Undergraduate Research Opportunity Programme in Science (UROPS)
Project Report

Project Title

By

Wong Song Ming

Department of Statistics and Data Science

Faculty of Science

National University of Singapore

2024/2025

Undergraduate Research Opportunity Programme in Science (UROPS)
Project Report

Project Title

By

Wong Song Ming

Department of Statistics and Data Science

Faculty of Science

National University of Singapore

2024/2025

Project ID: UXXXXX

Advisor: Professor Swapnil Mishra

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

placeholder

Example citation (Lamport, 1978).

Chapter 2

Related Work

placeholder

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 grouped into either injury or normal. The injury set includes radiographs with untreated 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 Table 3.1.

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 Table 3.2.

Set	Normal	Injury
Training	1957	1538
Validation	244	193
Test	186	251
Total	2387	1982

Table 3.1: Training Set

Set	Normal	Injury
Training	21935	14870
Validation	1533	1667
Total	23468	16537

Table 3.2: Pretraining Set

3.2 Data Processing

To improve the quality of the radiographs, all of the radiographs underwent preprocessing. Firstly, the images were 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 modifications to the architecture. 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.

Next, 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 2.5 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 the MURA dataset and 7 for the main dataset.

Lastly, the data was augmented with a combination of geometric transformations as follows: Random rotation between -60° and 60°, 75% probability of horizontal flip, scaling by a factor between 0.85 and 1.1 and translation along the x and y axis of between 0.02 and 0.12. This process artificially expands the training set to reduce overfitting and improve generalisation. Each image in the training set was augmented twice for the target dataset and once for the MURA dataset for a total of 10485 images and 73610 images respectively. Brightness and contrast transformations are also commonly used in image augmentation but were found to negatively impact performance and were thus excluded.

(insert examples of images)

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.

  (diagram to be inserted)

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 of 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 used are MedGemma 4B and LLava-Rad due to their high performance and relatively small size.

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. 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 technique used to finetune pretrained models which involves freezing the weights of the model and injecting lower rank matrices into the targeted parts of the model. During training, only these matrices are optimised instead of the original weights which greatly decreases the memory and time required. The matrices can then be merged with the original weights during inference to generate the desired output without increasing inference time.

As the target dataset consists solely of images, a prompt was added to each image to analyse the radiograph and classify it as A: 'No injury' or B: 'Fracture or dislocation'. The correct answer is either 'No injury' or 'Fracture or dislocation' depending on the class of the image. For both MedGemma and LLaVA-Rad, the pretrained weights were downloaded from Huggingface and used as a starting point for finetuning. However, for LLaVA-Rad, the checkpoint used for LLaVA-Rad is from the pretraining step which only tuned the image encoder. This is because the final step used LoRA instead of directly tuning the encoder and LLM.

The Adam-W adaptive optimiser was used instead of SGD as with the CNN models to increase the convergence speed. For MedGemma, the hyperparameters used were batch size of 32 divided into 8 accumulation steps, max epochs of 5, learning rate of 8e-5 with linear scheduling, weight decay of 1e-3, max grad norm of 0.03, LoRA rank of 16 and alpha of 32. For LLaVA-Rad, the different hyperparamters were max epochs of 4, learning rate of 1e-4 with cosine scheduling, LoRA rank of 32 and alpha of 64. For both models, the linear layers of the model and the embeddings were finetuned.

Chapter 4

Evaluation

4.1 Results

4.1.1 Evaluation Metrics

The models were evaluated using accuracy, precision, recall, F1 score and AUC-ROC (for the CNN models only). The first 4 metrics are calculated based on the true positive (TP), true negative (TN), false positive (FP) and false negative (FN) values at a particular prediciton threshold. AUC-ROC is calculated from a plot of the sensitivity of a model against specificity at different prediction thresholds. The equations are as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.1}$$

$$Precision = \frac{TP}{TP + FP} \tag{4.2}$$

$$Recall = \frac{TP}{TP + FN} \tag{4.3}$$

$$F1\,score = \frac{2 * TP}{2 * TP + FP + FN} \tag{4.4}$$

$$Sensitivity = \frac{TP}{TP + FN} \tag{4.5}$$

$$Specificity = \frac{TN}{TN + FP} \tag{4.6}$$

4.1.2 Results

The 4 CNN models were evaluated at a balanced threshold of 0.5 for all metrics except AUROC. Convnext achieved an accuracy of 92.5%, precision of 92.7%, recall of 92.2%, F1 score of 92.4% and AUROC of 0.976. EfficientNet achieved an accuracy of 88.7%, precision of 90.3%, recall of 87.0%, F1 score of 88.6% and AUROC of 0.971. MobileNet achieved an accuracy of 89.8%, precision of 86.5%, recall of 93.2%, F1 score of 89.7% and AUROC of 0.953. The Ensemble model achieved an accuracy of 93.1%, precision of 91.9%, recall of 94.3%, F1 score of 93.1% and AUROC of 0.978.

Model	Accuracy	Precision	Recall	F1 score	AUC-ROC
ConvNeXt	0.925	0.927	0.922	0.924	0.976
EfficientNet	0.887	0.903	0.870	0.886	0.971
MobileNet	0.898	0.865	0.932	0.897	0.953
Ensemble	0.931	0.919	0.943	0.931	0.978

Table 4.1: CNN Results

(insert confusion matrix when i've done it)

For MedGemma and LLaVA-Rad, the predictions are obtained from the generated text output and does not have a threshold like the CNN models. Hence, AUC-ROC could not be calculated. MedGemma achieved an accuracy of 86.6%, precision of 86.2%, recall of 87.0% and F1 score of 86.6%. LLaVA-Rad achieved an accuracy of 90.7%, precision of 87.9%, recall of 93.8% and F1 score of 90.8%.

Model	Accuracy	Precision	Recall	F1 score	AUC-ROC
MedGemma	0.866	0.862	0.870	0.866	-
LLaVA-Rad	0.907	0.879	0.938	0.908	-

Table 4.2: VLM Results

(insert confusion matrix when i've done it)

4.2 Discussion

placeholder

Chapter 5

Conclusion

5.1 Summary

placeholder

5.2 Future Work

placeholder

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