COVID19 classification using pretrained mobilenet

Abstract：

In this project, we explore the application of machine learning techniques for image classification of COVID-19, viral pneumonia, and normal chest X-rays. We use a dataset consisting of 137 images of COVID-19 and 317 total images including viral pneumonia and normal chest X-rays, structured into test and train directories. The goal is to develop an accurate and robust model that can distinguish between the different classes of images with high precision and recall. We use PyTorch, a popular deep learning framework, to build and train a convolutional neural network (CNN) model. We preprocess the images by resizing them to a standard size and normalizing pixel values, and split the dataset into training, validation, and testing sets. We train the model on the training set and optimize it using the validation set. Finally, we evaluate the model on the testing set and report its performance in terms of accuracy, precision, recall, and F1-score. Our results demonstrate that the CNN model can achieve high accuracy and satisfactory performance in distinguishing between COVID-19, viral pneumonia, and normal chest X-rays. This project demonstrates the potential of machine learning for automated and accurate diagnosis of COVID-19 and related respiratory diseases from medical images.

Keywords：

COVID-19, Image classification, Convolutional neural network (CNN), Machine learning

Introduction:

Deep learning has revolutionized the field of artificial intelligence, enabling machines to learn from data and make predictions or decisions based on that learning. One area where deep learning is increasingly being applied is in medical imaging, where it has shown promise for accurate and automated diagnosis of diseases such as COVID-19. In this work, we explore the use of a pre-trained MobileNet model for image classification of COVID-19, viral pneumonia, and normal chest X-rays using a dataset of medical images.

MobileNet is a family of neural network architectures designed for mobile and embedded devices. The architecture is optimized for low-latency, low-power consumption, and small memory footprint. These networks use depthwise separable convolutions, which drastically reduce the number of parameters and computation required compared to traditional convolutional neural networks. This makes MobileNet a highly efficient neural network that can run on mobile devices with limited computational resources. Moreover, MobileNet models are highly customizable and can be trained for a wide range of applications including image classification, object detection, and semantic segmentation. Due to their efficiency and versatility, MobileNets have become increasingly popular in the research community and industry for developing real-time applications on mobile devices.

MobileNetV3 is the latest iteration of this family, and it includes two variants: MobileNetV3-Large and MobileNetV3-Small. MobileNetV3-Large is designed for high accuracy tasks and is suitable for applications where accuracy is the most important consideration. This variant achieves state-of-the-art performance on various computer vision tasks, such as object detection and semantic segmentation. Compared to the previous versions of MobileNet, MobileNetV3-Large introduces several new features that improve its performance. For example, it includes a new activation function called Hard-Swish, which is faster and more accurate than previous activation functions. It also introduces a new architecture search method that can efficiently find the best network architecture for a given task. MobileNetV3-Large also uses some of the latest advancements in neural network design, such as squeeze-and-excitation modules and a

new version of depthwise convolutions. These improvements lead to better performance with fewer parameters, making MobileNetV3-Large ideal for mobile and embedded devices with limited computational resources.

Image classification is the process of assigning labels or categories to images based on their visual content. Convolutional neural networks (CNNs) are a type of deep learning model that have shown remarkable success in image classification tasks, including medical imaging. CNNs work by processing images in a hierarchical manner, extracting increasingly complex features at each layer. These features are then used to classify the image into one or more categories.

In this work, we use PyTorch, a popular deep learning framework, to build and train a MobileNetV3-large CNN model for image classification of COVID-19, viral pneumonia, and normal chest X-rays. We preprocess the images using techniques such as resizing and normalization to ensure they are suitable for input to the model. We then train the model using a dataset of medical images and evaluate its performance using metrics such as accuracy, precision, recall, and F1-score. The goal is to develop an accurate and robust model that can distinguish between the different classes of images with high precision and recall.

Datasource:

The dataset used in this work is a collection of medical images that have been gathered from publicly available sources. The dataset consists of 454 images that are structured into test and train directories, with each directory containing three classes of images: COVID-19, viral pneumonia, and normal chest X-rays. The images were released by the University of Montreal and are available on their publicly released GitHub account. The pneumonia data used in this dataset has been taken from the Radiological Society of North America (RSNA) website. Medical imaging datasets are critical resources for research and development in the field of medical diagnosis. They allow researchers and medical practitioners to study and analyze images in order to identify patterns and trends that may be indicative of a particular disease or condition. The availability of such datasets is therefore crucial for advancing medical knowledge and improving patient outcomes.

In this work, we aim to use the dataset to train a deep learning model for image classification of COVID-19, viral pneumonia, and normal chest X-rays. By accurately distinguishing between these three classes, we hope to contribute to the efforts of the medical and research community in combating the COVID-19 pandemic and related respiratory diseases. The dataset has been made available to the public in the hope of encouraging further research and development in the field of medical imaging. We hope that our work will inspire others to use and build upon this dataset, and that it will lead to the development of more accurate and effective diagnostic tools for COVID-19 and related respiratory diseases.

In the following sections, we will describe in more detail the methods used to preprocess and analyze the dataset, as well as the results obtained from training and testing the deep learning model on the dataset.

Preprocessing:

The first step in preparing the dataset for deep learning model training is preprocessing. This involves performing some basic operations on the images to standardize them and make them suitable for analysis. In this work, we perform several preprocessing steps to prepare the images for model training.

The first step in preprocessing is visualization. We visualize some of the images to gain an understanding of their characteristics and identify any features that may be relevant for classification. From the visualization, we find that there are no obvious features that can be identified using human eyes. However, we observe that all the images are 3-channels with variable height and width. This observation provides the basis for further image preprocessing.

To standardize the images, we use a set of data augmentation techniques that modify the images to create new variations of the same image. These variations can then

be used to train the deep learning model on a larger set of images, which can improve its accuracy. The data augmentation techniques used in this work include resizing, random horizontal flipping, and color jittering. The resizing operation is used to resize the images to a fixed size of 446x446. This helps to ensure that all the images are of the same size, which is necessary for training the deep learning model. The random horizontal flipping operation randomly flips the images horizontally with a probability of 0.5, which further increases the number of variations in the dataset. The color jittering operation is used to modify the color of the images. This operation randomly changes the brightness, contrast, saturation, and hue of the images to create new variations of the same image. This technique helps to increase the robustness of the model by allowing it to recognize the same image under different lighting conditions. Finally, we normalize the images to convert them to a standardized format. This involves converting the images to a normalized torch.FloatTensor and normalizing the mean and standard deviation of the dataset. The normalized dataset is then split into training and testing datasets for training and evaluation of the deep learning model.

Overall, the preprocessing steps used in this work help to standardize and normalize the images, making them suitable for training the deep learning model. By using data augmentation techniques, we are able to increase the number of images in the dataset and improve the accuracy of the model.

Model loading and setting:

Before training the model, there are several hyper-parameters to be set up. These hyper-parameters can affect the performance of the model during training and inference. In this work, the hyper-parameters are set as follows. The data is loaded into PyTorch's built-in DataLoader, which will handle the data loading during training. The train and test data are prepared using the ImageFolder function, which uses the directory address as an argument to make the dataset.

To prevent overfitting, the data is split into a training and validation set. The training set is used for training the model, while the validation set is used for evaluating the performance of the model on data it has not seen before. The data is randomly sampled using the SubsetRandomSampler function, which shuffles the indices of the data and splits them into training and validation indices. The training and validation indices are then used to create the training and validation samplers.

Next, a pretrained model is loaded for image classification. In this work, the MobileNet V3 Large model is used, which is a lightweight and efficient neural network architecture. This model is pre-trained on the ImageNet dataset, which is a large-scale image dataset that is widely used for pre-training deep neural networks. The model's output is changed to accommodate the classification task, which is to classify the images into one of three classes.

To train the model, a loss function and optimizer are specified. The loss function used in this work is the cross-entropy loss, which is commonly used for multi-class classification tasks. The optimizer used is stochastic gradient descent (SGD), which is a popular optimization algorithm used for training deep neural networks. To improve the performance of the optimizer, a learning rate scheduler is used. The learning rate scheduler decreases the learning rate of the optimizer at specific intervals during training, which can help the model converge faster and prevent overfitting.

Overall, setting the hyper-parameters and loading the model are important steps in the deep learning pipeline. These steps can significantly affect the performance of the model during training and inference. In this work, the hyper-parameters and model are chosen based on previous research and the characteristics of the dataset. By setting the hyper-parameters appropriately and loading a suitable model, the model can achieve better performance and generalize well to new data.

Model training and validation:

The goal of the training is to minimize the loss function that measures how different the model's predictions are from the true labels. The training is performed for a specified number of epochs, with each epoch consisting of a training phase and a validation

phase. During each epoch, the training phase updates the weights of the model by optimizing the loss on the training data using an optimizer. The optimizer updates the model's weights to reduce the loss function's value. The validation phase calculates the model's performance on the validation data without updating the weights of the model. The validation phase helps to determine if the model has overfit to the training data or if it generalizes well to new, unseen data.

In the training phase, the model is set to train mode, and the optimizer is cleared. The training data is fed to the model in batches, and the gradients are computed with respect to the loss for each batch. The optimizer updates the model's weights based on the gradients, and the training loss and accuracy are updated.

In the validation phase, the model is set to evaluation mode, and the validation data is fed to the model in batches. The validation loss and accuracy are computed without updating the model's weights. The output displays the training and validation loss and accuracy for each epoch during the training process of the neural network model. The training and validation loss are the error between the predicted output and the actual output, while the accuracy indicates the proportion of correct predictions out of the total number of predictions.

At the end of each epoch, the program prints out the epoch number, the training loss, validation loss, training accuracy, and validation accuracy. It also shows if the validation loss has decreased from the previous minimum value, and saves the model state if it does. In the output, we can see that the validation loss starts to decrease from epoch 1 to epoch 2, and this trend continues until epoch 7, with validation loss decreasing from 0.2291 to 0.0814. The validation loss then increases slightly in epoch 8, but it is still lower than the previous minimum. This result indicates that the neural network model is learning to generalize well, as it is able to perform well on data that it has not seen before. Moreover, we can observe that the learning rate is reduced after epoch 4 by a factor of 10, and then again after epoch 6 by another factor of 10. This stepwise decrease in learning rate is a common technique used in training neural networks to help avoid overfitting and improve convergence.

Model evaluation:

Model evaluation is an important step in the machine learning process. It involves assessing the performance of the model and determining how well it performs on unseen data. In this case, the model was evaluated using a test set of 66 images and 96.97% accuracy was achieved. The overall accuracy of the model on the test set was 97%, which indicates that the model performs well in distinguishing between the three classes. The precision, recall, and F1-score for each class show that the model performs exceptionally well in detecting Covid, with perfect precision and recall. The model also performs well in detecting Normal and Viral Pneumonia, with high precision, recall, and F1-score values. The weighted average F1-score for the model is 0.97, indicating that the model is highly accurate overall.

Conclusion:

In conclusion, we have built a Convolutional Neural Network (CNN) model for classifying chest X-ray images into three categories - Covid, Normal, and Viral Pneumonia. We used the pytorch and keras deep learning library and trained our model on a dataset of 216 images. We achieved a high accuracy of 96.97% on the test set of 66 images, which indicates the model's ability to accurately classify chest X-ray images. The model's confusion matrix and classification report show that the model performs very well for all three categories, with high precision, recall, and F1-score. The model has achieved 100% precision for Covid and Normal classes, and 91% precision for Viral Pneumonia class, indicating its ability to correctly predict the true positives without many false positives. The model appears to perform very well and can be a useful tool in identifying Covid and other respiratory conditions from chest X-ray images. Further work can be done to improve the model's performance, such as increasing the size of the dataset or fine-tuning the model's architecture.