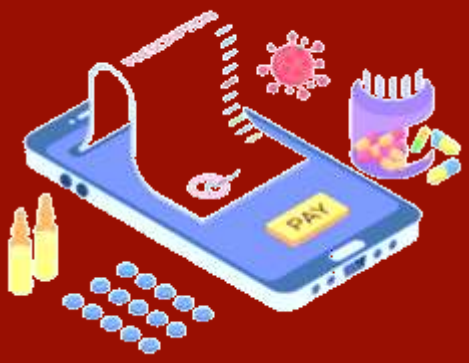
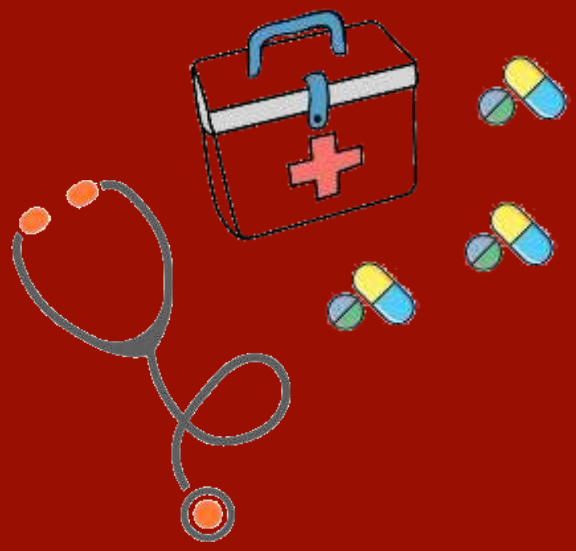





Smart Accident & Emergency Response System with Post-Accident Analysis

ACT II

Presented by :

Mina Edwar Dawood Elias

Narden Gerges Adward Amin

Nada Abdelmoneem Ahmed

Youssef Hany Ezzat Aly

Yousef Emad El Din Ibrahim

Dalia Abd Elazim Mohamed



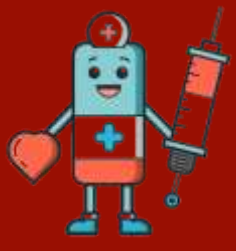


Table of Contents

List of Figures

List of Tables

List of Diagrams

1.Introduction

1.1 Problem Definition

One of the key challenges in accident response is the delay in assessing injuries due to a lack of immediate and structured information. Traditional methods rely on manual evaluation by first responders, which can slow down decision-making and resource allocation. Additionally, accurately detecting injuries in complex accident scenarios—such as poor lighting, varying injury severity, and occlusions—remains a challenge. A reliable, automated solution is necessary to improve the efficiency of emergency response teams and enhance injury assessment accuracy. **We aim to improve accuracy and generate more optimal reports, ensuring faster and more reliable medical decision-making.**

1.2 Background and Motivation

By integrating automated injury detection with a structured reporting system, we bridge the gap between accident scenes and emergency response teams. The system not only identifies injuries but also classifies them by age category and type, helping medical professionals prioritize treatment. Additionally, blood segmentation and advanced injury analysis, including wounds, cuts, and burns, provide crucial insights into the injured person's condition, allowing for better-prepared medical intervention before arrival at the hospital.

1.3 Overview of the Chapter

Injury detection and segmentation play a crucial role in emergency response and medical assistance, enabling quick and accurate identification of injuries. Our system automates the process of detecting and segmenting injuries, ensuring that first responders receive critical information. Our solution addresses this challenge by utilizing deep learning models to provide an efficient and precise injury assessment.

This chapter outlines the methodologies, models, and integration processes involved in our injury detection and segmentation pipeline. By combining injured detection, classification, and medical image segmentation, we aim to create a robust system that enhances emergency response efficiency. The following sections explore the research background, system modules, and the integration workflow, leading to an optimized reporting mechanism for accident scenes.

2.Research Background & Related Work

Research 1 :

Gender Recognition and Age Approximation using Deep Learning Techniques

Research 2 :

Detecting Non-Injured and Injured Humans in Thermal Imaging using YOLOv8

Research 3 :

Enhancing forensic blood detection using hyperspectral imaging and advanced preprocessing techniques

Research 4 :

Fully automatic wound segmentation with deep convolutional neural networks

Research 5 :

A Survey of Wound Image Analysis Using Deep Learning: Classification, Detection, and Segmentation

POC / Research	Research 1	Research 2	Research 3	Research 4	Research 5
Date	2020	2023	2024	2020	2022
Dataset	Adience datasets	a custom dataset of 2848 images is created, augmented, and annotated with the assistance of Roboflow.	A dataset for evaluating blood detection in hyperspectral images, Forensic Sci	Wound images segmentation, Images of wounds in human foot	1-Medetec datasets 2-BIP_US datasets 3-FUSeg dataset 4-Sarwebben 5-Chronic wound database 6-AHZ dataset 7-AHZ&UWM dataset
Task Type	Classification	Detection	Segmentation	Segmentation	Segmentation
Study Model	CNN	CNN YOLO	1-1D/2D/3D/HIS-CNN 2-PCA 3-FE	1-MobileNetV2 2-SegNet 3- VGG16 4-U-Net 5- Mask-RCNN	1-BI-CNN/ CNN/DCNN 2-DenseNet201 3- DFUNet/ _QUTNet/_ SPNet 4-VGG-19 5-U-Net 6-ResNet / 50/101 7-AlexNet
Feature Extraction	Edge detection Haar– like features	**NULL**	**NULL**	**NULL**	**NULL**

Metrics	Overall : 89.5%	(mAP)50:95 of 0.772	[1] 1D-CNN: 99.62% [2] 2D CNN: 99.99% [3] 1D-3D CNN:76%	[1] accuracy: 90.47% recall: 89.97%	[4] 85% [1] 87.7 % [5] 96.1 %

Table 1: Showing comparisons between previous related work

Research Takeaways

- **Feature Extraction Techniques:** Efficient methods like Haar-cascade and edge detection simplify feature extraction, helping CNNs in tasks such as gender detection.
- **Wound Segmentation:** Transfer learning with MobileNetV2 significantly enhances accuracy in wound segmentation by leveraging pre-trained models.
- **High Accuracy Models:** Mask R-CNN, with its advanced segmentation architecture, excels in detecting consciousness individuals with high accuracy.
- **Blood Detection:** Fast Extraction (FE) framework reduces hyperspectral imaging (HSI) complexity, enabling 2D-CNNs to achieve high accuracy in blood detection.

3.Modules

3.1 Injury detection

3.1.1 Data

Collection

Data was collected from different resources like Kaggle, Roboflow. The initial dataset (1) was 25620 images, after that an edge case was discovered, we will talk about it in trials, made us add 12671 images dataset (2) so the final data (3) is 38291 images.

Preprocessing

Each image was resized to 640 x 640 (width and height) pixels.

Labeling

We used Roboflow for some data and the auto-label grounding DINO model to start auto-labeling the images to 0 or 1 classes (Not-Injured & Injured respectively), We also started using local scripts for the other parts of data as we fixed some formats and converted them to YOLOv8 format, also some labels and converted them to our desired label, We also renamed each pic from 1 to and its corresponding label to maintain any data issues while merging the different data sources.

Augmentation

Augmentations applied on some of the datasets as some was collected with their augmentations, The type of augmentations that we did on the dataset was:

Grayscale 15%, Saturation -25% and 25%, Brightness -15% and 15%, Exposure -10% and 10%, Noise addition 0.1%.

Split

The **initial dataset (1)** was 84.5% train 10.22% valid and 5.28% test. The **second dataset (2)** was split into 85.5% train, 10.5% valid and test 4%. The **final dataset (3)** was split into 85% train, 10.36% valid and 4.64% test.

Samples

Figure 1: Injured & Not Injured Dataset

1a) Injured (Non fallen) *1b) Not Injured* *1c) Injured (Fallen)*

3.1.2 Model & Methodologies

Model Selection

After reading some papers about the Injury detection models using CNN, YOLO with different versions and also the Faster-RCNN we decided that the YOLOv8 from Ultralytics library will be the best fit to our problem as we need strong efficiency and speed in the inference on the images compared to the Faster-RCNN that takes more time to infer and in our case **each 1 second** is important, so we started by using the YOLOv8m.pt pretrained weights and started finetuning these weights with our data.

Architecture

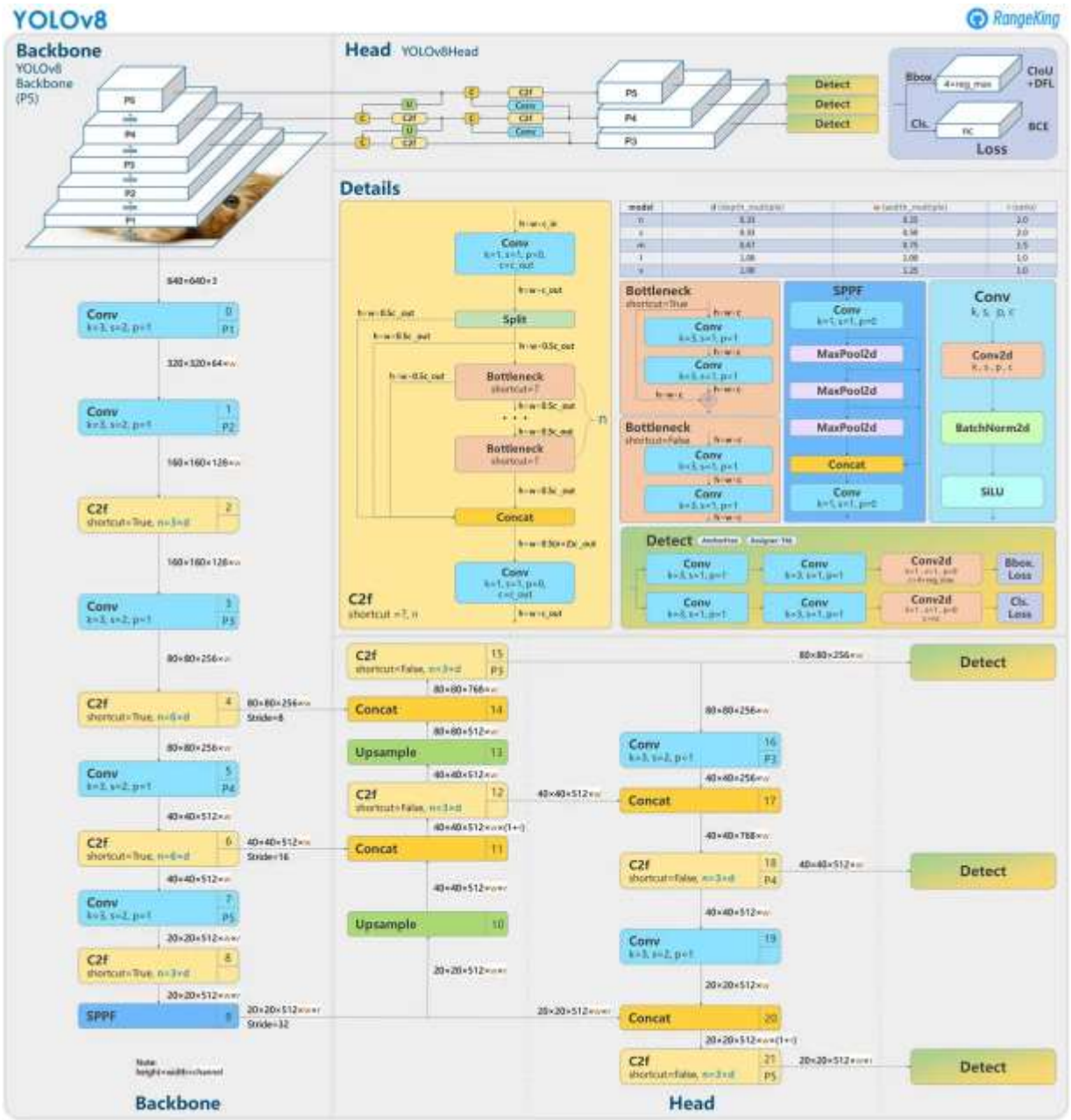


Figure 2: Architecture of YOLOv8 referenced from Ultralytics library

```
Model summary: 295 layers, 25,857,478 parameters, 0 gradients, 79.1 GFLOPs
(295, 25857478, 0, 79.06882560000001)
```

Training Process

We used Kaggle GPUs to start our training (fine-tuning) process for 50 epochs on the initial dataset and then another 50 epochs on the second dataset with imgsz=640, batch=32 and save_period=5.

```
print("Starting second stage of training (50 epochs)...")
train_results_stage2 = model.train(
    data=yaml_path,
    epochs=50,
    imgsz=640,
    batch=32,
    save_period=5,
    project=project_dir,
)
```

3.1.3 Output

Bounding Boxes and Confidence on each picture

3a) Injured person 3b) Injured person 3c) Not Injured Person

Figure 3: Our Injury detection model outputs samples

Total Number of Injuries

We started counting boxes with class_id = 1 which is injured and started printing this count as we need this information in our injury report.

3.1.4 Evaluation

Our model shows incredible performance compared to another models in terms of mAP, mAP 50:95 as shown in the following figure:

Class	Images	Instances	Box (P	R	MAP50	MAP50-95):
All	1317	1503	0.989	0.972	0.985	0.898
Not-Injured	363	449	0.979	0.951	0.975	0.847
Injured	995	1054	0.998	0.993	0.995	0.949

Table 2: Shows the evaluation metrics of our injury detection model

3.1.5 Trials

Trial 1 : We start with the initial dataset which handled the case of fallen people images only and we started fine-tuning and our model performed well but after that we started testing some cases from

Class	Images	Instances	Box (P	R	MAP50	MAP50-95):
All	2619	11046	0.87	0.837	0.886	0.708
Not-Injured	1327	9358	0.831	0.716	0.793	0.539
Injured	1327	1688	0.91	0.957	0.979	0.878

Table 3: Shows the evaluation results of our first injury detection model trial

outside the distribution of the data like blood on the faces, persons semi-fallen (Injures in different positions) and we noticed that it didn't give the expected performance on them so we did a 2nd trial.

Trial 2 :

We started by collecting images of all the edge cases we found during the testing of trial 1 and then we started fine-tuning our trial 1 model with these images and it gave the expected performance while testing with excellent evaluation metrics.

3.1.6 Next Steps

We are currently working on deploying the model on a cloud service to link it with the incoming application.

3.2 Age categorization & Gender classification

3.2.1 Data

Collection

Data used was The UTKFace Dataset which contains 23708 images, labeled in the name of each image the age_gender_...etc.

Preprocessing

-Each image was resized to 224 x 224 (width and height) pixels.

-Each image was normalized /255.0 .

Labeling

We started extracting the gender and age labels of each image name and started splitting Males class 0 in dataset and Females class 1 in dataset in different folders also for the age to 4 categories folders: baby if age<=3, kid if 4 <= age <= 12, young if 13 <= age <= 35 else old >36.

Augmentation

The augmentations we did on the train were:

geometric transformations (rotation, shifts, shearing, zooming, flipping), **color adjustments** (brightness and channel shifting).

Split

Data was split into 80% train 20% validate.

For Gender:

```
Number of Male images: 12391
Number of Female images: 11317
Male to Female ratio: 1.09
```

```
Found 18967 images belonging to 2 classes.
Found 4741 images belonging to 2 classes.
```

For Age:

```
Training Set:
baby: 1516 images
kid: 1216 images
old: 6881 images
young: 9356 images
```

```
Validation Set:
baby: 378 images
kid: 303 images
old: 1720 images
young: 2338 images
```

3.2.2 Models & Methodologies

Model Selection

Several models were tested for Gender Classification and Age Categorization, including EfficientNet, ResNet-based architectures, custom CNNs, and MobileNetV2. While EfficientNet and ResNet provided strong feature extraction, they were computationally expensive. Custom CNNs lacked generalization.

MobileNetV2 proved to be the best choice, offering:

High accuracy for both tasks.

Lightweight design with fast inference.

Good generalization after fine-tuning.

Its efficiency and performance balance made it the optimal model for real-world use.

Transfer Learning Process

Using **MobileNetV2** as our base model with pretrained weights on **'imagenet',** We froze the layers and added the last layers on top of it to fit our problem which was classifying gender to 0 or 1 which is a binary classification with last layer with **1 unit and activation sigmoid**, then, we trained it for 20 epochs after that, we unfroze the base model layers and fine-tuned the whole architecture with 30 more epochs. Also, we used the same base model on the age categorization problem and we froze the layers and added the last layers on top of it with the final layer **4 dense units and activation softmax** for classifying 1 of 4 classes, then, we trained it for 20 epochs after that, we unfroze the base model layers and fine-tuned the whole architecture with 30 more epochs.

Final Architecture

Gender :

Layer (type)	Output Shape	Param #
mobilenetv2_1.00_224 (Functional)	(None, 7, 7, 1280)	2,257,984
global_average_pooling2d_2 (GlobalAveragePooling2D)	(None, 1280)	0
dense_4 (Dense)	(None, 256)	327,936
batch_normalization_2 (BatchNormalization)	(None, 256)	1,024
dropout_3 (Dropout)	(None, 256)	0
dense_5 (Dense)	(None, 128)	32,896
batch_normalization_3 (BatchNormalization)	(None, 128)	512
dropout_4 (Dropout)	(None, 128)	0
dense_6 (Dense)	(None, 64)	8,256
dropout_5 (Dropout)	(None, 64)	0
dense_7 (Dense)	(None, 1)	65

Total params: 2,628,675 (10.03 MB)
Trainable params: 2,593,793 (9.89 MB)
Non-trainable params: 34,880 (136.25 KB)
Optimizer params: 2 (12.00 B)

Age :

Layer (type)	Output Shape	Param #
mobilenetv2_1.00_224 (Functional)	(None, 7, 7, 1280)	2,257,984
global_average_pooling2d_1 (GlobalAveragePooling2D)	(None, 1280)	0
dense_4 (Dense)	(None, 256)	327,936
batch_normalization_2 (BatchNormalization)	(None, 256)	1,024
dropout_3 (Dropout)	(None, 256)	0
dense_5 (Dense)	(None, 128)	32,896
batch_normalization_3 (BatchNormalization)	(None, 128)	512
dropout_4 (Dropout)	(None, 128)	0
dense_6 (Dense)	(None, 64)	8,256
dropout_5 (Dropout)	(None, 64)	0
dense_7 (Dense)	(None, 4)	260

Total params: 2,628,870 (10.03 MB)
Trainable params: 2,593,988 (9.90 MB)
Non-trainable params: 34,880 (136.25 KB)
Optimizer params: 2 (12.00 B)

Figure 4: Gender Model architecture

Figure 5: Age Model architecture

3.2.3 Output

Gender Classified

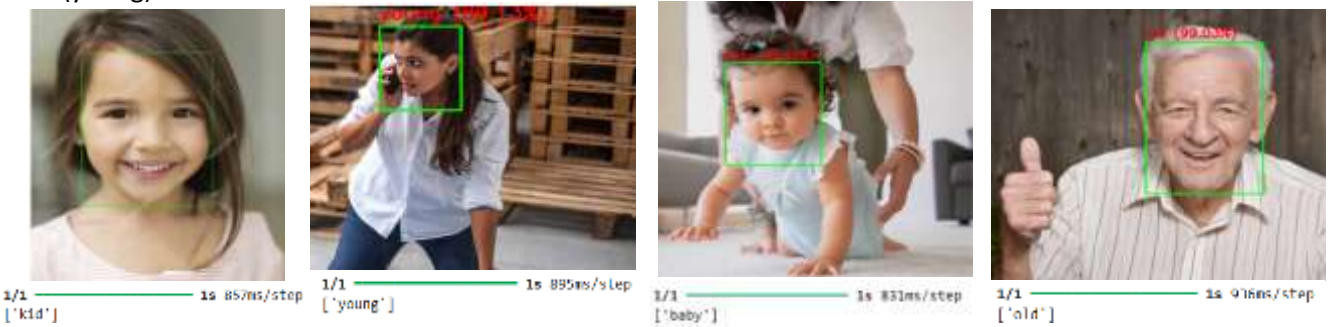
The image classified into class 0 (Female) or class 1 (Male).

6a) Face detected 6b) Female detected 6c) Face detected 6d) Male detected

Figure 6: Gender Classification output samples

Age Categorized

The image categorized into one of 4 categories: class 0 (baby) or class 1 (kid) or class 2 (old) or class 4 (young).



7a) kid-range 7b) young-range 7c) baby-range 7d) old-range

Figure 7: Age Categorization output samples

3.2.4 Evaluation

Figure 8: Shows evaluation metrics of Gender and Age Models

3.2.5 Trials

We did 3 trials in the Gender Classification models and 2 in the Age Categorization models:

Gender Trial 1:
we built our own architecture with 3 conv layers each one followed by maxpooling layer, then 2 dense layers with last one a sigmoid activation to classify to 0(Female) or 1(Male), this model was trained for 40 epochs and performed well in training and didnt generalize very well with 73 val accuracy only that's why we went to trial 2.

Gender Trial 2:
We used MobileNetv2 as a backbone model and added on top of it two dense layers between them a dropout layer with 0.5 dropout_prob, this model generalized well after fine tuning the frozen base layers and gave us about 79% val accuracy after training for 40 epochs.

Gender Trial 3:
We aimed to improve our trial 2 architecture so we added more dense layers with more units, we added 4 dense layers with Kernel regularizer, BatchNormalization and Dropout layers to reduce overfitting, this architecture succeeded in improving the val accuracy to 81%.

Age Trial 1:
We tried using EfficientNetB1 in the age categorization model and then fine tune it with our layers that we added with a final layer softmax with 4 units to classify to one of four categories, we evaluated it and we got 78% val accuracy.

Age Trial 2:
We shifted towards a MobileNetv2 as our backbone and added 4 dense layers and regularizers.l2 kernels also 3 dropout layers and 2 batchnormalization layers to reduce overfitting, this succeeded in improving the validation accuracy 82.3%.

3.2.6 Next Steps

We are currently working on improving the validation accuracy of the 2 models by modifying the architecture. After that, we will work on deploying the model on a cloud service to link it with the incoming application.

3.3 Blood segmentation

3.3.1 Data

Collection

We sourced blood-related images from Kaggle, Roboflow, and Google. During this process, we faced many problems like: other classes than blood, missing or incomplete labels, different blood itself classes and not accurate polygon regions instance segmentations, we dealed with them all and handled them to finally have our dataset with 8645 images.

Preprocessing

-Each image was resized to 640*640 (width and height) pixels. -

Labeling

We started manually segmenting the blood regions using the Roboflow Instance Segmentation tools to have accurate masks on the blood regions, We also fixed the labels where the class of blood is different from the class of bloodstain, …etc. and finally we mapped all the photos to the same category_id =1 using local scripts inside the JSON annotations file.

Augmentation

Augmentations we did on the train were:

Random Brightness, Random Flip (Horizontal), Random Saturation, Resize Shortest Edge, Random Contrast.

Split

The data collected was split into 6916 train images with 80% and 1729 validate images with 20%.

Samples

9a) Blood on person's face 9b) Blood on person's neck 9c) blood on floor

Figure 9: Blood dataset samples

3.3.2 Model & Methodologies

Model Selection

We had a lot of models to select from like UNet for Semantic Segmentation, DeepLab, YOLO + Segmentation head and finally MaskRCNN for instance segmentation.

Why did we choose the MaskRCNN as we needed to segment each blood region instance individual as some of them might be associated with different persons for semantic segmentation it just labels each

pixel to some class with no association for each instance.
We used Detectron2 Library and imported the ResNet101 to be our backbone and then we added the MaskRCNN on top of it with 1 ROI head classifier.

Architecture

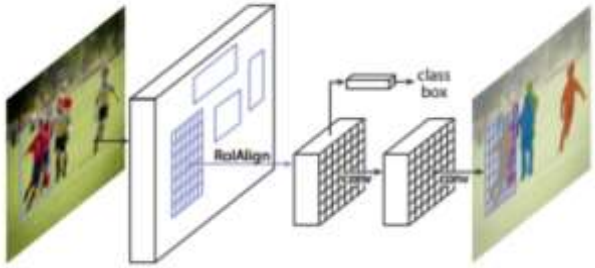


Figure 10: ROI idea referenced

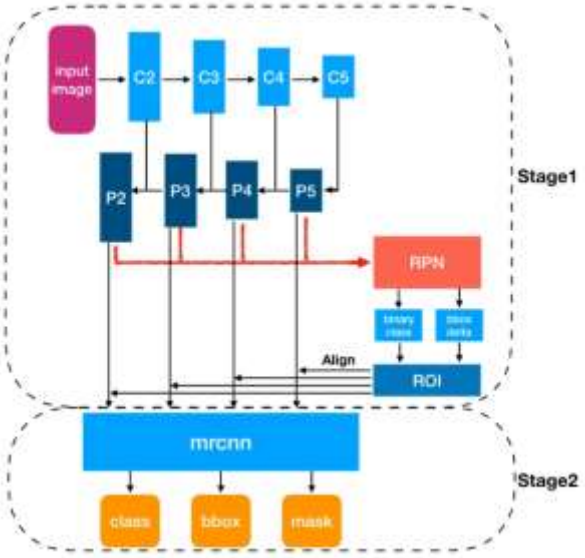


Figure 11: MaskRCNN architecture referenced

Training Process

We started training using Kaggle GPUs with Model configuration as the following:

```
cfg.DATALOADER.NUM_WORKERS = 4
cfg.SOLVER.IMS_PER_BATCH = 2
cfg.SOLVER.BASE_LR = 0.00025  # Learning rate
cfg.SOLVER.MAX_ITER = 10000  # Total iterations
cfg.SOLVER.STEPS = [7000, 9000]  # Learning rate decay steps
cfg.MODEL.ROI_HEADS.BATCH_SIZE_PER_IMAGE = 32
cfg.MODEL.ROI_HEADS.NUM_CLASSES = 1  # Only one class: Blood
cfg.TEST.EVAL_PERIOD = 500 #Evaluating every 500 epoch
```

3.3.3 Output

Blood regions masks

The model segments the regions of blood in the image.

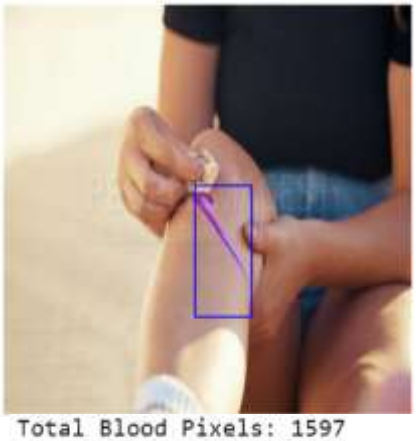
Total Blood Pixels (Segmented)

We count the pixels that have been masked to get the total pixels that may indicate more information about the severity of the case.

Blood Severity (Excessive, Moderate, No blood info)

We use the total blood pixels and compare it with a threshold and based on this we get the severity information.

12a) Excessive blood



12b) Moderate blood

12c) Excessive Blood

Figure 12: Blood Severity Classification different output samples

3.3.4 Evaluation

The training accuracy was **93 %**

The validation IOU and AP results was as following:

```
[02/13 23:24:47 d2.evaluation.coco_evaluation]: Evaluation results for segm:
|   AP   |  AP50  |  AP75  |  APs   |  APm   |  APl   |
|:------:|:------:|:------:|:------:|:------:|:------:|
| 31.934 | 56.071 | 31.796 | 13.305 | 34.540 | 39.420 |
[02/13 23:24:44 d2.evaluation.coco_evaluation]: Evaluation results for bbox:
|   AP   |  AP50  |  AP75  |  APs   |  APm   |  APl   |
|:------:|:------:|:------:|:------:|:------:|:------:|
| 39.148 | 60.873 | 41.250 | 18.042 | 39.413 | 50.267 |
```

Figure 13: Shows the evaluation metrics for blood severity model

3.3.5 Trials

Trial 1 : At first we tried only manual instances segmentation images but we had like 1260 images only for train and validation as the manually labeling takes so much time and effort, This trial was good but we aimed to increase the IOU and Segmentation accuracy.

Trial 2 : Secondly, we added more data the final dataset collected which is the 8645 images and with changing some parameters like the learning rate and the number of max iterations it succeeded in increasing the IOU and segmentation accuracy but we won't stop here.

3.3.6 Next Steps

-We are currently working on optimizing the validation accuracy and IOU, by searching for more data, training with more epochs and changing the hyperparameters and will update this documentation with the new results.

-We are also heading to deploy these models on the cloud services.

3.4 Advanced injury segmentation

3.4.1 Data

Collection

1.Data was collected from Roboflow . The used Dataset is composed of 2 dataset collections found on roboflow,The first dataset was 431 images, and the second was 723 images

Preprocessing

Each image was resized to 256 x 256 (width and height) pixels.

Labeling

the final dataset was cleansed and labeling was done and revised manually on roboflow, utilizing the powerful and quick labelling tools offered there. our final classes are as follows:
1-background 2-abrasion 3-bruise 4-cut 5-laceration 6-stab_wound

Augmentation

Our final dataset consisted of 1032 images, so the augmentation applied was not intensive
Brightness -15% and 15%, Horizontal Flip, Rotation between -10° and +10° .

Split

The **final dataset** consists of 2542 images, 89% training (2265),7% validation(174) and 4% test(103)

Samples

Figure 14: Different Injures samples

3.4.2 Model & Methodologies

Model Selection

Working on an injury segmentation task, a lot of good options were in mind, however the 2 most interesting to us were the mask R-CNN and U-net based architecture, and that is because of their relative simplicity and powerfulness in the field of semantic segmentation. We decided to use a U-net based backbone architecture because our data was not that large for the mask R-CNN to operate on and achieve high accuracies.

Our Model is a deep learning-based architecture designed for precise segmentation of

wound regions in images of injured people. It is derived from a combination of state-of-the-art techniques in semantic segmentation,
including U-Net, Atrous Spatial Pyramid Pooling (ASPP), Dual Attention Mechanisms, and Pyramid Pooling Modules.
The architecture is enhanced with residual connections, boundary refinement modules, and deep supervision to improve gradient flow, capture multi-scale context, and refine edge details.
The model is designed to deal with injury segmentation challenges, such as varying wound shapes, sizes, and textures, by leveraging multi-scale feature extraction and attention mechanisms.

Architecture

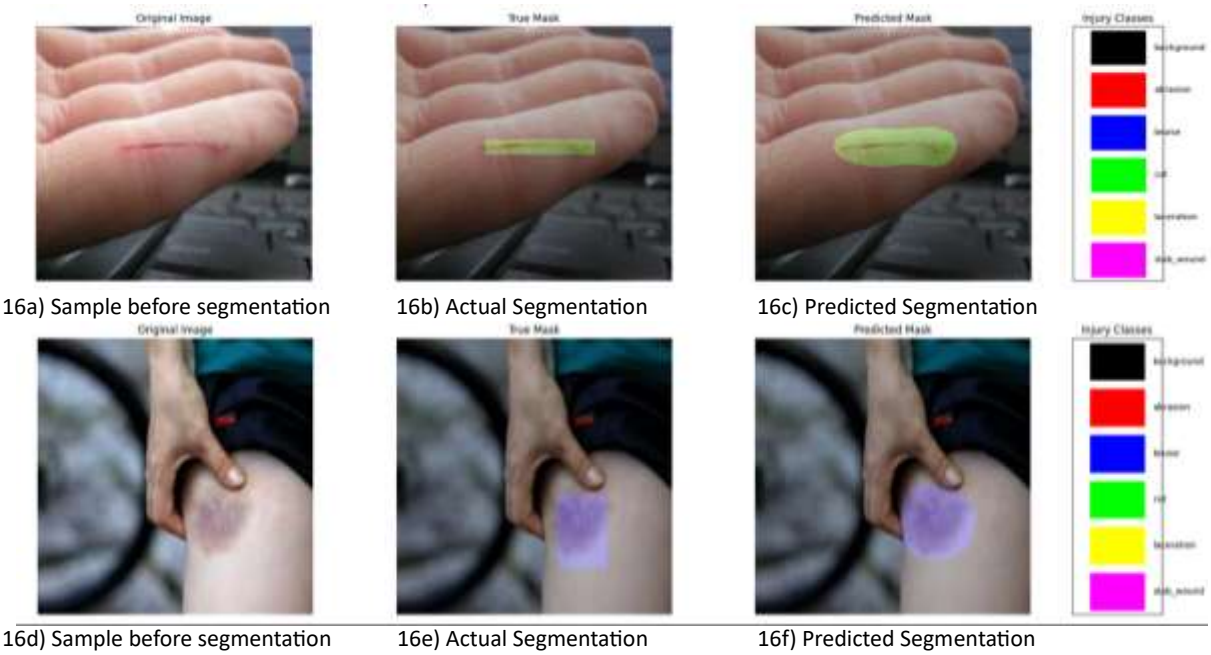
Layer Breakdown

Layer Type	Description	Output Shape
Input Layer	Input image (256x256x3).	256x256x3
Initial Conv Block	3x3 convolution, BatchNorm, Swish activation.	256x256x64
Residual Block 1	Two 3x3 convolutions, BatchNorm, Swish activation, skip connection.	256x256x64
Max Pooling 1	2x2 max pooling.	128x128x64
Residual Block 2	Two 3x3 convolutions, BatchNorm, Swish activation, skip connection.	128x128x128
Max Pooling 2	2x2 max pooling.	64x64x128
Residual Block 3	Two 3x3 convolutions, BatchNorm, Swish activation, skip connection.	64x64x256
Dual Attention	Spatial and channel attention applied to mid-level features.	64x64x256
Max Pooling 3	2x2 max pooling.	32x32x256
Residual Block 4	Two 3x3 convolutions, BatchNorm, Swish activation, skip connection.	32x32x512
Max Pooling 4	2x2 max pooling.	16x16x512
Bridge with ASPP	ASPP module with dilation rates (1, 6, 12, 18) and global context.	16x16x512
Upsampling Block 1	Bilinear upsampling, attention gate, residual block.	32x32x512
Upsampling Block 2	Bilinear upsampling, attention gate, residual block.	64x64x256
Upsampling Block 3	Bilinear upsampling, attention gate, residual block.	128x128x128
Upsampling Block 4	Bilinear upsampling, attention gate, residual block.	256x256x64
Pyramid Pooling Module	Multi-scale global context aggregation.	256x256x64
Boundary Refinement Module	Dilated convolutions for edge refinement.	256x256x64
Output Layer	1x1 convolution with softmax activation.	256x256xNUM_CLASSES
Auxiliary Outputs	Deep supervision outputs at intermediate layers.	Varies

Total params:
7,234,614 (27.60 MB)
Trainable params:
7,228,982 (27.58 MB)
Non-trainable params:
5,632 (22.00 KB)

Figure 15: Advanced Injury Segmentation model custom architecture

3.4.3 Output



16a) Sample before segmentation 16b) Actual Segmentation 16c) Predicted Segmentation

16d) Sample before segmentation 16e) Actual Segmentation 16f) Predicted Segmentation

Figure 16: Advanced Injury Segmentation output masks

The model takes an image and produces the output as shown in the previous images, with a colored segmented part and a color map to determine the final pixel segmentation result

3.4.4 Evaluation

```
7/7 ━━━━━━━━━━━━━━━━━━━━ 6s 775ms/step - conv2d_168_categorical_accuracy: 0.9137 - conv2d_168_multiclass_iou_
metric: 0.2260 - loss: 0.2539

Test Loss: 0.2596
Test IoU: 0.9084
```

Figure 17: Advanced Injury Segmentation evaluation results

Our model's test loss and IOU score were impressive given our limited dataset.

We did not depend on the model's scores only, this model was the one that gave the best actual results and performed well on unseen images, with different angles, lighting and varying sizes and zooming.

3.4.5 Trials

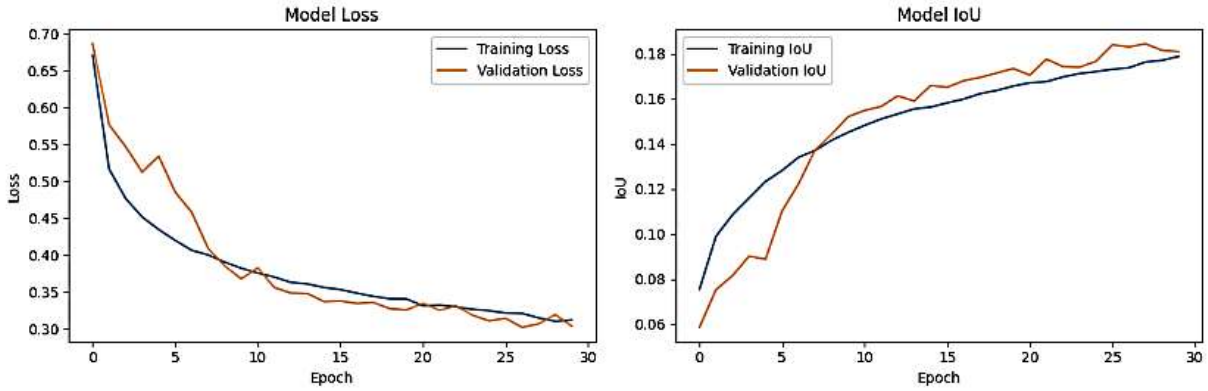
Trial one

This is a modified U-Net design for semantic segmentation that incorporates modern deep learning improvements. The key enhancements include:

The architecture uses Batch Normalization and LeakyReLU activation (α=0.2) to improve training stability and gradient flow. It implements Spatial Dropout (rate=0.3) for regularization and adds residual connections to help with gradient flow in deeper networks. The design also includes Squeeze-and-Excitation (SE) blocks to model channel relationships and improve feature representation.The network structure follows a typical encoder-decoder pattern:

Encoder: Progressively increases filters (64→512) with max pooling

Bridge: Maximum 512 filters with SE block
Decoder: Symmetric upsampling path (256→64) with skip connections




```
3/3 ━━━━━━━━━━━━━━━━━━━━ 2s 609ms/step - categorical_accuracy: 0.8964 - loss: 0.2900 - multiclas
s_iou_metric: 0.1876
```

Figure 18: Advanced Injury Segmentation Trial one evaluation results

```
Test Loss: 0.2894
Test IoU: 0.8953
```

In trial one , we trained for a total of 30 epochs, this was the result of the evaluation on the test dataset, and while it might seem good enough, the actual segmentation and visualization on the images was still a bit off, so we knew there was definitely better.
Multiple micro adjustments were made to the architecture and a few trials were made with some hyperparameter tuning to try and get the best possible output, but the previous remained the best so far

However, **the Second Trial** , which was the final architecture selected and talked about above, was the one that gave the best actual results, not only slightly better numbers in the evaluation.

3.4.6 Next Steps

A challenge we are facing is the dataset, since the regular injuries and wounds datasets are not as available in comparison to the medical datasets, and while the medical injury datasets might provide us with a huge number of images, it is always fixating on specific angles and a very large scale in terms of zoom , and they might be better suited for other medical segmentation purposes.

- We aim to try and search for extra datasets and retrain our already trained model and finetune with it.
- Try and solve the zoom problem, where images taken from afar will not have very good or accurate results.

3.5 Burn Segmentation

3.5.1 Data

Collection

Data was collected from Roboflow. The dataset contains total of 6097 images .

Preprocessing

Each image was resized to 640 x 640 (width and height) pixels with auto-orientation applied.

Labeling

The dataset has 4 classes, first , second and third degree burns and of course the background class, it was labeled using roboflow tools.

Augmentation

Augmentations applied on some of the datasets as some was collected with their augmentations, The type of augmentations that we did on the dataset was:

Saturation -25% and 25%, vertical and horizontal flip .

Split

The dataset is split into 92% train(5626), 4%test(242),4%validation(229)

Sample

Figure 19: Burn dataset samples of different degree level

3.5.2 Model & Methodologies

Model selection

As mentioned before in 3.4.2, our goal is to segment and identify a specific injury type (a burn in this case and its degree), so the same approaches and thinking strategies were applied here as well

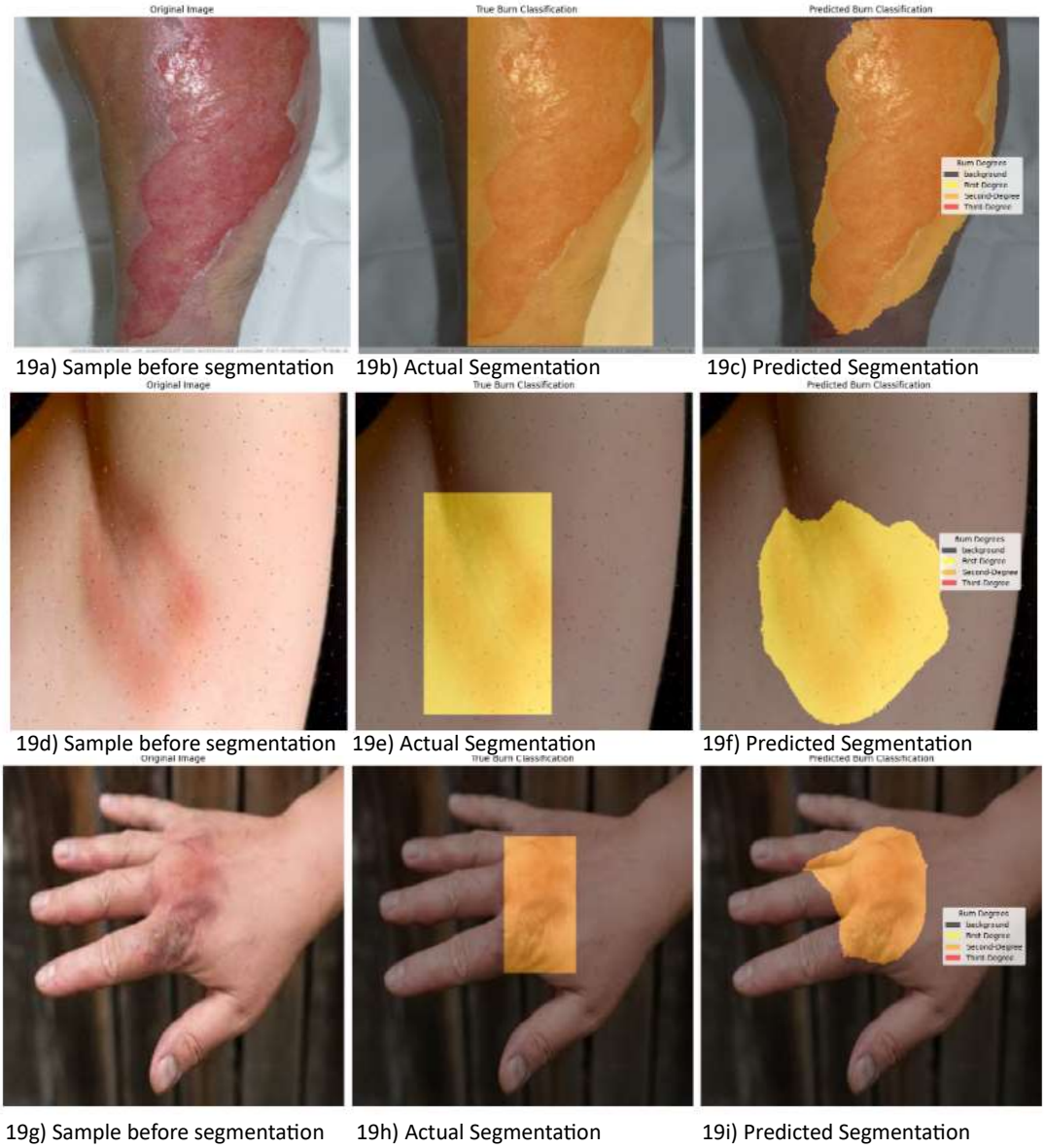
Architecture

The architecture that was tried as a first trial in the one of the aforementioned sections (3.4.5) worked extremely well on this dataset, the task and target is the same, but the difference in how each architecture performs is because of how each model responds to different dataset.

Training Process

We used Kaggle GPUs to start our training process for 30 epochs on the burns dataset with imgsz=256 x 256, batch=32 , initial_learning_rate=1e-4, decay_steps=1000, decay_rate=0.9 with callback monitoring validation loss with training stopping after 27 epochs

3.5.3 Output



19a) Sample before segmentation 19b) Actual Segmentation 19c) Predicted Segmentation

19d) Sample before segmentation 19e) Actual Segmentation 19f) Predicted Segmentation

19g) Sample before segmentation 19h) Actual Segmentation 19i) Predicted Segmentation

Figure 20: Burn Segmentation model outputs

3.5.4 Evaluation

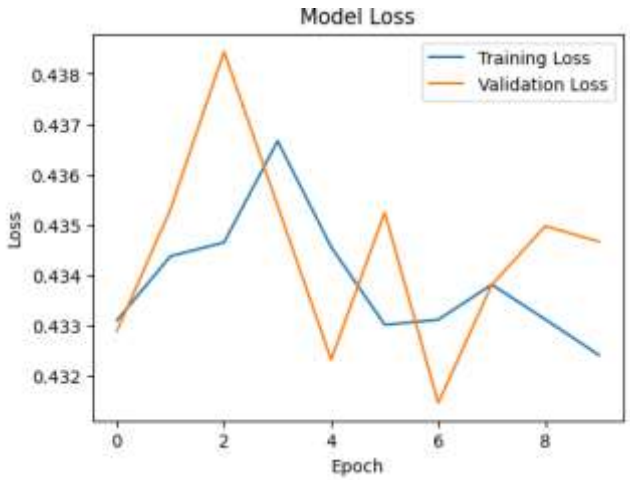
```
7/7 ━━━━━━━━━━━━━━━━━━━━ 4s 551ms/step - categorical_accuracy: 0.8048 - loss: 0.3845 - multi
class_iou_metric: 0.6637
```

Figure 21: Burn Segmentation evaluation results

```
Test Loss: 0.3783
Test IoU: 0.8081
```

3.5.5 Trials

Our currently working model's architecture was actually our first trial, and it worked seamlessly, but we to make use of the more advanced technique and deeper network architecture that was used for our final injury segmentation model, but to our surprise the data didn't perform so well, and the model overfit very quickly and didn't get nearly as good as the first time.




```
14/14 ━━━━━━━━━━━━━━━━━━━━ 12s 811ms/step - loss: 0.4466 - output_1_categorical_accuracy: 0.7166 - output_1_multiclass_iou_metric: 0.1868

Test Loss: 0.4525
Test IoU: 0.7120
```

Figure 22: Burn Segmentation Trial one evaluation results

And here is a sample of the model's performance on the test data:

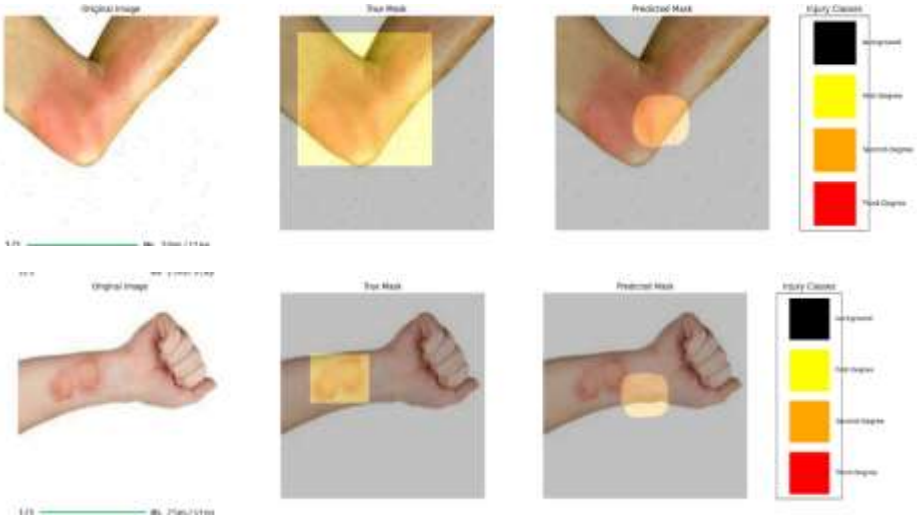


Figure 23: Burn Segmentation model outputs Trails

3.5.6 Next Steps

We have the same challenges as the previous injury segmentation model, and we intend to try and solve them the same way, by further fine tuning and searching for more robust and generalized datasets

3.6 Injury reporting

3.6.1 Overview

Our report provides essential accident details to help emergency teams respond more efficiently. By offering insights into the severity and nature of the accident, it ensures responders are well-prepared for the situation they will encounter. Additionally, the collected data supports advanced analysis, helping enhance emergency protocols, improve road safety, and ultimately save lives.

We will utilize our AI models to analyze images and determine whether individuals are injured. If injuries are detected, our system will extract and report key details, including the number of injured individuals, their gender, and age category. Additionally, our models will assess the presence of blood, evaluate its severity, and classify the types of injuries. All collected data will be compiled into a comprehensive report and promptly sent to emergency responders for immediate action.

3.6.2 Report Fields

The report will include some fields like the following:

Date	Time	LocationID(Area)	Images ID(Array)	Around No of Injuries	Gender	Age Category	Blood Severity	Advanced Seg		
								Burn	Cuts	Wound

Table 4: Shows the database fields for our injuries reported data

Date: contains the date of the accident that happened.

- **Time:** the timestamp of the accident.
- **LocationID:** each location has exact id in our database integrated with the database.
- **ImagesID:** the images that are associated with a specific LocationID.
- **Around No of Injuries:** **Number of estimated injuries** from our detection model.
- **Gender:** what are the genders included in this accident, Is it **Male, Female** or Both.
- **Age:** One of Four categories, **[baby, kid, old, young].**
- **Blood Severity:** Is it **Excessive** or **Moderate** or **No blood information**.
- **Advanced Segmentation:** Information fields like **Burn, Cuts, stabs, laceration, bruise, abrasion.**

3.6.3 Methodology

1. Limit the users entered photos to a 3 to 7 range.
2. Limit the images used in the models to a 7 at the same time from the same locationID.

Edge cases we might face in point 1,2:

Case 1:

If the user gave (7) photos for this location ID, enough to start generating the report if all of them provide injury detections (Model 1), if one or more of them didn't give any detections we wait 2 minutes to another user to enter images and we use our models then we generate the report.

Case 2:

If the user gave less than (7) photos we will wait 2 minutes to start running the models and generate our report.

Note: If we stucked in the loop of Case 2 and 1 together more than 1 time (4 mins) we reached our max time ~5mins and we start with the images queue even if less than 7 images.

---

3. After passing the model 1 process with the filtered images queue and **it gives no of injuries detected** → Parallel the models with the images.
4. Model 2 BloodMaskRCNN gives pixels if > quarter the bounding box around 25,000 pixel then the field is excessive, if less than <25k pixels moderate, if =0 then No blood info.

We gather all the information from our models and structure it in a report to be send to emergency response unit.

3.6.4 Next Steps

Advanced Data Analysis from the collected data with visualizations:

1. Total Accident % in Specific Areas in specific timestamp.
2. Total Number of Injuries due accident in a month, year.
3. Most affected Gender and Age categories from the accidents
4. Frequent times that accident happen during the day 6-7 pm example, Frequent seasons that accident happen (Date).

More information could be gathered from this data collected and sent to the authorities.

4.Integration

4.1 Flow diagram

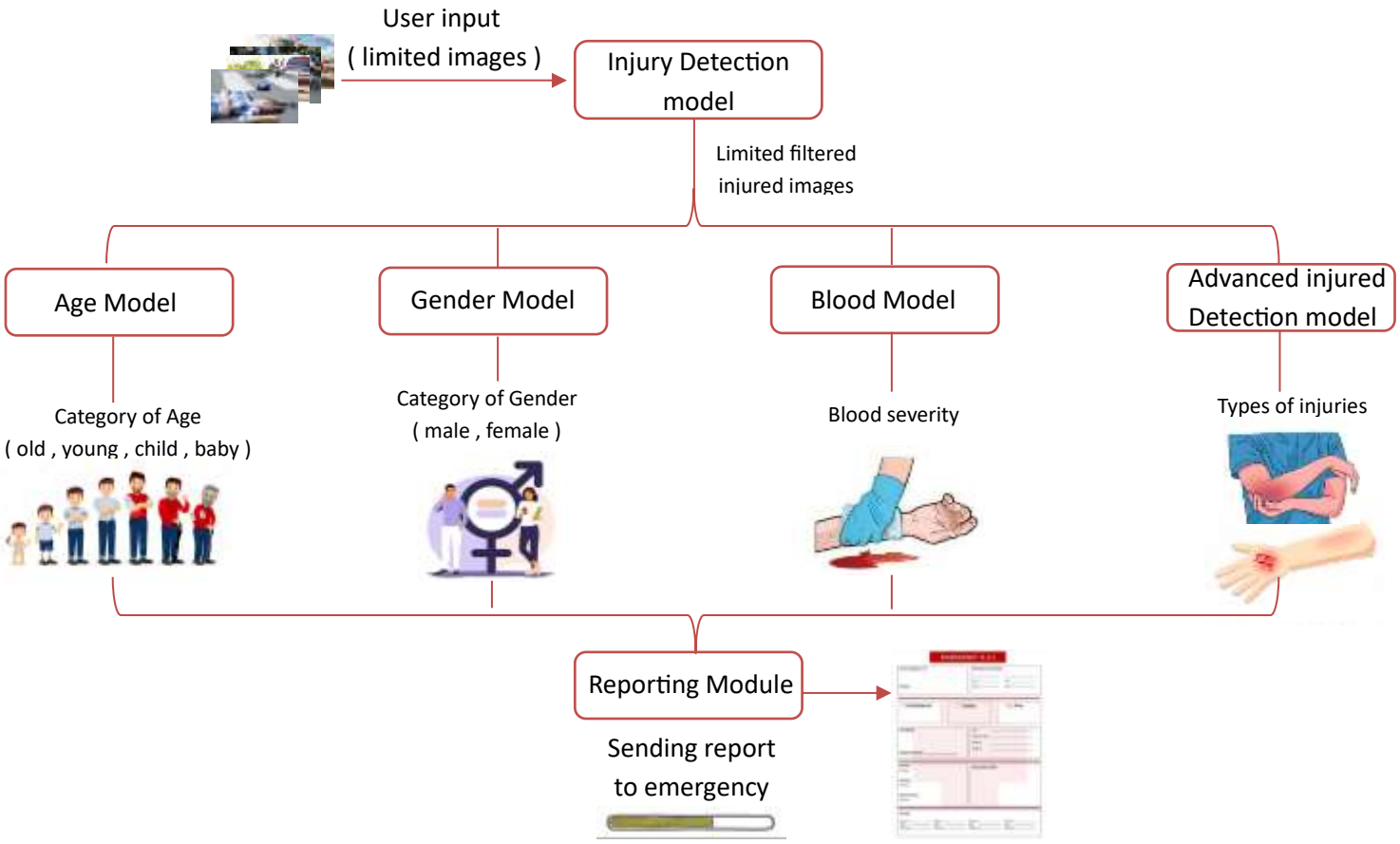
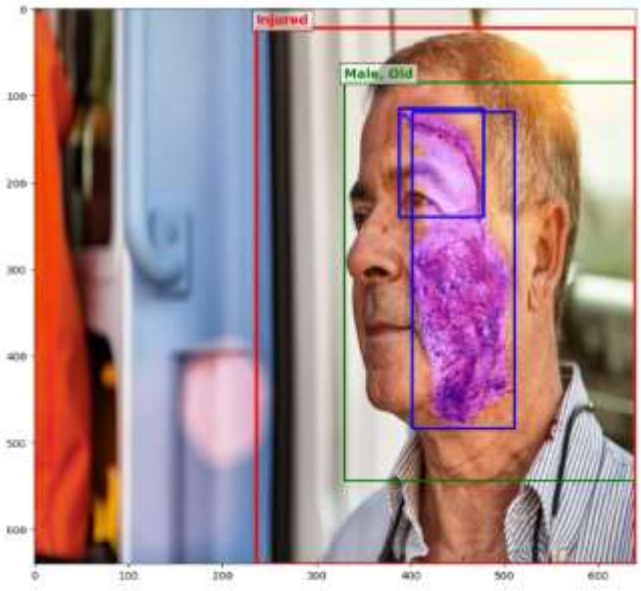


Diagram 1: The flow diagram of our injury detection, segmentation and reporting process

4.2 Samples




```
Total Injuries Detected: 1
Total Blood Pixels: 28760
Blood Severity: Excessive
Total Pixels in Image: 409600
1/1 ━━━━━━━━━━━━━━━━━━━━ 2s 2s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 2s 2s/step
Detected Face in Injured Area by DNN: Gender: Male, Age: Old
1/1 ━━━━━━━━━━━━━━━━━━━━ 3s 3s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 3s 3s/step
```

Figure 24: Sample of Part I of our integration process

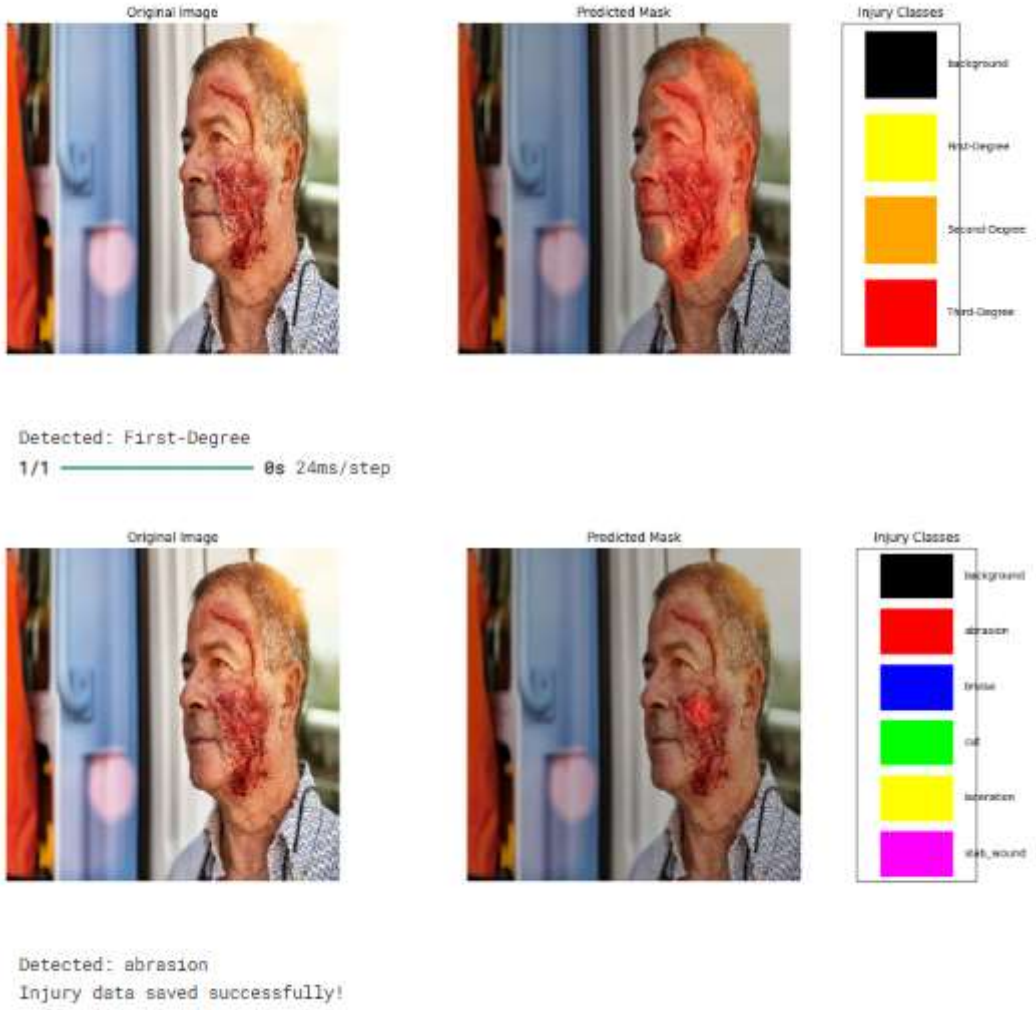


Figure 25: Sample of Part II of our integration process

5.Conclusion

In this chapter, we explored the development of an intelligent injury detection, segmentation, and reporting system that enhances emergency response through deep learning and computer vision. By leveraging **YOLOv8 for detection, MobileNetv2 for classification, and Mask R-CNN for segmentation**, our approach ensures precise identification and assessment of injuries in real-time. The structured reporting system bridges the gap between accident scenes and medical teams, enabling faster decision-making and improving patient care.

This integration of AI-driven injury analysis marks a significant step toward revolutionizing emergency response. With automated assessments and data-driven insights, first responders can act more efficiently, ensuring that every second counts in critical situations.

We're taking emergency response to the **next level!** 🌍 📡. In the next chapter, we unleash the power of **AI-driven mobile technology** to save lives **anytime, anywhere!** 📲 ⚡ **Stay tuned—because the future of first aid is just a tap away!** 🤖 🔥