**Introdution**

Respiratory diseases like COVID-19, tuberculosis, and pneumonia are serious global health threats. Chest radiological images help doctors detect and monitor these conditions early. Manual analysis of these images is time-consuming and can lead to inconsistent or inaccurate results.

**Motivation**

**1. Increasing Prevalence of Infectious Diseases:**

Infectious diseases such as tuberculosis, pneumonia, and COVID-19 require accurate and timely diagnosis for effective treatment and control, highlighting the need for advanced diagnostic tools.

**2. Need for Faster and Scalable Diagnostic Solutions:**

Due to this, medical images are processed much faster.

**3. Potential for Early Detection and Improved Patient Outcomes:**

Due to early diagnosis, its treatment is much faster.

**4. Reduction in Healthcare Costs and Resource Optimization:**

It will reduce the healthcare cost and improved efficiency.

**Background Study:**

**Challenges in Traditional Radiological Diagnosis:**

Traditional chest radiology faces many problem from detection diseases. That's why automated diagnostic important.

**Application of CNNs in Medical Imaging:**

In recent years, CNNs have been widely applied in medical imaging. The advantages of CNNs, such as high accuracy for the classification of infectious diseases in chest radiological images.

**Significance of the Study:**

Research into CNNs for classifying infectious diseases in chest radiological images is to unlocking their diagnostic challenges, and advancing medical image analysis.

**Methodology**

This section delves into the step-by-step procedure we have adopted, offering insights into our data handling practices, model developmental strategies, evaluation metrics, and introspective analysis techniques.

**1. Data Preparation:**

The dataset for this research is sourced from an open-access compilation of chest radiological images online. Categorized into COVID-19, Lung Opacity, Normal, Viral Pneumonia, and Tuberculosis, it offers a broad spectrum for investigation.

**2. Model Development and Training:**

Various deep learning models, including architectures like VGG19 and ResNet50, are fine-tuned on the curated dataset. Hyperparameters are adjusted to enhance performance.

**3. Model Evaluation and Comparison:**

Post-training, models are evaluated on a separate test set. Metrics like accuracy, precision, recall, F-score, and ROC-AUC offer a holistic performance view. A comparative analysis will be conducted to determine the most effective model.

**4. Model Introspection and Analysis:**

Understanding how deep learning models make decisions is crucial, especially in the medical field, where interpretability can be the bridge between artificial intelligence and clinical adoption. Visualization tools aid in demystifying the intricate operations within models.

**Data Collection:**

The process begins with **data collection**, where I utilized a comprehensive chest X-ray dataset comprising over 32,000 samples, categorized into five classes: COVID-19, Tuberculosis, Lung Opacity, Viral Pneumonia, and Normal. The dataset’s diversity is crucial for creating a robust model capable of accurately diagnosing multiple respiratory conditions.

**Preprocessing:**

Next, the data underwent **preprocessing**, a vital step to enhance image quality and consistency. I applied three key techniques:

1. **Image Sharpening** to enhance the clarity of key radiological features.

2. **Zooming** to focus on the central chest region, reducing noise and irrelevant artifacts.

3. **CLAHE (Contrast Limited Adaptive Histogram Equalization)** to improve image contrast and highlight subtle features.

These preprocessing steps were optimized to ensure the dataset was well-suited for deep learning models.

**Model Selection:**

For model selection, I evaluated four architectures:

1. **Custom CNN** - A proprietary model designed with spatial attention mechanisms and depthwise separable convolutions.

2. **VGG19** - A classical CNN known for its effectiveness in image classification tasks.

3. **ResNet50** - A deep residual learning framework that mitigates vanishing gradient issues.

4. **Vision Transformers (ViT)** - A cutting-edge model incorporating self-attention mechanisms for better interpretability.

**Model Evaluation:**

The models were evaluated using standard performance metrics:

• **Accuracy** to measure the overall correctness of predictions.

• **Precision** to evaluate how many predicted positives were actually correct.

• **Recall** to identify the model’s ability to detect all relevant cases.

• **F1 Score**, a balance between precision and recall.

**Output:**

Finally, the best-performing model produced the output, providing predictions for each chest X-ray. This system has the potential to assist clinicians in diagnosing respiratory diseases more effectively and efficiently.

Himel:

**Related Work: 7**

We have brought out some similar research papers related to our work. Here they worked with two or three models. They use the X-ray dataset and they have the high accuracy with 94%. In limitations they have the lacks comparative analysis of various models, and there's no mention of explainability techniques like Grad-CAM

**Gap Analysis:**

The CNN model works well to find infectious diseases. But sometimes it may not works due to its limitation of dataset.

CNNs are often criticized for being "black boxes," due to it’s limited classification decisions.

Sometimes CNN has limited exploration so that we need to use another model such as Resnet, DenseNet.

**Methodology:**

First we prepare the data, starting with a quality check to ensure accuracy. Next, we move to image preprocessing, followed by data augmentation to increase diversity in the training set.

In the second phase, we move into model development and training, using the prepared data to train our model.

Once trained, we conduct model evaluation and comparison to measure performance.

Finally, in the last phase, we perform model introspection and analysis.

**Conclusion:**

This study demonstrates that CNNs can accurately classify infectious diseases like pneumonia and tuberculosis from chest images and in future it will improve our healthcare system.

VGG19

**Training of Classification Layers:**

In the first row, the metrics during the training of classification layers are shown:

1. **Accuracy and AUC:** We see steady improvement for both training and validation, indicating that the model is learning effectively. By the 15th epoch, both metrics approach near-optimal levels, with validation performance closely following training.

2. **Loss:** The loss decreases significantly in both training and validation, suggesting better optimization. However, slight oscillations in validation loss indicate variability due to class imbalances.

3. **Precision:** Precision improves consistently, reflecting the model’s ability to accurately predict the positive cases.

**Fine-Tuning the Entire Network:**

The second row shows the metrics during fine-tuning:

1. **Accuracy and AUC:** While training accuracy and AUC improve further, validation performance plateaus and slightly diverges, suggesting **overfitting** during fine-tuning.

2. **Loss:** Training loss continues to decrease, but validation loss increases slightly, reinforcing the overfitting concern.

3. **Precision:** Training precision improves steadily, but validation precision remains relatively stable, showing that fine-tuning may not significantly enhance generalization.

Custom CNN

**History of Accuracy:**

The first graph shows the **accuracy trends**. Both training and validation accuracy improve steadily over the epochs, converging near **95% validation accuracy**. This demonstrates that the Custom CNN effectively learns to distinguish between the different disease classes with minimal divergence between training and validation, which indicates good generalization.

**History of Loss:**

In the second graph, the **loss** for both training and validation decreases sharply in the early epochs and stabilizes as training progresses. The low and stable validation loss suggests that the model avoids overfitting, which is a common challenge in deep learning for medical datasets.

Overall Performance:  
**Model Comparisons:**

1. **Custom CNN**:

• Achieved the highest **test accuracy of 97%**, demonstrating exceptional performance.

• Despite slightly lower precision (0.928) and recall (0.921) compared to other models, it performed consistently across all metrics.

• The slightly higher loss (0.208) reflects its ability to generalize effectively without overfitting.

2. **VGG19**:

• Delivered a strong performance with a **test accuracy of 94.5%** and high AUC (0.995).

• It achieved the best balance of **precision (0.946)** and recall (0.943), leading to the highest F1-score (0.945), making it highly reliable for balanced datasets.

3. **ResNet50**:

• Achieved a respectable **accuracy of 93.6%** and consistent performance across **precision, recall, and F1-score**.

• Its performance metrics, while slightly lower than VGG19 and Custom CNN, reflect its robustness for image classification.

4. **Vision Transformer (ViT)**:

• While ViT introduced innovation, its performance lagged behind with an **accuracy of 78.6%** and lower precision and recall.

• The higher loss (0.662) suggests challenges in handling the dataset compared to CNN-based architectures.

**CUSTOM CNN**

**Tell me about custom cnn here used..**

The **Custom CNN** (Convolutional Neural Network) described in the document is a specially designed architecture that incorporates advanced features to optimize its performance for classifying chest radiological images. Here's a detailed breakdown of its characteristics and functionalities:

**1. Architectural Features**

* **Spatial Attention Mechanisms**:
  + Enhances the model's ability to focus on the most relevant regions in the images, such as abnormal areas in the lungs, while reducing attention to irrelevant or noisy regions.
* **Depthwise Separable Convolutions**:
  + These convolutions decompose the standard convolution operation into depthwise and pointwise convolutions, significantly reducing computational cost and improving efficiency.
* **Skip Connections**:
  + Similar to ResNet, skip connections are used to mitigate the vanishing gradient problem in deep networks. They enable better gradient flow and improve the learning process by reusing features learned in earlier layers.
* **ResNet and VGG-like Blocks**:
  + Combines elements from popular architectures like ResNet (residual learning) and VGG (depth through smaller filters) for balanced performance.

**2. Training Strategy**

* **Training from Scratch**:
  + Unlike pre-trained models (e.g., VGG19, ResNet50), the Custom CNN is trained from the ground up on the dataset, allowing it to learn domain-specific features tailored to the chest X-ray data.
* **Learning Rate Scheduling**:
  + A learning rate scheduler adjusts the learning rate dynamically. For example, it decreases the learning rate when no improvement is observed in the validation AUC, stabilizing training and improving convergence.
* **Cross-Entropy Loss and Adam Optimizer**:
  + The model optimizes using categorical cross-entropy as the loss function, which is suitable for multi-class classification. The Adam optimizer facilitates efficient gradient descent.

**3. Performance Insights**

* **Loss Convergence**:
  + The model showed a consistent decline in training loss and achieved favorable convergence patterns in validation loss, avoiding significant overfitting.
* **Validation Loss Oscillations**:
  + Initial oscillations in validation loss were observed, which is typical for training complex models. These oscillations stabilized with learning rate scheduling.
* **Classification Challenges**:
  + The model faced difficulty in distinguishing between visually similar classes, such as 'Lung Opacity' and 'Normal,' due to overlapping radiological features.
* **High Accuracy and Balanced Metrics**:
  + An overall accuracy of 0.92 was achieved, with strong macro-averaged precision, recall, and F1 scores (~0.92), indicating balanced performance across all classes.

**4. Confusion Matrix and Misclassifications**

* **Lung Opacity vs. Normal**:
  + Misclassifications between these classes were the most significant due to their similar pixel intensity distributions.
* **COVID-19 Misclassifications**:
  + COVID-19 was sometimes confused with other conditions like 'Lung Opacity' or 'Viral Pneumonia,' reflecting overlapping radiographic features.

**5. Interpretability**

* The use of spatial attention mechanisms aids in highlighting the regions of the chest X-rays that contribute most to the model's decisions, making the model's predictions more interpretable.

**6. Advantages**

* **Optimized for Radiological Data**:
  + The architecture was customized to handle the nuances of medical imaging, such as subtle contrasts and variable patterns across diseases.
* **Efficiency**:
  + The use of depthwise separable convolutions and attention mechanisms enhances computational efficiency without sacrificing accuracy.

**7. Areas for Improvement**

* While the Custom CNN outperformed other architectures in some areas, it still struggled with certain challenging classes. This suggests potential for further optimization, such as more sophisticated pre-processing or integration with clinical data for better contextual understanding.

The Custom CNN demonstrates a thoughtful design tailored to the unique demands of medical imaging, striking a balance between computational efficiency, accuracy, and interpretability.

**What is the reason for custom cnn? and why we use custom cnn? what cnn things we have changed and here we know custom cnn?**

The use of a **Custom CNN** in this study stems from several motivations and the need to address specific challenges in medical imaging tasks, particularly in chest radiological image classification. Here's a detailed explanation:

**1. Reasons for Using Custom CNN**

* **Domain-Specific Challenges**:
  + Pre-trained models like VGG19 and ResNet50 are general-purpose architectures trained on datasets like ImageNet. While effective, these models may not fully exploit domain-specific patterns in chest radiological images.
  + Custom CNNs allow tailoring the architecture to better handle the nuances of chest X-rays, such as subtle contrasts and overlapping radiological features.
* **Overcoming Pre-Trained Model Limitations**:
  + Pre-trained models often require extensive fine-tuning and may still struggle with specific classes or tasks due to their generic design.
  + A Custom CNN can be designed from the ground up with specialized layers and mechanisms to focus on disease-specific features.
* **Incorporating Advanced Techniques**:
  + The inclusion of mechanisms like spatial attention and depthwise separable convolutions enhances the model's focus and efficiency, which pre-trained models might lack.
* **Interpretability and Clinical Relevance**:
  + Custom models can be designed to be more interpretable, aligning with the needs of medical professionals for transparent and explainable AI in clinical diagnostics.

**2. Why Use a Custom CNN?**

* **Flexibility in Design**:
  + Custom CNNs allow researchers to incorporate features specifically designed for medical imaging, such as spatial attention mechanisms to highlight disease-affected regions.
* **Efficiency**:
  + By using depthwise separable convolutions and optimized architectural components, the Custom CNN can achieve higher efficiency (reduced computation and memory requirements) compared to standard CNNs.
* **Focus on Domain-Specific Features**:
  + Medical images often exhibit unique characteristics (e.g., subtle patterns, noisy backgrounds) that require specialized processing. Custom CNNs can include operations like enhanced feature extraction to address these challenges.
* **Addressing Class Imbalances**:
  + By incorporating mechanisms like weighted loss functions or class-specific attention, a Custom CNN can better handle imbalanced datasets, which is common in medical imaging.

**3. Changes in the Custom CNN (Compared to Standard CNNs)**

* **Spatial Attention Mechanisms**:
  + Introduced to focus on the most critical regions in chest X-rays, ensuring the model prioritizes disease-affected areas while ignoring irrelevant regions.
* **Depthwise Separable Convolutions**:
  + A replacement for standard convolutions to reduce computational costs while maintaining representational power.
* **Residual Blocks**:
  + Borrowed from architectures like ResNet, these blocks allow better gradient flow and improve convergence, especially in deeper networks.
* **Skip Connections**:
  + Help to mitigate the vanishing gradient problem and reuse learned features, leading to faster and more stable training.
* **Custom Feature Extractors**:
  + Tailored filters and layers that are specifically designed to capture the characteristics of chest radiological images.

**4. Key Differences and What We Know About the Custom CNN**

* **Designed for Medical Imaging**:
  + Unlike generic CNNs, this architecture integrates features to address specific challenges in chest X-ray data, such as overlapping class features and subtle visual patterns.
* **Focus on Misclassification Reduction**:
  + Custom components are aimed at minimizing errors between visually similar classes like 'Lung Opacity' and 'Normal.'
* **Balanced Performance**:
  + Achieved a consistent accuracy (~92%) across all classes, with balanced precision, recall, and F1 scores.
* **Optimization**:
  + Adaptive learning rate scheduling and training from scratch ensure that the model is tuned specifically for the dataset.
* **Interpretable Design**:
  + Spatial attention maps provide insights into the model's focus areas, making predictions more transparent.

**Conclusion**

The Custom CNN was introduced to address the specific challenges of chest radiological image classification, such as class overlaps, subtle disease features, and the need for interpretability. By tailoring the architecture with spatial attention, depthwise convolutions, and residual blocks, the researchers enhanced both the model's efficiency and its ability to generalize across classes. The design reflects a thoughtful balance between leveraging deep learning principles and addressing domain-specific requirements.

**Why ViT model give us less accuracy?**

The **Vision Transformer (ViT)** model may provide less accuracy in this study for several reasons, which are largely related to the characteristics of ViT, the dataset, and the challenges specific to medical imaging tasks. Here's a detailed analysis:

**1. Characteristics of ViT and Its Limitations**

* **Patch-Based Processing**:
  + ViT divides images into fixed-size patches, processes them independently, and then aggregates the information. This approach can cause a loss of fine-grained local details, which are critical for medical images like chest X-rays, where subtle visual features can indicate disease.
* **Lack of Inductive Bias**:
  + Unlike CNNs, ViTs do not have built-in inductive biases such as spatial locality and translation invariance. These biases are crucial in medical imaging to capture patterns like textures and edges effectively.
* **Data-Hungry Nature**:
  + ViTs require large datasets to learn effectively because they lack the architectural constraints of CNNs. The dataset used in this study (32,687 samples) is relatively small compared to the vast datasets typically used to train ViTs, such as ImageNet-21k. This data limitation may lead to underfitting and lower accuracy.

**2. Challenges in Medical Imaging for ViT**

* **Subtle Differences Between Classes**:
  + Medical imaging datasets often involve classes with subtle differences in visual patterns, such as 'Lung Opacity' vs. 'Normal.' ViTs might struggle to discern these differences without the strong feature extraction capabilities of CNNs.
* **Class Imbalance**:
  + In this study, the dataset showed significant class imbalances, with some diseases having fewer samples than others. While CNNs can be adapted to handle this through techniques like class weighting, ViTs might be more affected due to their reliance on abundant data for learning.
* **Interpretability**:
  + ViTs rely on self-attention mechanisms that aggregate global context. This might not be as effective in identifying localized patterns (e.g., specific regions of abnormality in chest X-rays), leading to reduced performance.

**3. Dataset-Specific Factors**

* **Preprocessing Choices**:
  + The study employed advanced pre-processing techniques like CLAHE, sharpening, and zooming. These techniques enhance local features, which CNNs are inherently better at leveraging. ViTs may not benefit as much because they aggregate information globally.
* **Training Challenges**:
  + Training ViTs often requires sophisticated techniques like stronger regularization, data augmentation, and optimized learning rates. If these were not fully implemented or optimized for the ViT in this study, it could lead to suboptimal performance.

**4. Model Architecture Comparison**

* **Feature Extraction vs. Contextual Understanding**:
  + CNNs like VGG19 and ResNet50 excel at extracting hierarchical features, which are essential for distinguishing subtle patterns in medical images.
  + ViTs focus on contextual relationships between image regions, which may not be as beneficial for this task, where local details often dominate decision-making.
* **Parameter Count**:
  + ViTs typically have a higher number of parameters compared to CNNs, making them harder to train effectively on smaller datasets. Overfitting or underfitting can result if the dataset size and training process are not adequately matched to the model's capacity.

**5. Specific Observations from the Study**

* **Overfitting or Underfitting**:
  + The ViT model may have underfitted the dataset due to insufficient data or inappropriate training settings.
* **Interpretability Issues**:
  + The study highlighted interpretability challenges with ViTs, where attention maps might not align well with clinically relevant regions, reducing confidence in the model's predictions.

**6. Recommendations for Improving ViT Performance**

* **Data Augmentation**:
  + Use aggressive data augmentation techniques to increase the effective size of the dataset and provide more training examples.
* **Transfer Learning**:
  + Fine-tune a ViT pre-trained on large, diverse datasets like ImageNet-21k to leverage prior knowledge and improve performance.
* **Hybrid Models**:
  + Combine ViTs with CNN components (e.g., hybrid architectures) to capture both global context and local details effectively.
* **Class Imbalance Mitigation**:
  + Use advanced techniques such as oversampling minority classes, synthetic data generation, or focal loss to address class imbalance.

**Conclusion**

The **ViT model's lower accuracy** in this study is primarily due to its inherent design characteristics, reliance on large datasets, and challenges specific to medical imaging tasks. While ViTs have shown promise in general computer vision tasks, their application to medical imaging may require larger datasets, better training optimization, or hybrid architectures to achieve competitive performance.

 **GlobalAveragePooling2D**: Reduces the spatial dimensions to a single vector.

 **Dense**: Fully connected layers for learning new features.

 **Dropout**: Helps prevent overfitting by randomly setting inputs to zero.

 **Dense (output layer)**: Number of units corresponds to the number of target classes, and activation is softmax for multi-class classification.

**Explanation of Custom Layers**

* **GlobalAveragePooling2D**: Converts feature maps into a single vector per image, reducing spatial dimensions.
* **Fully Connected Layers**: Add complexity to learn task-specific features.
* **Dropout**: Regularization technique to reduce overfitting.
* **Output Layer**: Tailored to the number of target classes in the dataset.
*  **epochs**: Number of training iterations over the entire dataset (100 epochs in this case).
*  **lr\_reduction\_factor**: Factor by which the learning rate is reduced (multiplied by 0.5).
*  **lr\_patience**: Number of epochs with no improvement before reducing the learning rate.
*  **min\_lr**: Minimum learning rate to prevent it from becoming too small.
*  **early\_stopping\_patience**: Number of epochs with no improvement before stopping training.
*  **best\_model\_path**: Path to save the best model weights during training.

**Detailed Explanation of Custom Layers**

1. **GlobalAveragePooling2D**:
   * **Purpose**: Reduces the spatial dimensions of the feature maps output by VGG19 into a single vector per feature map. This reduces the number of parameters and avoids overfitting.
   * **Why?**:
     + Reduces the complexity of the model.
     + Preserves the global context of the features extracted by the base model.
2. **Dense(256, activation='relu')**:
   * **Purpose**: Fully connected layer with 256 neurons to learn task-specific patterns.
   * **Activation Function**: ReLU introduces non-linearity and prevents vanishing gradients.
3. **Dropout(0.5)**:
   * **Purpose**: Reduces overfitting by randomly setting 50% of the neurons to zero during each training step.
4. **Dense(128, activation='relu')**:
   * **Purpose**: A second fully connected layer with 128 neurons to increase learning capacity and capture additional patterns.
5. **Dropout(0.5)**:
   * **Purpose**: Adds regularization again to prevent overfitting in the smaller, high-capacity layer.
6. **Dense(num\_classes, activation='softmax')**:
   * **Purpose**: Output layer tailored for multi-class classification.
   * **num\_classes**: The number of output classes (e.g., 3 if the task involves predicting one of 3 categories such as normal, pneumonia, or another condition).
   * **Activation Function**: Softmax ensures the output is a probability distribution over the classes.

**Summary of Fine-Tuning Workflow**

1. **Load ResNet50**: Pretrained on ImageNet, excluding the top layers.
2. **Freeze Base Model**: Train custom layers while keeping base model layers unchanged.
3. **Add Custom Layers**: Adapt the model for the specific classification task.
4. **Train Custom Layers**: Focus on training the new layers while leveraging pretrained features.
5. **Unfreeze and Fine-Tune**: Gradually fine-tune the entire model with a lower learning rate.
6. **Callbacks**: Optimize training with learning rate reduction, early stopping, and checkpointing.

**Grad-CAM Workflow**

Grad-CAM highlights areas in the input image that contributed most to the model's prediction by using the gradients of the class score concerning the feature maps of the last convolutional layer.

**Steps in the Notebook:**

1. **Identify the Final Convolutional Layer**:
   * Grad-CAM requires the feature maps from the final convolutional layer of the model.

python

Copy code

last\_conv\_layer = model.get\_layer("conv5\_block3\_out") # Example for ResNet50

**Compute Gradients**:

* Gradients of the predicted class score concerning the feature maps of the last convolutional layer are calculated.

Grad-CAM (Gradient-weighted Class Activation Mapping) and Guided Grad-CAM are visualization techniques used to interpret the predictions of deep learning models by highlighting important regions in the input images. Here's how these techniques likely work in your notebook:

**Grad-CAM Workflow**

Grad-CAM highlights areas in the input image that contributed most to the model's prediction by using the gradients of the class score concerning the feature maps of the last convolutional layer.

**Steps in the Notebook:**

1. **Identify the Final Convolutional Layer**:
   * Grad-CAM requires the feature maps from the final convolutional layer of the model.

python

Copy code

last\_conv\_layer = model.get\_layer("conv5\_block3\_out") # Example for ResNet50

1. **Compute Gradients**:
   * Gradients of the predicted class score concerning the feature maps of the last convolutional layer are calculated.

python

Copy code

grad\_model = tf.keras.models.Model([model.inputs], [last\_conv\_layer.output, model.output])

with tf.GradientTape() as tape:

conv\_outputs, predictions = grad\_model(input\_image)

predicted\_class = tf.argmax(predictions[0])

loss = predictions[:, predicted\_class]

grads = tape.gradient(loss, conv\_outputs)

1. **Weight the Feature Maps**:
   * The gradients are averaged spatially to obtain the weights for each feature map.

python

Copy code

pooled\_grads = tf.reduce\_mean(grads, axis=(0, 1, 2))

1. **Generate the Heatmap**:
   * The feature maps are weighted by the pooled gradients and summed to create a heatmap.

python

Copy code

heatmap = tf.reduce\_sum(tf.multiply(pooled\_grads, conv\_outputs), axis=-1)

heatmap = tf.maximum(heatmap, 0) / tf.math.reduce\_max(heatmap) # ReLU and normalize

1. **Overlay Heatmap on the Input Image**:
   * The heatmap is resized and overlaid on the input image to highlight regions of importance.

python

Copy code

import matplotlib.pyplot as plt

import cv2

heatmap = cv2.resize(heatmap.numpy(), (input\_image.shape[2], input\_image.shape[1]))

heatmap = cv2.applyColorMap(np.uint8(255 \* heatmap), cv2.COLORMAP\_JET)

superimposed\_image = heatmap \* 0.4 + input\_image

plt.imshow(superimposed\_image)

**Guided Grad-CAM Workflow**

Guided Grad-CAM combines Grad-CAM with Guided Backpropagation, which visualizes pixel-level contributions.

**Steps in the Notebook:**

1. **Guided Backpropagation**:
   * Modify the gradient computation to ensure only positive gradients are backpropagated through ReLU activations. This filters noise and highlights relevant features.

A screenshot of a black screen

Description automatically generated

**What Happens in the Notebook?**

1. **Grad-CAM**:
   * Gradients of the output class score are computed with respect to the last convolutional layer.
   * A heatmap is generated and overlaid on the input image.
2. **Guided Grad-CAM**:
   * Gradients are computed while filtering out negative influences using Guided Backpropagation.
   * The Grad-CAM heatmap is combined with the guided gradients to create a detailed visualization.

**1. Spatial Attention**

The spatial\_attention function applies attention to highlight important spatial regions of the input features.

**Steps:**

1. **Convolution for Attention**:
   * If the input feature map has multiple channels, a Conv2D layer with 1 output channel and a sigmoid activation is applied.
   * This generates a spatial attention map, where values closer to 1 indicate higher importance.
2.  channel\_attention = Conv2D(1, kernel\_size, padding='same', activation='sigmoid',
3. name=f'spatialAttention\_conv\_{layer\_num}')(input\_feature)

**Element-Wise Multiplication**:

* The spatial attention map is multiplied with the input feature map, enhancing important regions while suppressing others.

**2. VGG-Like Block with Spatial Attention**

The vgg\_block function mimics a VGG-style convolutional block, consisting of two convolutional layers followed by batch normalization, ReLU activation, and max pooling.

**Additional Step: Spatial Attention**

* Spatial attention is applied **before pooling** to refine the feature maps.

python

Copy code

x = spatial\_attention(x, layer\_num=f'prePoolVGG\_{block\_num}')

return MaxPooling2D(name=f'vggBlock\_{block\_num}\_pool')(x)

**3. Residual Block**

The simplified\_residual\_block function incorporates **skip connections**, inspired by ResNet, to address the vanishing gradient problem and allow for efficient training of deeper networks.

**Steps:**

1. **VGG Block**:
   * The input feature map passes through a vgg\_block.
2. **Skip Connection**:
   * A shortcut (bypass connection) skips the vgg\_block and directly adds its output to the vgg\_block output.
3. **Adjust for Mismatches**:
   * If there is a mismatch in the spatial dimensions or number of channels between the input and the output of the vgg\_block, the skip connection is adjusted using:
     + Conv2D: For channel mismatch.
     + MaxPooling2D: For spatial mismatch.
4. **Addition**:
   * The adjusted skip connection is added to the output of the vgg\_block.

python

Copy code

**4. Custom Architecture**

The custom\_architecture function builds the overall model.

**Steps:**

1. **Input Layer**:
   * The model takes an input tensor of shape (224, 224, 3).
2. **Initial Spatial Attention**:
   * Spatial attention is applied to the input before feeding it into the first block.
3. **Stacked Residual Blocks**:
   * A series of simplified\_residual\_blocks is used, each with an increasing number of filters:
     + Block 1: 32 filters
     + Block 2: 64 filters
     + Block 3: 128 filters
     + Block 4: 256 filters
4. **Global Pooling**:
   * A GlobalAveragePooling2D layer reduces the spatial dimensions to a single vector.
5. **Fully Connected Layers**:
   * Two dense layers with 1024 neurons each and ReLU activations are added for feature extraction.
   * A Dropout layer with a dropout rate of 50% is used for regularization.
6. **Output Layer**:
   * A final dense layer with 5 neurons (one for each class) and a softmax activation provides class probabilities.

**What Does This Architecture Achieve?**

1. **Spatial Attention**:
   * Enhances the focus on important regions in the input images.
2. **Residual Connections**:
   * Helps in training deeper models by addressing gradient issues and improving feature reuse.
3. **VGG-Like Blocks**:
   * Captures fine-grained spatial features through small convolutional filters.
4. **Global Pooling and Dense Layers**:
   * Extracts high-level features and performs classification.

A screenshot of a black screen

Description automatically generated

**1. CLAHE (Contrast Limited Adaptive Histogram Equalization)**

**Algorithm:** CLAHE is an enhanced version of Adaptive Histogram Equalization (AHE) designed to improve contrast in images while avoiding amplification of noise.

1. **Divide the Image into Tiles:**
   * The input image is divided into small regions called tiles (e.g., 8x8 or 16x16).
2. **Apply Histogram Equalization Locally:**
   * For each tile, compute the histogram and redistribute pixel intensities to equalize the contrast locally.
3. **Limit Contrast Enhancement:**
   * Apply **contrast limiting** by clipping the histogram at a predefined threshold (clip limit) to prevent over-amplification of noise.
   * The excess pixels from clipping are redistributed across the histogram bins.
4. **Interpolate Between Tiles:**
   * To avoid boundary artifacts, bilinear interpolation is used to smooth transitions between neighboring tiles.

**Applications:**

* Medical imaging (e.g., X-rays).
* Enhancing low-light or hazy images.

**2. Zoom (Image Scaling)**

Zooming involves resizing an image, either enlarging or shrinking, while maintaining visual quality.

**Algorithms:**

1. **Nearest-Neighbor Interpolation:**
   * Simplest algorithm.
   * Assigns the nearest pixel value to the target pixel.
   * Fast but can produce blocky artifacts for enlargement.
2. **Bilinear Interpolation:**
   * Considers the 2x2 pixel grid surrounding the target pixel.
   * Computes the weighted average based on distances.
   * Produces smoother results than nearest-neighbor.
3. **Bicubic Interpolation:**
   * Considers a 4x4 pixel grid surrounding the target pixel.
   * Uses cubic polynomials for interpolation.
   * Produces smoother and higher-quality results compared to bilinear interpolation.
4. **Lanczos Resampling:**
   * Uses sinc functions for interpolation over a larger pixel neighborhood.
   * Provides high-quality results but is computationally expensive.

**Applications:**

* Digital zoom in cameras.
* Resizing images for deep learning models or displays.

**3. Image Sharpening**

Image sharpening enhances edges and details in an image by emphasizing high-frequency components.

**Algorithms:**

1. **Unsharp Masking (USM):**
   * Most common sharpening algorithm.
   * Steps:
     1. Create a blurred version of the image (using Gaussian blur).
     2. Subtract the blurred image from the original image to isolate high-frequency details.
     3. Add the high-frequency details back to the original image with a scaling factor: Isharpened=Ioriginal+α×(Ioriginal−Iblurred)I\_{\text{sharpened}} = I\_{\text{original}} + \alpha \times (I\_{\text{original}} - I\_{\text{blurred}})Isharpened​=Ioriginal​+α×(Ioriginal​−Iblurred​) where α\alphaα is the sharpening strength.
2. **Laplacian Filter:**
   * Uses the second derivative of the image to detect edges.
   * The filter kernel emphasizes regions with rapid intensity changes (edges).
   * The Laplacian result is added back to the original image: Isharpened=Ioriginal−Laplacian(Ioriginal)I\_{\text{sharpened}} = I\_{\text{original}} - \text{Laplacian}(I\_{\text{original}})Isharpened​=Ioriginal​−Laplacian(Ioriginal​)
3. **High-Pass Filtering:**
   * Removes low-frequency components (smooth variations) while retaining high-frequency details.
   * Combine the high-pass filtered image with the original for sharpening.

**Applications:**

* Enhancing blurry images.
* Improving edge clarity in medical imaging, satellite images, and photographs.