Hand and Associated Body Localisation

A Project as a Course Requirement for

Master of Technology in Computer Science

Bikash Ranjan Padhy

22554

Department of Mathematics and Computer Science

Prasanthi Nilayam Campus

March 2024

Dedicated At His Lotus Feet

1. Abstract

We investigate a novel problem: identifying hands and determining the position of the person that each hand detects. Numerous subsequent tasks, including hand tracking, augmented reality, gesture detection & sign language interpretation & human-computer interaction and hand contact estimation, benefit from this work. In unrestricted conditions, it can be difficult to associate hands with persons because there may be several people in the image with different overlaps and occlusions. We present an end-to-end trainable convolutional network that is able to identify a person's body location in addition to their hands. Our approach first detects a collection of hands and bodies, and then it predicts association scores between them by utilizing the overlap scores between hand and body bounding boxes. For this study, we use a new, difficult dataset called BodyHands, which consists of unconstrained photos of hands with annotations indicating the associated body locations. The dataset consists of images of crowded places, where hands and bodies are occluded too, making it very challenging. At the end, we provide the findings from the investigation and the spectrum of possibilities it raises.

SRI SATHYA SAI INSTITUTE OF HIGHER LEARNING
(Deemed to be University)

Dept. of Mathematics & Computer Science
Prasanthi Nilayam Campus

CERTIFICATE

This is to certify that this Project titled Hand and associated body localization submitted by Shri Bikash Ranjan Padhy, 22554, Department of Mathematics and Computer Science, Prasanthi Nilayam Campus is a bonafide record of the original work done under my supervision as a Course requirement for the Degree of Master of Technology in Computer Science.

.....
Dr. S Balasubramanian
Project Supervisor

Countersigned by

.....
Dr. (Ms.) Y Lakshmi Naidu
Head of the Department

Place: Puttaparthi

Date: 25-03-2024

Declaration

The work embodied in this thesis titled- Hand and associated body localization was carried out by me under the supervision of Dr. S Balasubramanian in the Department of Mathematics and Computer Science, Sri Sathya Sai Institute Of Higher Learning, Prasanthi Nilayam. The work is original and the results embodied in this thesis have not been submitted in part or full to any other University or Institute for the award of any Diploma or Degree.

Bikash Ranjan Padhy

Bikash Ranjan Padhy

22554

II M.Tech Computer Science

Prasanthi Nilayam Campus

Place: Puttaparthi

Date: 25th of March, 2024

2. Acknowledgements

I thank Bhagwaan Sri Sathya Sai Baba for being my constant guide, strength and companion all through this project and beyond,

my guide Dr. S Balasubramanian for guiding me and encouraging me to learn, explore while nudging me forward at all times,

my Parents without whose support and love my life and this project would neither have any existence nor any meaning,

the DMACS faculty of the Sri Sathya Sai Institute of Higher Learning for the constant availability and accessibility of the facilities and resources.

Contents

3. Computers –Images– Humans

With the development of computers in the the modern era of advance technology, where computers have become part of the human life as if they were another limb of ours, it is imperative that computers understand us humans through our appearance and movements to the maximum extent possible with as much accuracy and detail as possible. Although understanding human thoughts through words is equally as important, this project doesn't focus on it. Images are the best way to teach a computer how a human looks and videos show a computer how humans interact with their surroundings through motion and with other human beings as well. Luddites may say that giving a thorough understanding of the human world to computers may not serve our best interests and, in the worst case may even turn out to be counter productive for the human race, we can not limit out imagination and exploration because of concepts with very low probability and fears of a part of our society. Computers, hitherto, have seldom harmed human beings without being tempered or sabotaged deliberately, manually. Therefore, the argument can be settled by looking at history and concluding that such a case has not arisen with a concerning level of threat.

To understand humans and how they interact with their surroundings, it's important to understand our limbs- hands and legs. Hands are limbs we use to interact with most objects in our surroundings and also among each other legs, on the other hand, are more related to human motion. Leg movements predominantly require video-data for a computer to understand.

Detection of objects in images is very helpful to us in various real life scenarios. With autonomous vehicles, we see that without detecting the object in the view, a self-driving vehicle is could be a killing machine in the world. Surveillance becomes very each with another helping hand like an intelligent computer who can identify, locate and track objects, individuals and unusual activities, without this, a person might have to sit and focus on a monitor screen for hours for the same purpose- wasting

human time and energy while increasing chances of human errors and increasing labour costs. While locating tumours, lesions, an intelligent computer becomes helpful in diagnosis, and thus assists in patient treatment. To assist in industry quality control, an artificially intelligent model can be trained to remove sub-standard products. Satellite images can be scanned by an artificially intelligent model for details that are difficult or tedious for the human eye to detect- for example survivors of a natural disaster, signs of wild life, debris of a crashed plane, enemy entrapments that are too small and several such examples exist where satellite imaging plays a pivotal role. In human-computer interactions- gesture recognitions, facial recognitions, and other forms of human-computer interactions, in such natural interactions with devices detection is crucial. It becomes absolutely necessary for robots to know what objects are present in their surroundings to produce a plausible reaction. Otherwise there will be robots who might feed us flower pots as food in automated-futuristic restaurants. While monitoring monitoring ecosystems, tracking wildlife, and studying changes in land use AI models that can detect such changes aid in conservation efforts and understanding such environmental impacts. In AR/VR scenarios, the computer that is creating the surrounding images cannot be wrong in detection. It must not treat a bicycle as a motorbike and a car as a tank. Just imagine being classified as a terrorist because you own a car that looks too fancy for the model that a soldier is using in his VR headset of his sophisticated military equipment to search for a terrorist who ran away with a tank.

Hand-detection and body-detection have witnessed significant advancements, dr-riven by breakthroughs in deep learniing, CV that is, computer vision, and the avail-ability and feasibility of opens source, large-scale annotated datasets. These advances have propelled for the development of more accurate, efficient, and robust models for detecting and recognizing hands and bodies in images and videos. Although tracking an object eg body, car, hand across frames of a video has been studied extensively, associating two different objects after accurately detecting them has not been explored in the light of curiosity it as much. This project is a study on detecting and associating the hands and bodies present in an image. By associating- we mean that the neural network model in this project will be able to detect and map the correct body with an

instance of correctly detected hand as accurately and effectively as possible.

3.1 Deep Learning Architectures

One of the key drivers of progress in hand and body detection is the adoption of deep learning architectures. Convolutional Neural Networks (CNNs)[15], particularly variants like ResNets[8] have demonstrated exceptional performance in feature extraction and hierarchical representation learning. These architectures have been adapted and fine-tuned for specific tasks related to hand and body detection.

3.2 Transfer Learning and Pre-trained Models

Transfer-learning has played a crucial role in improving the performance of hand and body detection models. Researchers often leverage pre-trained models on large-scale datasets, such as ImageNet, which have more generic categories to initialize their networks. This allows models to learn general features from diverse visual data before being fine-tuned on specific tasks such as hand and body detection tasks. Transfer learning contributes to faster convergence and better accuracy, especially when training datasets for hand and body detection are limited.

3.3 Why Hand and Body Association

Hand and body association in computer vision and machine learning involves detecting, localizing, and connecting human-hands to their corresponding body parts in images and videos. This process has versatile applications, from healthcare to entertainment, making it a fundamental component of gesture recognition, sign language interpre-

tation, human-computer interaction, and augmented reality. Accurate hand-to-body mapping is essential in these fields, enhancing human-computer interfaces and their ability to understand and respond to human actions.

3.4 Motivation

In CV or computer vision, hand analysis is a crucial issue with applications in gesture, action, sign languagee recognition, human comprehension, and action. Applications involving augmented and virtual reality also depend on the visual analysis of human hands. This activity is helpful for comprehending scenes and identifying actions, particularly in photographs and films that have numerous people. For instance, recognizing individuals is useful when deciphering hand gestures used in human-to-human communication- knowing who is speaking what is very important. Hand-body association supports individuals using hand-held tools in production environments and aids in the designing of safety applications. We need to map the right body to the hand in domains like inter-human interactions in the same AR environment, to find the culprit in a very long surveillance footage, to interpret sign language where many humans are interacting.

Hand and body detection in an image have been worked on extensively for years. We have seen models that detect keypoints in humans, faces present in the image, bodies that appear in the image, hands that appear in the image along with hand-gestures that appear in the image etc. While hand detection, hand position estimation & hand tracking and hand contact estimation have all been researched extensively by the CV-Computer Vision community, hand-body association has not received much attention.

4. Problem Statement

Our goal is to create a DNN- deep neural network model that can recognize hands and bodies, correlate each detected hand instance in the input image with the matching body, and is end-to-end trainable. Given an image I with dimensions H $\times$ W $\times$ 3, our objective is twofold:

- Identify and locate the bounding boxes for hands (H) and bodies (B) within the image. Each bounding box, representing the spatial extent of an object, is described by a four-dimensional vector indicating its position.
- Associate each detected hand $h \epsilon$ H, with a body $b \epsilon$ B while satisfying the following constraints:
  - Each hand $h \epsilon$ H is associated with exactly one body $b \epsilon$ B.
  - Each body $b \epsilon$ B can be associated with at most two hands in H.

  The model is composed of convolutional layers.

5. Studying the existing literature

5.1 ResNet

The basis of all the RCNN papers dicussed above is the ResNet[8] architecture. This architecture was the first one to use skip connection to allow the backward flow of

gradients during the backpropagation in an epoch while training.

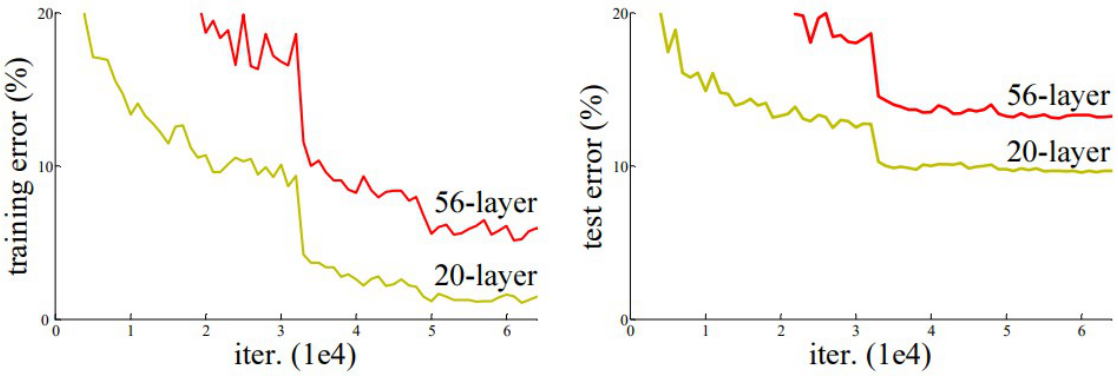


Figure 1: Performance: 20-layer versus 56-layer architectural comparison[8]

As we can see from the graph, contrary to the belief that deeper architectures perform better than shallower ones, both the training and testing errors are higher for a 56-layer model than those of a 20-layer model.

The authors concluded that the explanation of the error rates, after a thorough analysis, is the vanishing/exploding gradients for deeper networks. A novel architecture known as the Residual Network was introduced by Microsoft Research experts in 2015 through the proposal of ResNet. This was a sensation at that time in the world of deep neural networks

Residual Network: They intorduced the idea of residual blocks in the neural netwowk architecture to address the issue of vanishing/exploding gradients. This network makes use of a method known as skip connections. By omittting a few levels in between, skip connections link the activationns of one layer to deeper layers. This is a

residual block's foundation. These residual blocks are stacked to create resnets.

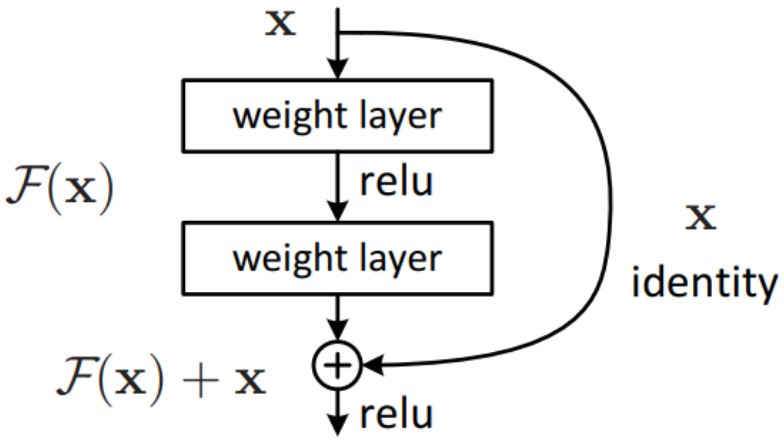


Figure 2: The residual block[8]

The idea behind this network is to let the network fit the residual mapping rather of having layers learn the underlying mapping. Thus, let the network fit rather than using, say, H(x), the initial mapping: F(x) := H(x) - x which gives H(x) := F(x) + x.

$$F(x) := \mathcal{H}(x) - x$$

$$\mathcal{H}(x) := F(x) + x$$

Figure 3:

These simple yet significant skip connections solved the problem of vanishing gradients in deep neural networks and thus gave birth to a new series of RCNN models that we saw above, and more that have not been covered here. The first resnet that was desgined to compare between vgg networks and resnets is shown below in the figure.

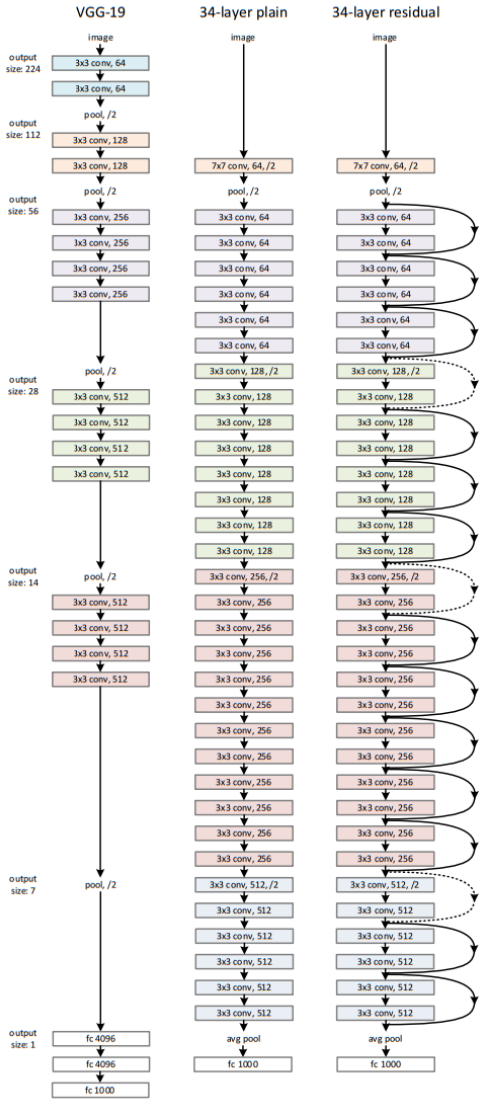


Figure 4: Plain Net-34 vs ResNet-34 architecture[8]

What makes ResNet such a prominent model? We can determine that this model was very successful because, in the 2015 classification competition, its ensemble placed first with an error of only 3.57% at the ILSVRC-ImageNet Large Scale Visual Recognition Challenge. Furthermore, in the ensuing ILSVRC and COCO competitions of 2015, it also won first place in ImageNet detection, ImageNet localization, COCO detection, and COCO segmentation.

5.2 Region-Based CNN

The paper on region-based convolutional networks RCNN[4] is one of the first papers to start talking about object detection along with considerably good results but the model used a very time consuming and computation extensive method of selective search. Selective search is a method that was used by the authors of this paper to select proposed bounding boxes(from about 2000 candidates) using SVM(Support Vector Machines) for classification. The boxes are hierarchically grouped based on their color, size, shape, texture.

For each proposed region, 4096 dimension features are extracted using a feature extractor; they used alexnet for feature extraction. To reject as many region proposals as possible, they used greedy non max suppression which rejects all the boxes that have a IoU(Intersection over Union) less than 0.5. The IoU metric is used even now as a very effective tool to measure the accuracy of the proposed regions, more IoU score for a box means the proposed box and the target box have a higher overlap/ intersection, ie more accurate proposals. Despite having slow training speed the fact that RCNN model performed 30% better gave it the spotlight and it attracted the attention of many researchers of that time.

5.3 Fast Region-Based CNN

The FastRCNN network model[3] by R. Girshick improved upon the features that were making the training of RCNN very slow. They used a single stage training process

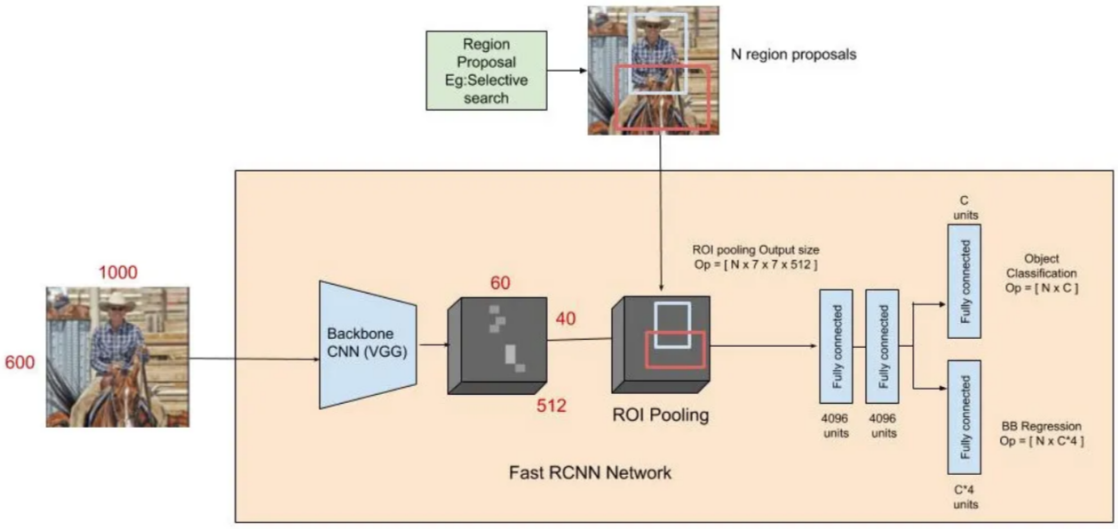


Figure 5: RCNN Architecture[4]

instead of the multistage RCNN model. All the network layers are updated during training. They used VGG network for feature extractioon. They used a batch size of 2 and each batch has 64 region proposals. Like RCNN, 25% of these proposals have IoU more than 0.5, ie they strongly match a ground truth region/ object. A box with label=0 is a background class. They could achive simultaneous training of the entire module by using only CNN processing upto ROI Pooling which enabled shared computation and memory for all the ROIs for an image. Following the CNNs in the ROI Pooling layer there are Box regression and softmax loss function for box classification and regression.

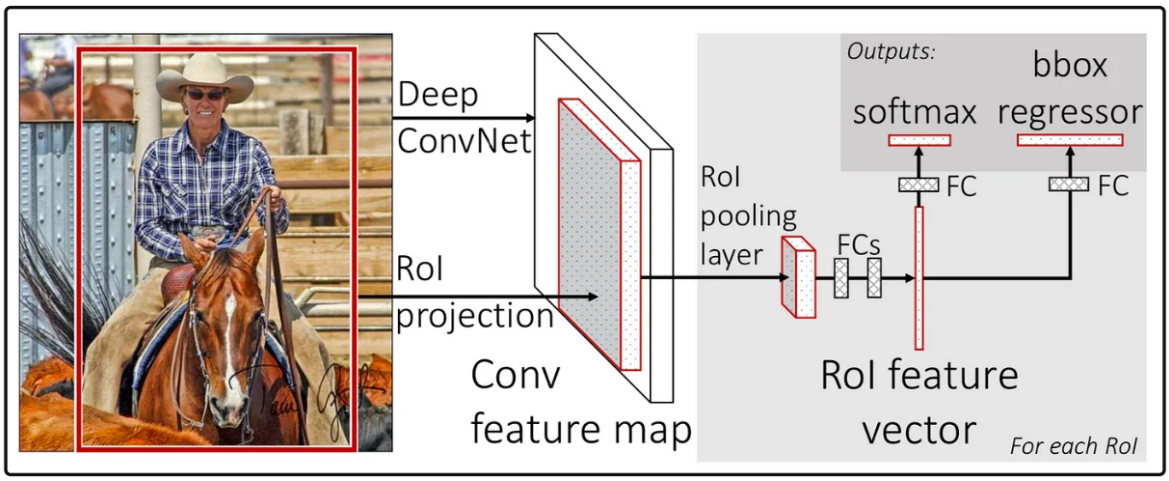


Figure 6: Fast RCNN Architecture[3]

5.4 Faster Region-Based CNN

FasterRCNN [16], on the other hand, uses a Region proposal Network that makes it very efficient in terms of memory and computation , reducing the search space substantially. They used a Region Proposal Network which is essentially a fully convolutional network that generates anchor boxes with many aspect ratios and scales. It tells the later layers of the model where to look. That was the most important contribution of Faster-RCNN. Next, it introducedd the concept of achoor boxes- while other methods included the use of pyramid of images or pyramid of filters.

5.5 Mask Region-Based CNN

MaskRCNN [7] further improved the detection results by using mask bits, ROI Align instead of ROI Pooling and combiniing the mask generation process and box reegression for better detection accuracy. It outputs a binary mask along with bounding box coor-

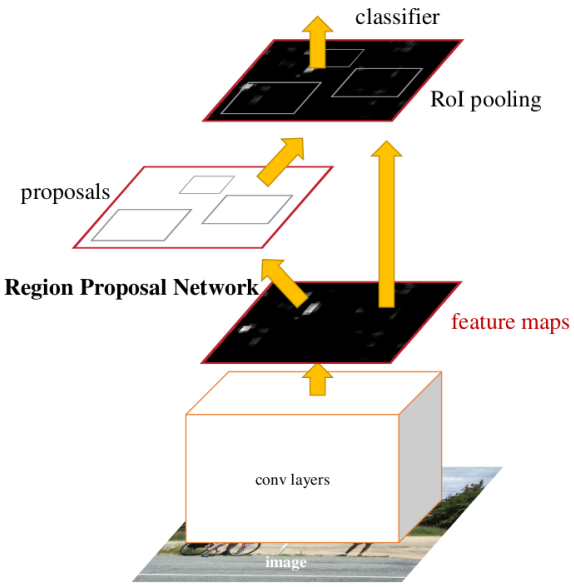


Figure 7: Faster-RCNN Architecture[16]

dinates and the corresponding class label. The masks have a binary cross entropy loss associated wtih them. This contributes to instance segmentation. The ROIAlign takes the float values and uses bijective interpolation to get the most accuarate pixel values to get box regression deltas and thus the IoU score is better than that of ROIPooling for that region. Furthermore, as the total loss is the sum of the mask loss, box loss, and classification loss ie, L = Lcls + Lbox + L$mask$, the mask head serves as a performance enhancer for the overall detection in addition to being an adjunct to the instance segmentation phase.

5.6 Detectron2

Detectron2 is a cutting-edge, open-source deep learning neural networks' framework, meticulously crafted for the task of object detection and instance segmentation in images. Its origin is in a foundation built with PyTorch. Pytorch is a popular framework celebrated for its prowess in machine learning and deep learning achievements. Through this symbiotic relationship, Detectron2 inherits PyTorch's pliability and efficiency, providing us an amalgamation of innovation and performance of the best of the pytorch's and some of its own, modules, classes and methods.

Modular structure: The coreof Detectron2 is a modular architecture which is

very user-friendly in terms of its adaptability and extensibility. It has components that are meticulously designed- backbone networks for feature extraction to region proposal methods for anchor box generation, matching, classification, and regression- invites us welcomingly to create solutions tailored precisely to our needs. This modular structure acts as a catalyst for creativity, and higher productivity; thus empowering us to construct pipelines fine-tuned to our customised needs, specific to the use cases.

Pre-trained Models: Detectron2 is adorned with a good number of pre-trained models, that have been trained on colossal datasets like COCO. This bolsters transfer learning, creating an efficient shortcut for us to train models not from scratch but from a point of head-start. This serves as a springboard that helps us towards unparalleled performance with very minimal efforts. Through transfer learning, the fastidious and strenuous journey from inception to deployment is shortened, thus strengthening the understanding and implementation of cutting-edge object detection systems.

Efficient: Detectron2 is all about efficiency, and its toolkit includes optimised pipelines for both training and inference. Using the combined power of several GPUs, distributed training guarantees quick convergence on large datasets, and GPU acceleration speeds up inference so that models go through inference stages more quickly than ever before.

Sophisticated methods: A intricate yet well-integrated set of methods adds even more depth to Detectron2's quiver of capabilities. There are a plethora of backbone architectures to choose from, each offering a distinctive viewpoint on the visual world. Techniques for enhancing datasets, from the commonplace to the exotic, are prepared to breathe new life into them and add a diverse range of elements. Sophisticated assessment criteria, carefully calibrated to the subtleties of object recognition, act as sentinels, warding off mediocrity and complacency.

Within the hazy world of open-source, Detectron2 is an example, a lighthouse of creativity, maintained by the combined efforts of a passionate community. Its path, a monument to the unwavering spirit of teamwork, is revealed with every contribution, pull request, and problem that is fixed. The potential of Detectron2, a portent of a future in which object detection rises above its present limitations to unprecedented

heights of capacity and comprehension, grows along with the community.

5.6.1 Bounding Boxes and Detectron2

Detectron2 is a prime example of current computer vision, executing object recognition using bounding boxes with careful precision and efficiency. Its core is an advanced architecture that is adept at identifying the shapes of things in pictures and enclosing them in bounding boxes.

Step 1: With the help of convolutional neural networks (CNNs), Detectron2 sets out on a feature extraction adventure, carefully examining each pixel. These retrieved features operate as the foundation upon which the object detection structure is built, providing the framework with the information it needs to traverse the visual terrain.

Step 2: The first step in the procedure is to identify candidate regions of interest (ROIs), each of which could be a signpost for the existence of an object. To determine their fidelity, these ROIs are subjected to a stringent assessment that examines them in light of numerous criteria. Bounding boxes are applied to those that are judged worthy, and these boxes precisely outline the detected object's spatial extent.

Step 3: Still, the trip doesn't end with simple discovery; it goes further: classification and refinement. Every object that is spotted is categorised and given a label that summarises what it is among the visual noise. Bounding boxes are also refined, fine-tuned with mathematical accuracy to capture the actual form of the object.

Detectron2 weaves a tapestry of object recognition through this complex combination of convolution, classification, and refining; each bounding box is a monument to its strength and complexity. This framework creates a visual symphony in which objects appear out of the pixelated ether, accompanied by bounding boxes that announce their arrival in the digital domain.

Feature Extraction:

To extract hierarchical features from the input image, a sequence of convolutional

operations is performed using the pre-trained backbone network (such as ResNet). This backbone is responsible for goinf through all the pixels in the image get the feature maps that most likely will contribute in the bounding box classification, regression tasks.

Region Proposal Network:

Region proposals, or candidate bounding boxes that are probably to contain objects of interest, are generated by the RegionProposalNetwork. These suggestions are chosen based on the convolutional features' chance of containing objects. Each proposal box has an objectness score and regression deltas associated with itself. a high objectness score means that the box has a higher chance of containing an object as mentioned in the target/ ground truth annotations. Regression deltas are the values in coordinates terms that tell how much the box should move and in which direction for it to be more aligned with the target ie the ground truth box. The box regression loss and box classification loss help this part of the network train.

ROI Pooler:

By extracting features from the convolutional feature maps corresponding to each region proposal (ie the Region of Interest, ROI) and spatially aligning them to a given size, the ROI Pooler plays a critical role in object detection. Regardless of their initial sizes or aspect ratios, characteristics taken from various sections of an image can be processed and compared uniformly thanks to this spatial alignment.

Usually, a rectangular portion of the input image correlates to a region proposal that is generated. Convolutional feature maps, however, could differ from the input image in terms of aspect ratio and spatial resolution. The ROI Pooler corrects this misalignment by converting the region proposal's coordinates to match the grid of the feature maps. There are 2 famous approaches to do this we'll see them in detail in further sections of this thesis.

The ROI Pooler executes a pooling operation to extract features inside the area once the coordinates of the region proposal are aligned with the feature maps. In

this process, the region is divided into a predetermined number of spatial bins, and statistics (such average/ max pooling) are computed separately within each bin. A fixed-size feature map that depicts the proposed region is the end product.

The ROI Pooler may use interpolation algorithms (eg bilinear interpolation) to handle sub-pixel accuracy in order to guarantee that the pooling operation appropriately captures the spatial information within the region proposal. Even in cases when the borders of the region proposal do not precisely match the feature map grid, this interpolation guarantees that features from the original convolutional feature maps are sampled appropriately. The interpolation operation tries not to exclude relevant cells and tries to exclude as many irrelevant cells as possible.

The ROI Pooler is engineered to achieve computational efficiency by utilizing optimal pooling operation implementations to swiftly extract features from several area proposals concurrently. For real-time or high-throughput object identification applications, when processing speed is critical(eg self driving vehicles), this efficiency is essential.

The ROI Pooler in the Detectron2 pipeline typically operates after the Proposal Network generates region proposals, but before those proposals advance to the subsequent stages for bounding box regression and classification. In order to facilitate future processing, this guarantees that the features derived from the region suggestions are suitably aligned and standardized in the expected shape.

Classifcation- BoxHead: An essential part of Detectron2's object detection pipeline is the BoxHead. It is positioned after the ROI Pooler and gets the spatially aligned features that are taken out of every region proposal. This guarantees that the input features are customized to the objects of interest that the region proposal network has discovered.

The moment as the ROI Pooler sends the pooled features to the BoxHead, it begins a sequence of modifications aimed at extracting relevant data needed for object classificaiton. Frequently, these conversions entail putting the features through fully linked layers, which educate themselves to extract relevant discriminative characteristics for the job at hand.

The BoxHead adds non-linearity to the feature representations by applying non-linear activation functions, such as Sigmoid or ReLU (Rectified Linear Unit), after the feature transformation stage. This non-linearity improves the model's capacity to identify intricate links and patterns in the data, allowing for more precise classification.

For every region proposal, the BoxHead produces a probability distribution over preset object classes. The BoxHead turns ie transforms the extracted characteristics into class probabilities, or the chance that each proposition will fall into a different object category, using more fully connected layers and softmax activation function.

Loss: The BoxHead calculates the classification task's loss concurrently with the classification result. This loss measures the difference between the ground truth labels and the projected class probabilities, giving the model feedback during training to improve its classification performance.

The BoxHead works in tandem with the bounding box regression part of the object identification pipeline, even though its primary goal is classification. The object detection system's final output is formed by pairing the class probabilities supplied by the BoxHead with the revised bounding box predictions obtained by the bounding box regression.

Backpropagation is used throughout the training process to adjust the BoxHead's parameters, which maximizes the network's capacity to recognize and classify items correctly. Based on the calculated classification loss, this optimization procedure iteratively modifies the parameters using optimisation methods like stochastic gradient descent (SGD) or Adam.

Bounding Box Regression: The set of layers in FastRCNNOutputLayers as a whole is responsible for bounding box regression in detectron2's pipeline. The FastRCNNOutputLayers are located after the feature extraction and classification phases of Detectron2's object detection pipeline. Its job is to refine the bounding box coordinates suggested by the region proposals after receiving the spatially aligned data from the ROI Pooler.

Bounding box regression is fundamentally embodied by the FastRCNNOutputLayers. It functions by forecasting changes (offsets ie bounding box regression deltas)

to the coordinates of the region proposals produced by the network of region proposals. By making these changes, the original bounding box coordinates should be more closely aligned with the actual spatial extent of the objects- ie with the target bounding boxes in the annotated ground truth.

The features recovered by the ROI Pooler are transformed within the FastRCNNOutputLayers prior to being used in bounding box regression. This is done in order to extract relevant data needed for regression, this usually entails running the features through fully connected layers and using non-linear activation functions, such ReLU.

The expected offsets (modifications ie bounding box regression deltas) to the bounding box coordinates are the main output of the FastRCNNOutputLayers. The bounding box coordinates are adjusted for every region suggestion, and this process improves the localization accuracy of the items that are detected.

Concurrently with the regression outputs, the FastRCNNOutputLayers computes the loss associated with bounding box regression. By measuring the difference between the target or ground-truthh bounding box coordinates and the predictted bounding box modifications, this loss gives the model feedback during training to improve its regression performance.

Although the main focus of the FastRCNNOutputLayers is bounding box regression, it works in tandem with the object detection pipeline's classification step, that is, with the BoxHead. The final output of the object detection system is formed by pairing the class probabilities supplied by the classification head with the revised bounding box predictions obtained by the bounding box regression.

Backpropagation is used during training to adjust the FastRCNNOutputLayers' parameters, which maximizes the network's capacity to precisely refine bounding box coordinates. Based on the calculated regression loss, this optimization procedure iteratively modifies the parameters at each epoch using optimisation methods like stochastic gradient descent (SGD) or Adam.

NMS and Postprocessing The object detection pipeline in Detectron2 uses Non-Maximum Suppression (NMS) to improve bounding box predictions and eliminate redundant detections following the BoxHead and bounding box regression stages.

The way NMS works is that iteratively, it chooses the bounding box with the highest confidence score (classification score) and suppresses (discards) bounding boxes that overlap significantly with the selected box but have lower confidence scores. This overlap is quantified by IoU, or intersection over union. Through this procedure, duplicate detections are removed and a more condensed set of final predictions is produced by ensuring that only the most confident and non-overlapping bounding boxes remain.

Following NMS, further post-processing procedures may be used to fine-tune the output of the object detection pipeline and improve overall system performance. One frequent post-processing step is to filter out bounding boxes with confidence ratings less than a specific threshold. This thresholding guarantees that only detections with appropriate confidence are maintained, lowering the risk of false positives. Another post-processing phase could include applying constraints or heuristics to the bounding box predictions in order to enforce additional constraints or rules. For example, bounding boxes having aspect ratios outside of a preset range may be removed, or spatial restrictions may be used to ensure that bounding boxes remain within the image's limits. Furthermore, post-processing processes may include techniques like instance segmentation, which delineates the borders of identified objects by breaking them down into individual pixels or areas.

NMS and post-processing processes are seamlessly included into the object detection pipeline after the BoxHead and bounding box regression phases. They work with the revised bounding box predictions created by these phases to ensure that the system's final output is refined, accurate, and optimized for downstream tasks like object tracking or semantic segmentation.

The settings and hyper-parameters controlling the NMS and post-processing stages can be adjusted and modified during training, just like other parts of the object detection pipeline. The system performance can be maximized by identifying the ideal parameter settings through the use of techniques like grid search or cross-validation.

In the area of detection and segmentation of images, MaskRCNN [7] has performed very well with the help of ResNets[8] and Feature Pyramid Networks [11]. MaskRCNN is actually based on the FasterRCNN network model[16], which is based

on the FastRCNN network model[3] by R. Girshick, which substantially improved the selective search method used by RCNN[4] network. FastRCNN uses selective search along with a Roi Pooling. Whereas FasterRCNN uses a Region proposal network[16] along with Roi pooling for detection bounding box regression and classification. One of the novel ideas of He et al. was the Roi align [7] which uses bilinear interpolation [6] instead of taking floor of the float value of the feature maps during the adaptive pooling operation. The globally acknowledged Deep Learning textbook [5] explains the basics of deep learning and advanced topics alike with clarity. Detectron2 API, created by Facebook AI Research, was used in an attempt to implement the technique described in the paper for **Hand Detection and Hand-Body Association in the Wild by Narasimhaswamy et al** [14].

6. Methodology

6.1 Dataset

While a number of datasets (e.g., [12, 13]) have annotations for the locations of the hands, the associated body locations are not marked. Using human pose datasets [1, 2, 10], which contain the positions of human body joints, is an additional choice. Nevertheless, hands' bounding box annotations are absent from these datasets. A dataset with hand and body locations is proposed by Zhou et al. [17]; however, the dataset is developed indoors. Furthermore, the public cannot access their dataset. By hand-keynoting photos from the COCO dataset, Jin et al[9]. create the COCO-WholeBody dataset.

The **Bodyhands public dataset** offers high-quality, unrestricted images with annotations that show where hands and bodies are located in each picture. This large dataset comprises 20,490 assigned photographs, of which 1,629 are put aside for testing. This dataset is a valuable resource for training and assessing models, with roughly 57,000 hand occurrences and 63,000 body instances. The different levels of overlap and occlusions in the photos make it difficult to navigate this dataset, which

might cause issues for models trying to learn from the data efficiently.

With respect to annotations, every image has an associated XML file named after it. Every object in the image is characterized with attributes in these annotation files, such as name (which indicates if it is a hand or a body), ID number, bounding box coordinates (which show the object's location in the image), and segmentation coordinates (if relevant). In order to create relationships between hands and bodies in the dataset, each object's name and body ID are taken into account jointly. Furthermore, when a hand is the only object in the picture, that hand counts as the body for that specific hand instance.

Figure 8: An example image from the dataset

6.2 Model Architecture

We use a ResNet50 as a backbone network and a FPN (Feature Pyramid Network) to extract feature maps at various stages.

```
</object>
<object>
        <name>body</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <difficult>0</difficult>
        <body_id>2</body_id>
        <bndbox>
                <xmin>671</xmin>
                <ymin>119</ymin>
                <xmax>925</xmax>
                <ymax>681</ymax>
        </bndbox>
</object>
<object>
        <name>hand</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <difficult>0</difficult>
        <body_id>2</body_id>
        <bndbox>
                <xmin>686</xmin>
                <ymin>130</ymin>
                <xmax>730</xmax>
                <ymax>195</ymax>
                <x1>722</x1>
                <x2>686</x2>
                <x3>694</x3>
                <x4>730</x4>
                <y1>195</y1>
                <y2>190</y2>
                <y3>130</y3>
                <y4>135</y4>
        </bndbox>
</object>
```

Figure 9: An example XML file structure from the dataset

Feature Pyramid Network: An attribute Neural network architecture such as the Feature Pyramid Network (FPN) in PyTorch are frequently employed for applications involving object detection and semantic segmentation. By utilizing features at several resolutions, it seeks to overcome the difficulty of detecting things at various scales.

Backbone for Feature Extraction: An example of a feature extraction backbone in an FPN architecture is a Convolutional Neural Network (CNN) such as ResNet or VGG. After processing the input image, this backbone creates a set of feature maps with various spatial resolutions. Bottom-Up Pathway: In the bottom-up route, the feature maps from the backbone are processed to form a series of feature pyramids. This entails applying convolutional layers or other techniques to augment the representation of features at different scales. Top-Down Route:

Higher-resolution feature maps are produced concurrently in the top-down pathway by applying interpolation andor upsampling techniques to the feature maps. These higher-resolution feature maps are then blended with the matching lower-resolution fea-

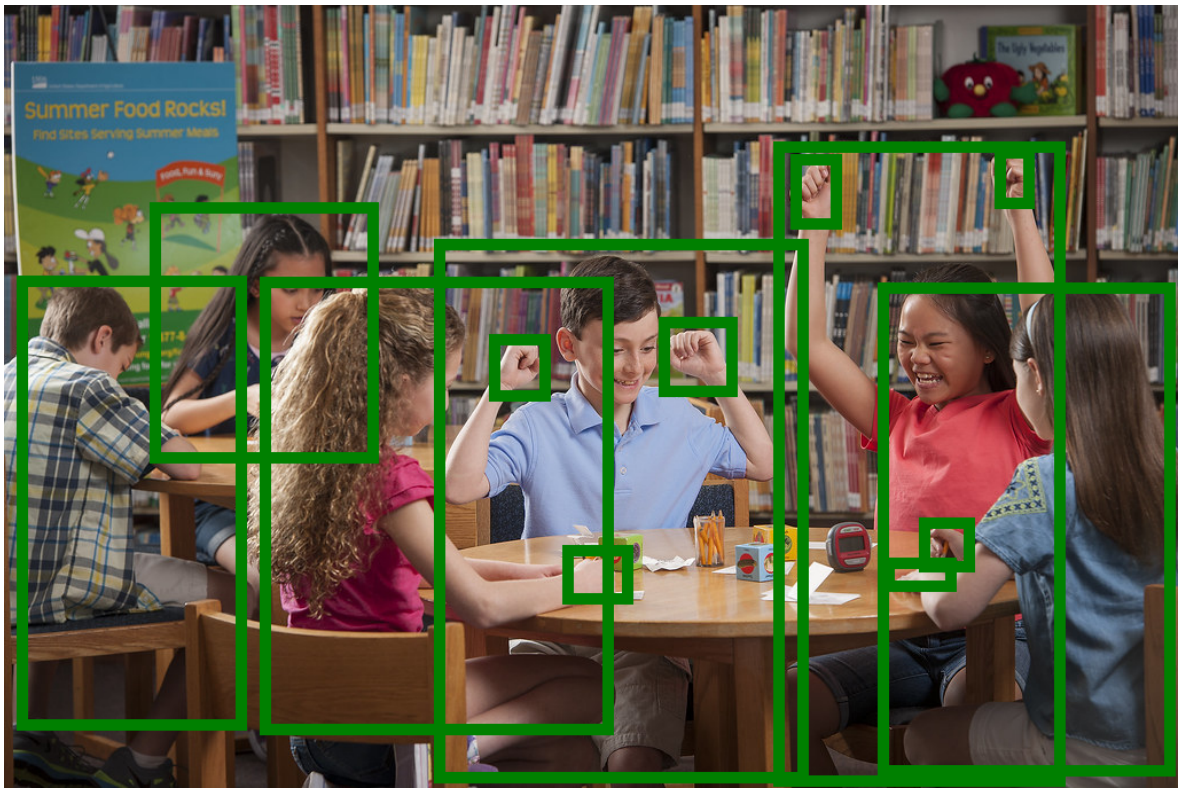


Figure 10: An annotation-visualised image from the dataset

ture maps from the bottom-up pathway. Fusion of Features: Feature maps from the top-down pathway and the bottom-up pathway are fused at each level of the pyramid. This fusion procedure often utilizes element-wise addition or concatenation to merge features from multiple resolutions effectively. Skip Connections: The FPN's final output consists of feature maps at multiple resolutions that form a feature pyramid. These feature maps are then used as input to subsequent tasks like object detection or semantic segmentation. Skip connections are commonly used to connect feature maps from the backbone to corresponding levels of the pyramid. This allows the network to access features at different resolutions and capture both local and global context. Layers of Output: A feature pyramid is formed by feature maps at various resolutions in the FPN's final output. Then, these feature maps are fed into other tasks, like semantic segmentation or object detection. Training and Optimisation: Typically, annotated datasets are used in conjunction with supervised learning approaches to train FPNs. To minimize an appropriate loss function, such as mean squared error for regression tasks or cross-entropy loss for classification tasks, they are optimized using methods

like stochastic gradient descent (SGD) or Adam.

The output of this hybrid feature extractor is sent to the **RPN** (Region Proposal Network). RPN takes anchor boxes and the feature maps from feature extractor as input and produces region proposals- which are candidate bounding boxes for bounding box regression. Each anchor box with an object in it has non-zero objectness score and is called a positive anchor box. In the case wherein multiple boxes intersect or overlap with a ground-truth box, NMS (Non-maximum Suppression) is used to get rid of redundant anchor boxes. Regression deltas for each anchor box are values by which coordinates of the box should change to get more intersection with the ground-truth boxes. The ground-truth box with which an anchor box overlaps is called the target box for that anchor box.

NMS: Non-Maximum Suppression (NMS) is a fundamental post-processing technique used in object detection tasks to reduce redundant bounding boxes and pick the most confident predictions. In PyTorch, NMS can be implemented using the torchvision.ops.nms method. We elaborate briefly how NMS functions in PyTorch for detection. Input: NMS takes as input a collection of bounding boxes along with their accompanying confidence scores. The coordinates of each bounding box (xmin, ymin, xmax, ymax) are used to represent it, and the confidence score indicates how likely it is that the box contains an object of interest. Rank Order Using confidence Score: Sorting the bounding boxes in descending order according to their confidence scores is the first stage in the NMS process. This guarantees that the boxes with greater confidence ratings are handled initially. Picking the Box with the Most Confidence: The first candidate to be included in the final list of detections is the bounding box with the highest confidence score. When compared to other boxes, this one is regarded as a ”seed” or ”anchor.” Limiting and IoU Estimation: NMS compares the Intersection over Union (IoU) of each remaining box with the greatest confidence box first. The overlap between two bounding boxes is measured using the area of union (AoU), which is computed as the area of intersection divided by the area of union. Repetitive and suppressed boxes have an IoU larger than a predetermined threshold, usually 0.5 or higher. Elimination of Superfluous Boxes: When redundant boxes greatly overlap the chosen anchor box, they are suppressed, or taken out of the equation. Among the

overlapping boxes, only the box with the greatest confidence score is kept. Repeat or Iterate: Until all boxes have been compared to one another and superfluous boxes have been suppressed, the process is repeated. As a result, a collection of non-overlapping bounding boxes with individual object detections is produced. Output: A list of chosen bounding boxes, along with a confidence score for each, is the NMS's output. These boxes reflect the final detections after reducing redundancy. Implementation and Documentation: Within PyTorch, the bounding boxes, their confidence ratings, and the IoU threshold are inputs to the torchvision.ops.nms function. Following NMS, it returns the selected box indices.

Next, the proposed regions with different shapes are resized to a particular size to be input to the box and mask heads, using adaptive pooling in the ROI Align part of the architecture. MultiScaleROIAlign is used for this project from the pytorch library.

ROI Pooling: One method for extracting fixed-size feature maps from feature maps of different sizes is ROI pooling. In object identification frameworks, it is usually used after the Region Proposal Network (RPN). Feature maps produced by the backbone network and a collection of region proposals (bounding boxes) produced by the RPN are the inputs used in ROI pooling. Each region proposal is divided into a fixed-size grid by ROI Pooling, which then applies max pooling independently within each grid cell. For every region proposal, this procedure generates fixed-size feature maps, irrespective of the region's initial size. Classification and bounding box regression layers thereafter get the fixed-size feature maps produced by ROI Pooling.

Multi-Scale ROI Alignment: In order to solve the misalignment problems brought on by quantization in RoI Pooling, Multi-Scale RoI Align is an enhancement over RoI Pooling. Multi-Scale RoI Align uses bilinear interpolation to sample features at non-quantized positions inside the region proposals, as opposed to quantizing the region proposals into a fixed-size grid. Multi-Scale ROI Align can better maintain spatial information and align features with object boundaries because to this interpolation, which enhances detection performance. Similar to how ROI Pooling works, Multi-Scale ROI Align adds interpolation steps to sample features at sub-pixel locations within each grid cell. Both Multi-Scale RoI Align and ROI Pooling are implemented as modules in

the torchvision of PyTorch.operations module:

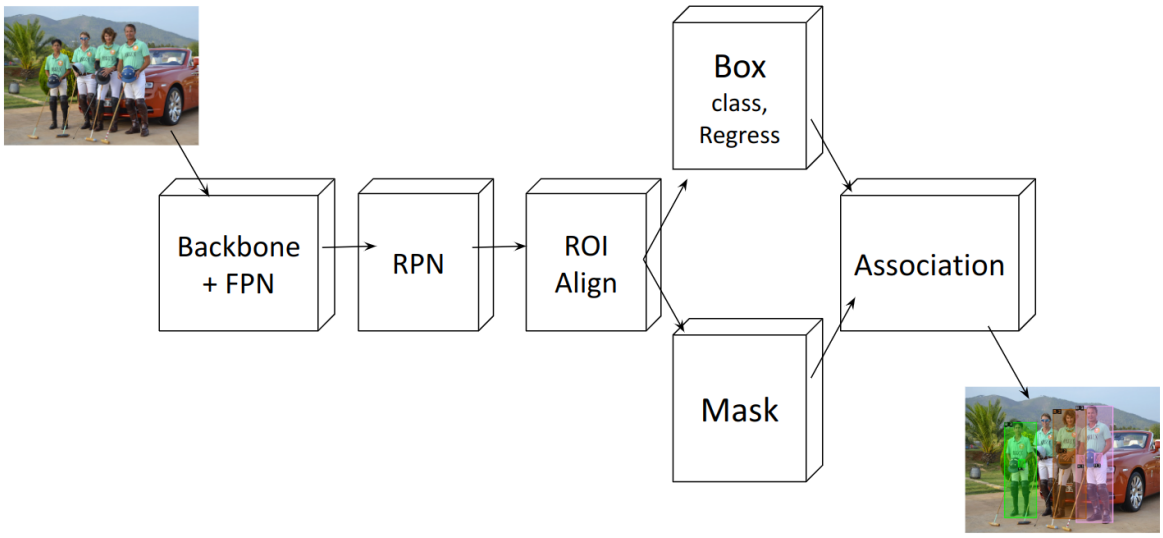


Figure 11: Model Architecture

The RoI Pooling is implemented by torchvision.ops.roi_pool. Multi-Scale ROI Align is implemented by torchvision.ops.roi_align.

Unlike ROI pooling, ROI align does not discard much of relevant information because it does not use quantisation, it uses float values instead of integer values to include/ exclude pixel values that align with the region proposals in the feature map. This is calculated using bilinear interpolation before the pooling operation. The 7x7 fature map output of ROI align is then sent to Box head and Mask head. The mask head consists of deconvolution layers that upsample the mask to original image size since the mask should be on the image whose size is bigger than the feature map. The box head is responsible for processing region-based features to perform classification and bounding box regression tasks for object detection.

The Mask Head and Box Head are two essential parts of our model that are in charge of producing instance masks and bounding box predictions, respectively. We discuss each of these elements in little more detail.

Mask Head: The Mask Head is a module in the Mask R-CNN architecture responsible for creating pixel-wise segmentation masks for each detected object instance.

To create high-resolution feature maps, the Mask Head in our model consists of a sequence of convolutional layers followed by upsampling operations. After that, a final convolutional layer with a sigmoid activation function is applied to these feature maps, producing a final binary mask for each instance of a recognized object. In order to maximize the mask prediction during training, the output masks are compared to ground truth masks using an appropriate loss function (such as binary cross-entropy loss).

Box Head: Within the Mask R-CNN architecture, another module called Box Head is in charge of estimating bounding boxes and objectness scores for every instance of a detected object. The Box Head in our model is composed of many fully connected and/or convolutional layers layered after a fully connected layer. Bounding box coordinates (xmin, ymin, xmax, ymax) and objectness ratings for each suggested region of interest (RoI) make up the Box Head's output. In order to maximize the box prediction during training, the output bounding boxes are compared to ground truth bounding boxes using an appropriate loss function (such as smooth L1 loss or IoU loss).

The model we employ incorporates the Mask Head and Box Head into the overall Mask R-CNN architecture, along with auxiliary components, a backbone network (in this case, ResNet-50), and a Region Proposal Network (RPN) for producing region proposals. A combination of loss functions that take into account both mask prediction and bounding box regression tasks are used to train the model end-to-end. Together, these elements enable Mask R-CNN to identify objects, provide precise bounding boxes, and create segmentation masks at the pixel level for each object that is discovered.

Association: After the detection of hands and bodies is done by the detector part of the model, the bounding boxes along with their class-labels are checked for areas of overlap. Since the dataset has only that kind of images and wherein hand-boxes are completely inside the body-boxes and the model has learnt to detect bodies and hands in the same manner, the next part of our model checks for hand box and body box overlap. For a hand, all the body boxes and also its own hand-box(in the case where there are no bodies but only hand(s) in the image) are candidates for association. The candidate-box with the highest area of overlap with the hand box gets associated with the hand-box. The association is done on the indices of the boxes- eg hand1 and hand1

get associated with body1; hand2 and hand2 get associated with body2 and so on. This is done using bipartite mapping aka Hungarian mapping. This has been implemented in the scipi library as "linear_sum_assignment" function in the "optimize" module. We maximise the mapping of the hand-box to the body box such that the overlap between them is maximised.

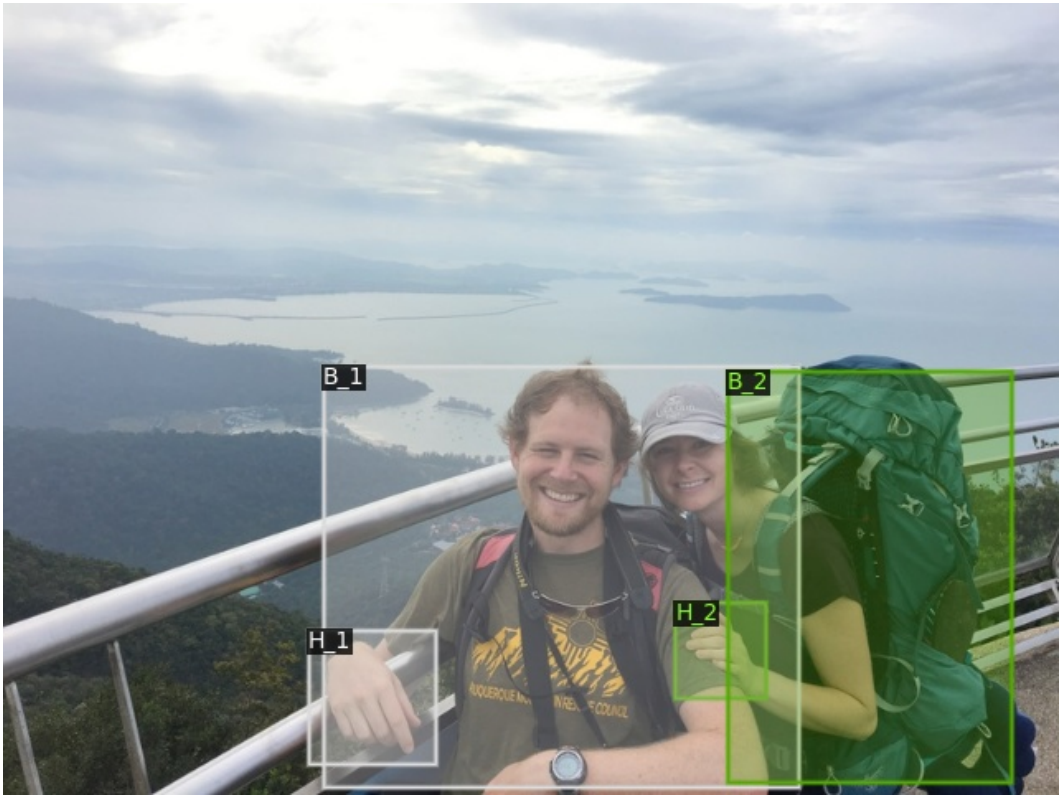
Bipartite Matching with Hungarian algorithm: The Hungarian algorithm, also called the Hungarian method or the Munkres algorithm, is a combinatorial optimization algorithm used to solve the assignment problem, which entails finding the optimal assignment of agents to tasks with minimum total cost or maximum total profit. The Hungarian algorithm is implemented by the linear_sum_assignment function from the scipy.optimize module. The problem formulation for the Hungarian algorithm is as follows: Given a cost matrix where each element represents the cost of assigning an agent to a task, the objective is to find the assignment that minimizes the total cost (or maximizes the total profit). Starting Point: The initial cost matrix used by the Hungarian algorithm may contain negative values that indicate profit. By deducting each value from the maximum value in the matrix, the costs can be turned into profits if the challenge involves maximizing profit. Row Diminution: The cost matrix is first subjected to row reduction using the method. The least value in each row is subtracted from each element in that row in this phase. Column Diminishment: Subtracting the lowest value found in each column from each element inside that column is the next step in the column reduction process. The matrix is reduced to a partially reduced matrix in this stage. Zero Assignment: Next, the procedure determines the reduced matrix's maximum number of independent zero entries. These zero items show possible work assignments for agents. Test of Optimality: The algorithm moves on to the next stage when an ideal assignment has been established and the number of zero entries equals the number of agents (or tasks). If not, Augmentation step of the algorithm is reached. Augmentation: The approach uses matrix augmentation to add more zeros if an ideal assignment cannot be determined. In order to achieve this, the cost matrix must be altered to add new zero entries while maintaining the ones that already exist. Repetition: Iteratively repeat steps Row reduction through Augmentation until the ideal assignment is discovered. The goal of each iteration is to lower the matrix's non-zero

entry count. Extracting the Solution: The procedure takes the assignments from the zero entries in the initial cost matrix and finds the optimal assignment/associations. In case of ties, the row with the lower index gets selected. The best way to solve the assignment problem is represented by these associations.

7. Results

We present from the many test images object, a few image outputs. The model performs well on most of the images except a few. The boxes imposed on the images are due to the classification, association done by the model.

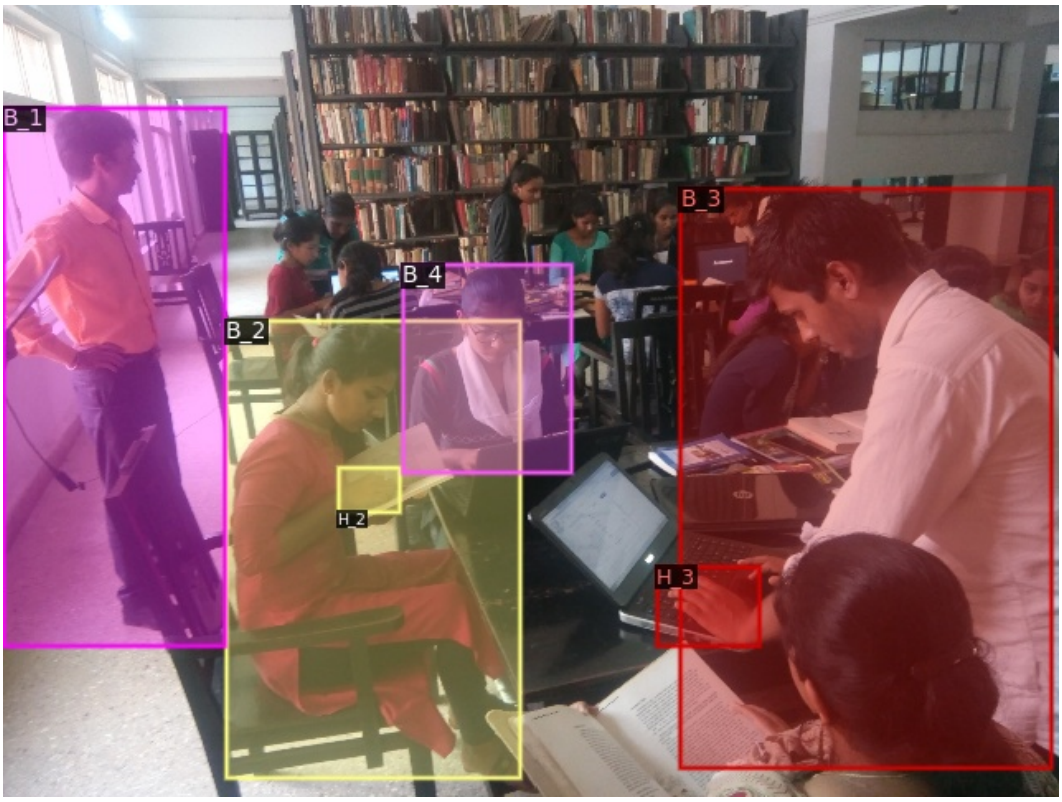

B_1
B_2
H_1
H_1
H_2
POLIC


B_1
B_2
H_2
H_1


B_1
B_3
B_2
H_1


B_1
H_1
B_2
H_2


B_2
B_1
H_2
CERVANTES
H_1


B_1
B_3
B_4
B_2
H_2
H_3


B_3
H_3
B_1
B_4
B_5
H_1
B_2
H_2
H_1


B_2
H_2
B_1
H_1
B_3


B_1
B_2
H_1
H_2


B_3
B_1
B_2
H_3
H_1
H_2


B_1
B_2
H_1
B_3
H_3
H_2

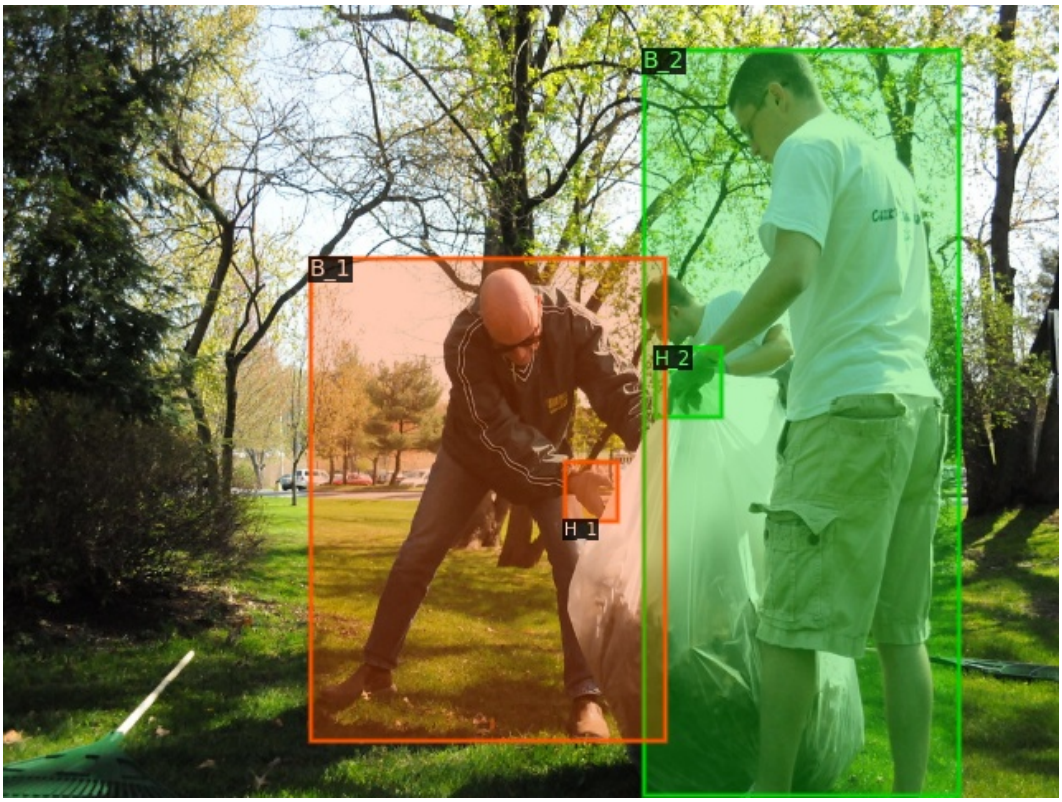

B_1
H_1
H_2


B_2
B_4
B_3
B_1
H_2
H_1
H_3


B_5
B_2
B_4
B_1
B_3
H_2
H_3
H_4
H_1


B_3
B_2
B_1
H_2
H_3
H_1


B_1
B_2
H_1
H_2


B_2
B_1
H_2
H_1

8. Conclusion

As part of this Project, the fascinating problems we have explored in the field of computer vision and machine learning is the complex task of hand and body association, which seeks to identify and associate human hands with their respective body parts in a variety of contexts. Using cutting edge techniques such as MultiScaleROIAlign, our method coordinates the generation of region proposal anchor boxes, ready for additional optimization in the Box and Mask heads of our model architecture that follow. While the RoiPooling layer pools the features from the image, the box head and mask head work in parallel to produces output boxes and masks. Although our object recognition and classification model performs admirably in a variety of images, it is still apparent that there are a few situations that call for improvements. In particular, the model's association section presents a major area that is ready for improvement; the smooth relationship between hands and their corresponding body parts requires careful consideration and optimization.

9. Future Objectives

The outcomes obtained from the model's performance in detection and segmentation are, as the results indicate, not far from satisfactory. This performance gap presents an opportunity for improvement. Following the initial steps of detecting and segmenting hands and bodies within the image, the next critical challenge lies in the association or mapping of these elements. The results of association welcome a lot of creativity and novelty. The fact that there is room for improvement indicates that there is room for creative and ground-breaking approaches to properly solve this issue. By exploring novel methodologies and advanced approaches, we can aspire to achieve significantly more precise, reliable, and coherent associations between hands and bodies in the visual data. This quest for better performance is necessary for a wide range of applications, including augmented reality, gesture detection & sign language interpretation & and human-computer interaction, where accurate and smooth human body-to-hand mapping is a fundamental need.

Bibliography

[1] Mykhaylo Andriluka, Leonid Pishchulin, Peter Gehler, and Bernt Schiele. 2d human pose estimation: New benchmark and state of the art analysis. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3686–3693, 2014. doi: 10.1109/CVPR.2014.471.

[2] Mykhaylo Andriluka, Umar Iqbal, Eldar Insafutdinov, Leonid Pishchulin, Anton Milan, Juergen Gall, and Bernt Schiele. Posetrack: A benchmark for human pose estimation and tracking, 2018.

[3] Ross Girshick. Fast r-cnn. *CoRR*, abs/1504.08083, 2015. URL http://www.cv-foundation.org/openaccess/content_iccv_2015/papers/Girshick_Fast_R-CNN_ICCV_2015_paper.pdf.

[4] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation, 2014.

[5] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[6] K.T. Gribbon and D.G. Bailey. A novel approach to real-time bilinear interpolation. In *Proceedings. DELTA 2004. Second IEEE International Workshop on Electronic Design, Test and Applications*, pages 126–131, 2004. doi: 10.1109/DELTA.2004.10055.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '16, pages 770–778. IEEE, June 2016. doi: 10.1109/CVPR.2016.90. URL http://ieeexplore.ieee.org/document/7780459.

[9] Sheng Jin, Lumin Xu, Jin Xu, Can Wang, Wentao Liu, Chen Qian, Wanli Ouyang, and Ping Luo. Whole-body human pose estimation in the wild. *CoRR*, abs/2007.11858, 2020. URL https://arxiv.org/abs/2007.11858.

[10] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2015.

[11] Tsung-Yi Lin, Piotr Dollár, Ross B. Girshick, Kaiming He, Bharath Hariharan, and Serge J. Belongie. Feature pyramid networks for object detection. In *CVPR*, pages 936–944. IEEE Computer Society, 2017. ISBN 978-1-5386-0457-1. URL http://dblp.uni-trier.de/db/conf/cvpr/cvpr2017.html#LinDGHHB17.

[12] Supreeth Narasimhaswamy, Zhengwei Wei, Yang Wang, Justin Zhang, and Minh Hoai Nguyen. Contextual attention for hand detection in the wild. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9566–9575, 2019. doi: 10.1109/ICCV.2019.00966.

[13] Supreeth Narasimhaswamy, Trung Nguyen, and Minh Hoai. Detecting hands and recognizing physical contact in the wild, 2020.

[14] Supreeth Narasimhaswamy, Thanh Nguyen, Mingzhen Huang, and Minh Hoai. Whose hands are these? hand detection and hand-body association in the wild. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4879–4889, 2022. doi: 10.1109/CVPR52688.2022.00484.

[15] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks. *CoRR*, abs/1511.08458, 2015. URL http://arxiv.org/abs/1511.08458.

[16] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett, editors, *NIPS*, pages 91–99, 2015. URL http://dblp.uni-trier.de/db/conf/nips/nips2015.html#RenHGS15.

[17] Huayi Zhou, Fei Jiang, and Ruimin Shen. Who are raising their hands? hand-raiser seeking based on object detection and pose estimation. In Jun Zhu and Ichiro

Takeuchi, editors, *Proceedings of The 10th Asian Conference on Machine Learning*, volume 95 of *Proceedings of Machine Learning Research*, pages 470–485. PMLR, 14–16 Nov 2018. URL https://proceedings.mlr.press/v95/zhou18a.html.