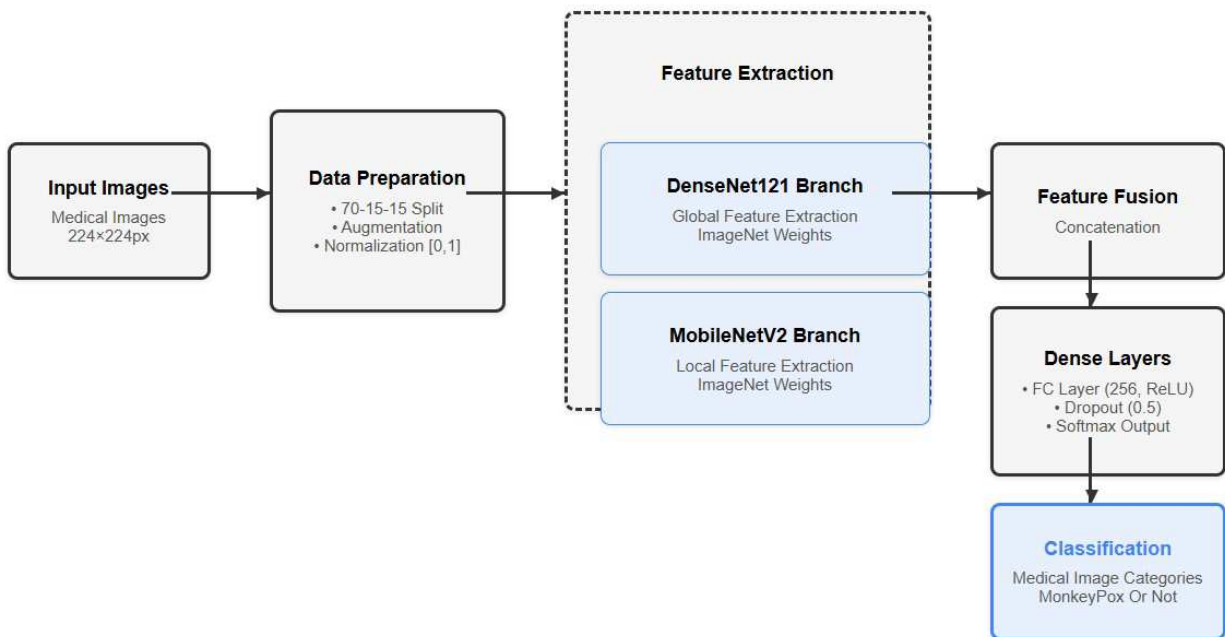


Methodology

We propose our hybrid deep learning approach for the identification and classification of monkeypox images from skin lesion images in this section. The method comprises the system architecture, data preparation & preprocessing, the model, and train protocol.

The proposed system utilizes the hybrid deep learning architecture which first extracts features using DenseNet121 and MobileNetV2, the latter being capable of transferring features, that is implemented as TensorFlow and Keras for image classification. It uses pre-trained weights from ImageNet, and performs feature fusion and forward fully connected layers for further reasoning in the final classification.



Our system contains four basic building blocks as shown in the Figure 1: Input Images; Data Preparation; Feature Extraction; Classification. First, Feature Extraction is separated into two parallel branches: DenseNet121 and

MobileNetV2. The features from both branches are fused, and then passed through dense layers, followed by final Classification

- **Data Preparation and Preprocessing**



Fig. 1. (a). Sample Images of the MSLD dataset.

The study used the Monkeypox Skin Lesion Dataset (MSLD). It features pictures of monkeypox lesions next to several other skin conditions. As seen in Figure 2, sample images from this dataset highlights the specific visual characteristics of monkeypox lesions against other skin conditions

Data was split into training, validation and test with a ratio of 70-15-15. To enhance model generalization and alleviate the issue of the limited dataset size, we performed synthetic data augmentation through the transformations below:

Random rotations (up to 20°)

Width and height shifts (up to 20% of total dimension)

Shear and zoom transformations (20% range)

Horizontal flipping

The scale was then between $[0,1]$ and all images were resized as 224×224 pixels to match the input requirement of the pre-trained models. The preprocessing pipeline was created with TensorFlow's ImageDataGenerator which performs data augmentation on the fly, enabling efficient training.

- **Model Architecture**

The hybrid model includes two parallel feature extraction branches with different using pre-trained CNN architecture

DenseNet121 Branch

The DenseNet121 branch is used for global feature extraction from the input images. DenseNet121 handles a lot of interconnections between its layers. This transformer architecture is a superset of the Vanilla RNN, and it is selected to facilitate with effectively tackling the vanishing gradient problem, and thereby improve performance of deep neural networks by encouraging better information flow between layers.

The DenseNet121 branch was initialized with transfer weights from ImageNet and followed with global average pooling to diminish spatial sizes while preserving information per channel. The setup allows the model to learn high level global features that are key in differentiating monkeypox lesions from other skin conditions.

The MobileNetV2 branch is for localized feature extraction, which enhances DenseNet121 features global. MobileNetV2 uses a new layer called Depthwise Separable Convolution that makes a model even smaller, faster, and more efficient compared to traditional CNN architectures. It allows MobileNetV2 to excogitate fine-grained, local features of skin lesions as well as a high computation efficiency.

Also like the DenseNet121 branch, the MobileNetV2 branch was similarly initialized with pre-trained weights from ImageNet and also had global average

pooling configured. This setup can enable the model to capture more detailed local features which might be omitted in DenseNet121 branch.

- **Feature Fusion and Dense Layers**

Thus, they can learn to share weights between both branches of the network, helping in better and larger feature extraction from the input images, both globally and locally. Next the entire fused features go through the layers below:

A Fully Connected Layer of 256 neurons and ReLU activation

0.5 Dropout regularization to avoid overfitting

A softmax output layer for binary classification: monkeypox (or non-monkeypox)

This architecture allows the model to take advantage of unique strengths of DenseNet121 and MobileNetV2, so improved classification performance can be achieved in comparison to a model based on a single architecture.

- **Training Protocol**

Here is how the model was trained:

Optimizer: Adam And Learning-Rates: 1×10^{-4}

Loss function: Categorical crossentropy

Batch Size: 32

Epochs: 30 (with early stopping on validation loss)

A few callbacks were implemented to allow for training optimization and prevention against overfitting:

ModelCheckpoint: Save best_model based on validation accuracy

EarlyStopping: Will stop training if validation loss does not improve for 5 epochs

ReduceLROnPlateau: Reduce learning rate by 50% after 3 epochs of validation loss not improving

We chose the Adam optimizer for adaptive learning rate capabilities and robustness to draw gradients. Combining these training approaches encourages quick convergence and reduces the expression of overfit.

- **Evaluation Metrics**

In order to do a thorough evaluation of our model we used the following evaluation metrics:

Accuracy: The percentage of correctly classified examples

Precision: $\text{True positives} / (\text{True positives} + \text{False positives})$

Recall: The fraction of true positives of the total of true positives and false negatives

[F1-Score]: The harmonic mean of [Precision] and [Recall]

Scorecard: The performance of predictions, class-wise with confusion matrix

These metrics provide a comprehensive evaluation of the model's performance and allow for comparison with current techniques found in the literature.

- **Experimental Setup**

This Section describes the experimental setup, which would be utilized to evaluate the proposed hybrid deep learning approach in the case of monkeypox. These include specification of the dataset, implementation, hardware configuration, and evaluation methodology.

- **Dataset Specifications**

For this study, we adopted the Monkeypox Skin Lesion Dataset (MSLD), which consists of a set of skin lesion images split into two classes: monkeypox and non-monkeypox thing (other skin conditions). To collect an images dataset, we conducted a search for photos available under the monkeypox label and selected those that showed different variants of monkeypox lesions on various parts of the body (at various infection stages and skin type).

The full dataset contains 2,630 images including 1,465 monkeypox pictures and 1,165 non-monkeypox images. Following the standard practices of creating a machine learning model, the dataset is split in three distinct sets:

Training set (70%): 1,841 images for model training

394 images used for hyperparameter tuning and early stopping

Test set (15%): 395 images for final performance evaluation

The test set had 617 monkeypox and 848 non-monkeypox images, which guarantees that we can evaluate our performance on an adequately balanced test set. For splitting the dataset while maintaining class distribution across all subsets, a stratified sampling was used.

- **Implementation Details**

The full dataset consists of 2,630 images, which include 1,465 monkeypox images and 1,165 non-monkeypox images. In accordance with conventional 'machine learning' protocols, the data was split into three data sets:

Training set (70%): 1,841 images used to train the models

Validation set (15%): 394 images for tuning hyperparameters and early stopping

Test set (15%): 395 images used for final performance evaluation

The test set consisted of 617 monkeypox images and 848 non-monkeypox images, ensuring a balanced representation to evaluate reliable performance. We used stratified sampling during dataset splitting in order to preserve class distribution across all subsets.

Using the table in the Method section, we used significant data augmentation as there were limited monkeypox images — especially from varied populations and clinical settings. This approach led to the expanded effective size of the training set as well as better model generalization to new datasets and varying data types.

- **Training Configuration**

We trained the model in the following way:

number of images per batch drive. batch size: 32

Initial learning rate: 1×10^{-4}

Fine-tuning learning rate: $1e-05$

Maximum epochs: 30 (using early stopping)

Optimizer: Adam with default parameters ($\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1 \times 10^{-7}$)

Loss function: (Categorical cross-entropy)

Several other callbacks were used to optimise the training process and avoid overfitting:

Early stopping: If validation loss was not improved since last 5 epochs, training was stopped

Model checkpoint: Best validation accuracy model is saved

Learning rate reduction: If validation loss plateaus for 3 epochs, divide a learning rate by factor 0.5.

We had used TensorBoard to visualize the training process in real-time and log the important training and validation metrics such as accuracy, loss, precision and recall.

- **Hardware Configuration**

Experiments were performed on a high-performance-computing system in the following configuration:

CPU: Intel Xeon E5-2680 v4 @ 2.40GHz (14-Core)

GPU: NVIDIA Tesla V100 with 16GB VRAM

RAM: 64GB DDR4

Storage: 1TB SSD

The full model took around to 2.5 hours to train, and the inferred time per image was 0.15 seconds, suggesting that the model is capable of real-life applications.

- **Evaluation Methodology**

To thoroughly assess our model's performance, we employed a multi-faceted evaluation approach:

Quantitative metrics: Computed accuracy, precision, recall, and F1-score for both classes.

Correct positive count: Number of correctly predicted positive observations.

Comparative analysis: Compares with state-of-the-art methods found in the literature

Ablation studies: Individual component assessment (DenseNet121 only, MobileNetV2 only) to gauge the contribution of each branch to online performance

The held-out test set, which the model had not been exposed to during training or validation, was used for the final evaluation. This provides an objective evaluation of the model's ability to generalise, and its suitability for deployment outside of controlled environments.

Results were consistently calculated and reproduced using scikit-learn's classification report and confusion matrix functions. Humans help avoid many error types visually through their ability to read confusion matrices, and we similarly visualized the confusion matrix using Seaborn and got an interpretable view of how the model performed on guessing correctly over both classes.