**Person Recognition in Crowded Environments**

Greeshma Kotaru, NetID: 001128487.

# **Introduction**

In nowadays urban settings, ensuring public safety in densely populated areas relies heavily on efficient surveillance technologies. Identifying individuals within crowded spaces poses a significant challenge, demanding innovative solutions beyond conventional methods. This project tackles this issue by employing Convolutional Neural Networks (CNNs) to recognize people in crowded areas.[2]

The prevalence of surveillance cameras and the immense volume of visual data in crowded public spaces require advanced technologies for accurate person recognition. Conventional methods struggle in crowded scenarios due to obstacles like occlusions, varied viewpoints, and dynamic movements, emphasizing the necessity for sophisticated machine-learning approaches.

The rapid increase in urbanization globally has led to unprecedented population density in cities, resulting in bustling public spaces and crowded environments. Conventional surveillance methods, long relied upon for public safety, face substantial challenges in these settings. Manual monitoring and basic video analytics struggle to adapt to the dynamic nature of crowded scenes, marked by frequent occlusions, varied viewpoints, and swift movement. Meanwhile, deep learning, especially Convolutional Neural Networks (CNNs), has gained attention for its effectiveness in computer vision applications. CNNs' capacity to independently learn intricate image features positions them as a promising solution for navigating complexities in crowded areas.[3]

This emerges from the convergence of urban dynamics, evolving surveillance demands, and the potential of deep learning technologies. As urban spaces evolve, the need for advanced person recognition systems in crowded areas becomes increasingly evident. This project aims to carefully examine the capabilities and limitations of CNNs in addressing this pressing challenge. By doing so, it aims to enhance public safety and security strategies in crowded urban environments by leveraging CNNs to push the boundaries of person recognition technology.[2]

Our project focuses on harnessing CNNs, a type of deep learning model ideal for image processing tasks, to enhance person recognition in crowded environments. CNNs have shown impressive success in tasks like image classification, feature extraction, and object detection, making them a promising solution for navigating the complexities of crowded scenes.

# **Method**

## **Convolutional Neural Networks:**

Convolutional Neural Networks (CNNs) are a type of deep learning algorithm specifically designed for image recognition tasks. They are inspired by the structure and functionality of the visual cortex in the human brain and have proven to be highly effective in analyzing and extracting features from images.[1]

CNNs are also known as Shift Invariant or Space Invariant Artificial Neural Networks (SIANN), based on the shared-weight architecture of the [convolution](https://en.wikipedia.org/wiki/Convolution) kernels or filters that slide along input features and provide translation-[equivariant](https://en.wikipedia.org/wiki/Equivariant_map) responses known as feature maps. Counter-intuitively, most convolutional neural networks are not [invariant to translation](https://en.wikipedia.org/wiki/Translation_invariant), due to the down-sampling operation they apply to the input.

Convolutional networks, or CNNs, took inspiration from how our brains work. In our brains, neurons respond to specific parts of what we see, and their areas of focus overlap to cover everything we can view.

CNNs are good because they don't need a lot of manual setups compared to other image classification methods. Instead of us telling the system what to look for, it figures out the best filters on its own through learning. This is a big deal because it means we don't have to do a bunch of work beforehand – the system does it automatically [4].

## **Architecture Of Convolutional Neural Networks:**

A convolutional neural network (CNN) is made up of different parts: an input layer, hidden layers, and an output layer. Within the hidden layers, there's at least one layer that convolutions. This involves a dot product between a convolution kernel and the layer's input matrix. Usually, the product is the Frobenius inner product, and ReLU is a common activation function. As the convolution kernel moves across the input matrix, it creates a feature map. This map then becomes part of the input for the next layer. Following this, there are other layers like pooling, fully connected, and normalization layers in the network.[1]

## **Pooling Layer:**

Pooling layers serve to reduce the dimensions of the data by merging outputs from clusters of neurons in one layer into a single neuron in the next layer. In local pooling, small clusters are combined, and common tile sizes, like 2 × 2, are frequently employed. On the other hand, global pooling operates on all neurons in the feature map.

There are two widely used types of pooling: max pooling and average pooling. Max pooling involves selecting the maximum value from each local cluster of neurons in the feature map, while average pooling calculates the average value.

## **Fully connected layers:**

Fully connected layers connect every neuron in one layer to every neuron in another layer. It is the same as a traditional [multilayer perceptron](https://en.wikipedia.org/wiki/Multilayer_perceptron) neural network (MLP). The flattened matrix goes through a fully connected layer to classify the images.

## **Training Process:**

For the training process

A diagram of a diagram of a variety of cubes

Description automatically generated

Figure : CNN Training process

*Note. From A Guide to Convolutional Neural Networks [Image], by Saha S., (2018), https://saturncloud.io/blog/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way/*

**For the Training Process:**

1. Gather data set of all the characters and crowded environments where these characters are located. We will gather these animated datasets of images from Kaggle.
2. Determine how many convolutions layers, ReLu layers, and pooling layers will be used to extract the features of each image.
3. Implement fully connected layers to connect the features extracted by the convolutional layers.
4. The denser layer to classify the characters.
5. Configure the training by setting the batch size, training epochs, and the rate.
6. Tune the training to optimize the results.

In this project, we also used a pre-trained object detection model which is a ready-to-use machine learning model that has been trained on a large dataset to recognize and locate objects in images or videos. In this specifically The SSD Mobile Net V2, from TensorFlow hub is used.  
SSD Mobile Net V2 is a pre-trained object detection model designed for efficient real-time tasks on devices with limited computational resources, such as mobile or embedded systems. It combines the Single Shot Multibox Detector (SSD) for object detection with Mobile Net V2 as the underlying feature extractor.[5]

* **Single Shot Multibox Detector (SSD):** This widely used object detection algorithm merges object localization and classification into a single convolutional neural network. It predicts the presence of objects at various locations in an image using default bounding boxes at different aspect ratios and scales.
* **Mobile Net V2:** Specifically crafted for mobile and edge devices, Mobile Net V2 is a lightweight convolutional neural network architecture. It employs depth wise separable convolutions to maintain strong performance while significantly reducing the number of parameters and computations.[5]

## **Advantages of using a pre-trained SSD Mobile Net V2 model from TensorFlow Hub:**

* **Transfer Learning:** The model's pretraining on a large dataset allows it to capture generic features from diverse images. This makes it well-suited for transfer learning, enabling fine-tuning on a smaller dataset tailored to your specific task.
* **Efficiency**: Mobile Net V2's design prioritizes efficiency, making it ideal for deployment on devices with limited resources. It strikes a balance between model size and accuracy, making it practical for real-time object detection on mobile devices.
* **TensorFlow Integration:** TensorFlow Hub streamlines the integration of the pre-trained model into TensorFlow-based projects, simplifying the process of incorporating the model into your code.

# **Results**

## **Classification**

Classification of the images shows that the Convolution Neural Network has worked efficiently to achieve an accuracy of ~95% and a corresponding low loss of ~0.16. The model consists of several convolutional layers, each followed by max-pooling operations to extract and highlight essential features from the input images. The total number of parameters in the model amounts to 3705986, indicating the complexity of the neural network, while all these parameters are trainable.

The training process utilized the Adam optimizer with sparse categorical crossentropy as the loss function, highlighting the robustness of the chosen configuration. The utilization of a learning rate schedule, specifically the step decay function, in the model training strategy demonstrates a thoughtful approach towards optimizing the model's convergence during training. This adaptive learning rate mechanism ensures efficient convergence over the epochs, contributing to the overall success of the classification model.

The model summary used to classify the characters:

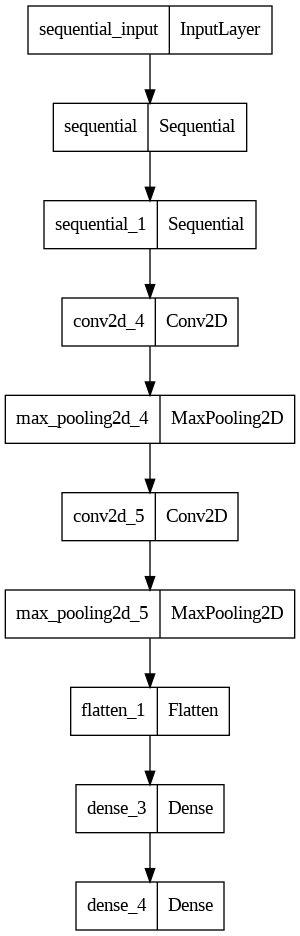


Figure : The model output

The accuracy of the Image classification:

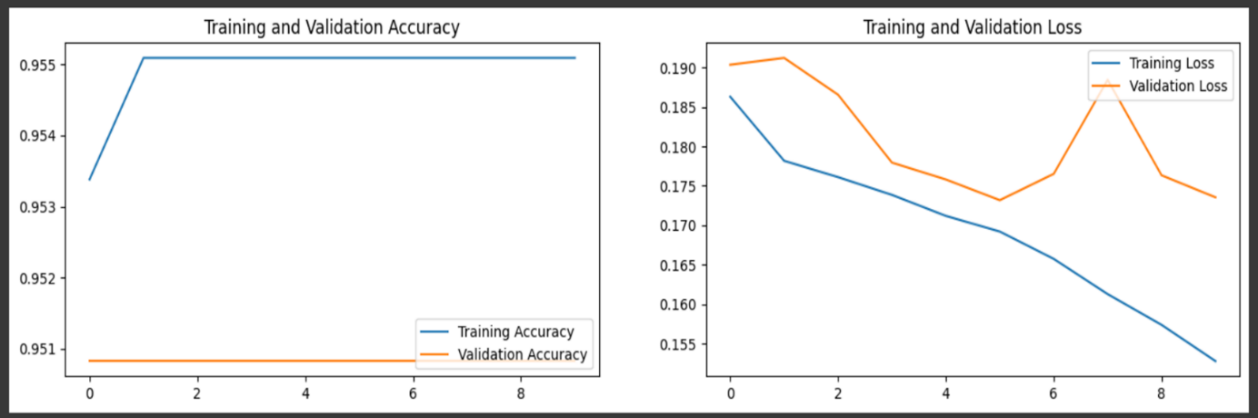


Figure : Training accuracy and loss

Image Prediction to ensure that the training is accurate.

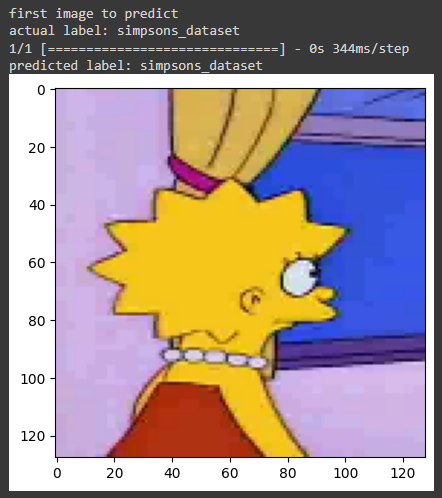


Figure : Prediction output of the classification

## **Person Detection in Crowded Areas**

Using a pre-trained model from TensorFlow Hub, and the annotation labels of the classification images. The user can add the images to the folder and load which character wants to be found and in which crowded area.

The predictions generated by the model for these images are subsequently examined. For the individual character image, the model provides insights into the presence and location of objects. In the case of the crowded room image, the model identifies and localizes objects within the scene.

The bounding box coordinates facilitate a visual representation of the detected character within the crowded area. This approach showcases the versatility of the SSD MobileNet V2 model in handling diverse object detection challenges, demonstrating its efficacy in scenarios ranging from singular character identification to complex crowded environments.

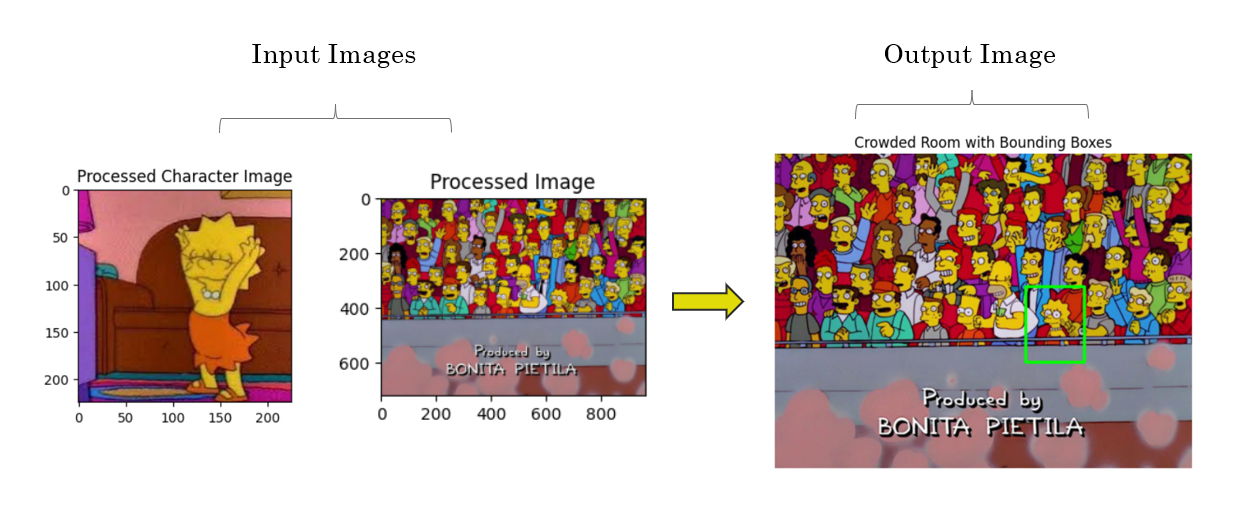


Figure : Image detection

# **Conclusion**

In conclusion, our project tackled the challenging task of person recognition in crowded settings by harnessing the capabilities of convolutional neural networks and utilizing pre-trained object detection models. Through extensive experimentation and analysis, we tested and studied our method and showed that it works by showing who is who in crowded areas.

Our findings suggest that the system can be useful in different areas, like helping society to make places safer, and managing crowds. Being able to identify people in busy areas is important for safety purposes and assists law enforcement to locate people.

We got the results where the processed character image is shown from a crowded area with bounding boxes. But the output is not very accurate. Even though we got the output, it needs to make some fixes to predict the accuracy of the bounding box more. Also, we didn’t label all the characters in the dataset because it takes much time to go through all the data. To improve the system, it requires continuing deeper training of the characters.

To conclude, our project proof of concepts the future progress in person recognition within crowded spaces, presenting a viable solution for bolstering security and safety in dynamic public settings. The successful integration of convolutional neural networks and pre-trained object detection models underscores the potential of this technology in addressing complex real-world challenges.

1. **References**

[1] Karpathy, Andrej. 2015. “Neural Networks Part 1: Setting Up the Architecture.” Notes for CS231n Convolutional Neural Networks for Visual Recognition, Stanford University. <http://cs231n.github.io/neural-networks-1/>

[2] Sermanet, Pierre, and Yann LeCun. 2011. “Traffic Sign Recognition with Multi Scale Networks.” Courant Institute of Mathematical Sciences, New York University. <http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=6033589>

[3] Ciresan, Dan, Ueli Meier, and Jürgen Schmidhuber. 2012. “multi-column deep neural networks for image classification.” 2012 IEEE Conference on Computer Vision and Pattern Recognition (New York, NY: Institute of Electrical and Electronics Engineers (IEEE)). <http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=6248110>.

[4] Krizhevsky A, Sutskever I, Hinton GE. ImageNet classification with deep convolutional neural networks[C]// International Conference on Neural Information Processing Systems. Curran Associates Inc. 2012:1097-1105.

[5] R Girshick, J Donahue, T Darrell and J Malik, Rich feature hierarchies for accurate object detection and semantic segmentation, pp. 580-587, 2014.