Computer Vision

Human-Computer Interaction

Dhritabrata Mitra

Abstract

In the current era, we see applications of computer vision everywhere from Object Recognition and Classification, Facial Recognition, Medical Image Analysis to much more. We already have self-driving cars and virtual reality headsets such as Apple Vision Pro which are making their user's life easy. In the coming years, this field and its application will only increase.

Introduction

Computer vision is a technology that enables computers to understand and interpret visual information from the world. It mimics human vision by using algorithms and models to analyze images or videos, extracting valuable insights, and making decisions. This field has applications in various domains, such as healthcare, automotive, and security. The primary goal is to teach computers to see and comprehend visual data, empowering them to recognize objects, track movement, and even understand the emotions on a person's face. With advancements in machine learning and deep learning, computer vision continues to evolve, enhancing its accuracy and expanding its range of applications, making it an integral part of modern technology.

In this paper, we are going to see how computer vision can be helpful in the medical field to automate the diagnosis and classification of various diseases. We are finding how to differentiate a person having normal lung and a person suffering from Pneumonia.

Advantages

1) It makes things easier and faster: Computer Vision helps clients and industries quickly check and access their products by using fast computers.

2) It's reliable: Unlike humans, computers and cameras don't get tired, so they work just as well all the time. They don't depend on things like feeling sick or being upset.

3) It's accurate: Computer Imaging and Computer Vision are very precise, ensuring that the final product is accurate.

4) It can be used in many areas: The same computer system can be used in different fields and activities. For example, it helps in factories with tracking supplies and in the medical industry with scanned images.

5) It reduces costs: Using Computer Imaging saves time and reduces errors, cutting down on the need to hire and train special staff. This means computers can do tasks that would normally need lots of workers, saving money.

Disadvantages

1) Need for experts: There is a big need for specialists in Machine Learning and Artificial Intelligence. These professionals understand how devices like Computer Vision work. They can also fix them when there's a problem.

2) Issues with depending on machines: While it's good to remove human errors in some cases, machines can't warn us when they're about to fail.

3) Problems with image processing: If a device fails because of a virus or software issue, Computer Vision and image processing might not work well. If we don't fix the problem, the device's functions can disappear. This could cause issues, like freezing production in warehouses.

To work with images, we will use Convolutional Neural Network. A convolutional neural network (CNN) is a special kind of neural network that learns to find important parts of data. It does this by using filters (or kernels). This helps avoid problems like vanishing gradients and exploding gradients, which were issues in older neural networks. Regularized weights are used over fewer connections to prevent these problems. For example, in a regular neural

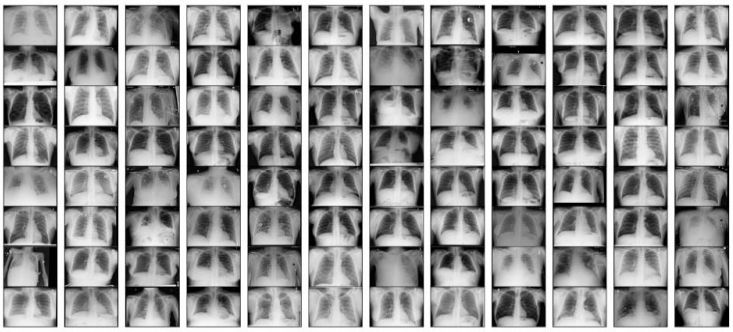
network, each tiny part of an image might need a lot of weight to process it. But with CNN, we can use fewer neurons to process smaller parts of the image, making it more efficient. This way, the network can understand bigger parts of the picture through different layers.

Even though CNNs are cool and work well for smaller images, it used to be expensive to use them on big, high-quality pictures. But now, with better computer parts like GPUs and smarter ways of doing the math (2D convolution), we can train big CNNs without costing too much. Also, there are lots of labeled examples in recent datasets like ImageNet, so we can train these big models without them getting too good at just the specific examples we used.

Dataset

This is a big dataset from the National Institutes of Health (NIH), and you can find it on Kaggle. It contains over 112,000 chest X-Ray pictures from more than 30,000 different patients. It's the biggest chest X-Ray dataset that is free to use by anyone in the world. The pictures are really accurate, more than 90% of the time, and most of them are good for computer programs that are learning about X-Rays without a lot of detailed guidance.

Each picture comes with helpful information written by computers (Natural Language Processing) so that the labels match up with the kinds of diseases mentioned in the X-Ray reports. This dataset is super helpful for people working on computer programs that are learning or understanding chest X-Ray images. If you're into machine learning or computer vision and you need X-Ray pictures of chests, this dataset is a goldmine.



Data Preprocessing

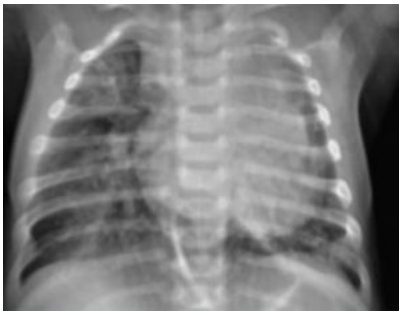
We often encounter messy and incomplete data in various forms like text, images, or videos. Data preprocessing is a crucial step where we clean and organize this raw information so that computers and machine learning systems can easily understand and analyze it. Machines prefer structured data represented as 1s and 0s, making calculations straightforward. However, real-world data is usually irregular and lacks uniform design, containing errors and inconsistencies. Therefore, before analysis, unstructured data such as text and images need to be cleaned and formatted to ensure accurate processing.

Using a data generator to load and preprocess images. We will normalize pixel values and set up data augmentation if needed. Split the training set into training and validation subsets so that we can check model performance.

The dataset employed in this study was partitioned into training and testing sets to ensure a comprehensive evaluation of the model's performance. The division was conducted using the train_test_split function from the scikit-learn library, which randomizes the allocation of samples while maintaining the overall distribution of classes. A commonly adopted split ratio of 80:20 was applied, allocating 80% of the data for model training and the remaining 20% for performance evaluation. This rigorous division facilitates the assessment of the model's ability to generalize to unseen data, a crucial aspect of ensuring the robustness and reliability of the proposed methodology.

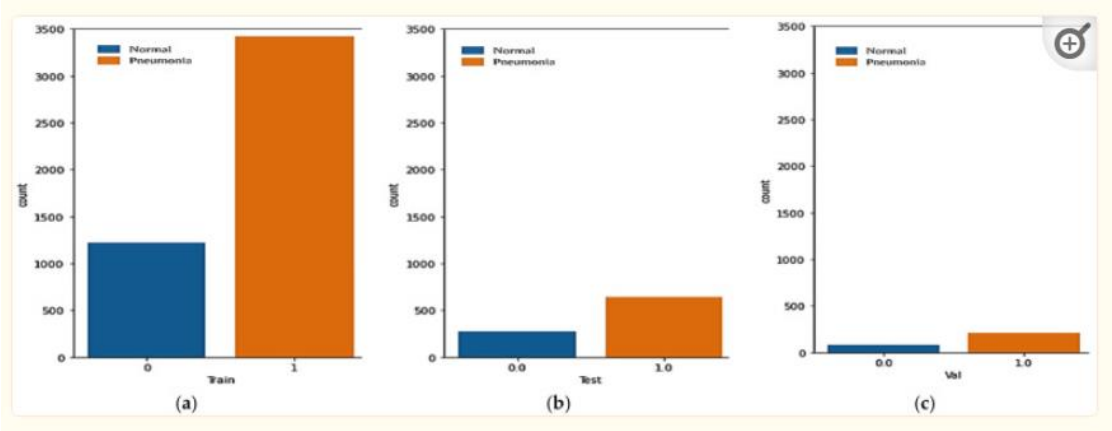
Difference between someone with normal lung and a person with Pneumonia

Normal



Pneumonia

Count plot depicting count for the pneumonia and normal dataset: (a) training set (b) testing set (c) validation set.



Data Splitting

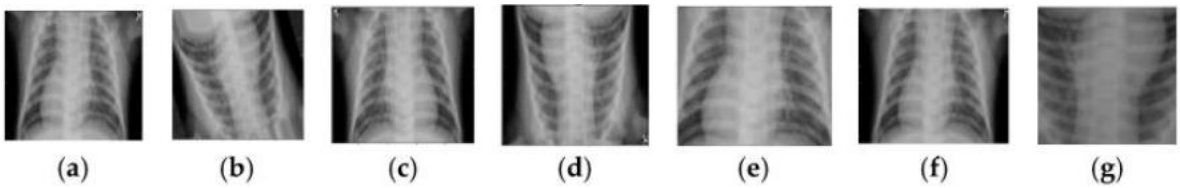
The details of data splitting on the training, validation, and testing classes of data are shown in the figure

Class	First Dataset		
	Train	Validation	Test
Pneumonia	3418	214	641
Normal	1224	81	278
Total	4642	295	919
	Second Dataset		
Pneumonia	1145	71	215
Normal	1145	71	215
Total	2290	142	430

Data Augmentation

CNN models require an extensive set of data sources to achieve optimal training and showcase enhanced performance, particularly on larger datasets. In cases where only a limited dataset is available, an approach is employed to artificially expand the dataset, thus helping to prevent overfitting. This technique, known as data augmentation, is commonly utilized. It involves

increasing the number of images by implementing various alterations while preserving the original class labels.



Model

All the models listed in figure are going to be used and the model providing the best accuracy will be selected.

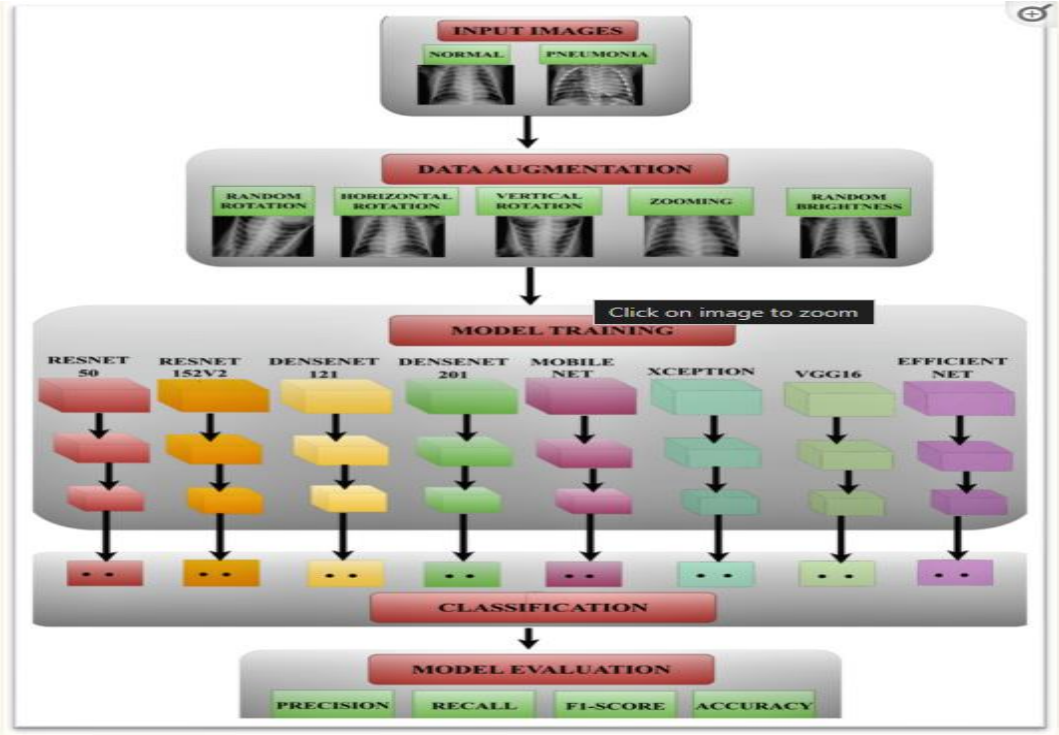
Descriptions of the pre-trained CNN models used

Model	Layers	Parameters (in Millions)	Input Layer Size	Output Layer Size
MobileNet	28	13	224 × 224 × 3	(2,1)
ResNet50	50	25.6		
ResNet152V2	164	60.4		
DenseNet201	201	20.2		
DenseNet121	121	8.1		
Xception	71	22.8		
VGG16	16	138		
EfficientNet	10	8.4		

Training

The picture shows how a computer system can help diagnose pneumonia without human intervention. The goal of this system is to sort chest X-ray images into two groups: normal and pneumonia. To make the computer better at learning, we use the regular X-ray pictures and add more variations to them. We also use models that have been trained before, along with these changed

images, to decide if a person has pneumonia. The next parts explain each step in more detail.



Testing and Validation

In the evaluation phase, the model underwent rigorous testing on an independent set, distinct from the training data. The objective was to assess the generalization capability of the proposed model to unseen instances. The test set, comprising [insert number] samples, was meticulously curated to encompass diverse cases representative of the real-world scenario. To measure the model's performance, standard metrics such as accuracy, precision, recall, and F1 score were employed. Additionally, confusion matrices were generated to gain insights into the model's classification behavior, including any tendencies for false positives or false negatives.

This meticulous evaluation process aims to provide a comprehensive understanding of the model's strengths and limitations, contributing to the robustness and applicability of the proposed methodology.

The performance of MobileNet is assessed across various optimizers by examining training and validation accuracies. Figure a illustrates the training and validation accuracy for the ADAM optimizer, Figure b depicts the same for ADADELTA, and Figure c shows the results for SGD.

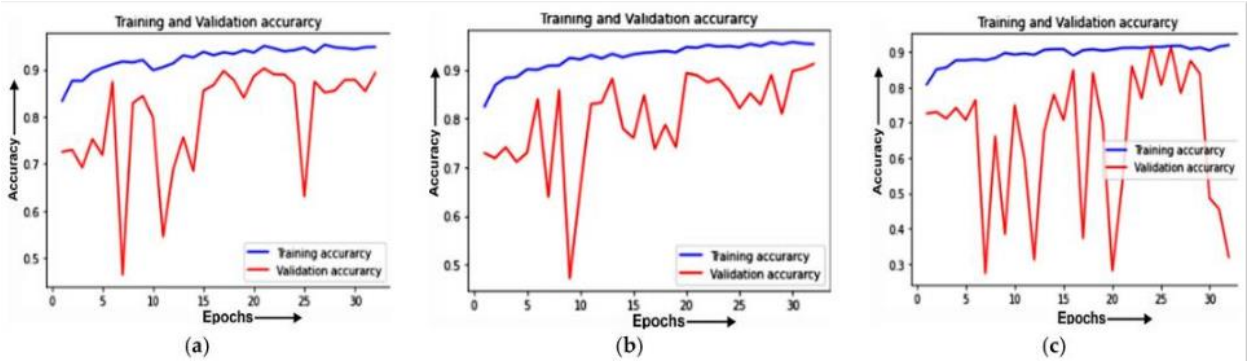
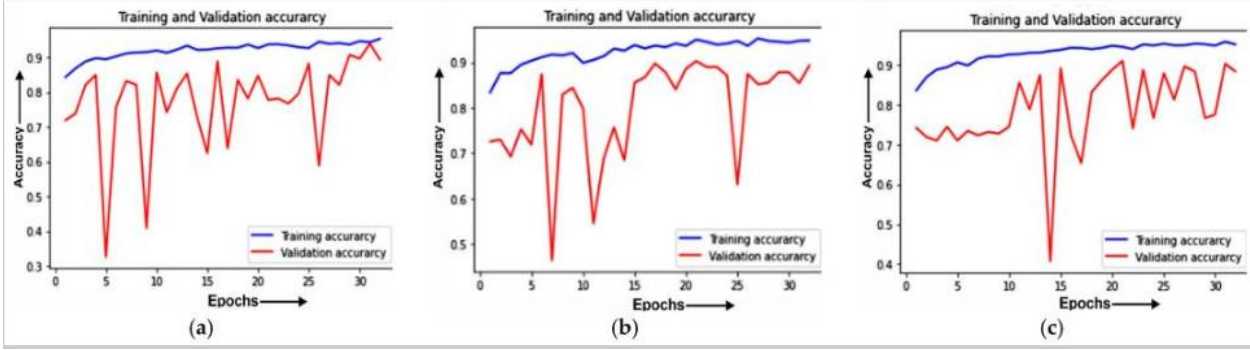


Figure a displays the training and validation accuracy using the ADAM optimizer. The figure indicates that the training accuracy remains constant between epochs 26 and 30, with both training and validation accuracies consistently above 90% and approximately 88%, respectively.

In Figure b, the training and validation accuracy for the ADADELTA optimizer are depicted. The figure illustrates an increasing accuracy trend from epoch 21 to 24. Validation accuracy experiences a sudden surge from 10% to 88%, followed by a stable increment. Training accuracy consistently stays above 90%, remaining constant from epoch 15 to 20.

Figure c showcases the training and validation accuracy with the SGD optimizer. The figure reveals a continuous increase in validation accuracy from epoch 5 to 30. Notably, both training and validation accuracy lines converge at epoch values 24 and 26, reaching 85%, indicating a consistent and steady improvement in accuracy.

Training and validation accuracy of MobileNet model for: (a) 16 batch size; (b) 32 batch size; (c) 64 batch size



The training and validation accuracies and losses of MobileNet were computed for different batch sizes. Figure a displays the training and validation accuracy for a 16-bit batch size, Figure b for a 32-bit batch size, and Figure c for a 64-bit batch size.

In Figure a, the training accuracy remains constant between epoch values 14 and 18. Meanwhile, the validation accuracy consistently increases from 35% to 85% across epoch values 4–6 and 7–11. Both training and validation accuracies are above 90% and approximately 88%, respectively. The lines intersect at a common point, reaching 82% accuracy at epoch value 31 after a consistent increase.

For a batch size of 32 (Figure b), the validation accuracy progressively increases from epoch value 11 to 24. The training accuracy rises by 5 points, from 90% to 95%, between epoch values 10 and 21.

Figure c illustrates the training and validation accuracy values for a batch size of 64. The training accuracy steadily increases from epoch value 1 to 30, with the validation accuracy exceeding 90%. This indicates the highest accuracy compared to the training and validation accuracies for other batch sizes.

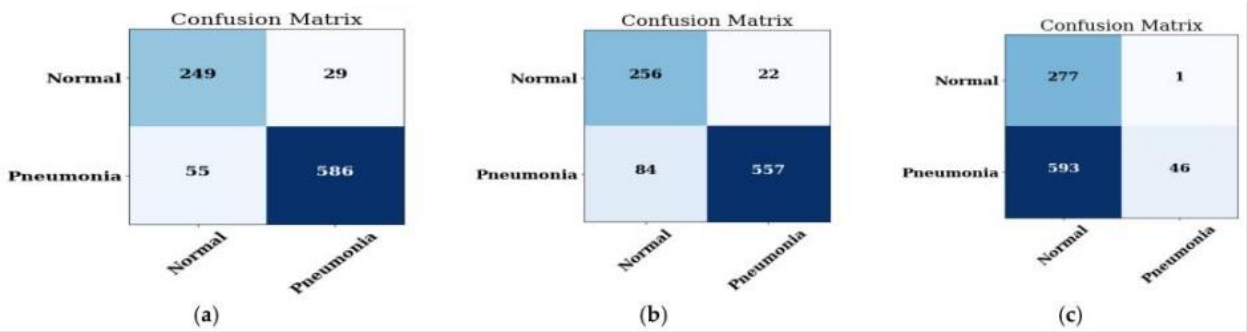
Confusion Matrix

The MobileNet model's prediction performance was assessed using three different optimizers: ADAM, ADADELTA, and SGD. The confusion matrices for these optimizers are depicted in Figures a, b, and c.

Upon analyzing the confusion matrices for normal and pneumonia image datasets, it becomes evident that the MobileNet CNN model achieves the highest accuracy when utilizing the ADAM optimizer, surpassing both ADADELTA and SGD.

Furthermore, an evaluation of accuracy, precision, recall, F1-score, and AUC score for the MobileNet model was conducted across all three optimizers: ADAM, ADADELTA, and SGD. This comprehensive assessment allows for a thorough understanding of the performance of each optimizer.

The confusion matrix of the MobileNet model on: the (a) ADAM optimizer; (b) ADADELTA optimizer; (c) SGD optimizer.



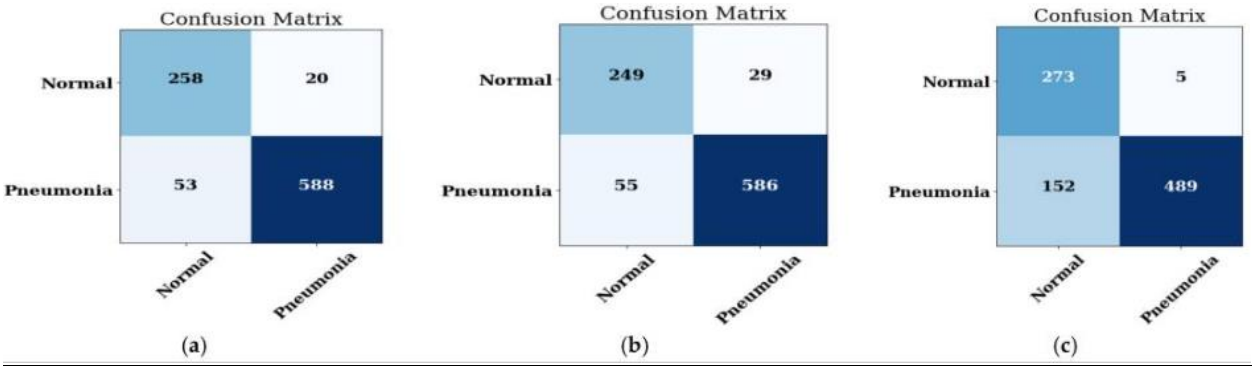
Comparison of eight CNN models in terms of confusion matrix parameters

Optimizer	Accuracy (%)	Precision (%)	Recall (%)	F1-Score	AUC
ADAM	90.85	95.28	91.41	91.41	0.933
ADADELTA	88.46	96.20	86.89	91.31	0.971
SGD	35.14	97.87	7.176	13.37	0.867

The prediction models are assessed based on pneumonia and non-pneumonia datasets, and the results are presented through confusion matrices. Figure a illustrates the matrix for MobileNet with a 16-bit batch size, Figure b for a 32-bit batch size, and Figure c for a 64-bit batch size.

Upon examination of the confusion matrices, it is observed that when predicting using normal and pneumonia image datasets, the MobileNet CNN model exhibits superior performance with a 16 batch size compared to the other two batch sizes—namely, 32 batch size and 64 batch size.

The accuracy, precision, recall, F1-score, and AUC score of the MobileNet model are then represented for the three batch sizes (16, 32, and 64), and the performance for each batch size is computed.



The confusion matrix of the MobileNet model on: (a) 16 batch size; (b) 32 batch size; (c) 64 batch size.

Batch Size	Accuracy (%)	Precision (%)	Recall (%)	F1-Score	AUC
16	92.05	96.71	91.73	94.15	0.980
32	90.85	95.28	91.41	93.31	0.970
64	82.91	98.98	76.28	86.16	0.971

Evaluation of MobileNet model for different batch sizes on different parameters.

Accuracy

Accuracy is a way to measure how well a model can correctly identify things in a set of data. It's calculated by dividing the number of correct identifications by the total number of things to identify.

Accuracy = (Number of correctly classified samples) / (Total number of samples)

If there are 1000 things to identify and the model gets 900 of them right, the accuracy is 90%. This means the model is correct about 90% of the time.

Accuracy = 900/1000 = 0.9 or 90%

Simply put, our model is right about 90% of the time when it tries to sort or categorize the data.

Precision

Precision is defined as the ratio of true positive samples to all the predicted positive samples. Mathematically, precision can be expressed as:

Precision = (True positives) / (True positives + False positives)

where True positives are the number of correctly classified positive samples and False positives are the number of negative samples that were incorrectly classified as positive.

If our model has predicted 100 samples as positive out of which 80 are positive and 20 are negative, then the precision of our model would be:

Precision = 80/(80+20) = 0.8 or 80%

This means that out of all the samples our model predicted as positive, it was able to correctly classify 80% of them as truly positive.

Precision is an important metric when we want to avoid false positives.

Recall

Recall is a way to see how good a model is at finding all the positive things in a group. It's like checking how many correct positive items the model found out of all the positive items there are. To calculate recall, you can use this formula:

$$\text{Recall} = (\text{True positives}) / (\text{True positives} + \text{False negatives})$$

For example, if a model correctly found 80 out of 100 positive items and missed 20 positive items, the recall would be:

$$\text{Recall} = 80/(80+20) = 0.8 \text{ or } 80\%$$

This means the model found 80% of all the positive items correctly.

F1-Score

F1-score is a number that shows how well a model is doing in terms of both precision and recall. It's like an average that balances these two measures. To calculate F1-score, you use this formula:

$$\text{F1-score} = 2 * ((\text{Precision} * \text{Recall}) / (\text{Precision} + \text{Recall}))$$

Precision is about how many of the predicted positive samples are actually positive, and recall is about how many of the actual positive samples are correctly predicted. If a model has a precision of 80% and recall of 85%, the F1-score would be 82%.

$$\text{F1-score} = 2((0.80.85)/(0.8+0.85)) = 0.82 \text{ or } 82\%$$

This means the model's overall performance is 82%, considering both precision and recall.

The MobileNet model's accuracy, precision, recall, F1-score, and AUC score were evaluated across three epochs: 16, 32, and 64. The performance metrics for each epoch were calculated and are presented graphically.

Epochs	Accuracy (%)	Precision (%)	Recall (%)	F1-Score	AUC
16	89.22	94.71	89.54	92.06	0.955
32	92.05	96.71	91.73	94.15	0.980
64	94.23	93.75	98.28	95.96	0.972

Evaluation of MobileNet models on different epochs on different parameters

Related Work

1) Very deep convolutional networks for large-scale image recognition - https://arxiv.org/pdf/1409.1556.pdf

2) ImageNet Classification with Deep Convolutional Neural Networks – https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

Conclusion

In summary, computer vision is important because it helps machines see and understand images, like how people do. Thanks to progress in computer vision, we've seen big improvements in areas like recognizing images, finding objects, and even in things like self-driving cars and medical image analysis. Looking ahead, more research and development in computer vision will likely make it even better, leading to new and helpful solutions for different industries and making our daily lives better.