Tuijin Jishu/Journal of Propulsion Technology
ISSN: 1001-4055
Vol. 44 No. 5 (2023)

Automated Human Detection in Images Using Deep Learning

[1] Srilaxmi , [2]Surekha Kamath, [3]Veena Mayya

[1] NMAMIT, Nitte
Nitte, India
[2] ICE department, Manipal Institute of Technology, Manipal Academy of Higher Education,
Manipal, 576104, India
Manipal, India
[3]Department of Information and Communication Technology, Manipal Institute of Technology,
Manipal Academy of Higher Education, Manipal, 576104, India

Abstract— Human detection is a vibrant and ever-evolving field of active research in the present context. This paper presents an automated approach utilizing deep learning techniques to identify and track humans within images. The primary objective of this work is to detect individuals within images. Machine learning models have found extensive utility in a variety of applications, including autonomous vehicles and speech recognition. In this paper, we introduce a deep learning-driven method for automated human detection, harnessing the power of convolutional neural networks (CNNs) and state-of-the-art object detection architectures. To boost the performance of our deep learning-based human detection system, we employ several strategies. These encompass the use of data augmentation techniques to expand the dataset, harnessing transfer learning to leverage pre-trained models, and implementing advanced optimization algorithms for model fine-tuning. Additionally, we investigate the effectiveness of different network architectures, loss functions, and hyperparameters to optimize detection accuracy. Our experimental results illustrate the effectiveness of deep learning in automating human detection. Our proposed approach attains high precision and recall rates, effectively pinpointing humans in a variety of challenging scenarios. The potential applications of our method span diverse domains, including surveillance systems, activity recognition, and human-robot interaction.
Keywords— Convolutional neural network Human activity, Human action classification, Machine learning.

1. Introduction

Classifying human images poses a formidable challenge within the domain of computer vision, requiring the identification of human subjects in images or videos. Transfer learning, a widely adopted technique in deep learning, involves reusing pre-trained models to address related problems. Among the manifold methods employed to augment deep learning models, transfer learning emerges as a prominent approach. The realm of deep learning applications is rapidly expanding, with Convolutional Neural Networks (CNNs) playing a pivotal role. CNNs find extensive use in diverse areas such as object detection, face recognition, video analysis, cancerous tissue segmentation for diagnostic purposes, pattern recognition, language processing, and more. When it comes to object detection, CNNs exhibit a superior level of accuracy compared to other deep learning networks. Within computer vision, human detection assumes a critical role, finding applications in surveillance, crowd analysis, autonomous driving, and beyond. Human detection, a subset of object detection, is primarily concerned with pinpointing the presence of humans within images. This intricate process involves scanning various regions within an image at multiple scales and comparing them against known human patterns. In the context of videos, the task of associating human detections over time to establish consistent paths or trajectories is known as human tracking, which holds immense importance in the domain of video surveillance. Human tracking and detection have substantial implications in surveillance, with potential applications spanning disaster response through the utilization of drones and military operations. Employing deep learning techniques, our system processes input images and furnishes binary classification results, indicating whether a human is present or absent. This model holds significant promise across various domains, offering robust solutions for human identification and tracking.

2. Related Work

Recently, Avaneesh Tripathi et al. [1] have elucidated the concept of automated human detection and tracking in the context of surveillance applications. Their paper introduced the innovative use of thermal imaging cameras as a solution to address the limitations of existing systems. In contrast to optical cameras, the utilization of thermal imaging enhances the system's detection capabilities, making it particularly effective for nighttime operations. Additionally, Sasa Sambolek et al. [2] discuss the critical role that automated human detection plays in Search Management Groups and Search and Rescue Teams (SAR) for locating and assisting injured or lost tourists in mountainous terrains. This study looked closely at a number of cutting-edge detection methods, such as YOLOv4, Faster R-CNN, Cascade R-CNN, and RetinaNet. It used the VisDrone benchmark and a custom dataset called SARD to model rescue situations. Wahyu Rahmaniar et al. [3] delved into the methodology of human detection using deep learning techniques. Their paper offers an extensive survey of recent advancements in human detection models tailored for deployment on embedded platforms. The emphasis is placed on the performance evaluation of several models, such as SSD MobileNet V2, NVIDIA Jetson, multiped, SSD MobileNet V1, PedNet, and SSD Inception V2, with a particular focus on their capabilities in edge computing. Nouar AlDahou et al. [5] explored the real-time detection of humans in aerial video sequences using deep learning models. Their approach incorporates automatic feature learning techniques, integrating optical flow, and various models. The research involved training and testing on the difficult UCF-ARG aerial dataset. The results of the tests showed that their methods worked well to solve problems with human detection.In summary, automated human detection using deep learning has made significant strides in recent years, resulting in advancements across a multitude of applications.

3. METHODOLOGY

A. Image Classification

As depicted in Fig. 1, the image classification framework utilizes a transfer learning approach. This methodology encompasses six distinct phases, each of which we will elaborate upon. All of these stages are executed by utilizing open-source tools, specifically TensorFlow, in conjunction with the Python programming language. The sequential steps encompass data collection and preprocessing, the selection of pre-trained models, the processes of fine-tuning or feature extraction, and the subsequent evaluation of model performance. The application of transfer learning emerges as highly advantageous in elevating the accuracy of models for classifying human images, particularly when confronted with limited or compact datasets.

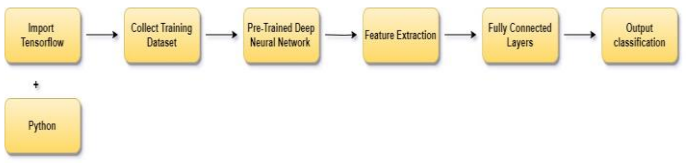


Fig. 1. Block Diagram of Image Classification

The initial step in human image classification involves the acquisition and preprocessing of the dataset. This dataset encompasses images or videos depicting humans engaged in various poses, activities, and environments. It may also include annotations or labels that signify the presence or absence of humans in each image or video.

Following this, we leverage transfer learning to adapt pre-existing models for the purpose of addressing the human image classification challenge. These pre-trained models are typically trained on extensive datasets such as ImageNet, which contains millions of images spanning numerous categories. The CNNs integrated into these models excel at extracting crucial features from images. In our approach, we opt for the VGG (Visual Geometry Group) model, which is a 16-layer CNN, harnessing the vast ImageNet database for this purpose.

Once the transfer learning strategy is chosen, the pre-trained model can be fine-tuned or employed to extract features from the dataset. During the feature extraction process, the pre-trained model is utilized to extract essential features from each image, and these features are subsequently utilized to train a novel classifier.

B. CNN Layers

Deep learning models are typically organized with multiple layers, each specifically designed to carry out particular computations. These layers play a defining role in shaping the architecture of the model and can come in various types. Here is an overview of common layers in deep learning and their respective importance:

Input Layer: This Serving as the model's initial layer, this component is responsible for receiving raw data input. While it doesn't engage in actual computations, it plays a critical role in defining the shape and size of the input data.

Convolutional Layer: This layer serves as the cornerstone of CNNs. It comprises a variety of filters, also known as kernels, each equipped with trainable parameters that are learned during the training process. These filters, typically smaller than the original input, convolve with the input data, resulting in the creation of activation maps. The primary purpose of this layer is to apply mathematical operations with these filters to the input data, effectively extracting essential features during the process.

$$output[i,j] = \sum(filter[k,l] \ * \ input[i+k, jl]) \qquad (1)$$

The output at position (i, j) is determined by Equation (1). In this equation, the term "filter" represents a set of filters, "input" refers to the input data, and "output" indicates the resulting output of the layer. This process involves element-wise multiplication, followed by the summation of the outcomes to produce a single output pixel. Convolutional layers are extensively employed in tasks related to image and video recognition due to their proficiency in detecting spatial patterns.

Max Pooling Layer: This layer diminishes the spatial dimensions of the input data through a downsampling operation. A critical form of pooling is known as max pooling, which selects the maximum value within each grouping of input values. The Equation (2) is utilized for the max pooling layer.

$$output[i,j] = \max\,(input[i * pool\_size{:}\, i * pool\_size + pool\_size, j * pool\_size{:}\, j * pool\_size + pool\_size\ ]) \qquad (2)$$

where pool_size is the size of the pooling window. Pooling layers are employed to decrease the dimensionality of the input data, simplifying the processing and analysis.

Activation layer: The layer that applies a nonlinear activation function to the output of the previous layer. The most common activation functions are sigmoid, ReLU, and tanh. The ReLU activation function is given by Equation (3).

$$output \ = \ \max(0, \text{input}). \qquad (3)$$

Activation layers are important because they introduce nonlinearities into the model, allowing it to learn complex relationships between the input and output.

Fully connected layer: This layer takes the output of the previous layer and applies a matrix multiplication operation to it, as depicted in Equation (4).

$$output = \text{activation}(\text{dot}(\text{input}, \text{weights}) + \text{biases} \qquad (4)$$

where input is the output of the previous layer, weights is a matrix of learned parameters, biases is a vector of learned biases, and activation is an activation function. Fully connected layers are used to learn high-level representations of the input data and are commonly used in classification tasks.

Loss Function : The loss function serves as the metric that compares the predicted output values to the target values, evaluating the neural network's performance in modeling the training data. When training, we aim to minimize this loss between the predicted and target outputs.

Binary Cross-Entropy Loss: This is the loss function used in binary classification models where the model takes in an input and has to classify it into one of two pre-set categories. In binary classification, there are only two possible actual values of y — 0 or 1. Thus, to accurately determine difference between the actual and predicted values, it needs to compare the actual value (0 or 1) with the probability that the input aligns with that category.

Optimizer Function: An algorithm that adapts the neural network’s attributes, learning rate and weights is known as an optimizer function. This function is used to improve accuracy and reduce the total loss.

Adam: Adam is a method that relies on adaptive estimation of first-order and second-order moments.

C. Model Architecture

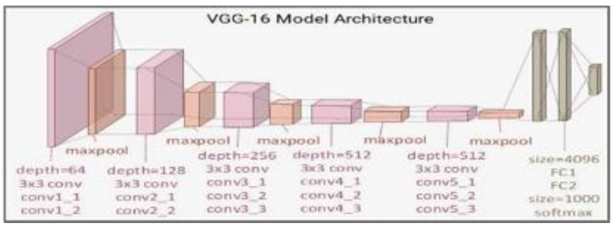


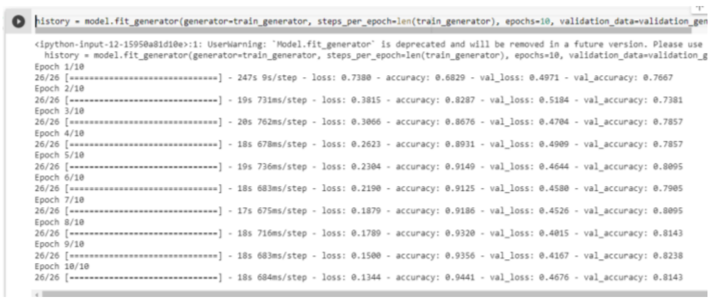
Fig.2 VGG16 Architecture Map [6]

The utilization of deep learning for human detection encompasses a variety of models and techniques. One widely employed method involves transfer learning using the VGG16 model[6], which is a sixteen-layer deep neural network incorporating essential convolutional neural network features. This architecture comprises crucial components such as convolution filters, three fully connected layers, and thirteen convolutional layers. In this approach, an appropriately sized image serves as the input. The initial two layers consist of 64 channels using a 3x3 filter size with identical padding. Subsequently, a max pool layer with a stride of (2, 2) is applied, followed by two convolution layers with 128 filters and a (3, 3) filter size. Another max-pooling layer identical to the preceding one is then used. The process continues with two convolution layers employing 256 filters and a (3, 3) size, followed by sets of three convolution layers with 512 filters and a (3, 3) size, each accompanied by a max-pooling layer. Throughout these layers, 1x1 pixel manipulation is employed to adjust the number of input channels. To preserve spatial features, 1-pixel padding, also known as "same padding," is applied after each convolution layer. After passing through these convolution and max-pooling layers, a (7, 7, 512) feature map is obtained, which is then flattened into a (1, 25088) feature vector. The subsequent steps involve three fully connected layers. The first layer takes the last feature vector as input, resulting in a (1, 4096) vector. Similarly, the second layer generates a (1, 4096) vector, while the third layer yields 1000 channels representing the 1000 classes of the ILSVRC challenge. The third fully connected layer utilizes the sigmoid function for classifying the 1000 classes. ReLU is employed as the activation function in all hidden layers, ensuring computational efficiency, faster learning, and reducing the risk of encountering vanishing gradient problems.

Transfer learning using the VGG16 model involves using a pre-trained Convolutional Neural Network (CNN) model as a starting point for a new task, such as image classification. Transfer learning with the VGG16 model involves several steps. Initially, the pre-trained model is loaded using libraries like Keras or TensorFlow. The pre-trained layers' weights are then frozen up to a specific layer, serving as fixed feature extractors and preventing overwriting during training. New layers are introduced on top of the frozen layers to tailor the model for the new task, such as image classification. These added layers, like a new fully connected layer for predicted class probabilities, fulfil the task requirements. The model is subsequently trained using the updated dataset, adjusting only the weights of the newly added layers. Optionally, fine-tuning can be performed by unfreezing some pre-trained layers, allowing their weights to adapt to the new dataset more effectively.

4. RESULTS AND DISCUSSION

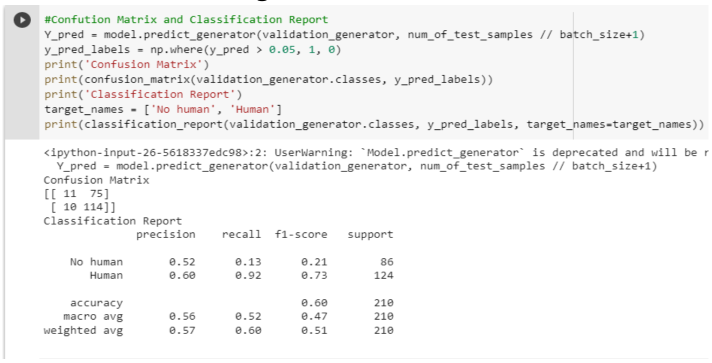
In this study, we utilize a dataset [7] comprising 1033 images. This dataset is split into two subsets, with 80% allocated for training and the remaining 20% for testing. Specifically, 823 images are used for training, and 210 images are set aside for testing. To enhance the training dataset, we employ a data augmentation technique, which involves creating modified versions of the dataset by manipulating the existing data. The dataset is then used to train the model. During the training process, the model learns to extract features from the images that are valuable for distinguishing between images containing humans and those that do not. After 10 training epochs, the model achieves a maximum training accuracy of 94%, and the highest validation accuracy is 82%, as depicted in Figure 3. Based on the results from the confusion matrix and classification report shown in Figure 4, the accuracy of the human detection model is determined to be 60%.



```
history = model.fit_generator(generator=train_generator, steps_per_epoch=len(train_generator), epochs=10, validation_data=validation_gen

<ipython-input-12-15950a81d10e>:1: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Please use
  history = model.fit_generator(generator=train_generator, steps_per_epoch=len(train_generator), epochs=10, validation_data=validation_g
Epoch 1/10
26/26 [==============================] - 247s 9s/step - loss: 0.7380 - accuracy: 0.6829 - val_loss: 0.4971 - val_accuracy: 0.7667
Epoch 2/10
26/26 [==============================] - 19s 731ms/step - loss: 0.3815 - accuracy: 0.8287 - val_loss: 0.5184 - val_accuracy: 0.7381
Epoch 3/10
26/26 [==============================] - 20s 762ms/step - loss: 0.3066 - accuracy: 0.8676 - val_loss: 0.4704 - val_accuracy: 0.7857
Epoch 4/10
26/26 [==============================] - 18s 678ms/step - loss: 0.2623 - accuracy: 0.8931 - val_loss: 0.4909 - val_accuracy: 0.7857
Epoch 5/10
26/26 [==============================] - 19s 736ms/step - loss: 0.2304 - accuracy: 0.9149 - val_loss: 0.4644 - val_accuracy: 0.8095
Epoch 6/10
26/26 [==============================] - 18s 683ms/step - loss: 0.2190 - accuracy: 0.9125 - val_loss: 0.4580 - val_accuracy: 0.7905
Epoch 7/10
26/26 [==============================] - 17s 675ms/step - loss: 0.1879 - accuracy: 0.9186 - val_loss: 0.4526 - val_accuracy: 0.8095
Epoch 8/10
26/26 [==============================] - 18s 716ms/step - loss: 0.1789 - accuracy: 0.9320 - val_loss: 0.4015 - val_accuracy: 0.8143
Epoch 9/10
26/26 [==============================] - 18s 683ms/step - loss: 0.1500 - accuracy: 0.9356 - val_loss: 0.4167 - val_accuracy: 0.8238
Epoch 10/10
26/26 [==============================] - 18s 684ms/step - loss: 0.1344 - accuracy: 0.9441 - val_loss: 0.4676 - val_accuracy: 0.8143
```

Fig. 2. Trained dataset.



```
#Confution Matrix and Classification Report
Y_pred = model.predict_generator(validation_generator, num_of_test_samples // batch_size+1)
y_pred_labels = np.where(y_pred > 0.05, 1, 0)
print('Confusion Matrix')
print(confusion_matrix(validation_generator.classes, y_pred_labels))
print('Classification Report')
target_names = ['No human', 'Human']
print(classification_report(validation_generator.classes, y_pred_labels, target_names=target_names))

<ipython-input-26-5618337edc98>:2: UserWarning: `Model.predict_generator` is deprecated and will be r
  Y_pred = model.predict_generator(validation_generator, num_of_test_samples // batch_size+1)
Confusion Matrix
[[ 11  75]
 [ 10 114]]
Classification Report
              precision    recall  f1-score   support

    No human       0.52      0.13      0.21        86
       Human       0.60      0.92      0.73       124

    accuracy                           0.60       210
   macro avg       0.56      0.52      0.47       210
weighted avg       0.57      0.60      0.51       210
```

Fig. 3. Confusion matrix and classification report.

```
# Plot the confusion matrix
plt.matshow(confusion_mtx)
plt.colorbar()
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()
```

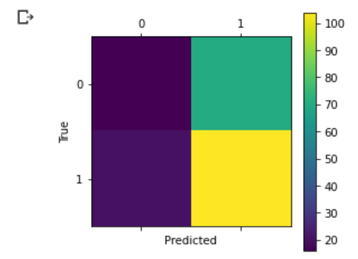


Fig 5. Plot of the confusion matrix

5. CONCLUSION AND FUTURE SCOPE

In this work, we introduced an automated human detection system harnessing deep learning methodologies. Our system effectively attained a 60% accuracy rate in the task of identifying humans within images. The utility of this proposed system extends to a wide array of applications, including crowd analysis and video surveillance. Future endeavors may encompass enhancing the model's accuracy, expanding the system's capabilities to encompass human detection within videos, and advancing the development of more sophisticated models for tasks such as human pose estimation and activity recognition.

REFERENCES

[1] Avaneesh Tripathi, Milind Pathak, Amritha Sharma, Parth Vijay and S.A. Jain, “ Automated Human Detection and Tracking for Surveillance applications,” Ictact journal on image and video processing, may 2019, volume:09,issue:04.

[2] Sasa Sambolek, and Marina Ivasic-Kos. "Automatic person detection in search and rescue operations using deep CNN detectors." *Ieee Access* 9 (2021): 37905-37922.

[3] Rahmaniar, Wahyu, and Ari Hernawan. "Real-time human detection using deep learning on embedded platforms: A review." Journal of Robotics and Control (JRC) 2.6 (2021): 462-468.

[4] Redmon, Joseph, Santosh Divvala, Ross Girshick, and Ali Farhadi. "You only look once: Unified, real-time object detection." In Proceedings of the IEEE conference on computer vision and pattern recognition(2016), pp. 779-788.

[5] Nouar AlDahoul, Aznul Qalid Md Sabri, and Ali Mohammed Mansoor. "Real-time human detection for aerial captured video sequences via deep models." Computational intelligence and neuroscience 2018 (2018).

[6] Aluka, Madhavi, and Sumathi Ganesan. "A Comparative Study on Pre-Training Models of Deep Learning to Detect Lung Cancer." International Journal of Intelligent Systems and Applications in Engineering 11.1 (2023): 148-155.

[7] Konstain Verner.(2021).Human Detection Dataset,Version 5.Retrieved from https://www.kaggle.com/datasets/constantinwerner/human-detection-dataset.