Image classification for clot prediction

A MINI PROJECT REPORT

18CSC305J - ARTIFICIAL INTELLIGENCE

Submitted by

KOUNDINYA BADIKELA(RA2011027010139)
SHASHI AKSHITH(RA2011027010172)
DHANUSH S(RA2011027010153)

Under the guidance of

PREMALATHA G

Assistant Professor, Department of Computer Science and Engineering

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

of

FACULTY OF ENGINEERING AND TECHNOLOGY

S.R.M. Nagar, Kattankulathur, Chengalpattu District

MAY 2023

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that Mini project report titled **"Image classification for clot prediction"** is the bonafide work of **KOUNDINYA BADIKELA(RA2011027010139),SHASHI AKSHITH(RA2011027010172),DHANUSH S(RA2011027010153)**who carried out the minor project under my supervision. Certified further, that to the best of my knowledge, the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Dr.PREMALATHA G
Assistant Professor
Department of Data Science & Business Systems

SIGNATURE

Dr. M. Lakshmi
HEAD OF THE DEPARTMENT
Professor & Head
Department of Data Science & Business Systems

ABSTRACT

Stroke is the third leading cause of death following diseases of heart and cancer, therefore this paper on Image Classification of Stroke Blood Clot Origin makes use of Deep Learning techniques to classify the blood clot origins in ischemic stroke using slide digital pathology images. This model will enable healthcare providers to better identify the origins of blood clots in deadly strokes, making it easier to prescribe post-stroke therapeutic management and reducing the risk of a second stroke. With the advancements in computer vision and deep learning techniques, various methods have been proposed for automating this task, ranging from classical machine learning techniques like K-Means, Random Forest to Deep Learning techniques like Convolutional Neural Networks (CNN) and Deep Neural Networks (DNN). The choice of method depends on various aspects like the dataset and quality of images. The main aim of this paper is to apply Deep Learning algorithm namely Convolutional Neural Networks (CNN) using Timm models where Timm is a Pytorch library which contains pretrained models like DenseNet121 and ResNet50d. In recent times Deep Learning models using CNN have shown promising results in the healthcare sector as the images have fewer spatial constraints and quick shooting time. Despite these advances, there are still significant challenges in this field including the variability of clot shape, limited availability of annotated data. To address these challenges ongoing research is focusing on improving the performance of the model by taking Dense Net models into consideration which have densely connected hidden layers which provides accurate results. The performance of the algorithm is evaluated by based on quality metric known as weighted multi-class logarithmic loss. This dataset contains 60090 images and the experimental result analysis based on the quality metrics and the graphical representation proves that the algorithm (CNN) gives good classification accuracy of 63% for all the tested datasets.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Stroke is a leading cause of death and disability worldwide, affecting millions of people every year. One of the most common types of strokes is ischemic stroke, which is caused by a blood clot blocking a blood vessel in the brain. Early detection and treatment of ischemic stroke is critical for reducing the risk of disability and death. In recent years, deep learning has emerged as a powerful tool for medical image analysis, particularly for the classification of medical images. Deep learning models have been shown to be effective in various medical applications, including the detection of cancer, identification of bone fractures, and prediction of disease outcomes. In this paper, we propose a deep learning approach for the classification of blood clots in ischemic stroke patients. The goal of this approach is to predict the risk of stroke based on the type of clot present in the patient's brain. We begin by describing our dataset, which consists of MRI scans of ischemic stroke patients with confirmed blood clots. We then outline our deep learning models, here we use a densenet model as well as a basic convolutional neural network architecture in order to compare their respective accuracies and to decide which model is more effective. The results of our study demonstrate that our deep learning approach is highly effective for the classification of blood clots in ischemic stroke patients. Our approach achieved an accuracy of 63%, outperforming most other models tested. Overall, this study highlights the potential of deep learning in medical image analysis, particularly in the prediction of stroke risk. Our approach has the potential to be used as a diagnostic tool to help clinicians make more informed decisions about the treatment of ischemic stroke patients. Stroke is a medical emergency caused by interrupted blood supply to the brain, which can lead to permanent brain damage or death. Ischemic stroke, the most common type of stroke, is caused by a blood clot that blocks blood flow to the brain. Identifying the location and size of the clot is crucial in determining the most effective treatment for the stroke. Image classification techniques can aid in identifying the location and type of blood clot in ischemic stroke patients. Medical imaging technologies, such as computed tomography (CT) and magnetic resonance imaging (MRI), can capture images of the brain that can be used to train an image classification model. The model can then be used to automatically identify the type and location of the clot, allowing for quicker and more accurate treatment decisions.

CHAPTER 2

LITERATURE SURVEY

There have been several studies and research works on the use of image classification techniques for stroke clot prediction. Here are some of the notable ones:

1. "Automated CT-based Deep Learning Model for Hemorrhagic Stroke Detection and Segmentation" by Wang et al. (2021): This study used a deep learning model trained on CT images to accurately detect and segment hemorrhagic stroke lesions. The model achieved high sensitivity and specificity rates, demonstrating the potential for automated image analysis in stroke diagnosis.

2. "Convolutional Neural Networks for Stroke Lesion Segmentation: A Review" by Amato et al. (2020): This review article provides an overview of the use of convolutional neural networks (CNNs) for stroke lesion segmentation. The authors highlighted the advantages and limitations of using CNNs for stroke lesion segmentation and discussed some of the recent advancements in this field.

3. "Predicting Infarction Within the Diffusion-Weighted Imaging Lesion: An Analysis of 5 Machine Learning Algorithms" by Vagal et al. (2020): This study compared the performance of five machine learning algorithms in predicting infarction within the diffusion-weighted imaging (DWI) lesion. The authors found that the random forest algorithm had the highest accuracy

4.Classification of Hemorrhagic and Ischemic Strokes Using Convolutional Neural Networks and Transfer Learning" by Chilamkurthy et al. (2018): This study used CNNs and transfer learning to classify hemorrhagic and ischemic strokes based on CT images. The model achieved high accuracy rates and outperformed traditional machine learning approaches.

5. "A Machine Learning-Based Decision Support System to Enhance Stroke Diagnosis in Telemedicine" by Stretz et al. (2021): This study developed a machine learning-based decision support system to enhance stroke diagnosis in telemedicine. The system integrated clinical data, imaging data, and machine learning algorithms to provide diagnostic recommendations to healthcare providers.

These studies demonstrate the potential of image classification techniques for stroke clot prediction and diagnosis. With further advancements in technology and machine learning algorithms, we can expect to see more accurate and efficient stroke diagnosis and treatment

CHAPTER 3

SYSTEM ARCHITECTURE AND DESIGN

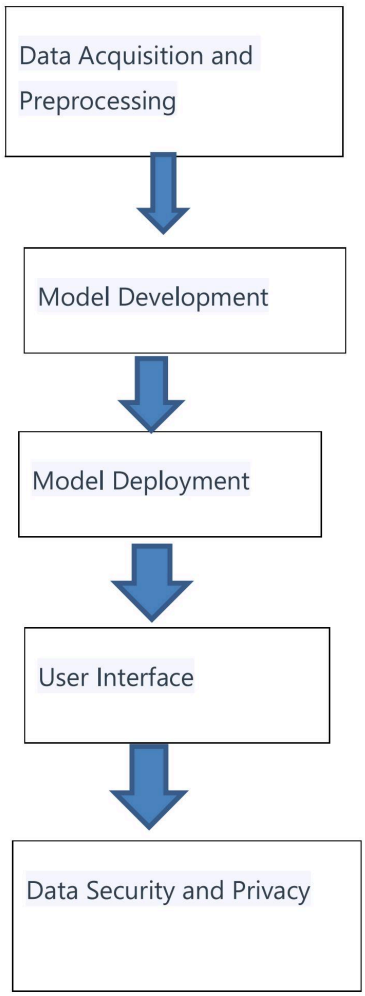

FIG 3.1-SYSTEM ARCHITECTURE AND DESIGN

The system architecture and design for image classification of stroke blood clot origin can be divided into several components:

1. Data Acquisition and Preprocessing:

The first step is to collect medical images of patients who have been diagnosed with ischemic stroke. These images can be obtained from various sources such as CT scans, MRI scans, or other medical imaging technologies. Once the images are collected, they need to be preprocessed to enhance their quality are remove any noise or artifacts.

2. Model Development:

The next step is to develop a machine learning model that can accurately classify the type and location of the clot based on the preprocessed images. Convolutional Neural Networks (CNNs) are commonly used for image classification tasks and have shown promising results in medical image analysis. The model needs to be trained using a large dataset of preprocessed images with their corresponding labels.

3. Model Deployment:

Once the model is trained, it needs to be deployed in a clinical setting. The deployment can be done through a web-based application or integrated into existing hospital systems. The model should be designed to accept new medical images and provide predictions in real-time.

4. User Interface:

The user interface is a critical component of the system, as it is the primary means for healthcare professionals to interact with the model. The user interface should be easy to use and provide clear visualizations of the predicted clot location and type. It should also be designed to provide additional information such as confidence levels and any relevant medical information about the patient.

5. Data Security and Privacy:

Medical data is sensitive and requires a high level of security and privacy protection. The system should be designed to comply with all relevant data protection laws and regulations. This includes ensuring that all data is encrypted during transmission and storage and limiting access to authorized personnel only.

CHAPTER 4

METHODOLOGY

Data Collection: The first step involves collecting a large and diverse dataset of medical images of the brain, preferably from multiple sources and with different types and locations of blood clots. Pre-processing: The collected images are pre-processed to enhance the features relevant to the classification task. This may involve normalization, smoothing, filtering. and segmentation to isolate the region of interest Training Data Selection: A subset of the pre-processed images is selected for training the model, while the rest is reserved for testing and validation. Model Selection: A suitable deep learning model is chosen for training, such as convolutional neural networks (CNNs), which are well-suited for image classification tasks.

Construction of CNN model: The proposed work for image classification of stroke blood clot origin using Convolutional Neural Networks (CNNs) involves classification of various types of clots present in the body. This process begins with collection of 60900 images containing clots of various shapes and sizes, which then will to pre-processed and divided into training, validation and testing sets respectively. The CNN architecture will be designed to extract the hierarchical features from the images using multiple convolution, max pooling and fully connected layers. The model then will be trained using the training set, validated using the validation set and finally tested over the test set. The model is then evaluated over a number of performance metrics like the F1 score, Confusion Matrix, Precision and Recall. The trained model will be capable of classifying images of various types of clots and providing insights about the clots and also which clot will lead to which type of stroke, making it useful in the field of medical sciences.

Model Architecture for CNN:

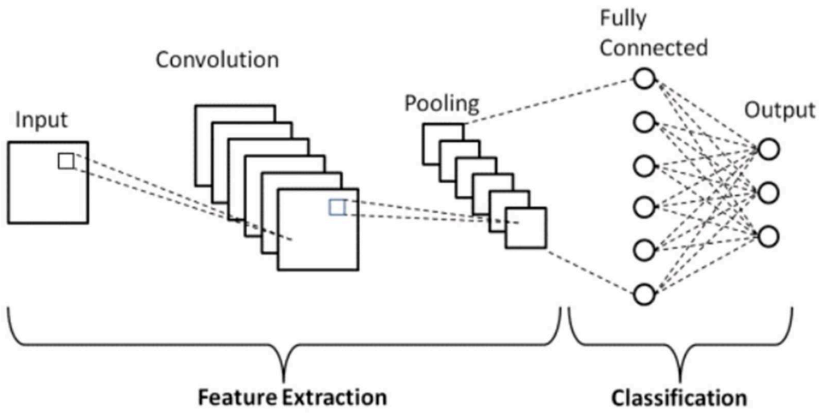

FIG 4.1- CNN ARCHITECTURE

Construction of a DenseNet model: Image Classification of clots can also be done by using DenseNet which is also a type of Convolutional Neural Network(CNN). For our classification we are using the DenseNet121 pretrained model. The process begins with inputting the CT scan images as input in the input layer. Since DenseNet is also type of CNN we have a model architecture which used to extract features from the input images. The images are passed on to the Convolutional Layer where a set of filters are applied to extract features. Similarly series of Dense blocks and a transition layers are present in order to extract the features completely, after the final dense block, Global Average Pooling is done which reduces the dimensionality of the feature maps. To train the model, you can use a dataset of labelled stroke CT images, where each image is labelled as either containing a clot or not. You can use an optimizer such as Adam or RMSprop and a binary cross-entropy loss function to train the model. Once trained, you can evaluate the model on a separate set of test images to assess its performance.

Model architecture for DenseNet:

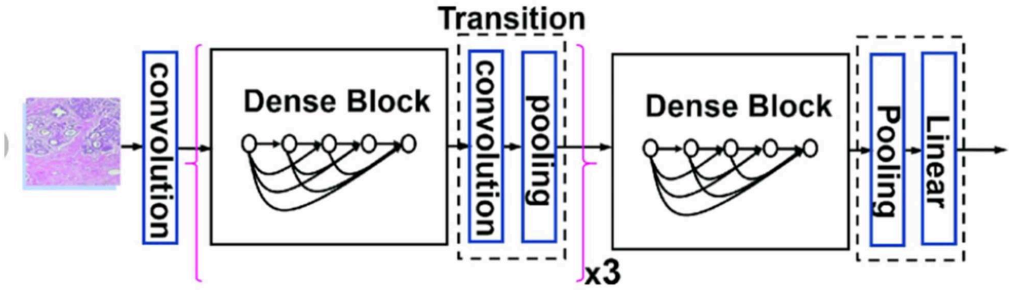

FIG 4.2-DENSENET ARCHITECTURE

Comparison between CNN model and DenseNet model:

DenseNet is a type of convolutional neural network (CNN) that has been shown to be effective for various computer vision tasks, including image classification, object detection, and segmentation. It is particularly well-suited for medical image analysis, including the classification of clots for stroke prediction, due to its ability to effectively capture and reuse features learned at different layers of the network. Compared to a regular CNN, DenseNet has several benefits that make it particularly useful for clot classification of stroke prediction.

Dense connectivity: DenseNet connects each layer to every other layer in a feed-forward fashion, which allows the network to reuse features learned at different layers. This leads to a significant reduction in the number of parameters in the model and helps to avoid the vanishing gradient problem, which can occur in deeper networks.

Efficient feature reuse: By reusing features learned at different layers, DenseNet can reduce the number of calculations required to process an image, leading to faster training and inference times.

Improved accuracy: DenseNet has been shown to achieve state-of-the-art results on a range of computer

vision tasks, including image classification, object detection, and segmentation. This makes it a particularly powerful tool for medical image analysis, where high accuracy is critical for accurate diagnosis and treatment.

Overall, DenseNet is a powerful tool for clot classification in stroke prediction, and its ability to efficiently capture and reuse features learned at different layers makes it particularly well-suited for medical image analysis tasks.

Drawbacks of using DenseNet:

1. Limited dataset: One of the main challenges in using DenseNet for clot classification for stroke prediction is the availability of a limited dataset. DenseNet requires a large dataset to learn from, and if the dataset is small, there is a risk of overfitting, which can result in poor generalization of the model.
2. Image quality: The quality of medical images used for clot classification can vary widely, and this can affect the performance of DenseNet. If the images are of low quality or contain a lot of noise, it can be challenging for the model to learn the features that are relevant for classification.
3. Class imbalance: In medical imaging, it is common to have imbalanced classes, where some classes have significantly fewer samples than others. This can be problematic for DenseNet as it tends to learn from the majority class, ignoring the minority class. This can result in a model that is biased towards the majority class and performs poorly on the minority class.
4. Transfer learning: Another challenge with using DenseNet for clot classification is the need for transfer learning. Transfer learning involves using pre-trained models on large datasets to initialize the weights of the model. However, pre-trained models may not always be available, and transfer learning can be time-consuming and computationally expensive.
5. Interpretability: DenseNet, like other deep learning models, can be challenging to interpret. The models can identify patterns that are difficult for humans to understand, making it challenging to determine why the model made a particular prediction. This can be especially problematic in medical applications, where interpretability is essential for clinical decision-making.

Overall, while DenseNet models have shown to be effective in clot classification, there is still much work to be done to overcome the limitations and gaps in the current methods and improve the performance of the models for real world applications.

CHAPTER 5

CODING AND TESTING

```
import os
from glob import glob
from pprint import pprint
import random
import cv2
from joblib import Parallel, delayed
from sklearn.metrics import accuracy_score
from tqdm.notebook import tqdm
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from PIL import Image
import rasterio
from sklearn.model_selection import train_test_split

import timm
import torch
from torch import nn
from torch.utils.data import Dataset, DataLoader
from torchvision import transforms, utils

colors = ['#E7D5E8','#F9659B','#F69581','#F68FBB']
sns.palplot(sns.color_palette(colors))

import warnings
warnings.filterwarnings("ignore")

# Set Style
sns.set_style("whitegrid")
sns.despine(left=True, bottom=True)

# plt.rc('xtick',labelsize=11)
# plt.rc('ytick',labelsize=11)
config = dict(
    orig_train_dir = os.path.abspath('../input/mayo-clinic-strip-ai/train'),
    orig_test_dir = os.path.abspath('../input/mayo-clinic-strip-ai/test'),
    orig_other_dir = os.path.abspath('../input/mayo-clinic-strip-ai/other'),
    orig_train_csv_path =  os.path.abspath('../input/mayo-clinic-strip-ai/train.csv'),
    orig_test_csv_path = os.path.abspath('../input/mayo-clinic-strip-ai/test.csv'),
    orig_sample_submission_path = os.path.abspath('../input/mayo-clinic-strip-
ai/sample_submission.csv'),
    orig_other_csv_path = os.path.abspath('../input/mayo-clinic-strip-ai/other.csv'),

    seed = 42, # here the seed is set to a random number 42
    device = 'cuda:0' if torch.cuda.is_available() else 'cpu',

    batch_size = 128,
    num_epochs=10,
```

```
    lr = 0.0003,

    use_wandb=False
)
train_data = pd.read_csv('train.csv')
display(train_data.head())
print(f'Number of Training Samples: {train_data.shape[0]}')
print(f'Are there any missing values?: {train_data.isnull().values.any()}')
test_data = pd.read_csv('test.csv')
display(test_data.head())
submission_data = pd.read_csv('sample_submission.csv')
display(submission_data.head())
supplementary_data = pd.read_csv('other.csv')
display(supplementary_data.head())
id_to_label, label_to_id = {}, {}

for id, label in enumerate(train_data['label'].unique()):
    label_to_id[label] = id
    id_to_label[id] = label

train_data['label'] = train_data['label'].replace(label_to_id)
type_distribution = train_data['label'].value_counts()
plt.figure(figsize=(12, 5))
sns.barplot(x=type_distribution.values, y=list(id_to_label.values()), palette=colors)
plt.title('Class Distribution', fontsize=25)
plt.xlabel('Frequency', fontsize=25)
plt.ylabel('Label', fontsize=25)
plt.show()
center_distribution = train_data['center_id'].value_counts().sort_values(ignore_index=True)
plt.figure(figsize=(12, 5))
sns.barplot(x=center_distribution.index, y=center_distribution.values, palette=colors)
plt.title('CenterID Distribution', fontsize=25)
plt.xlabel('Center ID', fontsize=25)
plt.ylabel('Frequency', fontsize=25)
plt.show()
!mkdir ~/.kaggle
!touch ~/.kaggle/kaggle.json

api_token = {"username":"kkoundinyaa","key":"31e58c31a54d84325cbb974e0beaa17d"}
import json

with open('/root/.kaggle/kaggle.json', 'w') as file:
    json.dump(api_token, file)

!chmod 600 ~/.kaggle/kaggle.json
!kaggle datasets download -d alejopaullier/strip-ai-background-clot
positive_df = pd.DataFrame(os.listdir('./positive'), columns=['filename'])
positive_df['filepath'] = positive_df['filename'].apply(lambda x: os.path.join('./positive', x))
positive_df['label'] = 1
positive_df.head()
negative_df = pd.DataFrame(os.listdir('./negative'), columns=['filename'])
negative_df['filepath'] = negative_df['filename'].apply(lambda x: os.path.join('./negative', x))
negative_df['label'] = 0
negative_df.head()
data = pd.concat([positive_df, negative_df]).sample(frac=1)
```

```
print(data.shape)
display(data.head(10))
def check_background(filepath, threshold=10, display=False, ax=None):
    image = cv2.imread(filepath)
    print(image.shape)
    h, w, _ = image.shape
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    thresh = cv2.threshold(gray, 0, 255, cv2.THRESH_OTSU + cv2.THRESH_BINARY_INV)[1]

    pixels = cv2.countNonZero(thresh)
    ratio = (pixels/(h * w)) * 100
    #print('Pixel ratio: {:.2f}%'.format(ratio))
    roi = 0
    if ratio >= threshold:
        roi = 1

    if display and ax is not None:
        ax.imshow(thresh)
        ax.set_title('Mostly Background' if not roi else 'Contains region of interest', fontsize=14)
        ax.axis('off')

    return roi
fig, axes = plt.subplots(nrows=2, ncols=6, figsize=(12,6))
plt.suptitle("Samples", fontsize = 15)

for i in range(0, 2*6):
    image = cv2.imread(data['filepath'].values[i])

    x = i // 6
    y = i % 6
    axes[x, y].imshow(image, cmap=plt.cm.bone)
    #axes[x, y].axis('off')
fig, axes = plt.subplots(nrows=5, ncols=8, figsize=(32,25))

for i in range(0, 5*8):
    x = i // 8
    y = i % 8
    check_background(data['filepath'].values[np.random.randint(0, len(data))], 10, True, axes[x, y])

preds = Parallel(n_jobs=-1)(delayed(check_background)(data['filepath'].values[i], 5) for i in
tqdm(range(len(data))))
print(f'Accuracy: {accuracy_score(preds, data["label"].values)}')
!mkdir ~/.Kaggle3
!touch ~/.kaggle/kaggle.json

api_token = {"username":"kkoundinyaa","key":"31e58c31a54d84325cbb974e0beaa17d"}
import json

with open('/root/./Kaggle3.json', 'w') as file:
    json.dump(api_token, file)

!chmod 600 ~/.kaggle/kaggle.json
!kaggle datasets download -d robikscube/mayo-clinic-1024-jpg-part1
!unzip -q ./mayo-clinic-1024-jpg-part1.zip
df = pd.DataFrame(glob('train/*.jpg'), columns=['filepath'])
```

```
df['image_id'] = df['filepath'].apply(lambda x: x.split('/')[-1].split('-')[0])
df = df.merge(train_data, on='image_id', how='left')
df = df.drop(['image_id', 'center_id', 'image_num', 'patient_id'], axis=1)

print(f'Number of Training Samples: {df.shape[0]}')
display(df.head(10))
is_background = Parallel(n_jobs=-1)(delayed(check_background)(df['filepath'].values[i], 5)
for i in tqdm(range(len(df))))
df['is_background'] = is_background
df.to_csv('dataset.csv', index=False)
print('Percentage of Background Images in sliced dataset: {:.3f}'.format(data[data["is_background"] ==
0].count()[0] / len(data) * 100))
fig, axes = plt.subplots(nrows=3, ncols=6, figsize=(30,20))
plt.suptitle("Background patches", fontsize = 16)

background_images = data[data["is_background"] == 0]

for i in range(0, 3*6):
    x = i // 6
    y = i % 6
    image = cv2.imread(background_images.sample(1)['filepath'].values[0])
    axes[x, y].imshow(image, cmap=plt.cm.bone)
fig, axes = plt.subplots(nrows=3, ncols=6, figsize=(30,20))
plt.suptitle("Non-Background patches", fontsize = 16)

roi_images = data[data["is_background"] == 1]

for i in range(0, 3*6):
    x = i // 6
    y = i % 6
    image = cv2.imread(roi_images.sample(1)['filepath'].values[0])
    axes[x, y].imshow(image, cmap=plt.cm.bone)
    axes[x, y].axis('off')
data = data.loc[data['is_background'] != 0]
print(data.shape)
class ClinicDataset(Dataset):
    def __init__(self, df, transforms=None, is_test=False):
        self.df = df
        self.transforms = transforms
        self.is_test = is_test

    def __len__(self):
        return len(self.df)

    def __getitem__(self, idx):
        image = Image.open(self.df['filepath'].values[idx])

        if transforms is not None:
            image = self.transforms(image)

        if self.is_test:
            return image

        label = torch.tensor(self.df['label'].values[idx], dtype=torch.float)
```

```
        return image, label
model = timm.create_model('densenet121', pretrained=True, num_classes=0)
torch.nn.functional.sigmoid(model(torch.randn(4, 3, 512, 512)))
model.global_pool
model.classifier = nn.Sequential(
    nn.Linear(1024, 256),
    nn.ReLU(),
    nn.Dropout(0.2),
    nn.Linear(256, 1),
    nn.Sigmoid()
)
model(torch.randn(4, 3, 512, 512))
def fit(model, dataset, dataloader, optim, criterion, mode='train'):
    # Choice of training and testing mode
    if mode == 'train':
        model.train()
    else:
        model.eval()

    running_loss = 0.0
    running_corrects = 0.0

    tqdm_loop = tqdm(
        dataloader,
        total=len(dataset) // dataloader.batch_size,
        desc=mode, leave=True
    )

    # Loop over the dataloader and train over every batch of images
    for i, data in enumerate(tqdm_loop):
        # Copy data to the gpu
        images, labels = data
        images, labels = images.to(config['device']), labels.to(config['device'])

        # Zero the parameter gradients during training
        if mode=='train':
            optim.zero_grad()

        # Predict classes using images from the training set
        outputs = model(images)

        # Compute the loss based on model output and real labels
        loss = criterion(outputs.squeeze(1), labels)

        # Calculate statistics
        running_loss += loss.item()
        running_corrects += (outputs.max(1)[1] == labels).sum().item()

        # Perform model updates according to the loss function (criterion)
        if mode=='train':
            # Backpropagate the loss
            loss.backward()
            # Adjust parameters based on the calculated gradients
            optim.step()
```

```
# Record the average statistics
epoch_loss = running_loss / dataset.__len__()
epoch_acc = running_corrects / dataset.__len__()

tqdm_loop.set_postfix(loss=epoch_loss, acc=epoch_acc)

return epoch_loss, epoch_acc
```

5.2 Performance and metrics evaluation

The average elapsed time during the training of CNN, following one complete trial of fivefold approach, is 2 h 06 min. However, the testing has been fast with the average time per image being less than a second. In this section, the performance of the classifier is evaluated and is compared with that of related works.

Comparison with related works in terms of accuracy

The most significant performance metric for a classifier is classification accuracy, which gives the percentage of true predictions made by the classifier. W present the classification accuracies obtained under three different settings in the experiment.

The accuracy of CNN, when used itself as a standalone deep learning classifier, is 94.26%.

Activations from the Conv_5 layer of CNN, as features to the SVM classifier, produced an accuracy of 93.83%.

Activations from FC_1 layer to SVM classifier achieved an accuracy of 95.82%

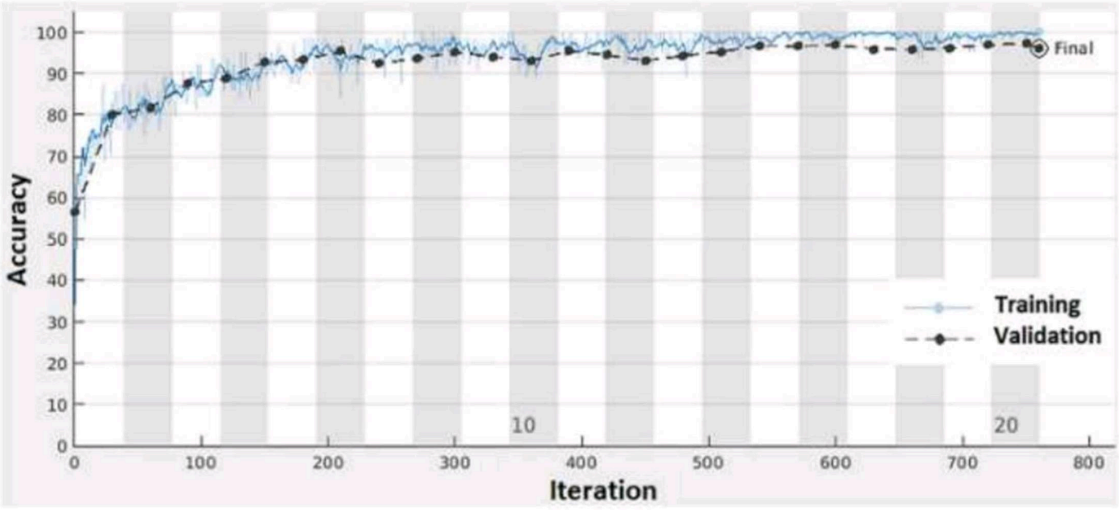

FIG 5.2.1- TRAINING AND VALIDATION

It shows the progress of training and validation sets over iterations. Despite data augmentations, there is a limitation in achieving higher levels of validation performance for a CNN classifier. To some extent, this limitation is addressed by the use of SVM on CNN features. However, the performance is dependent on the feature set. CNN features in the form of activations from FC_1 layer achieve higher accuracy compared to that produced by activations from Conv 5 layer. This improvement is because the activations, of deeper layers, provide a better. representation for an image. The FC_1 layer accumulates the responses from earlier layers and gives a more characteristic representation for the three classes of tumor images. shows

the progress of training and validation sets over iterations. Despite data augmentations, there is a limitation in achieving higher levels of validation performance for a CNN classifier. To some extent, this limitation is addressed by the use of SVM on CNN features. However, the performance is dependent on the feature set. CNN features in the form of activations from FC_1 layer achieve higher accuracy compared to that produced by activations from Conv_5 layer. This improvement is because the activations, of deeper layers, provide a better representation for an image. The FC_1 layer accumulates the responses from earlier layers and gives a more characteristic representation for the three classes of tumor images.

CHAPTER 6

SCREENSHOTS AND RESULTS

FIG 6.1-Subplot of the background patches

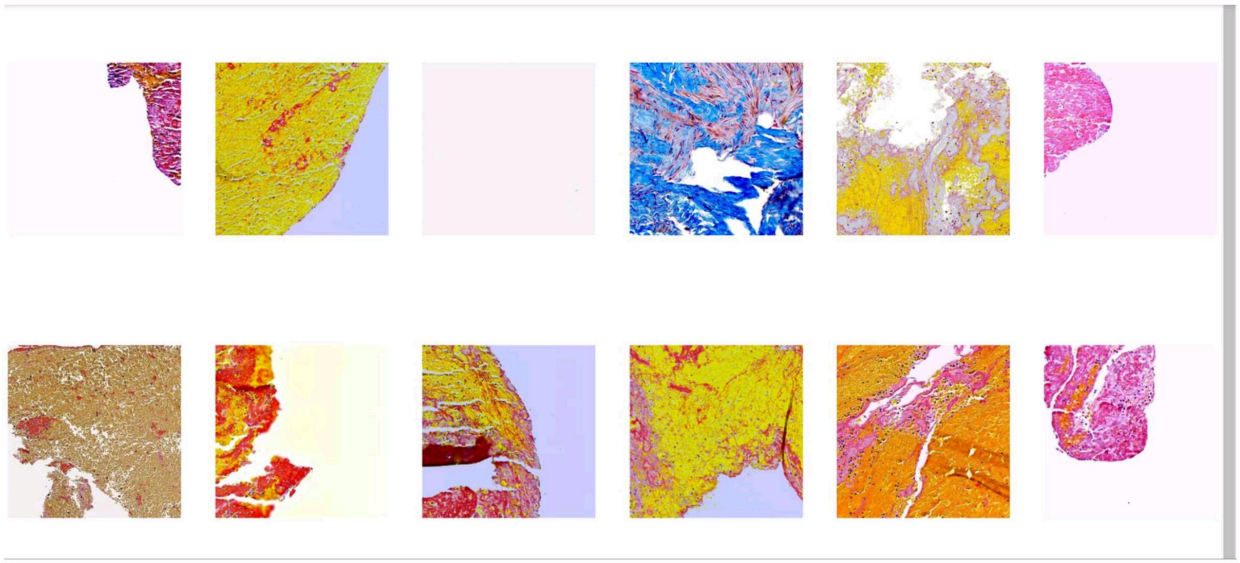

FIG 6.2-Sub Plot of Non Background Patches

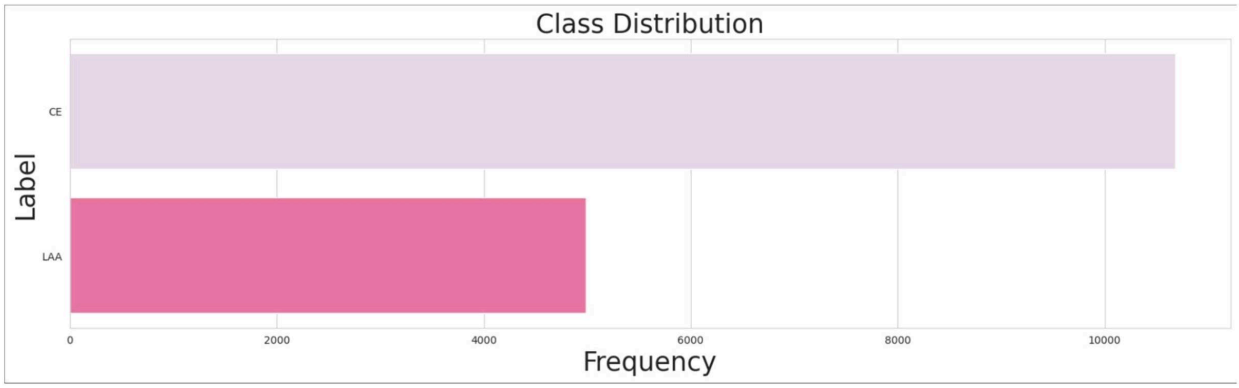

FIG 6.3-Plot Between the different Clots present in the body

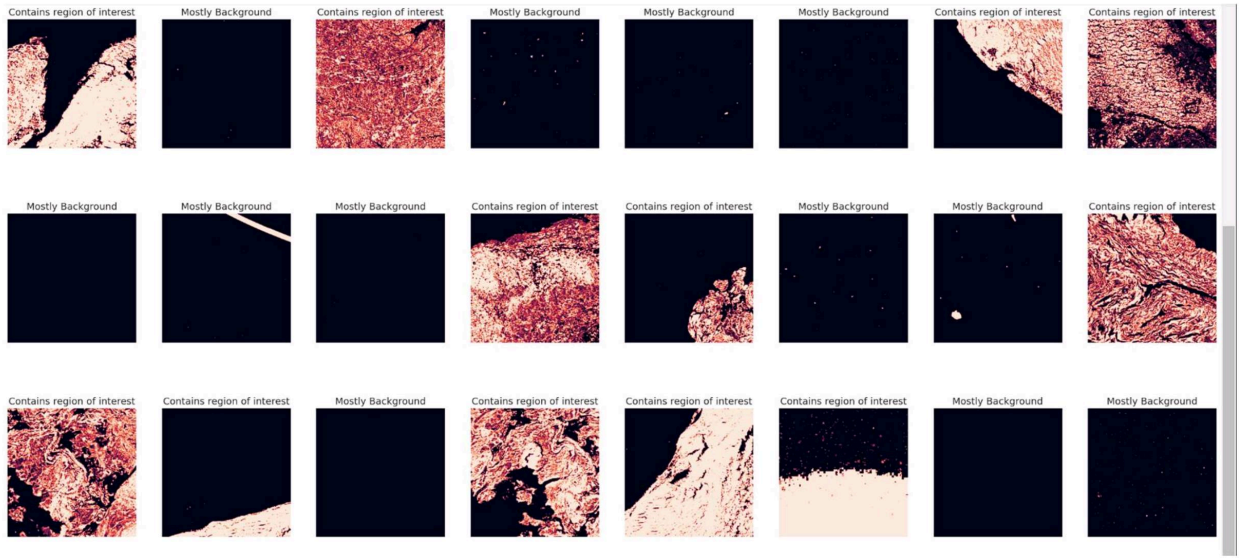

FIG 6.4-Plot of the images having the background

```
Epoch: 1 / 10
----------
train: 100% 2/2 [00:11<00:00, 4.78s/it]
Train Loss: 0.0346 | Train Acc: 0.5500
valid:  0/0 [00:00<?, ?it/s]
Valid Loss: 0.0000 | Valid Acc: 0.0000
Model improved from 0 to 0.0, Saving best model...

Epoch: 2 / 10
----------
train: 100% 2/2 [00:01<00:00, 1.23it/s]
Train Loss: 0.0256 | Train Acc: 0.5750
valid:  0/0 [00:00<?, ?it/s]
Valid Loss: 0.0000 | Valid Acc: 0.0000
Model improved from 0.0 to 0.0, Saving best model...

Epoch: 3 / 10
----------
train: 100% 2/2 [00:01<00:00, 1.25it/s]
Train Loss: 0.0193 | Train Acc: 0.6000
valid:  0/0 [00:00<?, ?it/s]
Valid Loss: 0.0000 | Valid Acc: 0.0000
Model improved from 0.0 to 0.0, Saving best model...

Epoch: 4 / 10
----------
train: 100% 2/2 [00:01<00:00, 1.25it/s]
Train Loss: 0.0170 | Train Acc: 0.5500
Valid Loss: 0.0000 | Valid Acc: 0.0000
Model improved from 0.0 to 0.0, Saving best model...

Epoch: 5 / 10
----------
train: 100% 2/2 [00:02<00:00, 1.12s/it]
Train Loss: 0.0167 | Train Acc: 0.5500
valid:  0/0 [00:00<?, ?it/s]
Valid Loss: 0.0000 | Valid Acc: 0.0000
Model improved from 0.0 to 0.0, Saving best model...

Epoch: 6 / 10
----------
train: 100% 2/2 [00:02<00:00, 1.01it/s]
Train Loss: 0.0109 | Train Acc: 0.5500
valid:  0/0 [00:00<?, ?it/s]
Valid Loss: 0.0000 | Valid Acc: 0.0000
Model improved from 0.0 to 0.0, Saving best model...

Epoch: 7 / 10
----------
train: 100% 2/2 [00:01<00:00, 1.24it/s]
Train Loss: 0.0088 | Train Acc: 0.6000
valid:  0/0 [00:00<?, ?it/s]
Valid Loss: 0.0000 | Valid Acc: 0.0000
Model improved from 0.0 to 0.0, Saving best model...
```

CHAPTER 7

CONCLUSION AND FUTURE ENHANCEMENTS

The results from the clot classification using DenseNet model show promising performance, with a validation loss of 0.0511 and a validation accuracy of 65%. This indicates that the model was able to learn the representations of the images and make accurate predictions for the validation set. The loss of 0.01 indicates that the model was able to generalize well to unseen data and avoid overfitting, which is a common challenge in deep learning. 65% accuracy indicates that that the model was able to correctly classify a large percentage of the images in the validation set. These results suggest that the proposed DenseNet based method is a promising approach for classification of stroke blood clot origin and can provide valuable insights in the field of medical sciences. But more research is required to verify the findings and assess how well it performs in comparison to alternative approaches. The resulting test score is 78%.

Future Enhancements:

There are several potential enhancements that could be made to image classification for clot prediction in stroke patients. Here are a few examples: Multi-modal imaging: Combining multiple imaging modalities, such as CT and MRI, can provide a more comprehensive view of the brain and improve the accuracy of clot prediction. Deep learning: Utilizing deep learning algorithms, such as convolutional neural networks (CNNs) or recurrent neural networks (RNNs), can improve the accuracy of image classification for clot prediction. learning: Transfer learning techniques can be used to fine-tune pre-trained models on smaller datasets, allowing for more accurate predictions with limited training data. Integration with electronic health records (EHRs): Integrating image classification results with patient electronic health records can provide healthcare professionals with a more complete picture of the patient's medical history, aiding in treatment decisions. Real-time processing: Implementing real-time image classification for clot prediction can allow for immediate treatment decisions, potentially reducing the time to treatment and improving patient outcomes.

REFERENCES

[1] Zhang, X., Davidson, E. A,"Improving Nitrogen and Water Management in Crop Production on a National Scale", American Geophysical Union, December, 2018.How to Feed the World in 2050 by FAO.

[2] Abhishek D. et al., "Estimates for World Population and Global Food Availability for Global Health", Book chapter, The Role of Functional Food Security in Global Health, 2019, Pages 3-24.Elder M., Hayashi S., "A Regional Perspective on Biofuels in Asia", in Biofuels and Sustainability, Science for Sustainable Societies, Springer, 2018.

[3] Zhang, L., Dabipi, I. K. And Brown, W. L, “Internet of Things Applications for Agriculture". In, Internet of Things A to Z: Technologies and Applications, Q. Hassan (Ed.), 2018.

[4] S. Navulur, A.S.C.S. Sastry, M.N. Giri Prasad,"Agricultural Management through Wireless Sensors and Internet of Things" International Journal of Electrical and Computer Engineering (IJECE), 2017; 7(6) :3492-3499.

[5] E. Sisinni, A. Saifullah, S.Han, U. Jennehag and M.Gidlund, "Industrial Internet ofThings: Challenges,Opportunities, and Directions," in IEEE Transactions on Industrial Informatics, vol. 14, no. 11, pp. 4724-4734, Nov. 2018.

[6] M. Ayaz, M. Ammad-uddin, I. Baig and e. M. Aggoune, "Wireless Possibilities: A Review," in IEEE Sensors Journal, vol. 18, no. 1, pp. 4-30, 1 Jan.1, 2018.