INTERNATIONAL JOURNAL OF PROGRESSIVE RESEARCH IN ENGINEERING MANAGEMENT AND SCIENCE (IJPREMS)

e-ISSN : 2583-1062

Impact Factor: 5.725

www.ijprems.com
editor@ijprems.com

Vol. 04, Issue 05, May 2024, pp: 1645-1651

CLASSIFICATION OF EYE-DISEASE THROUGH IMAGE PROCESSING AND DEEP LEARNING

Mrs. Ch. Srivatsa[1], Bolle Sharath[2], Gajula Tharun Kumar[3], Sanamandra Joel John[4], Chittumala Nipun Raj[5]

[1]Assistant. Professor Cse Dept Ace Engineering College Hyderabad, India.

[2,3,4,5]Cse Ace Engineering College Hyderabad, India.

DOI: https://www.doi.org/10.58257/IJPREMS34426

ABSTRACT

"Classification of Eye-Disease through Image processing and Deep Learning" is to develop a machine through deep learning algorithms to classify the eye diseases. Now-a-days everyone is suffering from different eye diseases which will lead to blindness. If we find the eye disease in the early stages we can easily cure the disease. Every eye disease will have different images if we can find the correct disease of our eye we can detect and we can cure the disease. We are using the CNN (Convolutional Neural Network) to classify the images. This CNN architecture will help to classify the large data set images.

1. INTRODUCTION

Introducing "Classification of Eye-Disease through image processing and deep learning." Eye diseases represent a significant public health concern worldwide which is effecting the millions of people.To address these challenges, there is a growing interest in leveraging advancements in image processing and deep learning techniques to develop automated systems. In this paper, we present a novel approach that combines image processing methods with deep learning architectures for the classification of eye disease images. We are using the preprocessed eye disease data set to classify the eye disease. The data set contains the different eye diseases like Acrima,Glaucoma,normal eye, cataract,retina disease. By leveraging a diverse dataset comprising images of various eye conditions, our proposed framework aims to achieve high accuracy in disease classification while minimizing false positives and false negatives.

2. OBJECTIVES

Develop a comprehensive understanding of the various types of eye diseases and their manifestations in retinal images.

Compare the performance of the proposed approach with existing methods and benchmarks, highlighting its advantages in terms of accuracy, computational efficiency, and generalization capability.

Providing the highest accuracy rate and the least error rate of the classification.

3. PROBLEM STATEMENT

Despite significant advancements in medical imaging technology, the diagnosis of eye diseases remains a challenging task, often relying heavily on manual examination by trained ophthalmologists. This manual process is not only time-consuming but also prone to subjectivity and human error, leading to delays in diagnosis and potentially affecting patient outcomes. Additionally, the increasing prevalence of eye diseases worldwide further exacerbates the demand for efficient and accurate diagnostic tools. In recent years, advances in image processing techniques and deep learning algorithms have shown promising results in medical image analysis, including the detection and classification of eye diseases. By leveraging these technologies, it is possible to develop automated systems capable of analyzing retinal images and accurately identifying different eye conditions, such as diabetic retinopathy, glaucoma, cataracts.

4. PROPOSED SYSTEM

We are proposing increasing the output accuracy by using the deep CNN architectures to classify the eye diseases. The classification of eye disease will be so much helpful for the doctors to cure the eye disease. We are using the kaggle data set which contains the different types of eye diseases. By taking that data set and combing with the deep CNN architecture we can get maximum accuracy of the classification.

5. HARDWARE AND SOFTWARE REQUIREMENTS

HARDWARE REQUIREMENTS:

- Processor – Intel i3 9th gen
- RAM – 4 GB (min)
- Hard Disk – 50 GB

INTERNATIONAL JOURNAL OF PROGRESSIVE RESEARCH IN ENGINEERING MANAGEMENT AND SCIENCE (IJPREMS)

e-ISSN : 2583-1062

Impact Factor: 5.725

www.ijprems.com
editor@ijprems.com

Vol. 04, Issue 05, May 2024, pp: 1645-1651

SOFTWARE REQUIREMENTS:

- Operating system – Windows 10, 11, mac os
- Coding Language – Python

6. TECHNOLOGY DESCRIPTION

Python

Python is a high-level, interpreted programming language known for its simplicity and versatility. Python emphasizes readability and ease of use, making it an excellent choice for beginners and experienced programmers alike. It features a clear and concise syntax, with significant whitespace used for code indentation, enhancing readability. Python supports multiple programming paradigms, including procedural, object-oriented, and functional programming, providing flexibility for various development tasks. It boasts a vast ecosystem of libraries and frameworks for diverse applications, such as web development, data analysis, machine learning, automation, and scientific computing. Python's popularity continues to grow, driven by its user-friendly nature, extensive community support, and wide adoption across industries.

7. PACKAGES USED

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Matplotlib is mainly used for plotting the plots, including line plots, scatter plots, bar charts, histograms, and more.

PyTorch is an open-source machine learning library primarily developed by Facebook's AI Research lab (FAIR). It provides a flexible and dynamic computational graph structure, making it suitable for deep learning and neural network research. PyTorch uses dynamic computation graphs, allowing for more flexible and intuitive model development compared to static graph frameworks.

Torchvision is a package built on top of PyTorch, specifically designed for computer vision tasks.

8. ALGORITHM

Loading Data: Loading the data set by providing the path of the data set.

Train-Test Split: Dividing the data set into two parts one is training data set and test data set

CNN Model: This CNN model will classify the images into their categories.

Confusion Matrix: we visualize the confusion matrix to understand the model's performance across different classes.

Metric Calculation: Finally, we calculate and print the accuracy, precision, recall, and F1 score.

Coding:

```
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sn
import torch
import torch.nn as nn
import torchvision
from torch.utils.data import DataLoader
from torchvision import datasets, transforms
from torchvision.utils import make_grid
from torchmetrics import Accuracy, ConfusionMatrix, Precision, Recall, F1Score
from tqdm import tqdm
def load_data():
t = transforms.Compose(
[
transforms.ToTensor(),
transforms.Resize((256, 256)),
]
)
return datasets.ImageFolder(root="C:\DATASET_101", transform=t)
dataset = load_data()
dataset
```

INTERNATIONAL JOURNAL OF PROGRESSIVE RESEARCH IN ENGINEERING MANAGEMENT AND SCIENCE (IJPREMS)

e-ISSN : 2583-1062

Impact Factor: 5.725

www.ijprems.com
editor@ijprems.com

Vol. 04, Issue 05, May 2024, pp: 1645-1651

```
# number of classes
NUMBER_OF_CLASSES = len(set(dataset.targets))
print(f"Number of classes: {NUMBER_OF_CLASSES}")
def display_image(image, label):
print(f"Label : {dataset.classes[label]}")
plt.imshow(image.permute(1, 2, 0))
# display the first image in the dataset
display_image(*dataset[0])
def train_test_split(dataset, train_size, random_state=42):
train_size = int(train_size * len(dataset))
test_size = len(dataset) - train_size
seed = torch.Generator().manual_seed(random_state)
train_dataset, test_dataset = torch.utils.data.random_split(
dataset, [train_size, test_size], generator=seed
)
return train_dataset, test_dataset
train_dataset, test_dataset = train_test_split(dataset, 0.8)
batch_size = 32
train_dataloader = DataLoader(
train_dataset, batch_size=batch_size, shuffle=True, num_workers=4
)
test_dataloader = DataLoader(
test_dataset, batch_size=batch_size, shuffle=False, num_workers=4
)
def show_batch(data_loader):
"""Plot images grid of single batch"""
for images, labels in data_loader:
fig, ax = plt.subplots(figsize=(16, 12))
ax.set_xticks([])
ax.set_yticks([])
ax.imshow(make_grid(images, nrow=16).permute(1, 2, 0))
break
show_batch(train_dataloader)
class CNN(nn.Module):
def __init__(self, NUMBER_OF_CLASSES):
super(CNN, self).__init__()
self.conv_layers = nn.Sequential(
nn.Conv2d(in_channels=3, out_channels=16, kernel_size=3, stride=2),
nn.BatchNorm2d(16),
nn.LeakyReLU(),
nn.MaxPool2d(kernel_size=2, stride=2),
nn.Conv2d(in_channels=16, out_channels=32,
kernel_size=3, stride=2),
nn.BatchNorm2d(32),
nn.LeakyReLU(),
nn.MaxPool2d(kernel_size=2, stride=2),
nn.Conv2d(in_channels=32, out_channels=64,
```

```
kernel_size=3, stride=2),
nn.BatchNorm2d(64),
nn.LeakyReLU(),
nn.MaxPool2d(kernel_size=2, stride=2),
)
self.dense_layers = nn.Sequential(
nn.Dropout(0.2),
nn.Linear(64 * 3 * 3, 128),
nn.ReLU(),
nn.Dropout(0.2),
nn.Linear(128, NUMBER_OF_CLASSES),
)
def forward(self, x):
x = self.conv_layers(x)
x = x.view(x.size(0), -1)
x = self.dense_layers(x)
return x
model = CNN(NUMBER_OF_CLASSES)
criterion = nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(model.parameters())
device = "cpu"
if torch.cuda.is_available():
device = "cuda:0"
elif torch.backends.mps.is_available():
device = "mps"
# A function to encapsulate the training loop
def batch_gd(model, criterion, optimizer, train_loader, test_loader, epochs):
model.to(device)
train_losses = np.zeros(epochs)
test_losses = np.zeros(epochs)
accuracy = Accuracy(task="multiclass",
num_classes=6).to(device)
for epoch in range(epochs):
train_loss = []
for inputs, targets in tqdm(train_loader, desc=f'Training... Epoch: {epoch + 1}/{epochs}'):
# move data to GPU
inputs, targets = inputs.to(device), targets.to(device)
# zero the parameter gradients
optimizer.zero_grad()
# Forward pass
outputs = model(inputs)
loss = criterion(outputs, targets)
# Backward and optimize
loss.backward()
optimizer.step()
train_loss.append(loss.item())
# Get train loss
```

INTERNATIONAL JOURNAL OF PROGRESSIVE RESEARCH IN ENGINEERING MANAGEMENT AND SCIENCE (IJPREMS)

e-ISSN : 2583-1062

Impact Factor: 5.725

www.ijprems.com
editor@ijprems.com

Vol. 04, Issue 05, May 2024, pp: 1645-1651

```
train_loss = np.mean(train_loss)
# Train accuracy
train_accuracy = accuracy(outputs, targets)
test_loss = []
for inputs, targets in tqdm(test_loader, desc=f'Validating... Epoch: {epoch + 1}/{epochs}'):
inputs, targets = inputs.to(device), targets.to(device)
outputs = model(inputs)
loss = criterion(outputs, targets)
test_loss.append(loss.item())
# Get test loss
test_loss = np.mean(test_loss)
# Test accuracy
test_accuracy = accuracy(outputs, targets)
# Save losses
train_losses[epoch] = train_loss
test_losses[epoch] = test_loss
print(f"Epoch {epoch+1}/{epochs}:")
print(
f"Train Loss: {train_loss:.2f}, Train Accuracy: {train_accuracy:.2f}")
print(
f"Test Loss: {test_loss:.2f}, Test Accuracy: {test_accuracy:.2f}")
print('-'*30)
return train_losses, test_losses
train_losses, test_losses = batch_gd(
model, criterion, optimizer, train_dataloader, test_dataloader, epochs=10
)
plt.title("Losess")
plt.plot(train_losses, label="Train loss")
plt.plot(test_losses, label="Test loss")
plt.xlabel("Epoch")
plt.ylabel("Loss")
plt.legend()
plt.show()
y_pred_list = []
y_true_list = []
with torch.no_grad():
for inputs, targets in test_dataloader:
inputs, targets = inputs.to(device), targets.to(device)
outputs = model(inputs)
predections = torch.max(outputs, 1)
y_pred_list.append(targets.cpu().numpy())
y_true_list.append(predections.cpu().numpy())
targets = torch.tensor(np.concatenate(y_true_list))
preds = torch.tensor(np.concatenate(y_pred_list))
confmat = ConfusionMatrix(task="multiclass", num_classes=NUMBER_OF_CLASSES)
cm = confmat(preds, targets)
sn.heatmap(cm, annot=True, fmt=".0f")
```

INTERNATIONAL JOURNAL OF PROGRESSIVE RESEARCH IN ENGINEERING MANAGEMENT AND SCIENCE (IJPREMS)

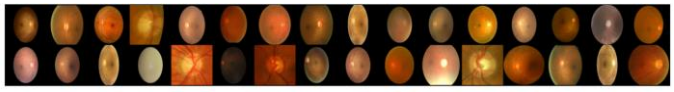
e-ISSN : 2583-1062

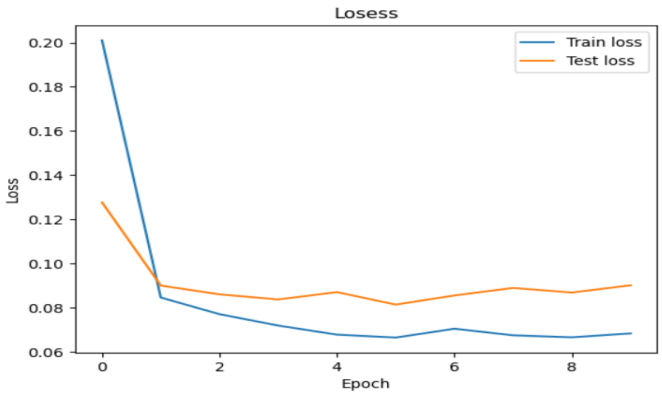
Impact Factor: 5.725

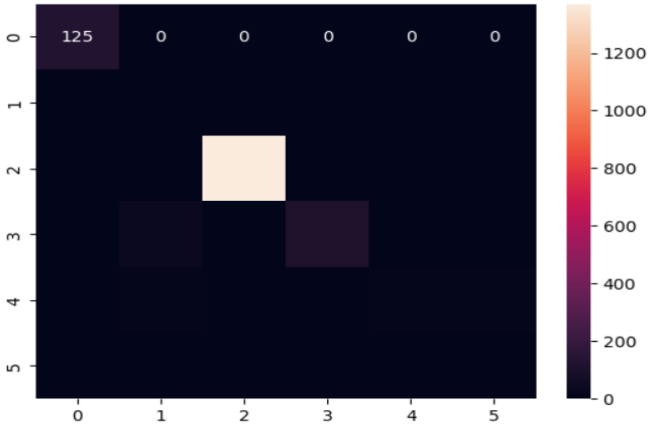
www.ijprems.com
editor@ijprems.com

Vol. 04, Issue 05, May 2024, pp: 1645-1651

```
plt.show()
accuracy = Accuracy(task="multiclass", num_classes=NUMBER_OF_CLASSES).to(device)
accuracy = accuracy(preds, targets)
print(f"Accuracy: {100 * accuracy:.2f}%")
precision = Precision(task="multiclass", average='micro', num_classes=NUMBER_OF_CLASSES)
precision = precision(preds, targets)
print(f"Precision: {100 * precision:.2f}%")
recall = Recall(task="multiclass", average='micro', num_classes=NUMBER_OF_CLASSES)
recall = recall(preds, targets)
print(f"Recall: {100 * recall:.2f}%")
f1 = F1Score(task="multiclass", num_classes=NUMBER_OF_CLASSES)
f1 = f1(preds, targets)
print(f"F1 Score: {100 * f1:.2f}%")
```







INTERNATIONAL JOURNAL OF PROGRESSIVE RESEARCH IN ENGINEERING MANAGEMENT AND SCIENCE (IJPREMS)

e-ISSN : 2583-1062

www.ijprems.com
editor@ijprems.com

Vol. 04, Issue 05, May 2024, pp: 1645-1651

Impact Factor: 5.725

9. TESTING

Collect a diverse and representative dataset of eye images containing various types and stages of diseases.

Preprocess the images to standardize their size, format, and quality. Common preprocessing techniques include resizing, normalization, and augmentation to enhance the model's generalization ability. Split the dataset into training, validation, and testing sets. The training set is used to train the model, the validation set is used to tune hyperparameters and monitor performance during training, and the testing set is used to evaluate the final performance of the trained model.

Regression Testing: Regression testing ensures that recent code changes or modifications do not adversely affect existing functionalities of the system. In the context of deep learning, regression testing involves re-running tests on the entire system after making changes to the model architecture, training procedure, or preprocessing techniques to verify that previously working functionalities still perform as expected.

Acceptance Testing: Acceptance testing involves testing the system's compliance with specified requirements and user expectations. In the case of an eye disease classification system, acceptance testing may involve presenting the system to domain experts, such as ophthalmologists, to evaluate its performance against real-world scenarios and validate its effectiveness in diagnosing eye diseases accurately.

Performance Testing: Performance testing evaluates the system's responsiveness, scalability, and stability under different conditions and workloads. For an eye disease classification system, performance testing may involve measuring the inference speed of the model, memory consumption, and resource utilization to ensure the system can handle a large number of images efficiently.

10. CONCLUSION

In conclusion, "Classification of eye disease through image processing and deep learning" will classify the data set containing the different eye diseases which will help the doctors to detect the eye disease at the early stages. By detecting eye disease at the early stage we can cure the patient from the vision loss. We achieved this by using the deep CNN architecture. Deep learning models exhibit scalability and generalization capabilities, enabling them to perform effectively across diverse populations and datasets. With proper training and validation, these models can adapt to variations in image quality, resolution, and patient demographics, making them versatile tools for diagnosing eye diseases in different clinical settings worldwide. By this paper we are concluding that we are getting the output accuracy of 96.14%.

11. REFERENCES

[1] https://ieeexplore.ieee.org/document/9402347
[2] https://ieeexplore.ieee.org/document/10242558
[3] https://ieeexplore.ieee.org/document/10125593
[4] https://ieeexplore.ieee.org/document/10417352
[5] https://ieeexplore.ieee.org/document/9783206