Undergraduate Research Opportunity Programme (UROP) Project Report

Project Title

By

John Doe

Department of Statistics and Data Science

Faculty of Science

National University of Singapore

2024/2025

Undergraduate Research Opportunity Programme (UROP) Project Report

Project Title

By

John Doe

Department of Computer Science

School of Computing

National University of Singapore

2024/2025

Project ID: UXXXXX

Advisor: Professor John Doe

Deliverables:

Report: 1 Volume

Abstract

Your abstract goes here, maximum 200 words.

Subject Descriptors:

D.2.4 Software/Program Verification

D.2.5 Testing and Debugging

Keywords:

Foo, Bar, Baz

Implementation Software and Hardware:

Hardware

Software

Acknowledgements

Your acknowledgements go here.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Example citation (Lamport, 1978).

1.1 Motivation

Chapter 2

Related Work

2.1 Design Overview

```
public class HelloWorld {
    public static void main(String[] args) {
        // Prints "Hello, World!" to the terminal window.
        System.out.println("Hello,␣World!");
    }
}
```

Figure 2.1: Hello World in Java

2.2 Testing

2.3 Problems Encountered

2.4 Limitations

Chapter 3

Materials and Methods

3.1 Dataset

The main dataset used in this project was collected from XXX hospital and consists of 4369 elbow radiographs in either the lateral or anterior posterior view. Each radiograph was named according to the type of injury present (supracondylar fracture, dislocation, contusion etc). Using this information, the data was labelled as either injury and normal. The injury set includes radiographs with fractures and/or dislocations while the normal set consists of all other radiographs. The data was further divided into training, validation, and test sets as shown in figure 3.1. (figure)

In addition, the MURA dataset from Stanford Machine Learning Group was used for pretraining the CNN models. This dataset consists of 40561 radiographs of 7 different bodyparts: elbow, finger, forearm, hand, humerus, shoulder and wrist. Each image is labelled as either positive (abnormality) or negative (no abnormality). Notably, the criteria for abnormality used in MURA differs from the criteria for injury in the main dataset. For instance, a contusion would be classified as normal in the main dataset but would be classified as abnormal in the MURA dataset. The division of the data into training and validation sets is shown in figure 3.2. (figure)

3.2 Data Processing

To improve the quality of the radiographs, all of the radiographs underwent preprocessing. First, the radiographs underwent Contrast Limited Adaptive Histogram Equalisation (CLAHE) using OpenCV. Adaptive Histogram Equalisation is an image processing technique which calculates histograms of the pixel intensities in different parts of an image. The pixel intensities are then redistributed using the histograms to improve the contrast of the image.

The implementation in OpenCV splits the image into a non overlapping grid and applies the technique subject to a clip limit which limits the amount of redistribution to reduce noise. The specific settings used were a clip limit of 2 and a grid size of 4.

Next, the radiographs were sharpened further using unsharp masking. First, each image was processed with a gaussian filter to produce a blurred version. The blurred image was then subtracted from the original image to produce a mask which is multiplied by a factor of 2 and added back to the original image. This process sharpens the image and increases contrast at edges. The sigma used for the gaussian filter was 5 for MURA dataset and 7 for the main dataset.

Next, the images are resized according to the model used. For the CNN models, the image size used for pretraining and training was 384 x 384 and 584 x 584 pixels respectively. As the models are fully convolutional, they could accept any image size without issue. For Medgemma and LLaVA-Rad, the images were resized to 896 x 896 and 518 x 518 respectively according to the requirements for each model. Lastly, the data was augmented with geometric transformations such as random rotations, scaling and translation. Brightness and contrast transformations were found to negatively impact performance and were thus excluded from this process.

3.3 CNN Models

3.3.1 Architecture

Convolutional Neural Networks (CNNs) are a type of neural network which use convolutional filters or kernels to perform convolutions on images and produce intermediate feature maps. These feature maps are then passed onto the following convolutional layers. At the final layer, the feature maps are pooled and flattened to produce a feature vector which is processed by a fully connected neural network to generate an output.

For this project, 3 CNN models were used: ConvNeXt, EfficientNetV2 and MobileNetV3. These models were chosen due to their excellent performance on ImageNet and low computational cost compared to similarly sized CNN models. After training the individual models, an ensemble model was created using feature fusion for the final classification. The framework used for the CNN models is outlined below.

- 1. Transfer Learning
  Transfer learning is a machine learning technique where a machine learning model is trained on one task and is then reused for another related task instead of starting again from scratch. This technique not only saves on training time but can also improve performance on the second task, especially if the amount of available training data is

limited.

In this case, the CNN models were initialised with weights which had been pretrained on ImageNet 1k. ImageNet 1k is a large dataset with over a million RGB radiographs divided into 1000 categories such as plants, animals and furniture and is commonly used as a source of transfer learning for image classification tasks. The weights for the models were obtained from Pytorch Hub which is a publicly available model repository.

- 2. Greyscale vs RGB
  ImageNet is a colour dataset with 3 channel (RGB) radiographs. Hence, the models cannot directly accept greyscale radiographs which only have 1 channel. This issue can be solved by duplicating the single channel twice to create a 3 channel image. However, this process is inefficient as it increases the memory cost of storing the image as well as the computational cost of processing the image through the model. To solve this, the first convolutional layer of each model was modified by summing the per channel weights of each kernel. This allows the models to directly accept greyscale radiographs without affecting the output of the first layer.

- 3. Convolutional Block Attention Module
  Convolutional Block Attention Module (CBAM) is an attention mechanism for CNN models which can be incorporated into an existing architecture and trained together with the rest of the model. CBAM takes in an intermediate feature map and infers a channel attention map and a spatial attention map. The intermediate feature map is then refined using element wise multiplication with the 2 attention maps. CBAM has been shown to improve the performance of CNNs on different computer vision tasks while incurring little additional computational cost.

  ConvNeXt was modified with 4 CBAM modules placed after each of the 4 ConvNeXt blocks. EfficientNet was modified with a single CBAM module between the FusedMBConv modules and the MBConv modules, as well as a second CBAM module after the final convolutional layer. MobileNet was modified with a single CBAM module after the final convolutional layer. The architecture for each model is shown below.

- 4. Ensembling
  Ensembling is a commonly used technique in machine learning which combines the output of several different base models to get a single model which outperforms the individual ones. This can be done through several methods such as a simple majority vote or training a model on the outputs of the individual models.

  In this case, feature fusion was used to combine the outputs of the 3 individual CNN models. This is done by concatenating the flattened feature vectors of each model before they are sent to the fully connected layer. By default, Convnext has 768 output channels while EfficientNet has 1280 output channels and MobileNet has 960 output channels. To ensure each model has equal input to the ensemble, the final convolutional layer of EfficientNet was modified to have 768 output channels and MobileNet had an additional convolutional layer added to reduce the channels from 960 to 768.

The concatenated feature vector with 2304 features is then normalised and passed to a dropout layer which randomly sets 30% of the features to 0 to mitigate overfitting. The feature vector is then used as the input for a small neural network to generate the final prediction.

3.3.2 CNN Training

As mentioned previously, ImageNet is a dataset of natural objects and is very different from the greyscale radiographs in our target dataset. Hence, it is not an ideal source of transfer learning as the features learnt from imagenet may not be useful. To address this issue, the MURA dataset was used for a pretraining step before training on the target dataset.

For pretraining, the Stochastic Gradient Descent (SGD) optimiser was used together with a cosine annealing learning rate scheduler with warm restarts. The training hyperparemeters were batch size 32, max epochs of 35, initial learning rate of 1.2e-3, 8e-4 and 1e-4 for ConvNeXt, EfficientNet and MobileNet respectively, momentum of 0.6 and weight decay of 1e-3. The cosine annealing scheduler was set to T_0 of 5 and T_mult of 2 which means the learning rate is decayed over 5 epochs in the first cycle, 10 epochs in the second cycle and 20 in the third.

Following pretraining, the pretrained weights of the 3 models were used for training on the target dataset. The hyperparameters for training were largely the same as pretraining. The differences were max epochs of 28, momentum of 0.8 and 0.7 for ConvNeXt and EfficientNet respectively and T_0 of 4. Lastly, for the ensemble model, the different hyperparameters were max epochs of 12, momentum of 0.8 and initial learning rate of 2e-4 for the fully connected layers and 5e-5 for the backbone.

3.4 Vision Language Models

3.4.1 Models

Vision language models (VLM) are multimodal generative models which combine a large language model (LLM) with a vision encoder which is usually a vision transformer model. The vision encoder processes image inputs into an embedding vector which has the same dimensions as the embedding generated by the LLM from the text input. The embeddings are then fused into a single embedding which combines both visual and textual information. This embedding is then passed on to the remainder of the LLM to generate a text output. This process allows VLMs to perform a large variety of tasks that involve image and/or text such as image captioning, question answering and image classification.

VLMs are typically pretrained on large generic datasets to create foundation models which can then be finetuned on downstream tasks. However, these datasets usually do not involve medical data which impacts the performance of these models on medical tasks. To address this issue, VLMs have been specifically trained on medical datasets for use in tasks such as medical report generation and diagnosis. For this project, the 2 VLMs chosen are Medgemma and LLava-Rad.

Medgemma is a collection of 2 publicly available foundation models created by google and trained for a wide range of medical related tasks such as radiology report generation and diagnosis. Medgemma is based on the Gemma 3 VLM and includes a 4 billion parameter and 27 billion parameter variant. Both variants use the SigLIP image encoder and have been trained on a large variety of medical data such as chest X-rays, histopathology slides and medical question answer pairs. LLaVA-Rad is another publicly available foundation model created by Microsoft and is focused on analysing chest X-rays and generating medical reports. Unlike Medgemma, LLaVA-Rad uses the BiomedCLIP image encoder and the Vicuna LLM and is trained on a large dataset of chest X-rays with corresponding radiology reports.

3.4.2 VLM Training

Finetuning the weights of an entire VLM is extremely computationally and memory intensive due to the large size of the model. Hence, a technique called Low Rank Adaptation (LoRA) was used instead. LoRA is a

Chapter 4

Evaluation

4.1 Hardware

4.2 Experimental Setup

4.3 Results

ID	Name	Score
1	Alice	88
2	Bob	92
3	Carol	85

Table 4.1: Example of a simple table.

Chapter 5

Challenges

Chapter 6

Conclusion

6.1 Summary

6.2 Future Work

References

Lamport, L. (1978). Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, *21*(7), 558–565. https://doi.org/10.1145/359545.359563

Appendix A - Something

Appendix B - Another Thing