CT Scan Insights: Unmasking Covid 19 using Convolutional Neural Networks

DS 5220 Supervised Machine Learning
Project Report

Northeastern University

Team:
Rohan Ojha(002253450)
Zankhana Mehta(002320268)

Abstract

This research explores the application of computational neural networks (CNNs) for automated detection of COVID-19 using chest CT images. Using deep learning techniques, we develop a robust classification system capable of distinguishing between positive and negative COVID-19 scans. Our approach demonstrates the potential of artificial intelligence in rapid medical diagnostics, offering a promising tool to support healthcare professionals during pandemic scenarios.

1. Introduction

1.1 Problem Statement

The COVID-19 pandemic has created an unprecedented global health challenge that requires rapid and accurate diagnostic methods. Traditional testing approaches often involve time-consuming and resource-intensive processes. This research investigates the potential of machine learning, specifically Convolutional Neural Networks (CNNs), to automate the detection of COVID-19 using chest CT images.

1.2 Significance

The detection of COVID-19 is crucial for effective pandemic management. Traditional testing methods can be time-consuming and resource intensive. Accurate detection of COVID-19 through imaging not only aids in early diagnosis but also reduces dependency on time-intensive testing methods such as RT-PCR. Automated image classification using machine learning offers the following:

a. Faster diagnosis and screening
b. Reduced human error
c. Potential for early detection
d. Scalable support for healthcare systems

1.3 Proposed Approach

The project implements a deep learning-based classification approach using convolutional neural networks (CNNs) to analyze CT scan images.
The methodology involves:

1. Data pre-processing: resizing the images to fit into transfer learning models
2. Transfer learning models: Transfer learning leverages pre-trained deep learning models to adapt knowledge from a similar domain, addressing the challenge of limited labeled datasets
3. Custom CNN models: Build custom CNN models by trying different architectures.
4. Performance evaluation using multiple metrics

1.4 Rational

1. CNNs have proven to be effective for medical image analysis due to the following:

- Ability to learn hierarchical features automatically
- Strong performance in pattern recognition
- Success in similar medical imaging tasks

2. Transfer learning can help overcome limited dataset size
3. Multiple model comparison will help identify most effective approach

1.5 Key Components and Limitations

Key components:
1. Dataset of CT scans (COVID-19 and normal)
2. Fine-tuning different transfer learning models on the dataset.
3. Developing a robust CNN model for COVID-19 classification. The aim is to achieve better accuracy than that of transfer learning models.
4. Evaluating model performance across multiple metrics
5. Demonstrating the feasibility of AI-driven medical diagnostics

Limitations: Limited diversity in the dataset may affect generalization.

2. Experiment setup

2.1 Dataset

We utilized a comprehensive CT scan dataset containing COVID-19 positive and negative images. Total 1152 CT scans are included in the data set, of which 576 belonged to COVID-19 patients and 576 to non-COVID-19 patients. It is a balanced dataset.

2.2 Model architecture

Transfer Learning models

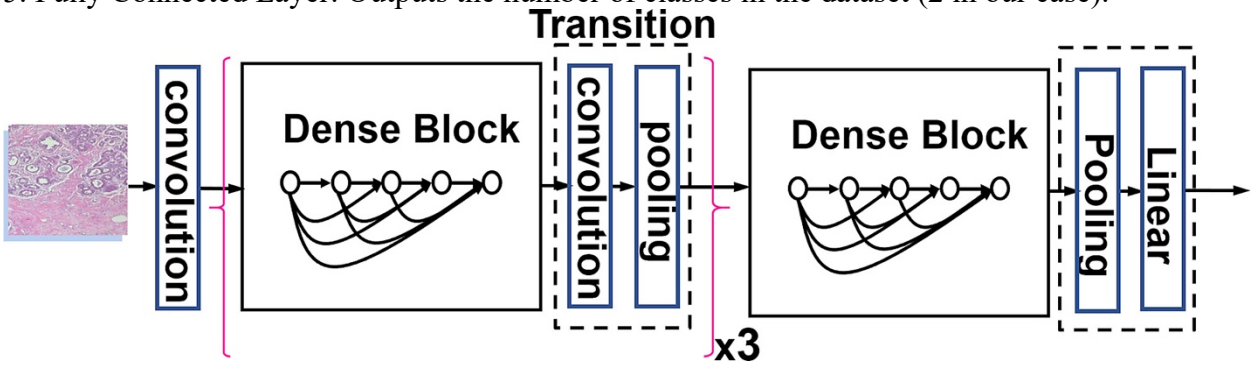
a. **Densenet121**: The DenseNet-121 architecture, a part of the DenseNet family (Dense Convolutional Networks), is designed to promote feature reuse, enhance gradient flow, and reduce the number of parameters.

- Below is a summary of its architecture:
  1. Dense Connectivity:Each layer is directly connected to every other subsequent layer in a "dense block." Feature maps from earlier layers are concatenated and passed as input to later layers. This enables feature reuse, which reduces the number of parameters and mitigates vanishing gradient issues.
  2. Composed of:
  Dense Blocks: Groups of convolutional layers where feature maps are concatenated.
  Transition Layers: Reduce the spatial dimensions and number of feature maps via Batch Normalization, 1x1 Convolutions, and Average Pooling.
  3. Compact Architecture: The 121 in DenseNet-121 refers to the total number of layers in the network.
- Layer Structure of DenseNet-121:

1. Input and Initial Convolution:7x7 Convolution with stride 2 and 64 output channels, followed by Batch Normalization, ReLU activation, and a 3x3 Max Pooling layer.
2. Four Dense Blocks:
Dense Block 1: 6 layers.
Dense Block 2: 12 layers.
Dense Block 3: 24 layers.
Dense Block 4: 16 layers.
Each layer consists of:
Batch Normalization
ReLU activation
1x1 Convolution (bottleneck layer)
3x3 Convolution
3. Transition Layers:

- Placed between dense blocks.
- Reduce feature map dimensions and number of channels.
- Includes Batch Normalization, 1x1 Convolution, ReLU activation, and 2x2 Average Pooling.

4. Global Average Pooling: Reduces feature maps to a single feature vector.
5. Fully Connected Layer: Outputs the number of classes in the dataset (2 in our case).



b. **ResNet18:** The ResNet-18 architecture is part of the ResNet family (Residual Networks), designed to solve the vanishing gradient problem and enable the training of very deep neural networks. ResNet-18 is one of the simplest ResNet models, featuring 18 layers.

- Key Characteristics:

1. Residual Connections: Introduces "skip connections" that bypass one or more layers. The output of a residual block is the sum of the input and the transformed input. These connections help preserve gradient flow and prevent vanishing gradients.
2. Layer Composition: 18 layers consisting of convolutions, Batch Normalization, ReLU activation, and residual connections.

- Layer Structure of ResNet-18:

1. Input and Initial Convolution:
A single 7x7 Convolution with 64 filters, stride 2.
Followed by Batch Normalization, ReLU activation, and 3x3 Max Pooling (stride 2).
2. Residual Blocks:
Four stages of residual blocks, with increasing feature maps and downsampling (when stride > 1).

Each block has 2 convolutional layers with Batch Normalization and ReLU, connected by skip connections.
3. Detailed Breakdown:
Stage 1: 64 filters, 2 residual blocks (each with 2 layers).
Stage 2: 128 filters, 2 residual blocks (downsample at the beginning of the first block using stride 2).
Stage 3: 256 filters, 2 residual blocks (downsample at the beginning of the first block using stride 2).
Stage 4: 512 filters, 2 residual blocks (downsample at the beginning of the first block using stride 2).
4. Global Average Pooling: Converts feature maps into a single feature vector.
5. Fully Connected Layer:Outputs the number of classes (2 in our case).

c. **EfficientNet:** The EfficientNet family of architectures is a series of deep learning models designed to achieve high accuracy while maintaining computational efficiency. It uses a principled approach to scaling the depth, width, and resolution of convolutional neural networks (CNNs).
   - Key Characteristics of EfficientNet:

   Compound Scaling:EfficientNet scales depth (number of layers), width (number of channels), and resolution (input image size) systematically using a compound scaling. This ensures a balance between all three dimensions for optimal performance and efficiency.
   Efficient Architecture: Built on MBConv (Mobile Inverted Bottleneck Convolutions) with depth wise separable convolutions, which reduce computational cost while maintaining accuracy.
   Baseline Network: EfficientNet-B0: The first and smallest model in the EfficientNet family, designed using Neural Architecture Search (NAS) to optimize the architecture for both accuracy and efficiency.
   - EfficientNet Architecture Components:

   1. Stem: A single 3x3 Convolution with stride 2 and Batch Normalization, followed by a Swish activation.
   2. MBConv Blocks: The core building blocks of EfficientNet.
   Consist of:
   Depthwise separable convolutions.
   Squeeze-and-excitation (SE) blocks for channel attention.
   Skip connections for efficient gradient flow (used when the input and output dimensions match).
   3. Compound Scaling: EfficientNet uses the baseline EfficientNet-B0 and systematically scales to larger models (B1 to B7) by adjusting depth, width, and resolution.
   4. Head: Global Average Pooling, followed by a Fully Connected Layer for classification tasks.
d. **Custom CNN architectures**: We have built several versions of our custom neural networks to see which one can provide us a better accuracy.

   Convolution layers: 2, 4 or 6
   Kernel size: 3
   Pool size: 2 or 3

Activation function: ReLu or LeakyReLu
HE initialization

Independent and blended channel processing are two strategies we came across for handling input channels in a convolutional layer. Let's investigate these concepts in detail:
1. Independent Channel Processing

In independent channel processing, each input channel is processed separately. This means the convolutional operation is applied independently to each channel, without mixing information across channels.

Characteristics:

- Separate Filters for Each Channel:
    - If an input has CCC channels (e.g., RGB has 3 channels), each filter processes these channels independently.
    - No mixing occurs between channels during this stage.
- Output:
    - Each channel's features are extracted separately, maintaining the channel's independence.
    - After processing, channels can be combined later in the architecture (e.g., through concatenation).

Advantages:

- Preserves the unique characteristics of each channel.
- Useful when channels have independent meaning (e.g., multi-spectral satellite imagery).

Applications:

- Medical imaging (e.g., processing different modalities like MRI or CT scans separately).
- Tasks where cross-channel correlations are less critical.

2. Blended Channel Processing

In blended channel processing, information from all input channels is combined during the convolutional operation. Filters process across the entire depth of the input, mixing channel information.

Characteristics:

- Depth-Wise Convolution Across All Channels:
    - Filters span the entire depth (number of channels) of the input tensor.

- The convolution operation blends features across channels.

- Output:
  - The output captures inter-channel dependencies, creating a more unified representation of the input.

Advantages:

- Captures inter-channel correlations and dependencies.
- Effective for natural images where the channels are highly correlated (e.g., RGB).

Applications:

- Standard image classification tasks (e.g., processing RGB images).
- Any task where the combined information of all channels is crucial.

We have also implemented HEInitialization which is a is a method for initializing weights in deep neural networks. It is particularly effective for layers that use activation functions like ReLU or its variants. The primary goal of weight initialization is to:

1. Prevent gradients from vanishing or exploding during training.
2. Ensure that the variance of activations is neither too large nor too small as the network goes deeper.

3. Experiment Results

3.1 DenseNet:

In this experiment setup we tried different optimizers - Adam Optimizer, AdamW optimizer, NAdam optimizer, SGD Optimizer, and Adamax Optimizer to the densenet architecture. Each optimizer has been tested on a series of 10 epochs, and the loss values for each epoch are displayed.

We can summarize the result of implementation as follows:

Optimizer	Test accuracy
Adam	67.24%
AdamW	66.81%
NAdam	56.90%
Adamax	65.95%
SGD	64.22%

We also applied a few additional custom layers before DenseNet with a hope to improve its accuracy, but it did not perform as per intuition and gave the test accuracy of only 56.03%.

We concluded this experimental environment with a maximum test accuracy of 67.24% with the adam optimizer.

3.2 ResNet

ResNet18 was trained for 10 epochs using Adam optimizer and it gave a testing accuracy of 51.72%. It also did not perform well and did not reach a satisfactory accuracy percent. But with these pre-trained model results we found an average figure of accuracy and hence a benchmark to train a custom model that outperforms these models.

3.3 Custom CNN

This has been our main playground where we have experimented with numerous approaches. We kept the optimizer constant as adam and the number of epochs has been consistent to a number of 10. A summary of findings is listed in the table below:

Blended Channel Processing

Convolution layer	Kernels	Activation function	Dropout	FC HL	Weight initialization	K-fold	Test accuracy
2	32-64	ReLU	0.5	1	No	No	61.21%
4	32-64-128-256	ReLU	0.5	1	No	No	67.24%
4	32-64-128-256	Leaky-ReLU	0.5	1	No	No	65.95%
6	32-64-128-256-512-1024	ReLU	0.5	2	No	No	64.22%
4	32-64-128-256	ReLU	0.3	2	Yes	Yes	62.50%

Independent Channel Processing

Convolution layer	Kernels	Activation function	Dropout	FC HL	Weight initialization	K-fold	Test accuracy
4	32-64-128-256	ReLU	0.5	2	No	No	70.69%
4	32-64-128-256	ReLU	0.3	2	Yes	No	75.43%
4	32-64-128-256	ReLU	0.25	2	Yes	Yes	75%
4	32-32-32-32	ReLU	0.5	2	No	No	70.26%
4	32-32-32-32	ReLU	0.5	2	No	No	69.83%
4	64-128-256-256	Leaky-ReLU	0.25	2	Yes	No	63.79%

4. Discussion

From the above experiment results we can derive a robust idea of the key findings and insights from the implementation of the various models for COVID-19 classification.

The DenseNet model was tested with several optimizers, including Adam, AdamW, NAdam, Adamax, and SGD. The best performance was achieved with the Adam optimizer, reaching a test accuracy of 67.24%. The NAdam optimizer had the lowest test accuracy at 56.90%, indicating it was not the most suitable for this task.

We extensively experimented with various custom CNN models, exploring different combinations of convolutional layers, kernels, activation functions, dropout rates, fully

connected layers, weight initialization, and k-fold cross-validation. The best-performing custom CNN model achieved a test accuracy of 75.43%, which outperformed the DenseNet and ResNet18 architectures. This custom CNN model used independent channel processing, 4 convolutional layers with ReLU activation, 0.3 dropout, 2 fully connected layers, and no weight initialization or k-fold cross-validation.

While the standard DenseNet and ResNet18 architectures provided reasonable results, the custom CNN model with independent channel processing demonstrated the highest test accuracy, showcasing the potential benefits of tailoring the model architecture to the specific problem at hand.

5. Conclusion

The research undertaken in this project has demonstrated the feasibility and potential of using deep learning techniques for automated COVID-19 detection from chest CT scan images. Through a comprehensive evaluation of various model architectures and optimization approaches, the study has produced promising results.
The key contributions of this work are:

1. Successful implementation of a custom Convolutional Neural Network (CNN) model that achieved a test accuracy of 75.43% in differentiating COVID-19 positive and negative scans. This custom model outperformed the performance of standard architectures like DenseNet and ResNet18.
2. Systematic exploration of different hyperparameters, including convolutional layer configurations, activation functions, dropout rates, and fully connected layer structures, to optimize the custom CNN model for the COVID-19 classification task.
3. Comparative analysis of multiple optimizers, such as Adam, AdamW, NAdam, Adamax, and SGD, on the DenseNet architecture, identifying the Adam optimizer as the most suitable for this problem.
4. Insights into the strengths and limitations of various model architectures, which can inform future research and development efforts in the field of AI-driven medical diagnostics.

The findings from this study contribute to the growing body of knowledge in the application of deep learning for COVID-19 detection, and the developed models have the potential to assist healthcare professionals in rapid and accurate diagnosis, particularly during pandemic scenarios.

As a future scope of the project we can implement ResNet50 and ResNet101 can be implemented. Additionally, we can also use Vision Transformers to improve accuracy.

The GitHub project link: https://github.com/zankhana46/CovidCTScanClassification
In the repository you can also find an excel sheet with a high-level summary of all the models that we have implemented, and there are illustrations to better explain the findings.

6. References

https://github.com/Dedepya/A-Deep-CNN-based-Diagnostic-Model-for-detection-of-COVID-19-from-CT-Scan-Images

https://pmc.ncbi.nlm.nih.gov/articles/PMC9188200/

https://www.sciencedirect.com/science/article/pii/S0010482522011258

https://pmc.ncbi.nlm.nih.gov/articles/PMC8715284/