# HeHealth “Mpox Screen Lite” AI tools to Support Africa’s Mpox Response Efforts

In response to WHO’s declaration of Mpox as a global emergency, HeHealth is prepared to assist with our AI-powered tools for rapid, on-device Mpox screening. The current challenge in Africa, particularly with Mpox Clade 1b, is the severe limitation of PCR testing in the field, which hampers effective surveillance, isolation, and resource allocation. Our AI models, YOLOv8n and MobileNetV2, are specifically designed to overcome these limitations. YOLOv8n, with its low computational demands, is ideal for resource-constrained environments, while MobileNetV2 offers robust performance across diverse settings. Both models can run without internet access on a simple mobile device, making them indispensable for frontline healthcare workers in remote areas. By significantly improving the accuracy of identifying probable cases, our AI tools can function as reliable surrogates for confirmed diagnoses, enabling more precise planning and allocation of resources. This approach not only enhances the immediate response but also supports long-term containment and management of the outbreak in regions with limited infrastructure.

## YOLOv8n Model

### Dataset Distribution

The dataset is balanced across three categories: Mpox, Other, and Normal, each consisting of 1200 images. The distribution for training, validation, and testing sets is as follows:

- Training Set: 75% (900 images per category): ( 50:50 Real : Synthetic)
- Validation Set: 10% (120 images per category)
- Test Set: 15% (180 images per category)

### Preprocessing Techniques

To enhance model robustness and generalization, the following preprocessing techniques were applied to the images:

- Flip: Images were flipped horizontally and vertically.
- Rotation: Images were rotated 90 degrees in both clockwise and counter-clockwise directions, as well as upside down.
- Saturation Adjustment: Saturation levels were adjusted between -10% and +10%.
- Brightness Adjustment: Brightness levels were adjusted between -10% and +10%.

### Model Configuration

The model employed is YOLOv8n (nano), suitable for deployment on devices with limited computational resources.

Framework and Libraries:

- Framework: PyTorch version 2.3.1
- Language: Python version 3.10.12

Model Specifications:
- Layers: 73
- Parameters: 1,438,723
- Model Weight Size: 5 MB

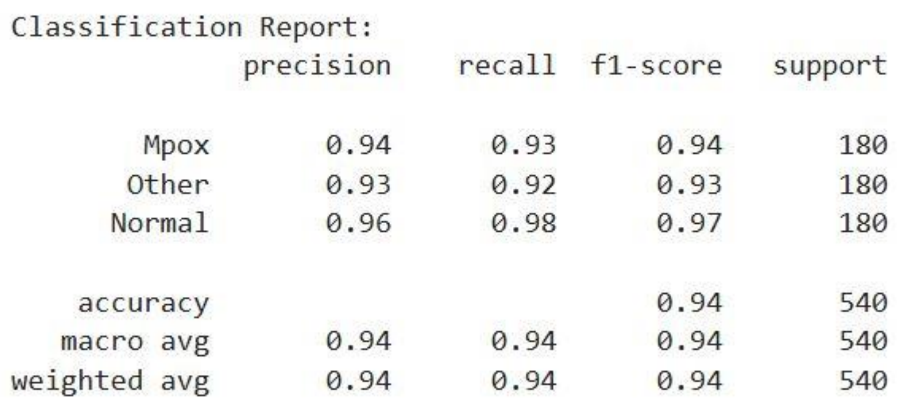
## Training Parameters

The model was trained using the following parameters:
- Epochs: 200
- Image Size (imgsz): 224x224 pixels
- Patience: 50 epochs for early stopping
- Dropout Rate: 20%
- Early Stopping: Training stopped early at epoch 115

## Performance Evaluation

| **Category** | **Precision** | **Recall** | **F1-Score** | **(test images )** |
|---|---|---|---|---|
| **Mpox** | 0.94 | 0.93 | 0.94 | 180 |
| **Other** | 0.93 | 0.92 | 0.93 | 180 |
| **Normal** | 0.96 | 0.98 | 0.97 | 180 |

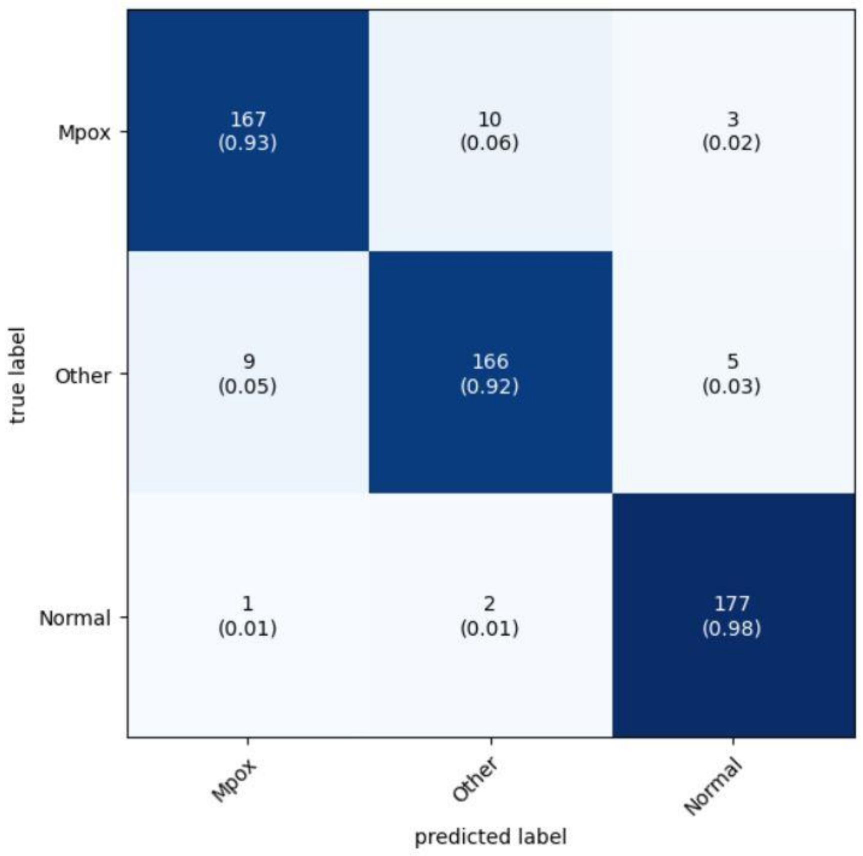
Overall Accuracy: 94%



```
Classification Report:
              precision    recall  f1-score   support

        Mpox       0.94      0.93      0.94       180
       Other       0.93      0.92      0.93       180
      Normal       0.96      0.98      0.97       180

    accuracy                           0.94       540
   macro avg       0.94      0.94      0.94       540
weighted avg       0.94      0.94      0.94       540
```



## MobileNetV2 Model

### Dataset Distribution

The dataset used for the MobileNetV2 model is the same as that for the YOLOv8n model, ensuring consistency in training and evaluation metrics. The distribution is as follows:

- Training Set: 75% (900 images per category)
- Validation Set: 10% (120 images per category)
- Test Set: 15% (180 images per category)

### Preprocessing Techniques

To prepare the images for training and ensure the model's generalization capabilities, several preprocessing techniques were applied:

- Flip: Images were flipped horizontally and vertically.
- Rotation: Images were subjected to rotations of 90 degrees in various directions.
- Saturation Adjustment: Saturation levels were tweaked to vary between -10% and +10%.
- Brightness Adjustment: Brightness levels were similarly adjusted between -10% and +10%.

## Model Configuration

- Framework: TensorFlow version 2.x
- Language: Python version 3.10.12
- Model: MobileNetV2, optimized for high accuracy with manageable computational and memory requirements.

## Model Specifications:

- Parameters: 3.4 million
- Model Weight Size: 35mb.

## Training Parameters

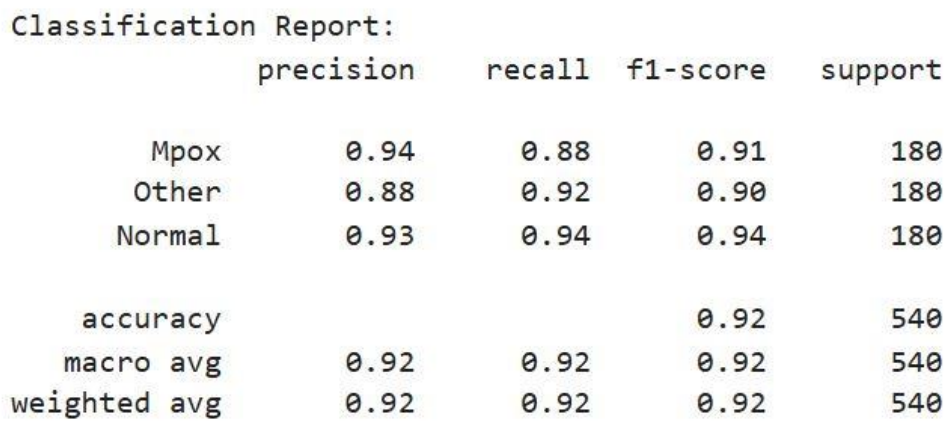
The model was meticulously trained with the following specifications to optimize performance:

- Epochs: 200, to sufficiently learn from the data without overfitting.
- Image Size (imgsz): 224x224 pixels, typical for MobileNetV2.
- Patience: Early stopping was set to trigger after 50 epochs without improvement.
- Dropout Rate: 20%, to prevent overfitting by randomly omitting units during training.

## Performance Evaluation

- Mpox: Precision: 0.94, Recall: 0.88, F1-Score: 0.91
- Other: Precision: 0.88, Recall: 0.92, F1-Score: 0.90
- Normal: Precision: 0.93, Recall: 0.94, F1-Score: 0.94

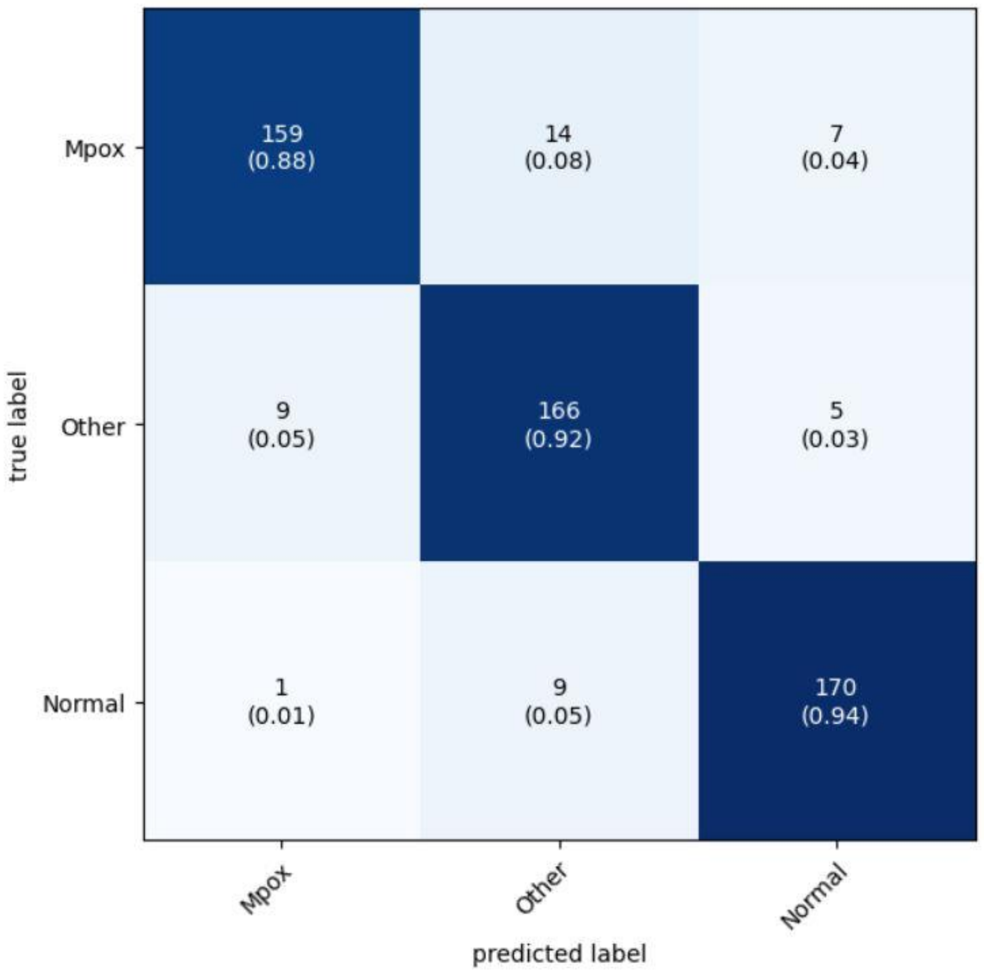
**Overall Accuracy: 0.92 across 540 test images**



```
Classification Report:
              precision    recall  f1-score   support

        Mpox       0.94      0.88      0.91       180
       Other       0.88      0.92      0.90       180
      Normal       0.93      0.94      0.94       180

    accuracy                           0.92       540
   macro avg       0.92      0.92      0.92       540
weighted avg       0.92      0.92      0.92       540
```



## Comparison of YOLOv8n and MobileNetV2

| **Feature** | **YOLOv8n** | **MobileNetV2** |
|---|---|---|
| Framework | PyTorch 2.3.1 | TensorFlow 2.x |
| Programming Language | Python 3.10.12 | Python 3.10.12 |
| Parameters | 1,438,723 | 3.4 million |
| Model Weight Size | 5 MB | 35 MB |
| Epochs | 200 (Stopped early at 115) | 200 (Stopped early at 90) |
| Image Size (imgsz) | 224x224 pixels | 224x224 pixels |

| | | |
|---|---|---|
| Patience for Stopping | 50 epochs | 50 epochs |
| Dropout Rate | 20% | 20% |
| Overall Accuracy | **94%** | **92%** |
| Processing Power | Low | Moderate |

## Pros and Cons of YOLOv8n and MobileNetV2

### Pros:

#### YOLOv8n:

Suitable for devices with limited computational resources due to lower model complexity.

Requires less storage space, making it easier to deploy on edge devices.

Higher overall accuracy and faster convergence during training.

#### MobileNetV2:

Provides robust performance with manageable computational requirements, despite being heavier than YOLOv8n.

Supported extensively by the TensorFlow ecosystem, which might be beneficial for integration with other TF-based applications or services.

### Cons:

#### YOLOv8n:

Being a smaller and lighter model might limit its ability to generalize across highly varied or unseen datasets compared to more complex models.

PyTorch ecosystem might not be as extensive as TensorFlow for some specific deployment scenarios.

#### MobileNetV2:

Larger model size which requires more storage and potentially more memory during inference.

Slower training times and possibly less efficient on devices with strict power or speed limitations.