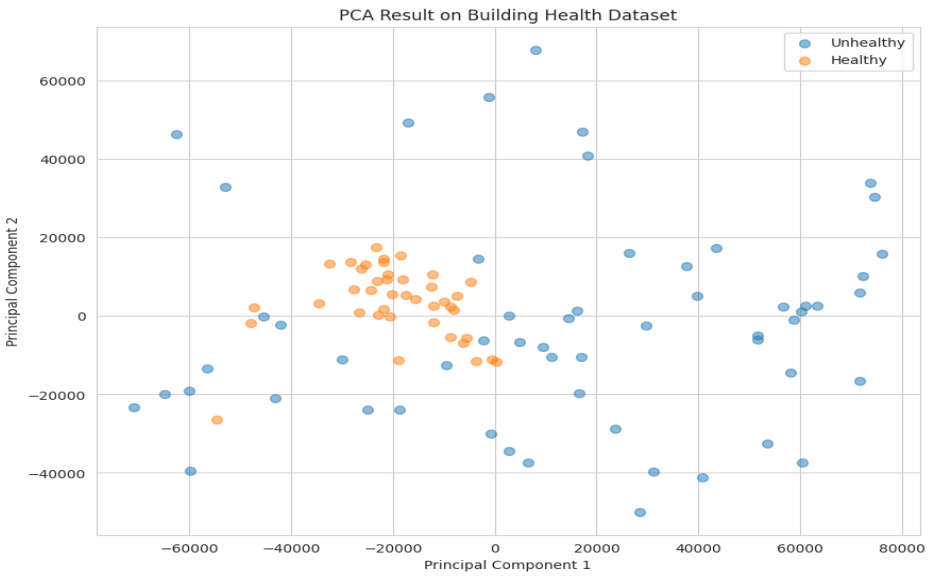
The initial report for Thermal Images Classification

Prepared by \\ Muhannad Mansour

- Principal component analysis:

Principal Component Analysis (PCA): Reduce the dimensionality of your images and visualize them in a 2D or 3D space. Helps in visualizing clusters or separations between classes.

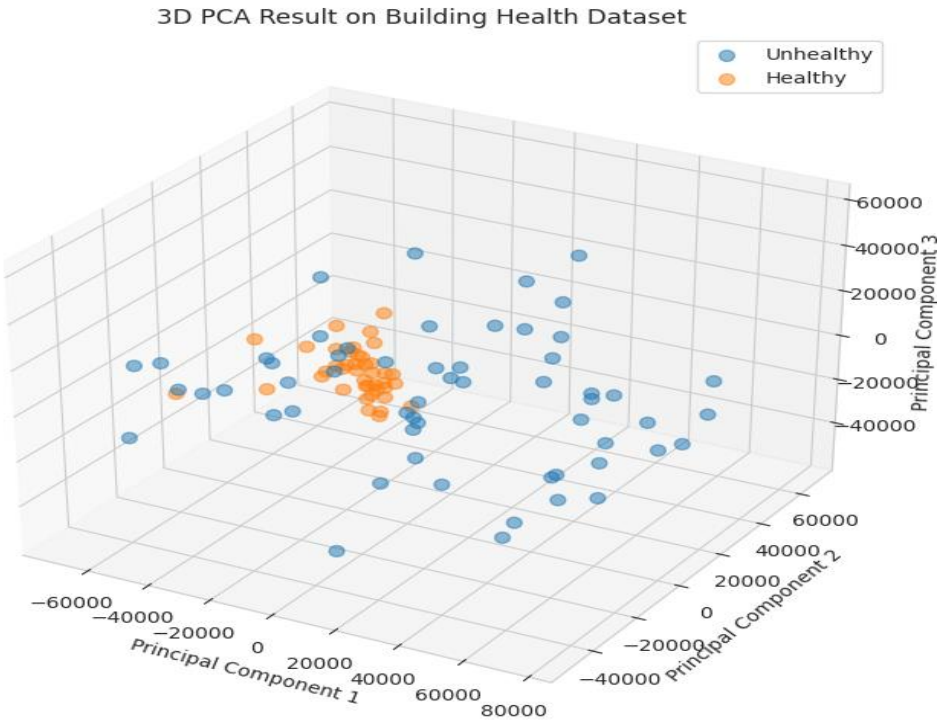


Spread of Unhealthy Images: The "Unhealthy" images are more spread out across the PCA plot. This could indicate a wider variety in the visual features of "Unhealthy" images compared to "Healthy" ones.- Overlap Between Classes: There's a clear overlap between the "Healthy" and "Unhealthy" classes in the reduced feature space. This overlap suggests that, based on the raw pixel values of the images, there's similarity between some of the "Healthy" and "Unhealthy" images. It's possible that a linear classifier (like Logistic Regression) might struggle to perfectly separate these two classes based on these principal components.

Density of Healthy Images: There's a dense cluster of "Healthy" images around the center of the plot. This suggests that a significant portion of the "Healthy" images are similar to each other in this reduced feature space.

Spread of Unhealthy Images: The "Unhealthy" images are more spread out across the PCA plot. This could indicate a wider variety in the visual features of "Unhealthy" images compared to "Healthy" ones.

Separation along Principal Component 1: While there's overlap, some of the "Unhealthy" images tend to have higher values on the Principal Component 1 axis, suggesting that this component does capture some variance that can be useful for differentiating the two classes.



- Images Data Augmentation Techniques Used:

Rescaling: All images are rescaled by a factor of 1./255 to normalize the pixel values between 0 and 1.

Rotation: Images are subjected to random rotations within a range of 15 degrees. This helps the model recognize images that might be oriented differently.

Zoom: Images can be randomly zoomed in or out within a range of 0.15. This introduces scale variability.

Width and Height Shift: Both horizontal and vertical random shifts are applied within a range of 0.15 times the width or height, ensuring the model can detect objects that aren't perfectly centered.

Shear Transformation: A shear transformation with a range of 0.15 is applied, which skews the image, helping the model to identify objects that may appear distorted.

Horizontal Flip: Images are randomly flipped horizontally. This is useful when there's no inherent left-right orientation in the data.

Fill Mode: The "nearest" fill mode is used, which fills the newly created pixels after transformations with the nearest existing pixel values.

Data Split for Training and Validation:

The data is split into training (80%) and validation (20%) subsets using the validation_split=0.2 parameter.

Image Input Specifications:

Target Size: Images are resized to a standard size of (224, 224) pixels.

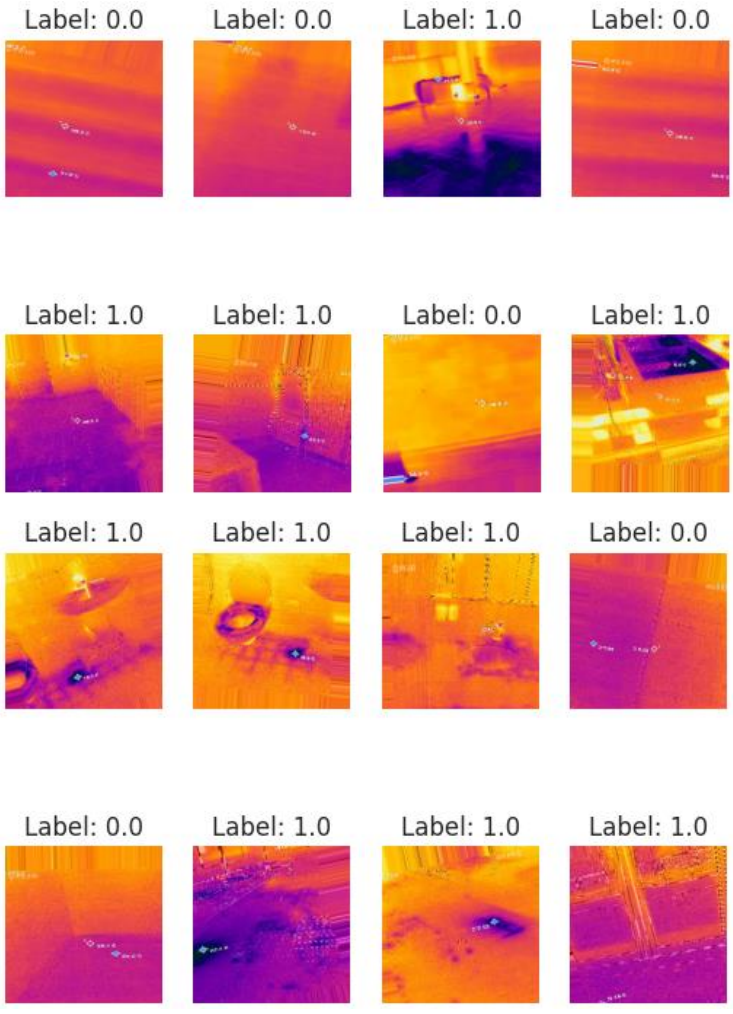
Batch Size: For efficient model training, data is loaded in batches of 32 images at a time.

Class Mode: The binary class mode indicates a binary classification task, where images can belong to one of the two classes.

Generators Created:

Training Generator: It generates augmented image batches from the training subset.

Validation Generator: It produces augmented image batches from the validation subset. It's crucial to note that while the validation data goes through the same augmentation processes, it's essential to have a separate, untouched test set for final model evaluation.



- Model Architecture:

MobileNetV2 as the Base Model: This model is a highly efficient architecture optimized for mobile and edge devices. It's being used here as a feature extractor by loading pre-trained weights on the ImageNet dataset and setting the include_top parameter to False.

Fine-tuning: The last 10 layers of the MobileNetV2 model are set as trainable. This means you're keeping the early layers of the MobileNetV2 frozen (to utilize the pre-trained weights) and only fine-tuning the deeper layers for the specific task at hand.

GlobalAveragePooling2D Layer: This layer is used to reduce the spatial dimensions of the output tensor from the base model.

BatchNormalization: Normalizes the activations of the current layer, which can help improve the speed, performance, and stability of the training process.

Dense Layers with Dropouts: A series of fully connected layers (Dense) with Dropout layers in between. The dropout layers randomly set a fraction of the input units to 0 at each update during training time, which helps prevent overfitting.

Final Dense Layer: Since it's a binary classification task (as indicated by the use of binary_crossentropy loss), the model has a single output neuron with a sigmoid activation function.

Training Configuration:

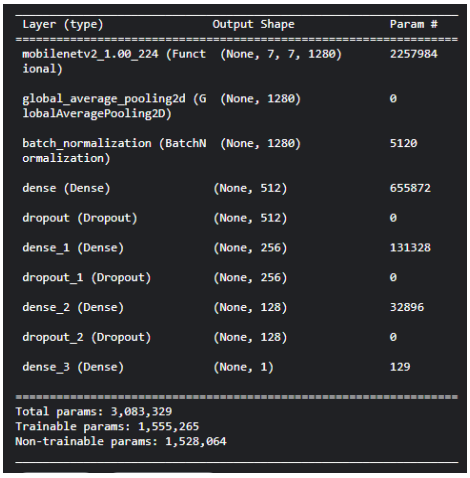
Adam Optimizer: A widely-used optimization algorithm that computes adaptive learning rates for each parameter.

Early Stopping: This callback will stop the training process if the validation loss doesn't improve for a set number of epochs (patience=5), preventing overfitting and saving time.

Model Checkpoint: This will save the model weights when the validation loss is at its minimum.

Learning Rate Scheduler: Adjusts the learning rate based on the epoch number, which can lead to faster convergence and improved generalization.

Class Weights: Since the data might be imbalanced, compute_class_weight is used to calculate weights for each class, ensuring that the model doesn't get biased towards the majority class.



Layer (type)	Output Shape	Param #
mobilenetv2_1.00_224 (Functional)	(None, 7, 7, 1280)	2257984
global_average_pooling2d (GlobalAveragePooling2D)	(None, 1280)	0
batch_normalization (BatchNormalization)	(None, 1280)	5120
dense (Dense)	(None, 512)	655872
dropout (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 256)	131328
dropout_1 (Dropout)	(None, 256)	0
dense_2 (Dense)	(None, 128)	32896
dropout_2 (Dropout)	(None, 128)	0
dense_3 (Dense)	(None, 1)	129

Total params: 3,083,329
Trainable params: 1,555,265
Non-trainable params: 1,528,064

- Model Training and Validation:

```
Epoch 1/10
3/3 [==============================] - 11s 1s/step - loss: 1.1305 - accuracy: 0.5570 - val_loss: 0.6786 - val_accuracy: 0.4211 - lr: 1.0000e-04
Epoch 2/10
3/3 [==============================] - 4s 1s/step - loss: 0.6737 - accuracy: 0.7089 - val_loss: 0.6605 - val_accuracy: 0.4737 - lr: 1.0000e-04
Epoch 3/10
3/3 [==============================] - 3s 1s/step - loss: 0.6615 - accuracy: 0.6962 - val_loss: 0.6307 - val_accuracy: 0.6842 - lr: 1.0000e-04
Epoch 4/10
3/3 [==============================] - 3s 1s/step - loss: 0.6276 - accuracy: 0.6835 - val_loss: 0.5813 - val_accuracy: 0.6842 - lr: 1.0000e-04
Epoch 5/10
3/3 [==============================] - 3s 1s/step - loss: 0.3775 - accuracy: 0.8101 - val_loss: 0.5066 - val_accuracy: 0.8421 - lr: 1.0000e-04
Epoch 6/10
3/3 [==============================] - 3s 950ms/step - loss: 0.4644 - accuracy: 0.7722 - val_loss: 0.5247 - val_accuracy: 0.8421 - lr: 5.0000e-05
Epoch 7/10
3/3 [==============================] - 3s 924ms/step - loss: 0.3352 - accuracy: 0.8481 - val_loss: 0.5674 - val_accuracy: 0.6842 - lr: 5.0000e-05
Epoch 8/10
3/3 [==============================] - 3s 1s/step - loss: 0.3711 - accuracy: 0.8101 - val_loss: 0.4894 - val_accuracy: 0.7895 - lr: 5.0000e-05
Epoch 9/10
3/3 [==============================] - 3s 1s/step - loss: 0.2809 - accuracy: 0.8608 - val_loss: 0.4648 - val_accuracy: 0.7895 - lr: 5.0000e-05
Epoch 10/10
3/3 [==============================] - 3s 1s/step - loss: 0.3564 - accuracy: 0.8101 - val_loss: 0.5270 - val_accuracy: 0.7368 - lr: 5.0000e-05
```

Training and Validation Accuracy Observation:

The training accuracy sees a noticeable increase from epoch 0 to epoch 2, then continues to fluctuate without a consistent upward trend from epochs 2 to 9. The validation accuracy, on the other hand, follows a similar initial rise but then shows a notable drop around epochs 3 and 4 before rising again. After epoch 6, it seems to stabilize but with slight fluctuations.

The divergence between the training and validation accuracy indicates that while the model is getting better at classifying the training data (overfitting), its performance on unseen data (validation set) is deteriorating. This suggests the model may be memorizing the training data rather than generalizing to new data.

Training and Validation Loss Observation:

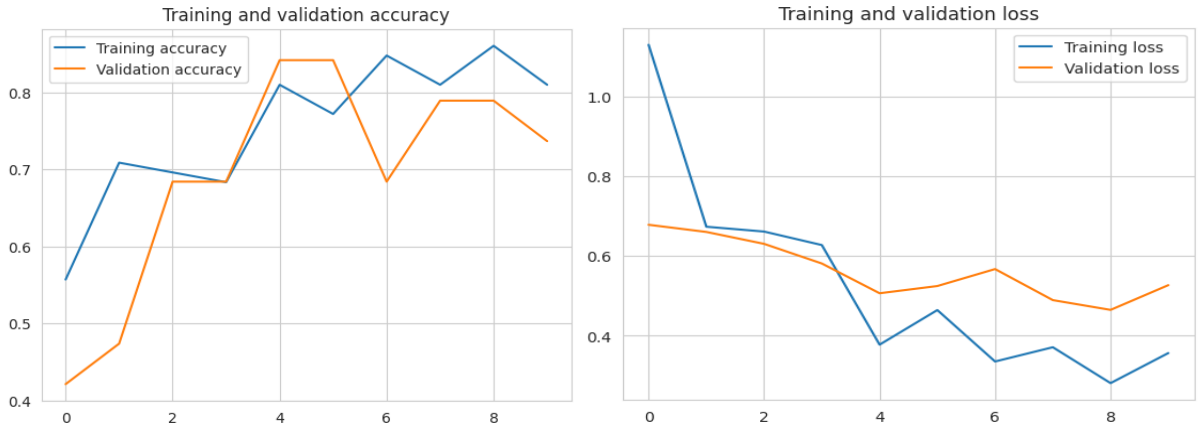
Training loss starts off very high but sees a sharp decrease till about epoch 2. Afterward, there's a gentle decline till epoch 5 and then some fluctuations. Validation loss starts slightly lower than the training loss and decreases in a similar pattern till epoch 2. It then fluctuates more than

The divergence between training and validation loss is a classic sign of overfitting. The model is becoming too specialized for the training data and losing its ability to generalize to new, unseen data.

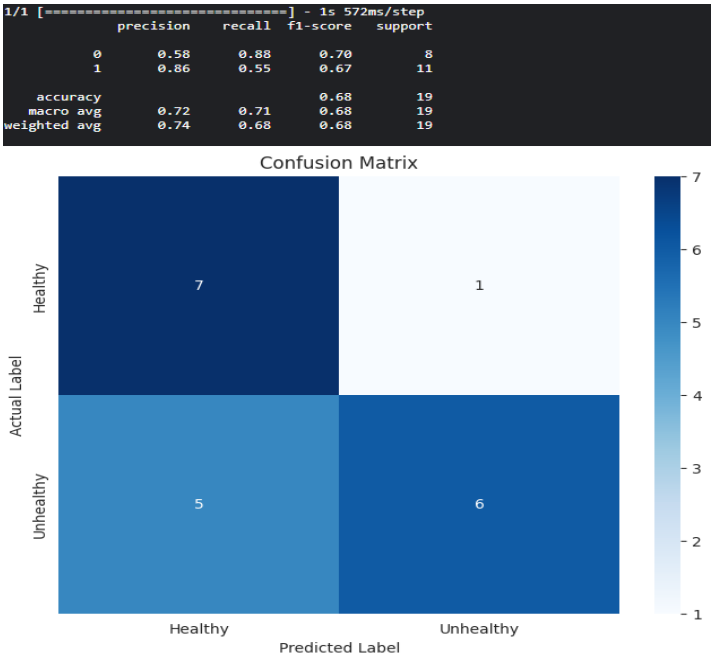
Overfitting Indicators:

While the training accuracy increases, the validation accuracy does not follow suit, especially between epochs 3 and 5. This divergence between training and validation accuracy could be an early sign of overfitting.

A similar observation can be made from the loss graphs, where the training loss decreases but the validation loss sees an increase in the same period (epochs 3-5).



Classification and Confusion Matrix Report:



Confusion Matrix:

True Positives (TP): The dark blue box in the bottom right indicates that 6 unhealthy buildings were correctly classified as unhealthy.

True Negatives (TN): The dark blue box in the top left indicates that 7 healthy buildings were correctly classified as healthy.

False Positives (FP): The Light white box in the top right indicates that 1 buildings were incorrectly classified as unhealthy when they were actually healthy.

False Negatives (FN): The Light blue box in the bottom left indicates that 5 unhealthy buildings were misclassified as healthy.

Observations:

For the "Healthy" label (0):

Precision is 0.58, indicating that 58% of the instances predicted as "Healthy" were actually healthy.

Recall is 0.88, suggesting that the model correctly identified 88% of all actual healthy instances.

F1-score is 0.70, which balances precision and recall.

For the "Unhealthy" label (1):

Precision is 0.86, indicating that 86% of the instances predicted as "Unhealthy" were actually unhealthy.

Recall is 0.55, meaning the model identified only 55% of all actual unhealthy instances.

F1-score is 0.67.

Accuracy: The model's overall accuracy on the validation set is 68%, suggesting it correctly classifies the health condition in 68% of the instances.

The recall for the "Unhealthy" label is considerably lower than for the "Healthy" label. It might be beneficial to aim for better detection of "Unhealthy" instances as misclassifying them could have significant implications depending on the application.

Balancing the Dataset: The differences in precision and recall might be due to an imbalanced dataset. Consider checking the distribution of the "Healthy" and "Unhealthy" labels in the dataset and consider oversampling the underrepresented class or under sampling the overrepresented class.

More Data: Acquiring more data can greatly benefit the model's performance, especially in deep learning where large datasets can lead to better generalization. More data provides a richer representation of the problem space, helping the model to better distinguish between "Healthy" and "Unhealthy.