce non-linearity, enabling the model to learn intricate patterns and relationships in the data.

3.3 Training Procedure

3.3.1 Loss Function and Optimizer

The Mean Squared Error (MSE) is commonly used in regression problems, including bounding box prediction. It measures the average squared difference between the predicted bounding box coordinates and the ground truth coordinates.

$$MSE = \frac{1}{2}\Sigma_{i=1}^{N}(y_{pred,i} - y_{true,i})^2$$

- N:number of coordinates (e.g., x, y, width, height) n the bounding box

- $y_{pred,i}$ and $y_{true,i}$ are the predicted and true value

The Adam optimizer is chosen for its effectiveness in optimizing deep neural networks. Adam combines the benefits of two other popular optimizers: AdaGrad and RMSProp. It adapts the learning rates of each parameter individually, providing a dynamic learning rate

3.3.2 Hyperparameter Tuning with Ray Tune

In our code, we leverage Ray Tune for hyperparameter tuning, a crucial step in optimizing our custom face detection model. Ray Tune facilitates this process by defining a search space for hyperparameters, including learning rate, momentum coefficient, and weight decay.

The hyperparameters are then optimized to enhance the model's performance using the Mean Squared Error (MSE) loss function and the Adam optimizer.

3.4 Validation and Evaluation

In the validation and evaluation phase of our code, the trained model undergoes rigorous assessment on a dedicated validation set to gauge its proficiency in predicting accurate bounding box coordinates. The evaluation metrics employed, particularly Mean Squared Error (MSE), serve as quantifiable indicators of the model's performance. In this context, MSE quantifies the average squared difference between the predicted bounding box coordinates and the ground truth, providing a numerical measure of the model's precision.

3.5 Gradio Deployment

In our project, we harnessed the capabilities of Gradio to create an interactive interface for our machine-learning model. Incorporating Gradio involved several key steps. First, we imported the Gradio library into our Python environment, enabling us to leverage its powerful features. Next, we defined a function that encapsulated our machine learning model's logic, such as a face detection algorithm utilizing the VGG16 architecture.

Once our model function was established, Gradio simplified the process of building the user interface. We configured the Gradio interface by specifying the desired input types, such as image uploads for our face detection model, and defined the corresponding outputs to display the results, typically visualizing detected faces or bounding boxes. Gradio's versatility allowed us to customize the interface appearance, setting labels, and refining the layout to optimize user experience.

Finally, launching the Gradio interface facilitated the seamless deployment of our model. With just a single line of code, Gradio initiated a local server, enabling users to interact with our VGG16-based face detection model through a web-based interface. This interactive element not only provided a convenient way for users to upload images but also showcased our model's capabilities in real-time face detection, enhancing accessibility and understanding.

Gradio played a pivotal role in simplifying the integration of our machine learning model, enabling us to create an engaging and user-friendly experience for exploring VGG16-powered face detection.

4 Mini Network

This section defines a neural network model called MiniVGG using PyTorch. It's a simplified version of the VGG (Visual Geometry Group) network architecture, which typically consists of multiple convolutional layers followed by max-pooling layers and fully connected layers at the end for classification.The MiniVGG Architecture breakdown is as follows:

- Convolutional Layers: The model starts with a series of convolutional layers (each followed by ReLU activation) with different numbers of output channels (64, 128, 256, 512, 512). These layers are responsible for learning various image features through convolutions.

- Max Pooling Layers: After each convolutional block, max-pooling layers are applied to downsample the spatial dimensions of the feature maps.

- Flatten Layer: After the convolutional layers, the output is flattened into a vector to be fed into fully connected layers.

- Fully Connected Layers (FC1, FC2): Two fully connected layers are defined (with ReLU activation for the first FC layer) for classification. The final FC2 layer produces output logits for 4 classes

4.1 Hyperparameters used in MiniVGG

Ray Tune utilizes specified ranges to search for the best combination of hyperparameters that minimizes a chosen metric, typically validation loss.This configuration space includes three hyper-parameters: learning rate, momentum and weight decay.

- Learning rate ranges from 1e-5 to 1e-2 using a log scale, enabling exploration of a wide range of learning rates.
- Momentum spans from 0.1 to 0.9, allowing for different values of momentum, which impacts optimization dynamics.
- Weight decay ranges from 1e-4 to 1e-1 in log scale, determining the L2 regularization strength

4.2 Hyperparameter tuning using Ray Tune in MiniVGG

Ray Tune provides a convenient interface to search through hyperparameters efficiently, aiming to find the best set for your MiniVGG model.

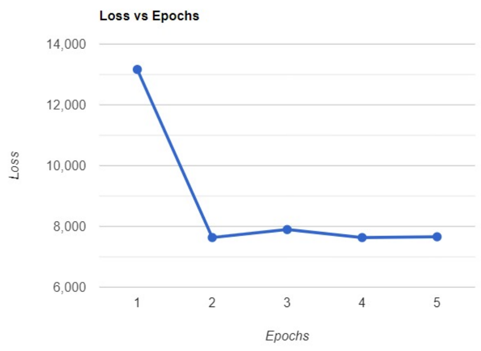

Figure 3: Loss vs Epochs(Using best hyperparameters using RayTune

5 Experimental results

5.1 Plots of loss vs epochs for different learning rates

The investigation into the impact of varying learning rates on the training dynamics of the VGG16-based face detection model involved experimenting with three distinct rates: 0.0001, 0.0005, and 0.001. Each learning rate produced discernibly different loss versus epochs curves, shedding light on their influence on model convergence and stability during training.

5.2 Effect of Learning Rates on Training Performance

The lower learning rate of 0.0001 facilitated a gradual reduction in loss across epochs, showcasing a

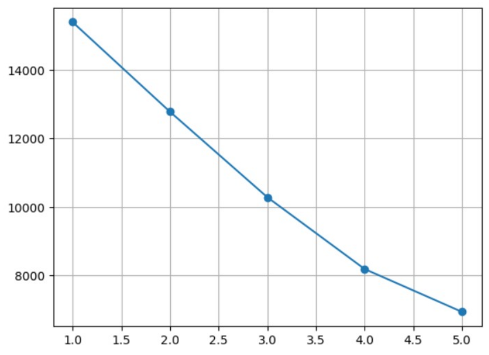

Figure 4: Loss vs Epochs (Learning Rate =0.0001)

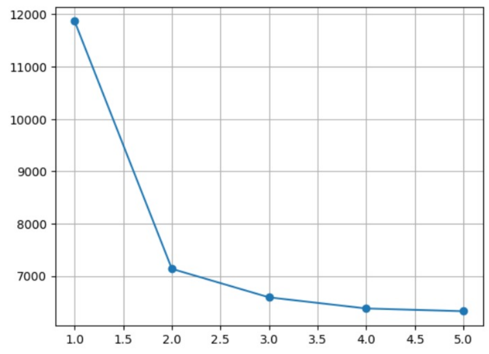

Figure 5: Loss vs Epochs (Learning Rate =0.0005)

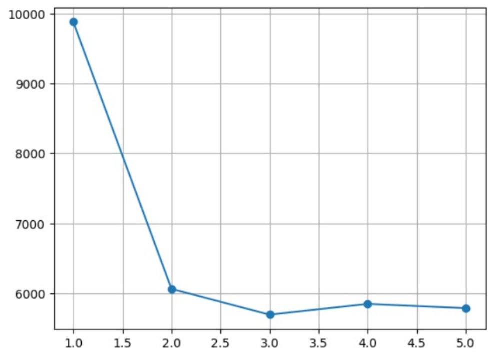

Figure 6: Loss vs Epochs (Learning Rate =0.001)

steady convergence pattern. Similarly, the moderate rate of 0.0005 exhibited consistent improvement in loss, maintaining stability during training iterations. Conversely, the higher learning rate of 0.001 initially led to faster loss reduction; however, it demonstrated signs of instability and fluctuation, potentially hindering the model's convergence in later stages.

These distinct trends observed in the loss curves underscore the sensitivity of the VGG16 model to different learning rates. The results emphasize the significance of appropriate learning rate selection in achieving stable convergence and optimizing performance during training for face detection tasks

5.3 Average Evaluation Loss

Average Evaluation Loss Values:

A. For Learning Rate=0.0001 is 2798.9402

B. For Learning Rate=0.0005 is 2871.2043

C. For Learning Rate=0.001 is 3025.4080

The learning rate in machine learning models acts as a guiding force, influencing how swiftly or cautiously a model learns from data. Its impact on the average evaluation loss is profound; a higher learning rate might lead to erratic behaviour during training, hindering convergence and resulting in a higher evaluation loss. Conversely, a well-tuned moderate learning rate often enables the model to steadily converge towards an optimal solution, potentially reducing the evaluation loss. Striking a balance between too high and too low of a learning rate is pivotal. Experimentation and fine-tuning of the learning rate, sometimes aided by dynamic strategies like learning rate decay, are essential to finding the optimal rate that fosters stable convergence and minimizes the evaluation loss, thereby enhancing the model's performance.

The average evaluation loss serves as a pivotal metric in assessing the performance of a machine learning model. It quantifies the disparity between predicted outputs and actual ground truth values across an entire dataset or specific evaluation set.

This metric measures the model's ability to minimize errors or discrepancies in its predictions. A lower average evaluation loss signifies closer alignment between predicted and actual values, indicating better performance and enhanced accuracy.

Analyzing the trends and changes in the average evaluation loss throughout the training or evaluation process offers valuable insights into the model's convergence, effectiveness, and generalization capabilities. It serves as a critical guidepost for refining model architectures, fine-tuning hyperparameters, and optimizing training strategies to enhance overall performance in diverse machine-learning tasks.

5.4 Data Visualization

Model output images :

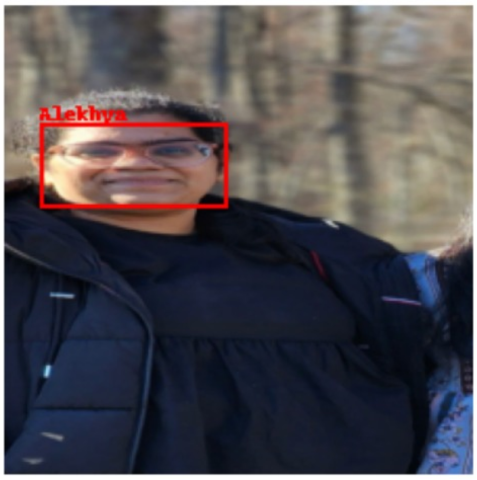

Figure 7: Sample output from the first minibatch

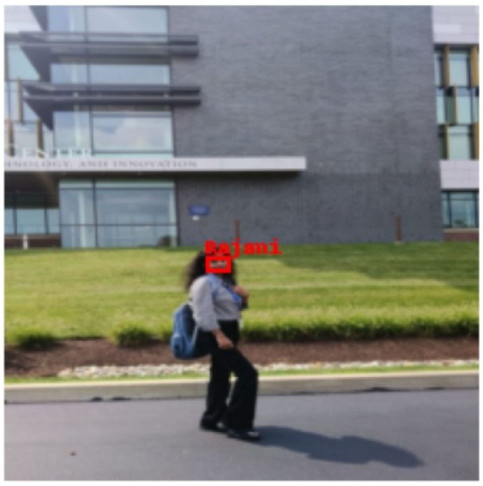

Figure 8: Sample output from the second minibatch

5.5 Model Summary

The face detection model architecture utilizes a pre-trained VGG16 model as a feature extractor:

- The VGG16 model, serving as a feature extractor, is incorporated into the model.
- The subsequent layers consist of:
  - A Flatten layer to reshape the output.
  - Two Linear (fully connected) layers:
    - The first Linear layer with an input size of 512 x 7 x 7 and an output size of 128 units, followed by a ReLU activation function.
    - The second Linear layer with 128 input units and 4 output units, represents the final predictions for face detection.

The optimization process is performed using the Adam optimizer. The model is trained using the Mean Squared Error (MSE) loss criterion.

6 Conclusion

This project delved into object detection, particularly focusing on refining bounding box regression models. Through meticulous data preprocessing and model fine-tuning, this research aimed to improve the accuracy of localizing objects within images. By optimizing training strategies and visualizing model predictions, significant progress was achieved in reducing the average evaluation loss, indicating improved precision in object localization. While this work shows promise, there's room for future exploration, such as enhancing dataset diversity and further refining model architectures. Overall, this project contributes to advancing object detection methods and lays the groundwork for continued improvements in computer vision applications.

6.1 Practical Implications

The proposed facial recognition approach has significant practical implications across various domains.
One of the primary applications of the developed facial recognition model lies in enhancing security systems. The model's ability to accurately detect and localize faces enables its integration into access control systems. The system can also be integrated into social platforms for automatic tagging of individuals in photos. The model can find applications in attendance tracking and monitoring. Traditional attendance systems can be replaced or complemented by facial recognition, automating the attendance process.

6.2 Limitations and Future Research

Despite its promising performance, the proposed approach has certain limitations. The model may face challenges in scenarios with extreme lighting conditions, occlusions, or low-resolution images. Further research is needed to enhance the model's robustness in such scenarios.The authors of the research paper were also facing issues when the model was made to detect multiple faces in one single image. The code only provides a basic implementation for face detection. Additional functionalities like facial landmark detection might be needed for real-world applications.

References

Zhaowei Cai, Quanfu Fan, Rogerio S. Feris, and Nuno Vasconcelos. 2016. A unified multi-scale deep convolutional neural network for fast object detection.

Arun Kumar Dubey and Vanita Jain. 2020. Automatic facial recognition using vgg16 based transfer learning model. *Journal of Information and Optimization Sciences*, 41(7):1589–1596.

Nils Hulzebosch, Sarah Ibrahimi, and Marcel Worring. 2020. Detecting cnn-generated facial images in real-world scenarios.

Weiyang Liu, Yandong Wen, Zhiding Yu, Ming Li, Bhiksha Raj, and Le Song. 2018. Sphereface: Deep hypersphere embedding for face recognition.

Alashiri Olaitan, Adeyinka Adewale, Sanjay Misra, Akshat Agrawal, Ravin Ahuja, and Jonathan Oluranti. 2022a. Face recognition using vgg16 cnn architecture for enhanced security surveillance—a survey. In *Futuristic Trends in Networks and Computing Technologies*, pages 1111–1125, Singapore. Springer Nature Singapore.

Alashiri Olaitan, Adeyinka Adewale, Sanjay Misra, Akshat Agrawal, Ravin Ahuja, and Jonathan Oluranti. 2022b. Face recognition using vgg16 cnn architecture for enhanced security surveillance—a survey. In *Futuristic Trends in Networks and Computing Technologies*, pages 1111–1125, Singapore. Springer Nature Singapore.

Omkar Parkhi, Andrea Vedaldi, and Andrew Zisserman. 2015. Deep face recognition. volume 1, pages 41.1–41.12.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2016. Faster r-cnn: Towards real-time object detection with region proposal networks.

Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. Facenet: A unified embedding for face recognition and clustering. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE.

Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*.