Assessing Fracture Healing with Artificial Intelligence:

Using Transfer Learning to Predict the *Radiographic Union Score for Tibial Fractures*

Author: Shen Zhou Hong

Supervisor: Georgios Mastorakis

May 5th, 2023

Declaration of Authorship

I, Shen Zhou Hong, declare that this thesis titled "Assessing Fracture Healing with Artificial Intelligence: Using Transfer Learning to Predict the *Radiographic Union Score for Tibial Fractures*" and the work presented in it are my own. I confirm that:

- Where I have consulted the published work of other authors, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- This thesis is entirely human work. No part of this thesis contains content generated by ChatGPT or other large language models.
- I have acknowledged all main sources of help.

Signed, Shen Zhou Hong.

Date: 2023-05-05.

Abstract

Keywords:

Acknowledgements

I would like to thank my supervisor, Georgios Mastorakis of Goldsmiths College, University of London, those support and encouragement were valuable aides throughout the journey of my thesis. Special thanks to Dr. Renan Castillo of the Johns Hopkins Bloomberg School of Public Health, those guidance planted the very seeds of this project. This endeavour would not have been possible without the work of my colleagues at the Major Extremity Trauma and Research Consortium (METRC), those research in the field of orthopaedic trauma yielded the very data that I build upon. I am profoundly grateful for the countless patients those participation in RetroDEFECT, Outlet, Pain, and Pacs, advanced our understanding of trauma.

Finally, I would like to thank my mother, Wang Yanqi, and grandmother Liu Xia, who raised me to pursue my passions fearlessly. Without them, I would not be the person that I am today.

Contents

Chapter 1

Introduction

Long bone fractures are a frequent effect of high-energy trauma [1], among which tibial fractures of the lower extremities are the most common. These fractures require long-term follow-up, where after initial fixation the fracture site must be re-examined at regular intervals for callus formation[1], bridging, and union [2]. Whelan et al's Radiographic Union Score for Tibial Fractures (RUST score) is a discrete 12-point scale that serves a common metric for the assessment of union from the lateral and antero-posterior[2] radiograph of a fracture [3]. This project proposes a means to automate the assessment of fracture healing, by using transfer-learning to develop a machine learning model to classify radiographs according to their RUST Scores.

1.1 Aims and Motivation

Non-union and delayed union are significant complications in fracture healing, one which results in heightened morbidity, loss-of-function, and infection risk [4]. As a result, it is important for physicians to determine non-union events so that further treatment and corrective surgery may be taken. Although non-union may be determined through a variety of clinical assessments (e.g. palpation, weight-bearing), the RUST score is emerging as a quantitative radiography-based measure with high consistency [5], biomechanical correlativity [6], and good guidance for postoperative rehabilitation [7]. However, in order to assess a fracture using the RUST score, a orthopaedic must examine at least two radiographs (one lateral, one anteroposterior) for callus formation — a non-trivial process.

Recent advances in deep learning, coupled with the increasing availability of large radiographic datasets (e.g. CheXpert, LERA, MURA) [8, 9, 10] offer the

[1] The development of cartilaginous material containing bone-forming cells.
[2] i.e. front-to-back.

possibility of automating the process of fracture classification. Certain research models such as Rajpurkar et al's DNN ConvNet are able to meet, or exceed radiologist-level performance for abnormality classification in specific anatomical domains [10], and as of 2021 commercial developments are beginning to see regulatory clearance[3] [11].

However, much of the current available literature[4] is focused on the mere detection and classification of fractures (i.e. abnormality detection). Such models either perform binary classification (e.g. "Is this a *normal* radiograph?"), multi-class (e.g. "Is this a *leg*, *arm*, or *knee* fracture?"), or localisation (e.g. "Where *is* the fracture on this radiograph?"). Comparatively less work has been done on the *characterisation* of radiographs, where the properties of a fracture are described [12]. The existence of this 'research gap' can be in part attributed to the absence of large radiographic datasets with quality labelling.

This gap in the field offers opportunity for further investigation, especially as it is not the mere presence of a fracture which informs medical decision-making, but rather its severity and properties. This study aims to build a machine-learning model which infers the RUST score of a fracture from a pair of radiographs, specifically utilising the technique of *transfer-learning* in order to address the challenges of working with small datasets. By building a model that infers the RUST score of a fracture from it's associated radiograph, we hope to demonstrate the feasibility of such an approach, and open the door to further investigation,

1.2 Objectives and Evaluation

The objective of this project is to develop an AI model using *transfer-learning* that is able to predict the RUST score of a pair of anteroposterior and lateral radiographs. Because our primary constraint is the size of our dataset (around 3,000 radiographs, see Dataset), we will be utilising the technique of transfer-learning to demonstrate the feasibility of training AI models on datasets of this order of magnitude. We will be using the InceptionV3 model architecture [13] as the base model with a custom classifier for our transfer-learning approach.

First, we will build and evaluate model that is end-to-end trained without the use of transfer-learning (Protocol I). This will serve as a baseline benchmark for our subsequent attempts. Next, we will evaluate two different transfer learning approaches based on the InceptionV3 model architecture. The first model will use InceptionV3 trained with the general-purpose ImageNet dataset [14] (Protocol II). The second model will use the same InceptionV3 architecture, but trained with the domain-specific RadImageNet dataset [15] (Protocol III). The development and comparative evaluation

[3] Authorisation for real-world clinical use by national health agencies like the Food and Drug Administration (FDA), Health Canada, *Conformité Européenne*, etc.

[4] A selection of which are analysed with commentary in chapter 2.

of these two base models for transfer learning will allow us to compare and contrast the use of a model pre-trained on a general-purpose dataset (ImageNet) versus a model pre-trained on a slightly smaller, but domain-specific dataset (RadImageNet).

Afterwards, we will select the best-performing base model and find the best-performing hyperparameters. This will be done via a hyperparameter search in two steps, called Regime I and Regime II. Regime I will be a random-search on batch size and dropout, while Regime II will be a grid search on learning rate and epsilon. Finally, the model trained with the best-performing hyperparameters will be evaluated on the hold-out test set, and the overall performance will be examined.

Chapter 2

Background Research

The use of artificial intelligence in the analysis of radiography predates deep learning, with early approaches reliant on predefined engineered features and handcrafted algorithms (e.g. edge detection, wavelet transform) [16]. With the advent of more powerful computer hardware and the democratisation of machine learning through open source frameworks, we see deep learning techniques being applied to the field of medical imagery. These studies often had to solve unique, domain-specific challenges — such as small datasets, the need for *evidence-based* labeling[1], and difficult image-preprocessing requirements. This project will face similar challenges. Hence, by taking a survey of existing literature, we may be better informed in overcoming these challenges.

2.1 Early Period: Small Datasets, Feature-Engineering

The first period of AI-based based fracture detection was characterised by the limitations of small datasets [2] , and innovative approaches aimed to overcome said limits. One example of work in this period was Cao et al's use of feature fusion in random-forests, which allowed multiple categories of features to be considered by the model [17]. Likewise, Dimililer's *Intelligent Bone Fracture Detection system* used meticulous image pre-processing, with Haar wavelet transforms and Sub-Variant Feature Transform (SIFT) [3] in order to extract invariances from the radiography image [18]. Both authors relied on a combination of domain-specific image pre-processing and feature-engineering, in order to compensate for limited datasets that they had. The limitations of their data further manifested in difficulties with model evaluation. In

[1] Labelling that is done by a clinical professional who is empowered to issue diagnoses.

[2] Often working with an individual hospital or medical center, these studies these studies generally had hand-annotated datasets of up to a hundred images, and seldom more than two hundred.

[3] Both functions are algorithms from the domain of signal processing, designed to compress spatio-temporal information in a manner that preserves invariances.

[18], only 100 labelled radiographs were available, making the use of a distinct hold-out validation set impossible. The lack of a separate validation set makes it difficult to judge whether or not the model performs without overfitting, weakening the study's conclusion. One possible mitigation that the author of [18] could have considered was k-fold cross-validation, which [17] does implement. In [17] Cao et al. implements 10-fold cross-validation over a data set of 145 radiographs, hereby yielding a more rigorous assessment of the model's performance. However, [17] did not use any data augmentation strategy, which may have been useful in light of the limited dataset available.

What lessons can we draw from these two studies? Radiographic imagery is highly hetereogenous, with individual x-rays mostly consisting of dark and light regions with sharp transitions in-between. Pre-processing steps such as scaling or down-sampling must capture these discontinuities faithfully, in order to prevent information-loss in the input. We can take particular inspiration from [18], who uses Haar wavelet transform in order to downsample inputs without loss of detail in the transition boundaries.[4] Likewise, we must use k-fold cross validation, and consider a data augmentation strategy for our own model. Overall, these two examples are representative of early, exploratory work in AI-based fracture detection, and they are useful to illustrate both solutions to working with small datasets (pre-processing, feature-engineering), as well as challenges (difficulties with validation). The context of the above studies help us better understand later work, which began with the advent of large, publicly-available datasets: the most important of them being the Stanford Musculoskeletal Radiography Dataset (MURA).

[4] According to Dimililer: "Unlike the discrete cosine transform, the wavelet transforms are not fourier based and therefore discontinuities in image data can be handled with better results using wavelets." [18]

2.2 MURA: Large Musculoskeletal Radiography Datasets

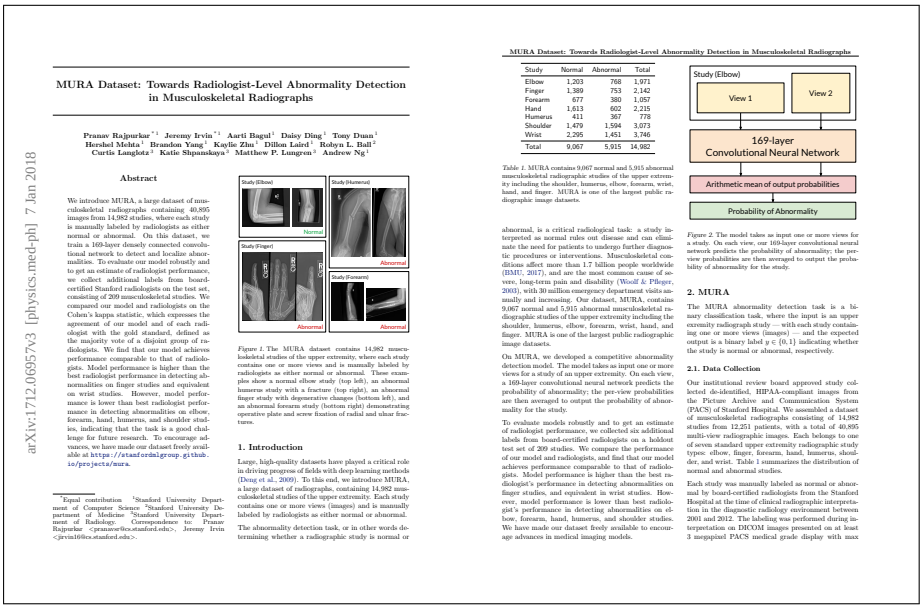


Figure 2.1: Thumbnail of article by Rajpurkar et al. [10]

MURA is one of the first large radiographic datasets focused exclusively on musculoskeletal imagery, as well as the name of a study conducted alongside said data by the Stanford Center for AI in Medical Imaging [10]. The MURA study marks a distinct landmark in the field of AI-assisted fracture classification, because it was the first to apply a deep-learning model on a large (40,561 radiographs), publicly available dataset. As a result, an examination of MURA allows us to contextualise our study in important ways: first, the success of MURA establishes the *possibility* of using deep learning to robustly analyse radiography. Rajpurkar et al. demonstrates near-radiologist levels of performance, validating the feasibility of our project in general. Secondly, the fact that the MURA model takes multi-view input imagery will help inform our own architectural design for processing both lateral and antero-posterior data. Finally, the limitations of MURA being a pure anomaly-detection system allows us to understand the need for a model which goes beyond binary classification, i.e. what our project is trying to accomplish.

To begin, it will be useful to look at MURA's architecture The MURA model is a 169-layer convolutional neural network which takes one or more views as input[5], and delivers a *probability of anomaly*. The final probability is the arithmetic mean of output probabilities from every view [10]. Every type of radiograph may have

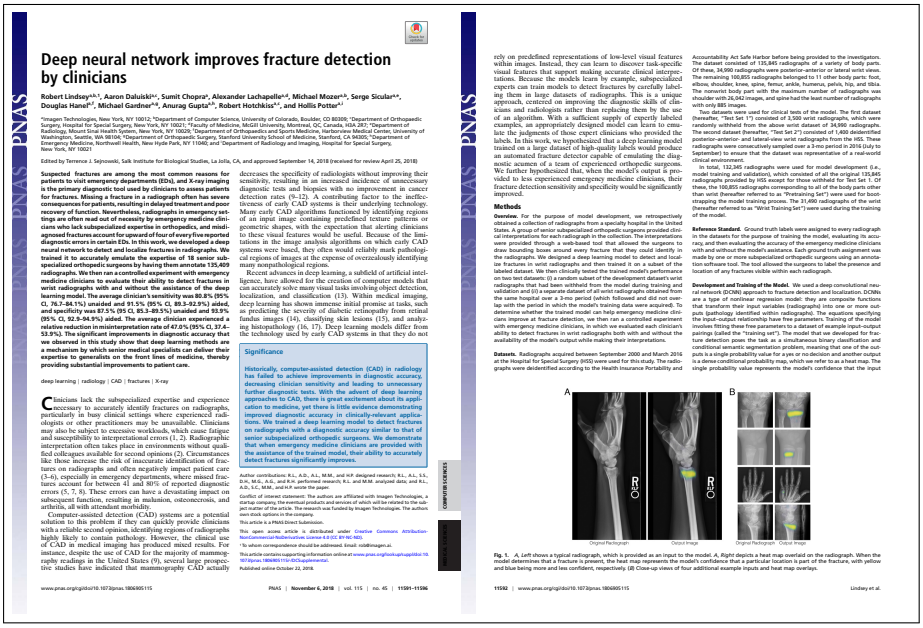
[5]Different views are radiographs of the same subject taken at different standard perspectives.

multiple standard views, and the model architect has a choice of either training a model on a single view, or combining information from views in some ensemble stage. For anomaly detection, MURA chose a fairly conservative approach of assessing a separate probability of anomaly for each view, and then finding their average. This approach will be similar to the one that our project must take: which is to look at both the lateral and anteroposterior view of the tibia.

Another aspect of MURA that is worth examining, is their evaluation process. The model was accessed in two different ways: first, the model's precision versus recall plotted out in a Receiver Operating Characteristic (ROC) plot, and then the Area-Under-Curve of the ROC (AUROC) was quantified. The AUROC is a common metric to assess diagnostic ability since it serves to quantify the precision-recall curve of the model. The MURA model had an AUROC of 0.929. Second, a panel of three radiologists were assembled to evaluate a set of radiographs, and their performance was compared against the model. This kind of competitive evaluation allowed the study to compare the model performance against human clinicians, yielding an informative baseline for the AUROC. With this information, we can contextualise the earlier AUROC value of 0.929, and see that the model performs slightly worse than humans.

The use of AUROC as an evaluative metric, as well as competitive evaluation with human clinicians, present an advancement in model evaluation compared with the two earlier studies. However, in practice any AI model in radiography will not aim to replace human radiologists entirely, but serve as an additional tool which augments a human clinician's own diagnostic ability. Thus, the MURA study's evaluation is not representative of what real-world deployment would look like. This is why we must turn to Lindsey et al's *Deep neural network improves fracture detection by clinicians*, for a more holistic example of model evaluation.

2.3 Lindsey et al: Refinements to Model Evaluation



Deep neural network improves fracture detection by clinicians

Figure 2.2: Thumbnail of article by Lindsey et al. [19]

In this study, a deep convolutional neural network is trained on a dataset of 31,490 wrist radiographs [19]. Like MURA, the study presents an AI-based fracture detection model, outputting both labels as well as a location heatmap. We include this study in the background research, for it's more holistic approach to evaluation, which simulates a real-world use-case. "Radiographic interpretation often takes place in environments without qualified colleagues available for second opinions" [19], the paper acknowledges, before proposing a model which aims to serve a possible 'second opinion.' After an initial evaluation which demonstrated an AUROC of 0.967 on certain datasets[6], Lindsey et al. proceeds to conduct a second experiment, where a group of clinicians were shown radiographs from the same test set, and tasked to evaluate the radiograph both with and without the model's assistance:

[6]For model testing, the study utilised two distinct categories of radiographs divided into 'Test Set 1' and 'Test Set 2' (a slight improvement over MURA), with the former containing a collection of singleton wrist radiographs, and the latter of wrist radiographs with anterio-posterior and lateral views.

> For each radiograph shown, the clinicians were asked whether or not a fracture was present. After a clinician made a response, the model's semantic segmentation prediction [i.e. heatmap] was shown overlaid on the radiograph; the model's clinical determination was shown as text, and the clinician was asked the same question again. [19]

This experiment simulates the workflow of a clinician when using a fracture-detection model as a part of a CAD workflow. By doing so, we evaluate the effectiveness of the model as an useful tool within a broader clinical practice. The study found that the "... sensitivity and specificity of the emergency medicine MDs were significantly improved with the assistance of the deep learning model." [19]

The key advantage of Lindsey et al's paper lies in its evaluative design, which our own project must take inspiration from. It may be impractical to setup a similar trial involving radiologists or clinical professionals,[7] but one feature that we do aim to replicate is the output of a heatmap which highlights the location of features that the model detects. For the Lindsey et al. and the MURA study, these heatmaps highlighted fracture-sights, whereas for our project they will highlight the sites of callus formation and bridging. By having this as a output, we will make the model's behaviour much more interpretable, and allow for integration into real clinical workflows.

[7] Such an addition to this project is, however, within the realm of possibility, in collaboration with the medical faculty at METRC.

2.4 Kim & MacKinnon: Cross-Domain Transfer Learning

Clinical Radiology 73 (2018) 439–445

Contents lists available at ScienceDirect

Clinical Radiology

journal homepage: www.clinicalradiologyonline.net

Artificial intelligence in fracture detection: transfer learning from deep convolutional neural networks

D.H. Kim*, T. MacKinnon

Medical Imaging Department, Royal Devon and Exeter Hospital, Barrack Road, Exeter EX2 5DW, UK

ARTICLE INFORMATION

Article history:
Received 28 May 2017
Accepted 14 November 2017

AIM: To identify the extent to which transfer learning from deep convolutional neural networks (CNNs), pre-trained on non-medical images, can be used for automated fracture detection on plain radiographs.

MATERIALS AND METHODS: The top layer of the Inception v3 network was re-trained using lateral wrist radiographs to produce a model for the classification of new studies as either "fracture" or "no fracture". The model was trained on a total of 11,112 images, after an eightfold data augmentation technique, from an initial set of 1,389 radiographs (695 "fracture" and 694 "no fracture"). The training data set was split 80: 10: 10 into training, validation, and test groups, respectively. An additional 100 wrist radiographs, comprising 50 "fracture" and 50 "no fracture" images, were used for final testing and statistical analysis.

RESULTS: The area under the receiver operator characteristic curve (AUC) for this test was 0.954. Setting the diagnostic cut-off at a threshold designed to maximise both sensitivity and specificity resulted in values of 0.9 and 0.88, respectively.

CONCLUSION: The AUC scores for this test were comparable to state-of-the-art providing proof of concept for transfer learning from CNNs in fracture detection on plain radiographs. This was achieved using only a moderate sample size. This technique is largely transferable, and therefore, has many potential applications in medical imaging, which may lead to significant improvements in workflow productivity and in clinical risk reduction.

© 2017 The Royal College of Radiologists. Published by Elsevier Ltd. All rights reserved.

Introduction

Demands on radiology services have increased dramatically in recent years causing a considerable strain on the workforce. In the UK, the number of computed tomography (CT) examinations increased by 29% between 2012 and 2015, whilst recruitment lagged behind, leaving 9% of consultant radiology posts vacant in 2015. Furthermore, in the same year nearly all radiology departments declared a failure to meet their reporting requirements.[1] In February 2016, there was an estimated backlog of 200,000 plain radiographs and 12,000 cross-sectional studies.[2] It is clear from these figures that improvements in reporting efficiency and workflow management are desperately needed if we are to avoid patient harm from delayed or missed diagnosis.

Artificial intelligence (AI) has the potential to address these issues. The wide adoption of electronic picture archiving and communications systems (PACS) has resulted in the development of one of the largest image data sets in existence. In the UK, there were 41 million imaging studies performed in 2016 alone.[3] These data lend themselves perfectly to machine learning.

Machine learning is a form of AI, which uses algorithms that iteratively improve, or learn, in response to training data in order to make autonomous predictions.[4] Supervised machine learning is one subtype, which relies on the provision of pre-labelled training data. In the field of general imaging and computer vision, deep learning is the leading machine learning tool.[5] Deep learning refers to techniques that build on developments in artificial neural networks in which multiple network layers are added to increase the levels of abstraction and performance.[6]

A simplified overview of the mechanics behind deep convolutional neural networks (CNNs) is provided in Fig 1. Neural networks aim to mimic the structure of the human brain. They use a series of interconnected "neurons", which collect, summarise, and/or transform data before passing the values to the next neuron in the sequence. This process culminates in an output layer, which can be used to formulate a prediction.

Constructing and training an effective neural network from scratch requires huge amounts of data. State-of-the-art image classification networks are frequently trained on data sets containing millions of images, facilitated by multiple computer servers, running continuously for several weeks.[7] This is not feasible for the majority of medical researchers. One method for overcoming this is to use a process called transfer learning. This is the process of adopting powerful and highly refined features from large existing pre-trained CNNs and using these as a starting point in training a new model for a different task. These low-level features, such as those illustrated in Fig 2, can be thought of as the basic building blocks of images such as lines and curves, and have been shown to be applicable to many different image-recognition tasks.[8] This technique can vastly reduce the computational requirements needed for network training and can deliver substantial performance benefits compared with training a CNN from scratch.[9]

Transfer learning has been relatively underutilised in the clinical setting despite the availability of vast amounts of image data. Two recent publications have demonstrated that transfer learning from pre-trained CNNs can produce human-level diagnostic results in the categorisation of skin lesions and in defining disease on digital retinal images.[10,11] Similar levels of diagnostic accuracy have yet to be reported in the analysis of plain radiographs; however, a few promising studies have recently been published. For example, one study retrained the GoogLeNet CNN for the detection of pathology in plain frontal chest radiographs, resulting in an area under the curve of between 0.861 and 0.964 for different chest radiograph features.[12] A different study used transfer learning from a pre-trained ImageNet CNN for the automated categorisation of osteoarthritis in knee radiographs.[13]

Plain radiographs are the most common radiological test with over 22 million studies performed in the UK in 2016.[2] A high proportion of these are plain extremity radiographs in the context of trauma. There is a strong case for developing automated strategies to improve efficiency and workflow management in this area considering the backlog in unreported studies and the fact that over £88 million was spent on outsourcing radiology reports in 2014–2015.[1] It is surprising therefore that, to the author's knowledge, there are currently no studies in the literature that have successfully applied transfer learning from pre-trained CNNs to the problem of fracture detection on plain radiographs. This proof of concept study aims to establish to what extent this is possible.

Materials and methods

This National Health Service (NHS)-based study was granted approval by the Health Research Authority in England. This study was retrospective and used only anonymised data, and therefore, ethics approval was not required.

Imaging study selection

Anonymised lateral wrist radiographs were obtained from the Royal Devon & Exeter Hospital for studies performed between January 2015 and January 2016. Radiographs were excluded if there was a plaster cast in place, if the growth plates of the wrist had not yet fused, or if the study demonstrated any fracture other than a fracture of the distal radius or ulna. The images were classified as either "fracture" or "no fracture" on the basis of the radiological report. This classification was checked and verified by a radiology registrar competent in the reporting of plain radiographs and with 3 years radiology experience. Images were also excluded if the single lateral projection was inconclusive for the presence or absence of fracture.

Image pre-processing

This resulted in a preliminary data set of 695 wrist radiographs demonstrating a fracture and 694 wrist radiographs demonstrating no fracture. The images were converted to JPEG by a trained radiologist ensuring the most appropriate windowing was selected. Images were then pre-processed by removing additional annotations such as the commonly used "Red Spot" annotation applied by radiographers at the time of image acquisition in order to reduce the possibility of over-fitting. Over-fitting describes the process by which the learning algorithm learns features that are not truly representative of the image category in an attempt to reduce the error in the learning process.

A data augmentation technique was used to amplify the data. This involved making a number of non-exact copies, or transformations, of each image. This served to provide the CNN with more training examples by incorporating the salient features in multiple orientations. The aim was to better reflect the real-world population of wrist radiographs,

Figure 2.3: Thumbnail of article by Kim and MacKinnon. [20]

Studies [10, 19] both rely on large, labelled radiographic datasets. While such datasets exist for fracture *detection* and fracture *classification*, the classification of fracture healing by their RUST scores is without precedence in literature, and to our knowledge there are no openly available datasets with the aforementioned labels. For our project, we will be using radiographic data internally collected by the Major Extremity Trauma Research Consortium (METRC), a research unit at the Johns Hopkins Bloomberg School of Public Health. Although this dataset has the RUST-scores we require, the number of radiographs we have access to is limited. Hence our approach is to utilise *transfer learning*, a process where the model is first trained on a larger, general-purpose dataset, before being fine-tuned on the study data. Thus, we turn to Kim and MacKinnon's *Artificial intelligence in fracture detection* for their innovative use of transfer learning. [20]

In Kim and MacKinnon's study, a dataset of 1,389 radiographs was available for a binary classification task (i.e. labels were "fracture" or "no fracture"). Instead of training a model *ab initio*, a CNN trained on a general purpose, non-radiographic dataset was re-applied on the radiography data. The authors used the Inception v3 network, an object-detection and image classification network [21] with the topmost layer fine-tuned on the radiographic data. [20]. By using a pre-existing model and

data augmentation[8], Kim and MacKinnon were able to achieve an AUROC of 0.954, a value that is within the same standard deviation of [10, 19], studies which both had much larger datasets for their models.

Our inclusion of Kim and MacKinnon's study serves as an important validation for the aims of our project. The METRC dataset will consist of around two to three thousand radiographs sourced from a handful of METRC studies. If Kim and MacKinnon's model is already able to achieve a robust performance with only 1,389 samples, than it is quite possible for our own project to achieve it's aims. The theoretical reasoning behind why transfer learning can achieve high levels of model performance is because the first layers of a CNN are primarily responsible for high-level feature extraction, e.g. edge detection, pattern recognition, with only the latter layers responsible for task-specific classification [20]. This does, however open up a further avenue of inquiry. Would model performance improve further, if the base model that the transfer learning is conducted from, was itself originally trained on a domain-specific dataset, like the MURA data from [10]? Our project will explore this possibility, through it's own application of transfer learning to the prediction of radiographic union with RUST scores.

[8]The authors of [20] ultimately generated 11,112 augmented samples from their initial set of 1,389 radiographs.

Chapter 3

Methodology

The primary objective of our study is to develop an AI model using the technique of transfer-learning, to automatically predict RUST scores from an input radiograph. In this chapter, we will first discuss our model design, contextualising the choices that we made in choosing our classifier, optimizer, and pre-training datasets.[1] Next, we will discuss the primary datasets used in the study, before proceeding to the define the three protocols that we will conduct, and the hyperparameter tuning regime. Afterwards, the evaluation and endpoints will be presented, as well as a discussion on patient privacy and ethical considerations.

3.1 Model Design

Transfer learning is a technique which uses a model trained upon a larger dataset first, before being applied to a smaller, task-specific dataset. Assuming our task-specific dataset (in our case, the RUST data from METRC) is fixed, the performance of our transfer-learning model is determined by the following factors:

1. The architecture of the original model.
2. The initial, large dataset that the original model is trained on.

Hence, in the context of our study design, we must first choose (or create) a model architecture, and select an initial large dataset to train our model on. Only then are we able to apply the transfer learning procedure (e.g. freeze model weights, add new classifier, fine tuning, etc) upon our task-specific dataset.

[1] i.e. the datasets that are used to train the base model, before transfer-learning.

3.1.1 Model Architecture Choices

The first decision that we must make in our experiment design is the choice of model architecture. According to a literature survey by Litjens, Kooi, Bejnordi et al., convolutional neural networks (CNNs) are the most common deep learning architecture deployed for medical image analysis, vastly outnumbering alternative methods such as Stacked Autoencoders (SAE), or Recurrent Neural Networks (RNNs) [22, p. 77]. This is expected, as CNNs exhibit strong performance with image classification tasks, and as our project is an image classification task (albeit with radiographs), we will be using a CNN.

The choice now remains to either design our own CNN from scratch, or use an existing CNN architecture. Although designing a CNN *ab initio* allows the possibility of further experimentation and potentially finer-grained control, such a *de novo* model is difficult to assess holistically: there would be no prior work in literature to serve as a basis for comparison. In contrast, if we use a well-documented, existing CNN as our transfer-learning foundation, we may compare the performance of our implementation against other instances of the same model architecture used for transfer-learning in different domains.

Thus, we will be using the InceptionV3 model, a convolutional neural network developed by Szegedy, Vanhoucke, Ioffe, et al. from Google [13]. Our choice of InceptionV3 is based upon a 2022 literature review of transfer-learning models for the domain of medical image analysis. According to Kora, Ooi, Faust et al., CNNs with a broad (as opposed to deep) network topology perform well in transfer-learning tasks [23], with models like InceptionV3 outperforming more parameterized models like AlexNet [24]. Indeed out of the 54 studies included in the review, Inception-style models both the most common (14 out of 54) and among the highest-performing. [23]

3.1.2 'Top-Classifier' Choices

Following the choice of our base model, we must then build a trainable classifier on top of the frozen base model layers. It will be the role of the classifier to transform the base model outputs into the 18-length one-hot encoded vector which will serve as our model's output. Top-classifiers are generally composed of a pooling layer, followed by layers of two or more densely interconnected neurons, with dropout. For our model, we have implemented the following classifier:

```
classifier: tf.keras.Sequential = tf.keras.Sequential([
    layers.GlobalMaxPooling2D(),
    layers.Dense(1024, activation='relu'),
    layers.Dropout(dropout_rate),
    layers.Dense( 512, activation='relu'),
    layers.Dropout(dropout_rate),
    layers.Dense( 256, activation='relu'),
    layers.Dropout(dropout_rate),
    layers.Dense(  18, activation='sigmoid')
])
```

Listing 1: Classifier Architecture (Github)

Our use of three `Dense` layers is fairly typical of classifiers built for transfer-learning applications. What does warrant a brief explanation, however — is the choice of a `GlobalMaxPooling2D()` layer as a means to flatten the two-dimensional output of the base model. Why MaxPooling, instead of AveragePooling? We use MaxPooling because it 'selects' the brightest pixels per block-size. This performs well for images that are primarily composed of light pixels on dark backgrounds, such as our radiographs.

3.1.3 Model Optimizer Choices

Finally, the last architectural decision we must make in our model design, is the choice of an optimizer. We have two possible approaches: we may either select an optimizer from *a priori* principles, or consider the selection of an optimizer to be a hyperparameter, and benchmark a variety of optimizers with our model on the data-set. Because we are already evaluating two variants of InceptionV3, the additional task of iterating through different optimizers will be infeasible given the time and compute constraints of the project.

Hence, our choice of an optimizer is determined by a review of available benchmarks and literature. The study and benchmarking of deep learning optimizers is a fairly recent field, initiated by the development of robust, reproducible benchmarks. Projects like Schneider, Balles, et Hennig's *DeepOBS: A Deep Learning Optimizer Benchmark Suite* allowed researchers to evaluate optimizers against an assay of realistic optimization problems, simulating common deep learning tasks and neural network architectures. [25] The availability of reproducible benchmarks allowed the first large-scale empirical experiments to be conducted on optimizer performance, culminating in Schmidt, Schneider, et Hennig's 2021 paper in optimizer benchmarking. [26] In an evaluation of 15 popular optimizers[2] across a total of more than 50,000 epochs of training, significant data on optimizer performance was gathered.

Of the eight optimization problem assays in the benchmark suite, our task of

[2] AMSBound, AMSGrad, AdaBelief, AdaBound, AdaDelta, Adam, LookaheadMomentum, LookaheadRAdam, Momentum, NAG, NAdam, RAdam, RMSProp, and SGD, respectively. The full list of results are available in their supplementary appendix.

radiographic image classification is bears closest resemblance to the CIFAR-10 benchmark: a CNN-based image classification task. According to the latest results available on the paper's website[3], the ADAM optimizer has a slight accuracy improvement over Momentum and SDG. However, the differences are so small that the author mentions:

> ... a practitioner with a new deep learning task can expect to do about equally well by taking almost *any method* from our benchmark and tuning it, as they would by investing the same computational resources into running a set of optimizers with their default settings and picking the winner. [26, p. 2]

Therefore, we will select the ADAM optimizer, and spend time on fine-tuning the model and hyperparameter choices, instead of devoting further resources to selecting an optimizer.

3.1.4 Pre-Training Dataset Choices

Now that we decided our model architecture, the next step is to select the initial, or pre-training dataset, that is used to train the base model.

ImageNet Dataset

Originally, InceptionV3 was trained on the ImageNet dataset: a general-purpose collection of more than 14 million everyday images. [14] The overwhelming size of the ImageNet dataset serves as a robust foundation for the InceptionV3 model, which exhibits strong performance in classification tasks upon it. However, the statistical characteristics of data in ImageNet is quite different from data in a typical radiography dataset. Hence, it remains a valid research question to ask whether a InceptionV3 model trained on a smaller, but more domain-specific dataset will exhibit better performance when applied to our transfer-learning task.

RadImageNet Dataset

Thus, we will also investigate RadImageNet, a collection of 5 million medical images composing of radiographic (CT), MRI, and ultrasound images. [15] As an open radiologic dataset developed specifically for transfer learning applications, a base model trained on RadImageNet may exhibit better performance on our radiography data, as the original network is trained upon images in a similar domain. However, the advantages of a similar domain is moderated by the corresponding smaller dataset size (5 million versus ImageNet's 14 million).

Therefore, in this project we will evaluate the use of a InceptionV3 model trained both on the ImageNet dataset, as well as the RadImageNet dataset as the base model for transfer learning. By comparing the performance of both models, we will be able to select the best-performing one for further development and refinement.

[3] https://deepobs.github.io/leaderboardP4.html

3.2 Dataset

Our transfer-learning models will be trained on a corpus of radiographic data sourced from a set of three METRC studies. [27, 28, 29] The data consists of pairs of radiographs (the anteroposterior view and the lateral view) along with a corresponding RUST score.

Dataset	Name	Samples
[27] RetroDEFECT	*Retrospective Study of the Treatment of Long Bone Defects*	1389
[28] Pain	*Pain Management & Long Term Outcomes Following High Energy Orthopedic Trauma*	1233
[29] Pacs	*Predicting Acute Compartment Syndrome using Optimized Clinical Assessment*	316
Sum		2938

Table 3.1: Sources of Radiographic Data with RUST labels in REDCap

The data is downloaded from the METRC Redcap database using the PyCap API library [30] and a set of Jupyter notebooks (see Github repository link in Appendix), and then processed into a Tensorflow `tf.data.Dataset` object.

3.2.1 One-Hot Encoding for Labels

RUST Scores will be one-hot encoded as 18-length label Tensors. Recall that our model must predict a RUST score from a pair of input radiographs. RUST scores are measures of orthopaedic union (i.e. healing), and ordinarily consist of two subscores for each view (the anterior-posterior and medial-lateral views), quantifying the development of bone calluses and bridging over the fracture line. The score components are as follows:

Radiographic Feature	Score
Fracture Line, No Callus	1
Fracture Line, Visible Callus	2
No Fracture Line, Bridging Callus	3
No Fracture Line, Remodeled.	4

Table 3.2: Radiographic Union Score for Tibial Fractures (RUST) Rubric

These score components[4] are then used to assess the features of a fracture from the

[4]Note that the original Whelan et al. paper [3] does not include a value for remodelled fractures, however as the METRC dataset includes this category, we will be using the modified RUST variant specific to Johns Hopkins for this study.

anterior, posterior, medial, and lateral cortices:

Subscore
Anterior Cortex
Posterior Cortex
Medial Cortex
Lateral Cortex

Table 3.3: RUST Scoring Instrument

Finally the resulting subscore components are summed in order to yield a RUST score for the fracture as a whole. For this study, our model is designed to predict every component subscore. Hence, the label will consist of a 18-value `tf.Tensor` with the shape (18,), consisting of 16 one-hot encoded values for the RUST subscores (four for each cortice), as well as two additional one-hot values to represent the view (anterioposterior or medial-lateral).

We explicitly choose to one-hot encode every component subscore of the RUST instrument in order to avoid bifurcating our already small dataset. Recall that it takes a pair of radiographs (one anteroposterior, one lateral) in order to score a fracture. If we chose to label each radiograph simply according to their component cortex sum (e.g. Anterior cortex score + Posterior cortex score), then we would have to develop two separate models, one for the anterioposterior views, and another for the lateral views. This would cut the effective size of our dataset in half, as there are two radiographs for every fracture.

3.2.2 Data Augmentation Strategy

Because we have a small dataset, a robust data augmentation strategy is needed to combat overfitting. In order to develop our data augmentation strategy, we followed suggestions from Bejani et Ghatee's *systematic review on overfitting control in shallow and deep neural networks* [31]. We may categorise data augmentation strategies to three broad families of methods: [32]

1. **Geometric Methods:** rotation, cropping, flipping.
2. **Photometric Methods:** noise, color-shifting, edge modification.
3. **'Complex' Methods:** generating artificial data using GANs, style transfer.

Of the three, 'complex' data augmentation methods such as generating artificial data points is inappropriate for our use case, as it will compromise the evidence-based labelling of our dataset. Hence, we may recourse to either geometric methods, or photometric methods. Originally, our intuition was that photometric data augmentation

methods would be of limited applicability to radiographs, which are entirely greyscale. Upon further investigation, it turns out that when evaluated individually, the most effective data augmentation strategy is cropping and rotation. [33] Our intuition was that cropping may not be appropriate, given that fracture-lines may lie on the peripheries of radiographs. Hence, we restrained ourselves to using only rotation, via random-flipping, as our method of data augmentation.

3.3 Evaluation

We will evaluate the performance of our models using a combination of K-fold cross-validation (for the initial evaluation and hyperparameter search), and a hold-out test set (for the final evaluation of the completed model).

3.3.1 Performance Metric: AUROC

The performance metric that we will be targetting is AUROC, the Area-Under-Curve of the Receiver Operating Characteristic (ROC) graph. This is the area underneath the precision-recall plot of the model, and is a common metric used to assess diagnostic accuracy. An AUROC of 0.50 indicates a diagnostic performance of no better than chance, while an AUROC of 1.00 indicates perfect performance.

3.3.2 K-Fold Cross-Validation

We will be using K-Fold Cross validation as a technique to get a more accurate assessment of our model performance. K-fold cross validation will be used during the initial evaluation of the different models (see section 3.4), as well during hyperparameter search. We will be using a k value of $k = 10$ during the initial evaluation, and then a smaller k value of $k = 6$ for each sample of the hyperparameter space during the hyperparameter search (see section 3.5)

K-Fold	Percentage (of 3000)	Samples
6	14%	420
10	8.5%	255

Table 3.4: Size of validation set, per k-fold value.

3.3.3 Hold-Out Test Set

A separate hold-out test set containing entirely unseen data will be reserved for a evaluation of the final model performance, after hyperparameter search. This hold-out test set will be kept strictly separate for final evaluation only, and will be used only once at the very end. With a dataset of $\approx 3,000$ samples, we will be setting aside 15% of the initial data (≈ 450 samples) for the hold-out test dataset.

3.4 Protocols

Now that we have finished discussing the model design, data, augmentation, and evaluation; it is time to define the study protocols. The experimental portion of this study will consist of three protocols:

1. Protocol I: Baseline: InceptionV3 end-to-end trained on RUST Dataset.
2. Protocol II: Transfer-learning w/ InceptionV3 trained on ImageNet.
3. Protocol III: Transfer-learning w/ InceptionV3 trained on RadImageNet.

All protocols will utilise the same data augmentation pipeline.

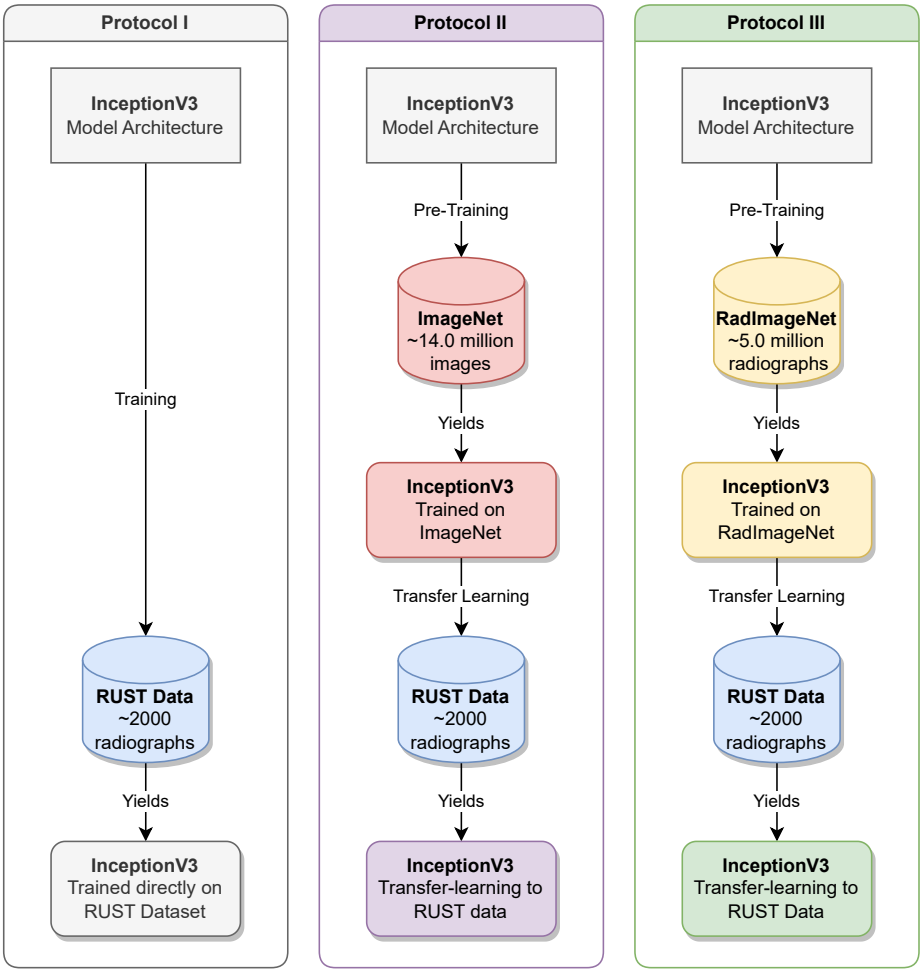


Figure 3.1: Overview illustrating the three model development protocols.

3.4.1 Protocol I: InceptionV3 End-to-End Trained

Protocol I is designed to help us establish a baseline level of performance that we will use to evaluate the feasibility of using transfer learning. Hence we will be using the same model architecture (InceptionV3) and classifier (see subsection 3.1.2), but instead of using transfer-learning, we will train our model directly on the RUST dataset. We may expect a model that has only marginal performance, as the full InceptionV3 model has 23.9M trainable parameters [13], which will almost certainly not converge on a dataset of only 3000 samples. The performance of the Protocol I model will be used as a baseline for our subsequent investigations in transfer learning.

The model will be evaluated using k-Fold cross-validation, with $k = 10$.

3.4.2 Protocol II: InceptionV3 with ImageNet

After establishing a baseline, we will move on to evaluating the use of ImageNet, or RadImageNet as our pre-training dataset for the transfer-learning model. Protocol II uses a base model (InceptionV3) with ImageNet weights. Here we use transfer-learning to train our classifier on the RUST dataset, and evaluate the AUROC of the resulting model using k-fold cross validation with $k = 10$.

3.4.3 Protocol III: InceptionV3 with RadImageNet

Finally, we instantiate a third model that uses the InceptionV3 architecture, except this time our pre-training dataset is RadImageNet. Here we use transfer-learning on the InceptionV3 model with RadImageNet weights, to train our classifier on the RUST dataset once more, and evaluate the AUROC of the resulting model using k-fold cross-validation with $k = 10$.

Once this is complete, we will choose the better-performing model out of Protocols II and III (Protocol I's performance will almost certainly be marginal at best), and select it for further hyperparameter optimisation. This will take place according to two steps defined in the following section.

3.5 Hyperparameter Search

Using whichever is the better performing model, we will now attempt to optimise it's performance further by searching for better hyperparameters. We have the following hyperparameters that are available for consideration: *dropout rate*, *batch size*, *learning rate*, and *epsilon*.[5] What is the best method for us to systematically find the best hyperparameters? A brief survey of machine learning literature informs us that a variety of methods are possible, including sophisticated approaches such as Bayesian optimization, and genetic algorithms. [34] However, due to time and resource limitations we will be implementing only random search and grid search. We will perform a

[5] A momentum factor similar to learning-rate decay specific to the Adam optimiser.

random search on the hyperparameter space defined by dropout and batch size, and then a grid search over a grid of different learning rates and epsilons.

This choice of performing hyperparameter search across two regimes is a practical compromise to the amount of compute resource that this project has. Although every hyperparameter influences every other hyperparameter, in practice it is rarely possible to conduct a hyperparameter search which varies over every hyperparameter under consideration at the same time. Instead, by constraining ourselves to two hyperparameters per search regime, we get to avoid the 'curse of dimensionality'[6]. Regime I is concerned with hyperparameters that are specific to the model, while Regime II is concerned with hyperparameters which are specific to the optimizer. Likewise, due to resource limitations, we will be performing k-fold cross validation for every search sample with a $k = 6$, instead of $k = 10$.

3.5.1 Hyperparameter Search Regime I

Regime I uses a stochastic search process where the hyperparameter hypothesis space is randomly sampled t times. We use random search for Regime I because random search performs better than grid search over large hypothesis spaces, where the process of randomly sampling the hypothesis space will yield better hyperparameters as t increases, in a manner similar to Monte Carlo methods in statistics. [35] We begin by defining the range of batch size from 16 to 2048, and the range of possible dropout rates from 0.00 to 0.50.

Given this domain of batch size versus dropout, we will randomly sample the hypothesis space 100 times, where each sample contains a batch size and dropout rate pairing. Using this sampled pair, we will conduct a k-fold cross-validation experiment with $k = 6$, where for every experiment the model will be trained for 20 epochs with said hyperparameters. The resulting average AUROC across the k-fold cross validation experiment will be used to assess the performance of said hyperparameters.

3.5.2 Hyperparameter Search Regime II

After concluding Regime I, we will use the best-performing dropout and batch size, and proceed on to Regime II. Here, we aim to find the best learning rate and epsilon ϵ (an Adam-specific momentum factor). Possible values for epsilon and learning rate span not merely a short interval (like those in Regime I), but across several orders of magnitude. Indeed, possible choices for learning rate can range from 10^{-4} to 10^{-1}, while possible choices for epsilon can range from 10^{-7} to 10^{-1}. Due to the sheer size of the search space, I judged random search to be infeasible. Instead, we will perform a grid search with a fixed set of learning rate and epsilon values, spanning each order of magnitude.

[6] The concept in which the size of a hypothesis space grows combinatorially ($n!$) with the number of free parameters, i.e. degrees of freedom, which are added.

Specifically, we will investigate the following learning rates: $10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}$ against the following epsilon values: $10^{-7}, 10^{-6}, 10^{-5}, \ldots, 10^{-1}$.

3.6 Endpoints and Final Evaluation

Once we have completed the Hyperparameter Search Regime II, we will have obtained the optimal set of batch size, dropout, learning rate, and epsilon for our model. Now will be the occasion for us to evaluate this final model against our hold-out test set (see subsection 3.3.3). To do this, we will first train a new instance of our model with said optimal hyperparameters, observing its validation AUROC. Next, we will evaluate the model against the hold-out test set, and note the test AUROC.

Once we have done so, we will be able to answer the following questions for our evaluation and conclusion:

1. *Does Transfer-Learning using the InceptionV3 model perform better than end-to-end training with the InceptionV3 model?*
2. *Does the Transfer-Learning Model with the RadImageNet weights perform better than the Transfer-Learning Model with the ImageNet Weights?*
3. *Is Transfer-Learning able to achieve a persuasive level of performance, as measured by it's AUROC?*

Protocol I and Protocol II will allow us to answer questions one and two. Once we have found the best-performing variant of InceptionV3, we will then begin the process of optimising model performance as measured through AUROC. The final model, as evaluated on the hold-out test set, will be used to answer question three. For the purpose of this study, we define a persuasive level of performance as a final AUROC that is greater than or equal to 0.80.

3.7 Ethical Considerations

Any research project or study that involves human subjects will necessitate ethical consideration. This section is only a brief discussion of ethical risks and mitigations. The aim of this project is to validate transfer learning as a technique to infer RUST scores from a small dataset of radiographs. Although we aim to achieve robust performance in order to demonstrate the feasibility of this technique as an avenue for further research, at no time do we imply to develop a *diagnostic* tool for clinical use. The overarching spirit of the study is to explore interdisciplinary applications of AI in medical imaging, in hopes of reducing clinical caseload for medical practitioners, leading to better standards of care for all patients overall. This study has passed Goldsmiths IRB review.

3.7.1 Human Subject Research, and HIPAA Compliance

This study works with radiographs collected from human subjects, that were a part of past or on-going METRC studies in high-energy trauma. This data is fully anonymised, and does not contain any personally identifiable information. As a part of Johns Hopkins Bloomberg School of Public Health's IRB requirements, all researchers working with human data, even anonymised data, must complete a Human Subject Research certification, and a Information Privacy Security certification. The author of this paper attests that they hold the aforementioned qualifications (see Appendix, Human Research Certification).

Chapter 4

Implementation and Analysis

In this chapter, we will present the implementation of the study methodology. Recall that the methodology has three components. We will begin with the establishment of an initial baseline, by creating and training a classical 'shallow' convolutional neural network based upon LeCun et al.'s 1998 LeNet model. [36] This classical CNN baseline will serve as the minimal performance standard that our model will aim to surpass. Next, utilising the InceptionV3 architecture which will serve as our transfer-learning base model, we will train an end-to-end (i.e. without transfer learning) model on our radiography dataset. This will serve as an additional baseline that will allow us to validate the transfer-learning *technique* against regular end-to-end training.

Following the establishment of these two baselines, we will proceed to begin an initial evaluation of two different transfer-learning base models. We will compare the performance of InceptionV3 trained with ImageNet weights [14], against InceptionV3 trained with RadImageNet [15] weights. This initial evaluation will help us explore whether a base model trained on the smaller, but domain-specific RadImageNet dataset will have any advantages over the larger, but general ImageNet dataset. We will select the better performing base model out of the two options, and proceed to optimize the model's hyperparameters.

Our model's hyperparameter search procedure consists of two steps, which we term hyperparameter search Regime I and hyperparameter search Regime II. As per our methodology, in Regime I we find the optimal batch size and dropout rate for our model. This is done using a stochastic search process where the hyperparameter space of the model is randomly sampled for t trials, where each trial consists of a k-fold cross-validation of the model with the selected hyperparameters. Once the optimal combination of batch size and dropout rate are found, we will set these hyperparameters

as constant and proceed to the second hyperparameter search regime. In Regime II we find the optimal learning rate and epsilon value ϵ for the Adam optimizer, by conducting a grid search over a selection of possible values.

4.1 K-Fold Evaluation

Before we begin, we must first implement our k-fold cross-validation routine. Since model performance is sensitive to the network's random weight initialisation[1] [37], our methodology requires k-fold cross-validation to be conducted on every experiment (i.e. model run). My implementation of the k-fold cross-validation process consists of two parts: a function which will divide the dataset into k folds, as well as a function that runs the k-fold cross-validation on the given model. The the `k_fold_dataset()` function is given as follows:[2]

```python
def k_fold_dataset(ds: tf.data.Dataset, k: int = 10) -> list[tuple[tf.data.Dataset,
↪  tf.data.Dataset]]:
    # First shard the given dataset into k individual folds.
    list_of_folds: list[tf.data.Dataset] = []
    for i in range(k):
        fold: tf.data.Dataset = ds.shard(num_shards=k, index=i)
        list_of_folds.append(fold)

    # Next, generate a list of train and validation dataset tuples
    list_of_ds_pairs: list[tuple[tf.data.Dataset, tf.data.Dataset]] = []
    for i, holdout_fold in enumerate(list_of_folds):
        ds_valid: tf.data.Dataset = holdout_fold

        # Select every fold except holdout_fold as the training folds
        training_folds: list[tf.data.Dataset] = list_of_folds[:i] +
        ↪  list_of_folds[i+1:]

        # ds_train size is (k-1)/k of the original dataset
        ds_train: tf.data.Dataset = training_folds[0]
        for fold in training_folds[1:]:
            ds_train = ds_train.concatenate(fold)

        ds_pair: tuple[tf.data.Dataset, tf.data.Dataset] = (ds_train, ds_valid)
        list_of_ds_pairs.append(ds_pair)

    return list_of_ds_pairs
```

Listing 2: Sharding dataset for K-Fold Cross Validation (Github)

One thing of note, is that our `k_fold_dataset()` function conducts all dataset-related operations using the Tensorflow's high-performance `tf.data.Dataset` API. This allows support for pre-fetch, caching, and other low-level optimisations. This function serves as a dependency which is called by `cross_validate()`, which runs the actual

[1] This is particularly true on small datasets with unbalanced classes like ours.

[2] The code listings provided in this document *are for illustration only.* The actual implementation is generally longer, and contains docstrings, debugging instrumentation, file I/O logic, as well as additional function arguments. Every listing will have a link to it's corresponding implementation in the git repository.

K-fold cross validation experiments on the given model:

```
def cross_validate(ModelClass: tf.keras.Model, ds: tf.data.Dataset, epochs: int = 50,
↪  batch_size: int = 128, k: int = 10) -> list[tf.keras.callbacks.History]:
    history_list: list[tf.keras.callbacks.History] = []
    train_valid_pairs: list[tf.data.Dataset] = k_fold_dataset(ds, k)

    for i, (ds_train, ds_valid) in enumerate(train_valid_pairs):
        # Reset tensorflow gradient tape
        tf.keras.backend.clear_session()
        model = ModelClass()
        model.compile(
            optimizer=tf.keras.optimizers.Adam(),
            loss=tf.keras.losses.BinaryCrossentropy(),
            metrics=metrics
        )
        history = model.fit(
            ds_train,
            validation_data=ds_valid,
            epochs=epochs,
            batch_size=batch_size,
        )
        history_list.append(history.history)

    return history_list
```

Listing 3: K-Fold Cross Validation (Github)

The output of every k-fold cross-validation experiment will be a 'history list' containing k `tf.keras.callbacks.History` objects. This `History` object will contain training and validation metrics which will be used to calculate the average metric over k folds:

```
def calculate_mean_metrics(kfold_metrics: list[dict[str, float]]) -> dict[str,
↪  list[float]]:
    # Initialise aggregate metrics with appropriate keys
    aggregate_metrics: dict[str, list[float]] = {}
    for fold in kfold_metrics:
        for metric in fold.keys():
            if metric not in aggregate_metrics:
                aggregate_metrics[metric] = []

    # Calculate the average metric per epoch for every fold
    number_of_folds: int = len(kfold_metrics)
    for metric in aggregate_metrics.keys():
        number_of_epochs: int = len(kfold_metrics[0][metric])
        for epoch in range(number_of_epochs):
            # A list of every value for that given metric in this epoch across folds
            values_per_epoch: list[float] = [x[metric][epoch] for x in kfold_metrics]
            mean_per_epoch  : float = sum(values_per_epoch) / number_of_folds
            aggregate_metrics[metric].append(mean_per_epoch)

    return aggregate_metrics
```

Listing 4: Calculating Mean Metrics from K-Fold Data (Github)

The above code now completes the prerequisites necessary for data gathering.

4.2 Establishing Baseline Performance Targets

In this section, we will establish the baseline performance targets for our transfer-learning model by training and developing two models which will represent alternative approaches to the problem of multilabel classification on a small dataset. The baseline models will be: a 'shallow' CNN following LeCun et al.'s classical 1998 LeNet architecture [36], and an InceptionV3 model that is directly end-to-end trained on our radiography dataset. We explicitly choose the above two models as our baseline for comparison, because they each help validate a different aspect of this project: whether a deep neural network is appropriate for the task in the first place, and whether the *technique* of transfer learning is appropriate for our dataset. The second question of whether or not our technique is necessary is why we train a version of our model's architecture directly on the radiography data, in order to obtain a performance measure of using the same model architecture *without* transfer learning. At minimum, our transfer-learning model must achieve a better performance (as measured by it's AUROC score) over the two baseline models.

The performance of the baseline models will be measured as the highest observed *average* AUROC, found using k-fold cross-validation with $k = 10$. The value of $k = 10$ is chosen because the resulting per-fold training and validation splits are no larger than a conventional train, test, and validation split of 70%, 15%, 15%, where:

- Training and Validation Set (`ds_train + ds_valid`): 2490 (85%):
  - K-Fold Cross-Validation, $K = 10$:
    - Training Set: 2241 (~76%)
    - Validation Set: 249 (~8.5% per fold)
- Hold-out Test Set (`ds_test`): 441 (15%)

Larger k values yield a more thorough measurement of a model's performance at the cost of additional computational costs, while lower k values risk lowering the training-validation split ratio until the training set is too small for adequate training. For this initial evaluation, as we wish to yield a benchmark for baseline performance, we will be using a k value of 10. For the hyperparameter search regime, we will be using $k = 6$ in order to lower computational costs.

4.2.1 Shallow Convolutional Neural Network

For the first benchmark, we begin by implementing the shallow convolutional neural network described by LeCun et al in [36] in Tensorflow. Our implementation follows the original paper, with a slight modification in the final classifier, in order to output the 18-vector one-hot encoded label predictions. Note the presence of only two convolutional layers — this is typical for early CNNs of that period.

```python
class LeNet1998(tf.keras.Model):
    def __init__(self, **kwargs):
        super().__init__(**kwargs)

        self.input_layer: tf.Tensor = layers.InputLayer(input_shape=(299, 299, 3))
        self.data_augmentation: tf.keras.Sequential = tf.keras.Sequential([
            layers.RandomFlip(seed=RNG_SEED),
        ])

        self.lenet1999: tf.keras.Model = tf.keras.Sequential([
            layers.Conv2D(6, kernel_size=5, strides=1,  activation='tanh',
            ↪  padding='same'),
            layers.AveragePooling2D(),
            layers.Conv2D(16, kernel_size=5, strides=1, activation='tanh',
            ↪  padding='valid'),
            layers.AveragePooling2D(),
        ])

        self.classifier: tf.keras.Sequential = tf.keras.Sequential([
            layers.Flatten(),
            layers.Dense(1024, activation='relu'),
            layers.Dense(18, activation='sigmoid')
        ])

        self.model: tf.keras.Sequential = tf.keras.Sequential([
                self.input_layer,
                self.data_augmentation,
                self.lenet1999,
                self.classifier
        ])

    def call(self, inputs):
        return self.model(inputs)
```

Listing 5: The LeNet 1998 Shallow CNN Model (Github)

We implement our version of the LeNet architecture by subclassing `tf.keras.Model` class, which is then passed on to our `cross_validate()` function to be evaluated. This entire experiment is conducted within a Jupyter notebook which is made available as a self-contained, reproducible unit within the project repository (Github). Running the experiment yields our first AUROC to performance graph:

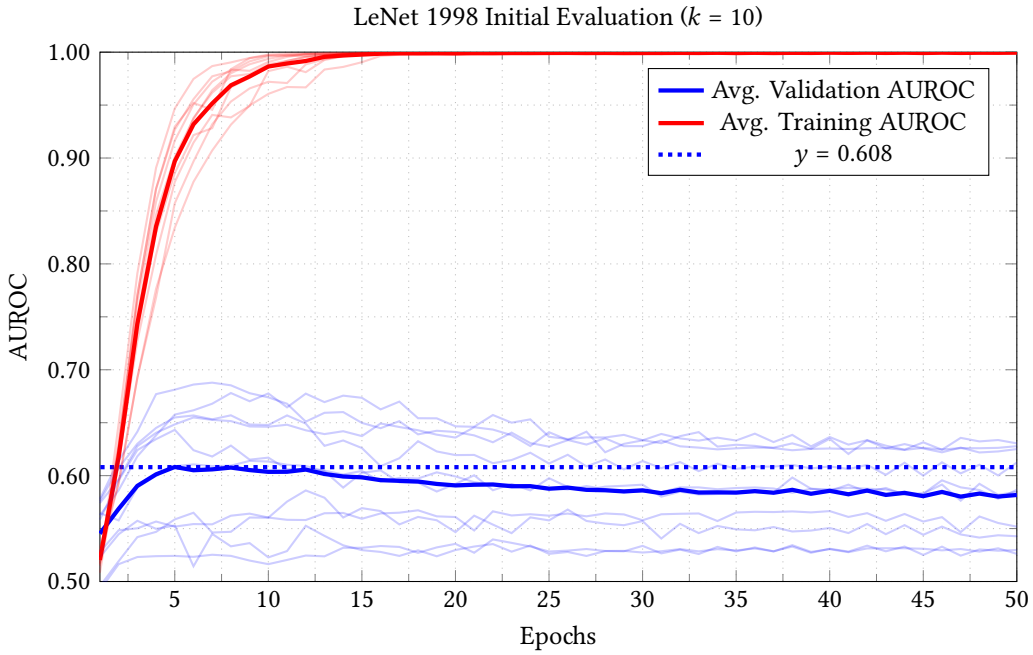


Figure 4.1: Baseline shallow CNN based on the LeNet 1998 architecture

The bold lines in the chart represent the *average* training (red) and validation (blue) AUROC, as measured after performing k-fold cross-validation on 10 folds (k = 10). The transparent lines indicate the observed training and validation AUROC per each individual fold: this per-fold performance has been charted in order for us to better observe the consistency of model performance per epoch. Variations in performance per fold is due to a combination of different random starting conditions (due to random weight initialisation at the start of a model's training), as well as variances in floating-point calculations.

What information does our data for the LeNet model tell us? First, we can observe severe overfitting: by epoch 10, performance on the training set asymptomatically approaches 1.0 (as quantified by AUROC). However, the validation performance remains minimal: generally averaging around 0.60, with certain instances of the model performing little better than chance (0.50). This information indicates that a classical 'shallow' CNN lacks the representational power to extract the features necessary to perform classification on our dataset. Indeed, it appears that the LeNet model fails to converge at all. This is to be expected: and our experiment yields a minimal baseline AUROC value of 0.608 that our subsequent models must beat. Likewise, by demonstrating that classical 'shallow' CNNs are unable to solve our problem, we make the

case for using a 'deep' neural network: in the form of the InceptionV3 architecture, which we will explore in the following section.

4.2.2 End-to-End Training with InceptionV3

Having validated the necessity of using a deep convolutional neural network to solve this *multiclass*, *multilabel* classification task, our next question would be: "is it necessary to use the technique of *transfer-learning* on our dataset, or would a regular end-to-end training process suffice?" Although the small size of our dataset indicates that transfer learning is appropriate, it is important for us to validate our assumptions through empirical data. Hence, we arrive at the establishment of the second baseline model: end-to-end training InceptionV3 on our dataset. Let us start by defining our base model:

```
class TransferLearningModel(tf.keras.Model):
    def __init__(self, dropout_rate: float, **kwargs):
        super().__init__(**kwargs)

        self.input_layer: tf.Tensor = layers.InputLayer(input_shape=(299, 299, 3))
        self.data_augmentation: tf.keras.Sequential = tf.keras.Sequential([
            layers.RandomFlip(seed=RNG_SEED),
        ])

        self.inceptionv3: tf.keras.Model = tf.keras.applications.InceptionV3(
            include_top=False,
            weights='imagenet'
        )
        self.inceptionv3.trainable = False

        self.classifier: tf.keras.Sequential = tf.keras.Sequential([
            layers.GlobalMaxPooling2D(),
            layers.Dense(1024, activation='relu'),
            layers.Dropout(dropout_rate),
            layers.Dense( 512, activation='relu'),
            layers.Dropout(dropout_rate),
            layers.Dense( 256, activation='relu'),
            layers.Dropout(dropout_rate),
            layers.Dense(  18, activation='sigmoid')
        ])

        self.model: tf.keras.Sequential = tf.keras.Sequential([
            self.input_layer,
            self.data_augmentation,
            self.inceptionv3,
            self.classifier
        ])

    def call(self, inputs):
        return self.model(inputs)
```

Listing 6: Model Class for InceptionV3 (Github)

We define a `class TransferLearningModel` which will be instantiated by every k-fold validation trial. Note that for this particular experiment, as we are establishing an

end-to-end trained baseline, we will be setting `self.inceptionv3(weights=None)` and the attribute `self.inceptionv3.trainable = True`. Naturally in the actual implementation (Github) this is done through an argument in the class constructor, however the listing is simplified for the purpose of size and readability. So what happens now when we run the kfold experiment (Github)?

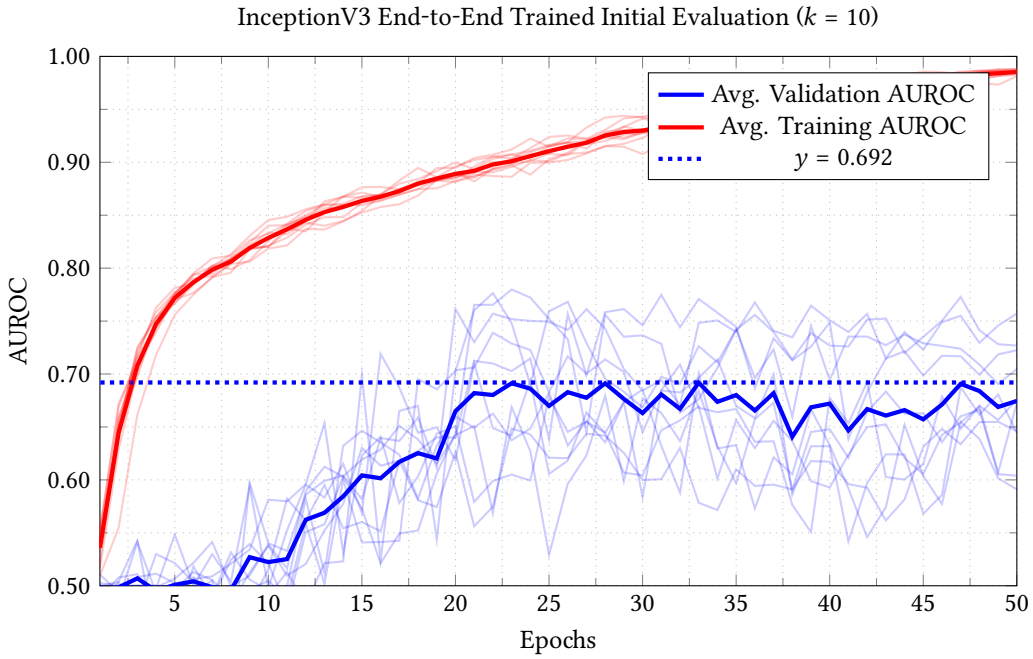


Figure 4.2: InceptionV3 Model Trained on Study Data.

We can observe that the end-to-end variant performs marginally: achieving an average AUROC of 0.692. However, upon a closer examination it is clear that the validation AUROC of each individual k-fold trial is highly erratic. The large spikes in validation AUROC indicates a failure to converge, as the dataset is too small for the number of tunable weights in the model. The InceptionV3 model has 189 layers, with a combined total of 23.9 million trainable weights: representing a parameter space several orders of magnitude larger than our dataset. The highly erratic validation AUROC is only a symptom of the model's inability to converge, and demonstrates clearly that regular end-to-end training is insufficient and inappropriate for our dataset.

4.2.3 Baseline Metrics

Having completed assessing our two baseline models, we are left with the following metrics that will help us in our own evaluation:

Baseline	Validation AUROC
Random (no better than chance)	0.50
Classical Shallow CNN (LeNet)	0.61
End-to-End Model (InceptionV3)	0.69

Table 4.1: Baseline Benchmarks

The first level of performance that our subsequent models are expected to achieve is a validation AUROC > 0.50. As a measurement of classifier performance, an AUROC of 0.50 indicates performance no better than chance (i.e. the same as choosing by random). If we are unable to meet the minimum baseline of > 0.50, then our entire approach may be unrealistic and infeasible. The second baseline that we must achieve is a performance of > 0.61. For that is the best performance measured from a classical 'shallow' CNN. As deep neural networks, with their dozens (if not hundreds) of layers incur a computational cost that is an order of magnitude above classical 'shallow' CNNs, if our model is not able to exceed the performance of a regular CNN, it will be better to develop a regular CNN instead. Finally, the last baseline that we established allows us to validate the suitability of the transfer-learning technique. If our model is unable to meet an AUROC of > 0.69, then we will be better served to train our model architecture directly on our dataset.

With this information in hand, it is time for us to embark on the second part of our study: developing a transfer-learning model to infer the RUST score of radiographs of long-bone fractures. In the transfer learning technique, a base model is trained on a larger dataset, before having it's weights frozen, and then used as a component of a classifier trained on the task-specific dataset. A key decision in this process is the choice of the larger dataset that the base model will be trained on. Thus, we are lead to the second part of our study: evaluating the performance of InceptionV3 trained on ImageNet, and RadImageNet.

4.3 InceptionV3 with Transfer Learning

Let us begin by evaluating the performance of the RadImageNet weights. Following the same procedure as we did earlier, we instantiate a InceptionV3 model with our classifier, and set the attribute `self.inceptionv3(weights='radimagenet.h5')` and `self.inceptionv3.trainable = False`. This class instance now has InceptionV3 with pre-trained ImageNet weights, which we have frozen. Now when we run the training process, only our classifier will be trained. We conduct the experiment in the following Jupyter notebook (Github).

4.3.1 Base Model Trained on RadImageNet Dataset

The model trained on RadImageNet weights achieves an average validation AUROC of 0.706. Note how unlike the version of InceptionV3 that was end-to-end trained, the per-fold validation AUROC is fairly consistent: we do not see any large spikes in validation performance. Likewise, although the model begins to exhibit overfitting after epoch 15, the degree of overfitting is relatively well controlled.

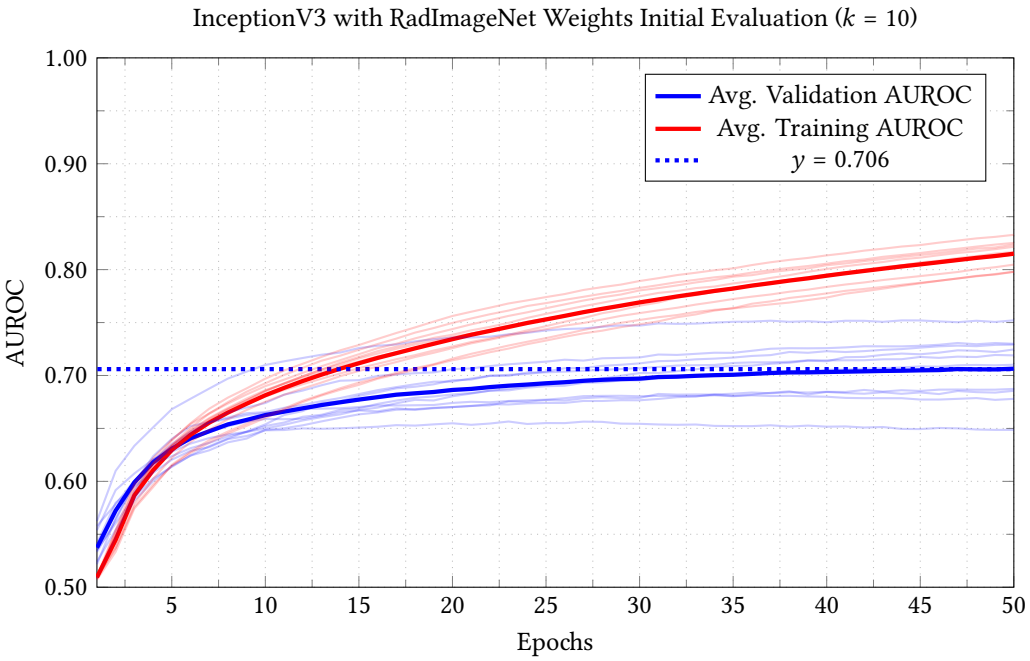


Figure 4.3: InceptionV3 with RadImageNet Weights

This preliminary information helps inform us that the technique of transfer learning is appropriate for our use case and dataset. All that follows now is for us to evaluate the ImageNet weights, and compare their performances together.

4.3.2 Base Model Trained on InceptionV3 Dataset

We now conduct the same experiment in a separate Jupyter notebook with ImageNet weights (Github). Recall that the difference between RadImagenet and ImageNet is that former contains approximately 4.1 million images [15], while the latter contains around 15.0 million [14]. While the RadImageNet dataset is exclusively sourced from medical imagery, including radiographs — the vastly larger ImageNet dataset has the potential to perform better, simply because the model was trained on a larger dataset. Does this assumption hold true?

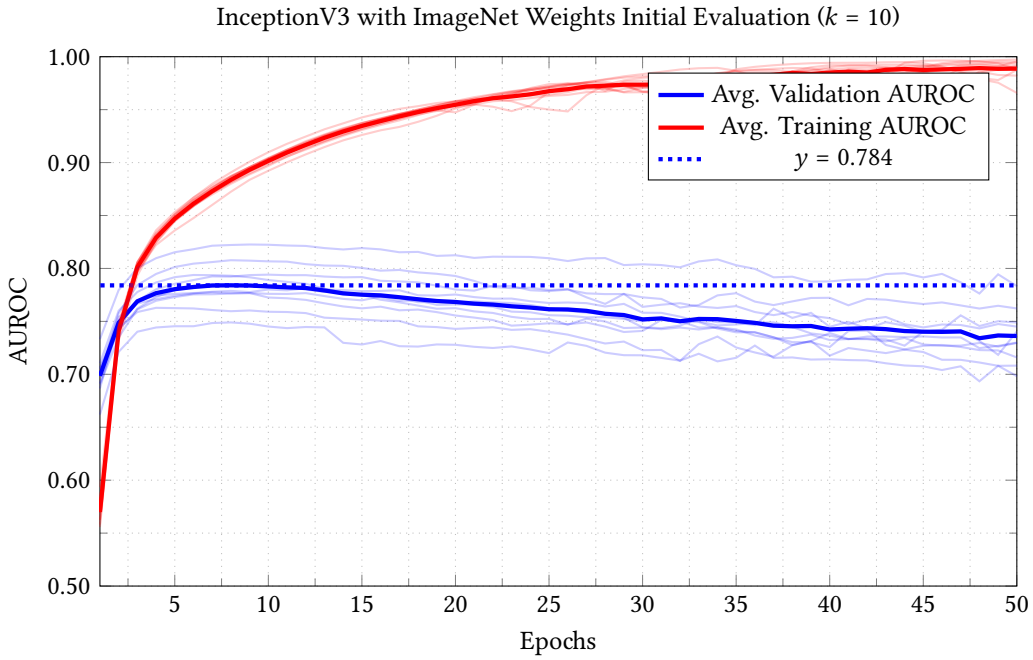


Figure 4.4: InceptionV3 with ImageNet Weights

Our data indicates that the InceptionV3 weights yield a higher validation AUROC of 0.784, in comparison to the RadImageNet model with an AUROC of 0.706. This means that although the RadImageNet dataset was more domain-specific to our needs (i.e. medical radiography classification), it appears the sheer size of the ImageNet datset yielded weights which performed better.

4.3.3 Comparison between RadImageNet and ImageNet

However, despite this difference in 'naive' (i.e. untuned, without hyperparameter optimisation), the performance characteristics of both models are quite different. Observe how the overfitting profile of the RadImageNet model is less severe than that of ImageNet, despite achieving a lower validation AUROC overall. Likewise, while the validation AUROC of the ImageNet model begins to decrease past epoch 15 due to overfitting, the validation AUROC of RadImageNet is still growing by epoch 50. Although our methodology calls for us to select the better performing model out of both of them, the information shown here offers room for further investigation — which can be the subject of a future study.

At this point, having assessed the 'naive' performance of both the RadImageNet weights and the regular ImageNet weights, we will select the better-performing

ImageNet weights as the basis for our transfer learning model. Going forward, we are now ready to tackle our problem directly: and begin the process of hyperparameter search.

4.4 Hyperparameter Search

Recall that our methodology calls for a two-part hyperparameter search (see section 3.5), consisting of Hyperparameter Search Regime I, and Hyperparameter Search Regime II. Regime I is concerned with finding the best combination of batch size and dropout, while Regime II is for the best learning rate and epsilon ϵ for the Adam optimizer.

4.4.1 Hyperparameter Search Regime I

To begin, let us construct a search function which conducts t trials, where during each trial a random portion of the hypothesis space is sampled, and then evaluated using k-fold validation with $k = 6$. Recall that our methodology specifies the use of a slightly lower k value as a concession to the amount of compute resources available. Given 20 epochs of training per k-fold, and 6 folds per trial, this hyperparameter search regime conducts a total of 12,000 epochs of training over the course of 2 days (Github).

```
def hyperparameter_search(trials: int, kfolds: int = 6, epochs: int = 20) ->
↪  list[dict[str, Union[int, float, list[tf.keras.callbacks.History]]]]:
    search_results: list[dict[str, any]] = []

    for trial in range(trials):
        # Randomly pick hyperparameter options
        rng = np.random.default_rng()
        batch_size  : int   = rng.integers(16, 2048, endpoint=True)
        dropout_rate: float = rng.uniform(0.0, 0.5)

        # Conduct K-Fold cross-validation with given hyperparameters
        results: list[tf.keras.callbacks.History] = cross_validate(
            TransferLearningModel,
            ds_train_and_valid,
            k=kfolds
            epochs=epochs,
            batch_size=batch_size,
            model_kwargs={"dropout_rate": dropout_rate},
        )

        search_results.append({
            "batch_size"  : batch_size,
            "dropout_rate": dropout_rate,
            "history_list": k_fold_results
        })

    return search_results
```

Listing 7: Hyperparameter Search Regime I (Github)

The results of this hyperparameter search are then plotted, with the maximum observed validation AUROC represented as the colour of the data point in the following scatter

plot. Note that the color-map is scaled according to the highest and lowest observed average validation AUROC.

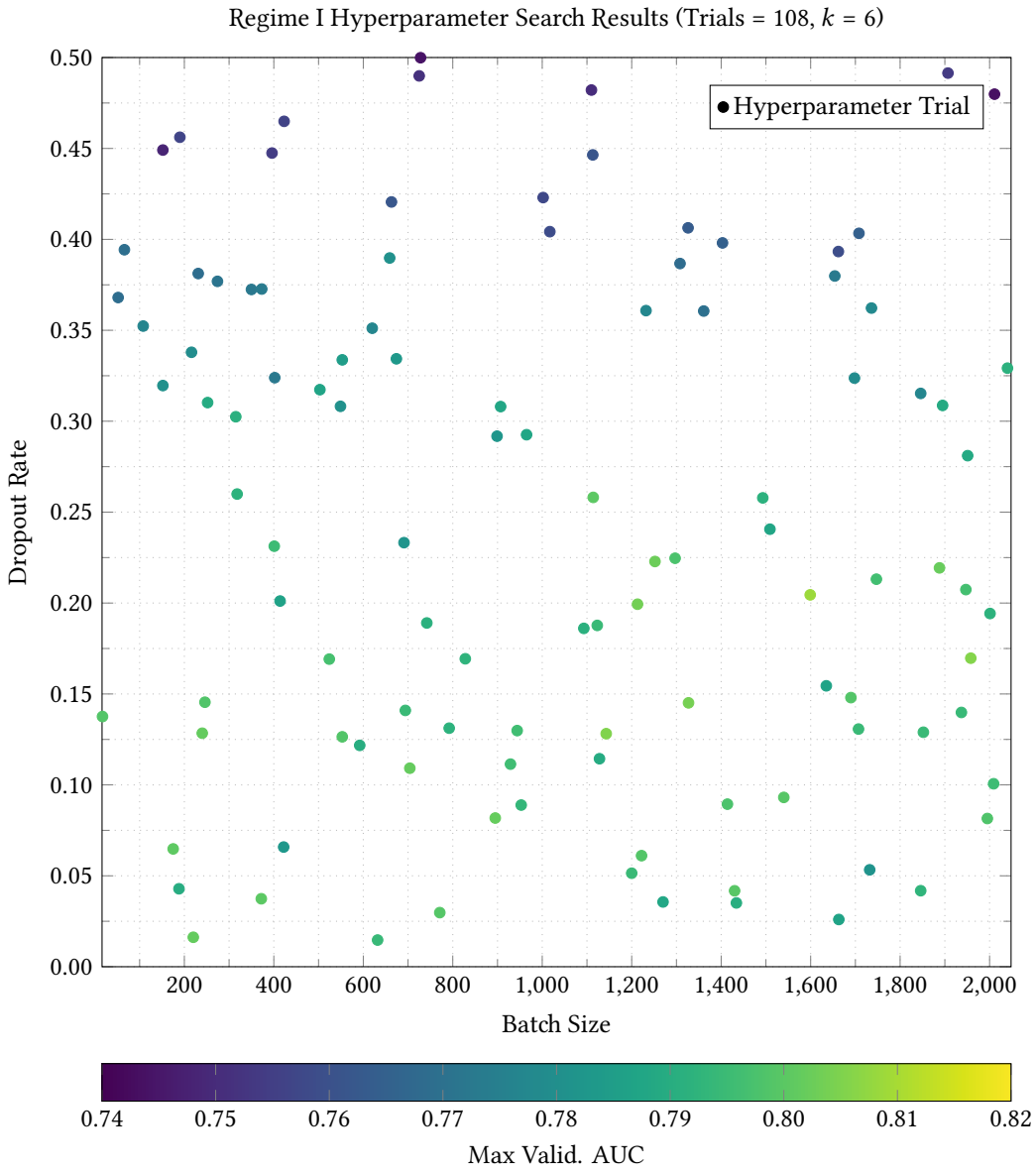


Figure 4.5: Results for the Hyperparameter Search Regime I

What were some of the learning rate and batch size combinations that we found? The following table lists out the top ten hyperparameter options.

Batch Size	Dropout Rate	Max Validation AUROC
1599	0.20450534699237183	0.8082009057203928
1958	0.16968093634911133	0.8052726884682974
1143	0.12816312484235065	0.8047231038411459
1327	0.14511673948299006	0.8034322460492452
1213	0.19936088970848825	0.8029904663562775
1252	0.22284567287235169	0.8021498719851176
1888	0.21933964138972928	0.8016549249490103
895	0.08177228898982053	0.8012685179710388
2009	0.10061578442603508	0.7949380973974863
401	0.23126040688192015	0.794610470533371

Table 4.2: Top Ten Hyperparameter Options for Regime I

As we can from a glance, a reasonable batch size seems to hover between 1500 to 2000. Likewise, a good dropout value seems to be around 0.20. This corresponds the colour gradient that we can see in the scatter plot. As an additional visualisation, we will also graph a random assortment of nine hyperparameter choices from our hypothesis space sample:

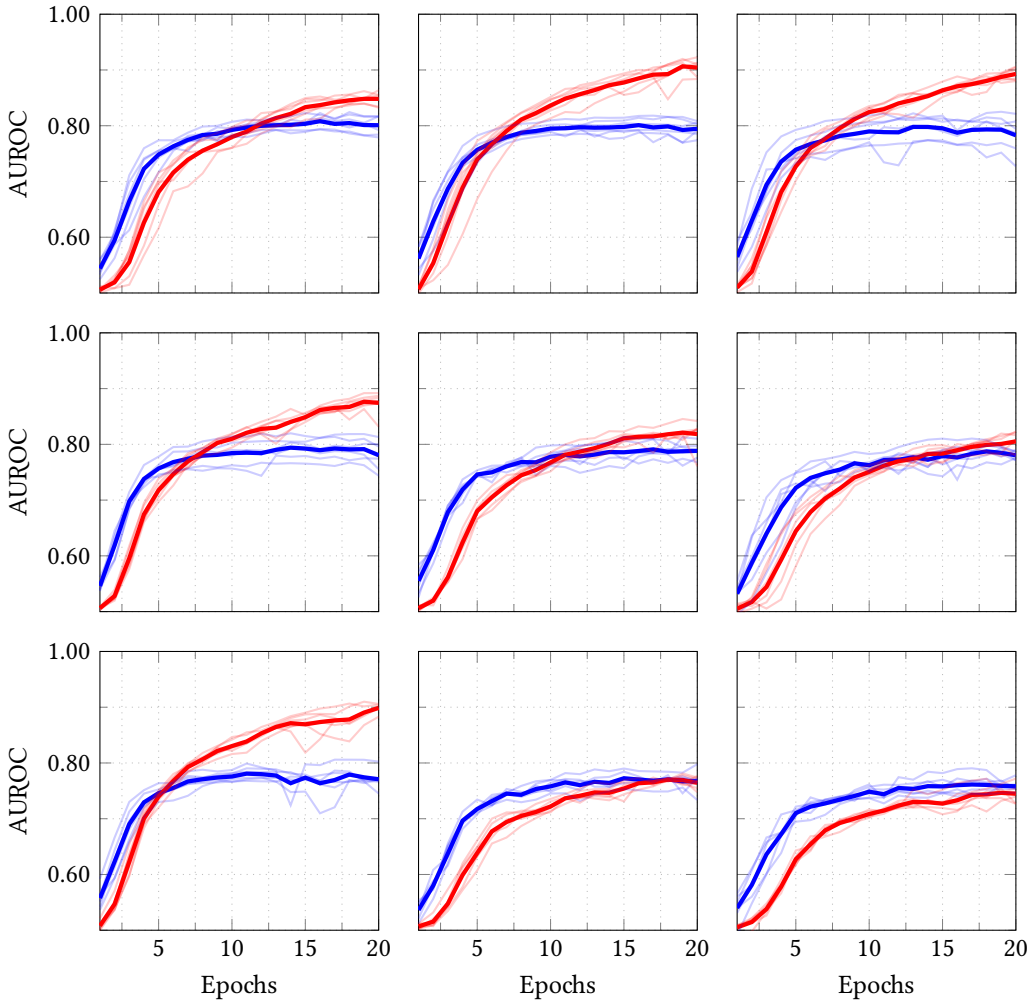


Figure 4.6: Random examples of models from hyperparameter search regime I.

The graphs above represent a selection of nine batch size and dropout combinations, sorted by best-performing to worse-performing. The first graph represents the top-performing hyperparameter combination, which we will graphs in greater detail below:

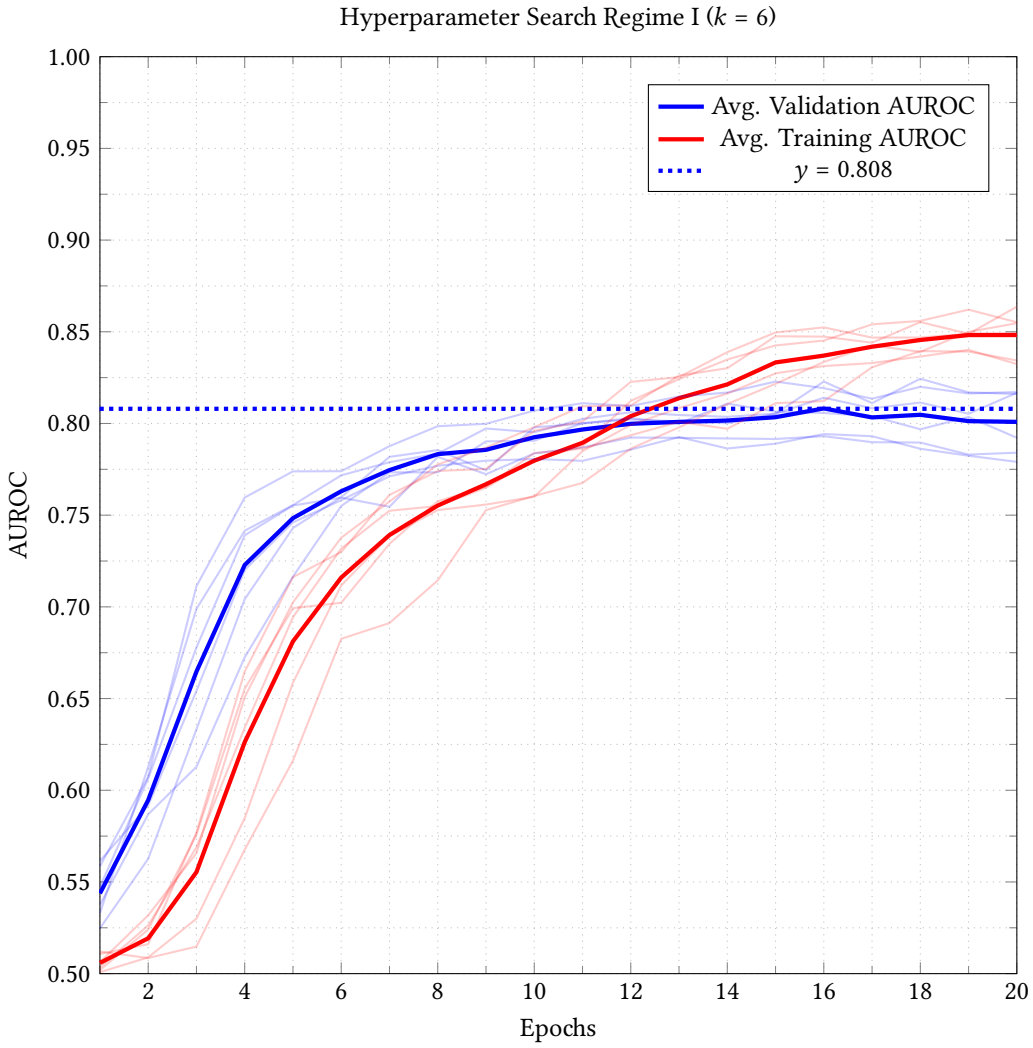


Figure 4.7: Best performing model in Regime I

Thus, we complete the first hyperparameter search regime, obtaining the hyperparameter options of 1599 for batch size and 0.205 for dropout. Going forward in all subsequent models, we will use those values, after rounding them up slightly to the nearest whole figure: `batch_size: int = 1600, dropout_rate: float = 0.20`. With this information, we may now move on to the second hyperparameter search regime, where we find the best hyperparameters for the Adam optimizer.

4.4.2 Hyperparameter Search Regime II

Recall that for Regime II, we must find the best learning rate and epsilon factor for the Adam optimizer. We begin by implementing a grid search function, as seen below. We sample the grid at discrete intervals for both learning rate and epsilon:

```python
def learning_rate_gridsearch(kfolds: int = 6) -> list[dict[str, Union[int, float,
↪  list[tf.keras.callbacks.History]]]]:
    # Grid i: 1.0 × 10^-1 ≤ learning_rate ≤ 1.0 × 10^-4
    learning_rates: list = [1 * np.float_power(10, -exp) for exp in range(1, 5)]
    # Grid j: 1.0 × 10^-1 ≤ epsilon_rate ≤ 1.0 × 10^-8
    epsilon_rates : list = [1 * np.float_power(10, -exp) for exp in range(1, 9)]

    search_results: list[dict[str, Union[int, float,
    ↪  list[tf.keras.callbacks.History]]]] = []
    for i, learning_rate in enumerate(learning_rates):
        for j, epsilon_rate in enumerate(epsilon_rates):
            # Conduct K-Fold Experiment
            k_fold_results: list[tf.keras.callbacks.History] = cross_validate(
                TransferLearningModel,
                ds_train_and_valid,
                k=kfolds,
                epochs=EPOCHS,
                batch_size=BATCH_SIZE,
                model_kwargs={"dropout_rate": DROPOUT_RATE}
                optimizer_kwargs={"learning_rate": learning_rate, "epsilon":
                ↪  epsilon_rate},
            )
            search_results.append({
                "learning_rate": learning_rate,
                "epsilon_rate" : epsilon_rate,
                "history_list" : k_fold_results
            })

    return search_results
```

Listing 8: Hyperparameter Search Regime II (Github)

Running the above search routine in a Jupyter notebook (Github) on our hosted compute provider, we obtain the following results:

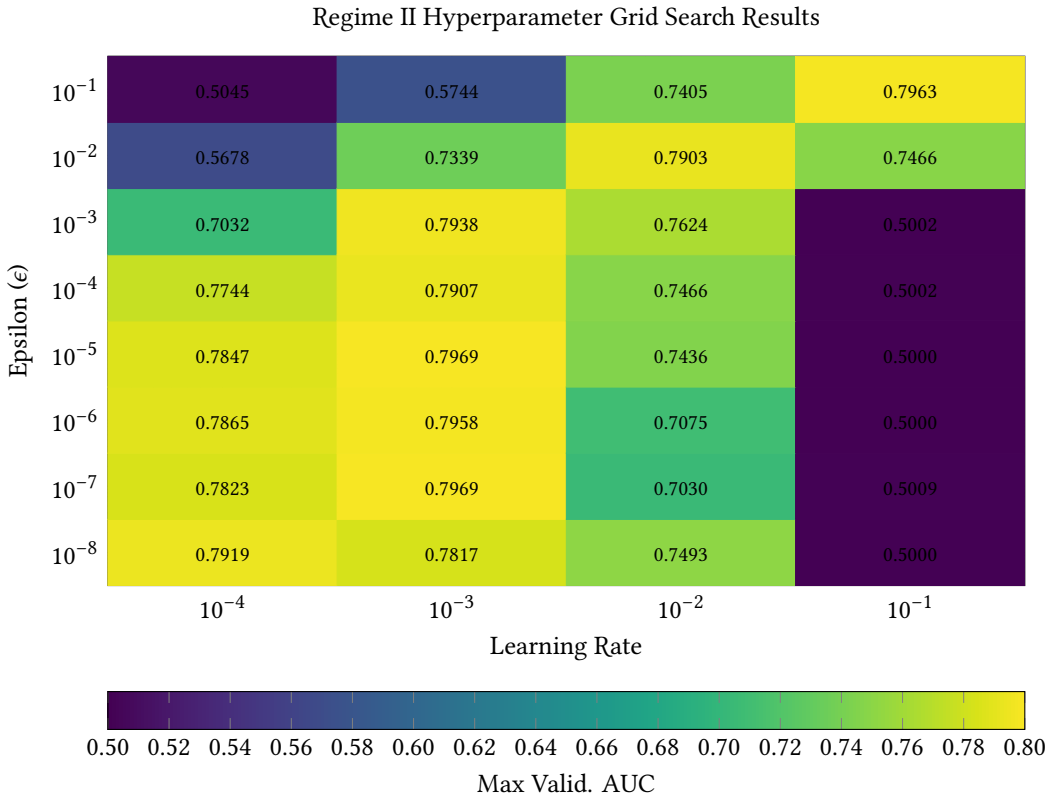


Figure 4.8: Results for the Hyperparameter Search Regime II

Learning Rate	Epsilon ϵ	Max Validation AUROC
1.0×10^{-3}	1.0×10^{-7}	0.7969321310520172
1.0×10^{-3}	1.0×10^{-5}	0.7969251374403635
1.0×10^{-1}	1.0×10^{-1}	0.7962859372297922
1.0×10^{-3}	1.0×10^{-6}	0.7957556545734406
1.0×10^{-3}	1.0×10^{-3}	0.7938481668631235
1.0×10^{-4}	1.0×10^{-8}	0.7918900847434998
1.0×10^{-3}	1.0×10^{-4}	0.7907490432262421
1.0×10^{-2}	1.0×10^{-2}	0.7903381884098053
1.0×10^{-4}	1.0×10^{-6}	0.7865246037642161
1.0×10^{-4}	1.0×10^{-5}	0.7846748034159342

Table 4.3: Top Ten Hyperparameter Options for Regime II

4.4.3 Final Hyperparameters

Thus, we have completed our hyperparameter search. After implementing and running Regime I and Regime II (Github), we were able to find the following hyperparameter options for our final model:

Batch Size	Dropout Rate	Learning Rate	Epsilon (ϵ)
1600	0.20	1.0×10^{-3}	1.0×10^{-7}

Table 4.4: Final Hyperparameters for InceptionV3 with ImageNet Weights

The implementation, analysis, and search of our model hyperparameters are now complete. Now we are ready for the final evaluation of the complete model on our hold-out test set. This will be done in the final part of our project, in the Part IV evaluation.

Chapter 5

Evaluation

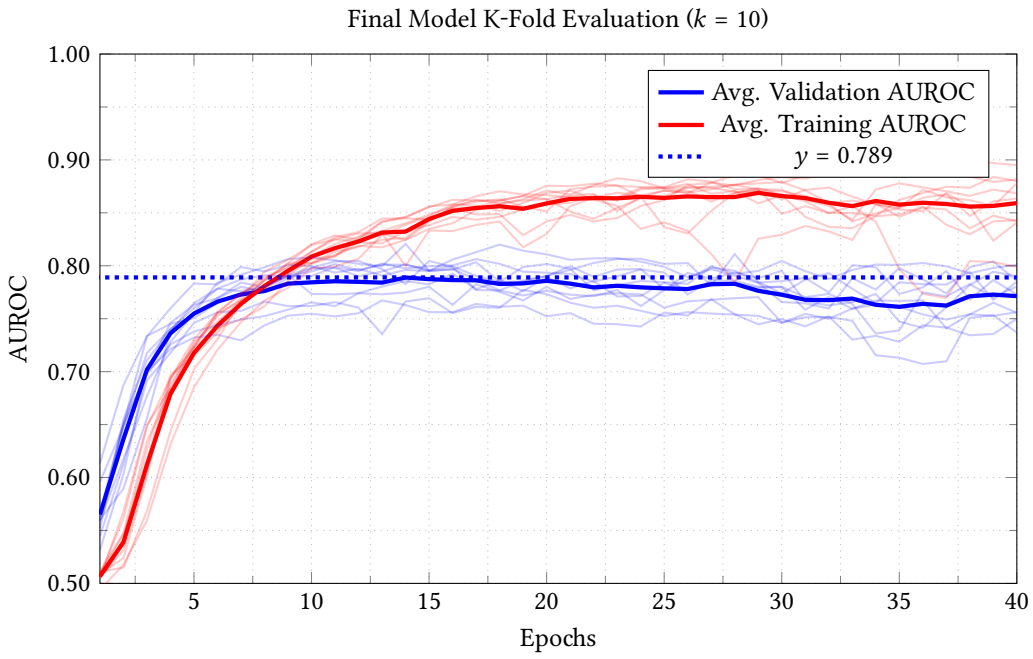


Figure 5.1: K-Fold Evaluation for Final Model with Best Hyperparameters.

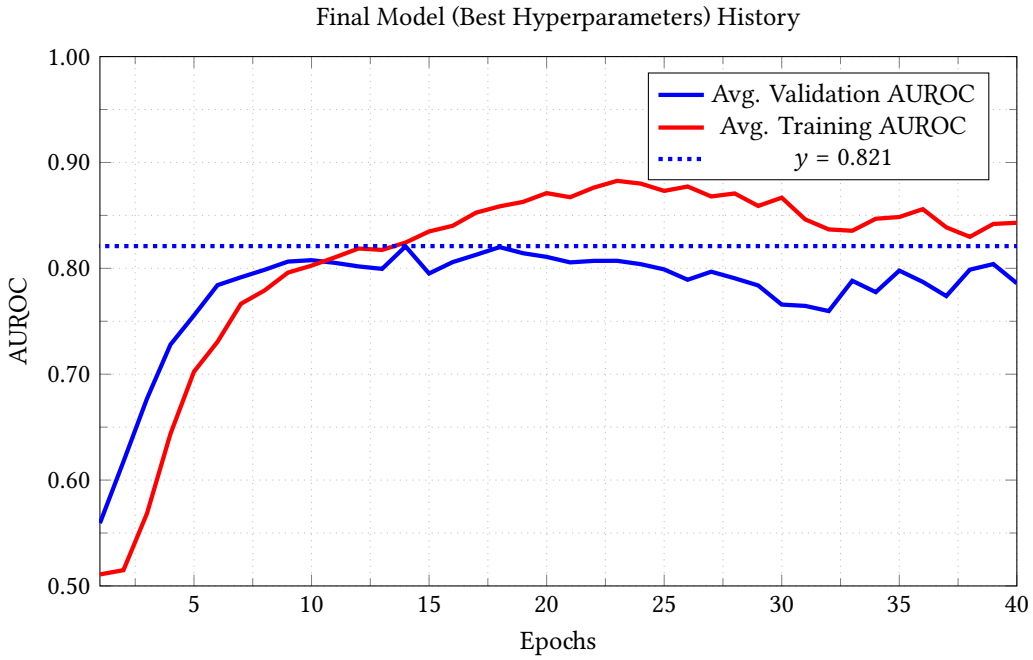


Figure 5.2: Training History of the Final Model (with Best Hyperparameters)

Chapter 6

Conclusion

Appendix A

Additional Materials

A.1 Project Code and Github Repository

All of the implementation details, model architecture, and data are made available in this project's Git repository (Github). Every code listing contains a link to the specific implementation, and different experiments also contain links to their corresponding Jupyter notebooks where the code was originally run. As a part of this project's commitment to reproducibility, all Jupyter notebooks are documented, and readers are encouraged to follow along and run the experiments for themselves. For further information, please see the repository `README.md`.

https://github.com/ShenZhouHong/radiography-ai-project/

A.1.1 Initial Evaluation Models

Jupyter notebooks used to run the initial evaluations of LeNet 1998, InceptionV3 with end-to-end training, and initial transfer learning models:

https://github.com/ShenZhouHong/radiography-ai-project/tree/master/python/initial-evaluation

A.1.2 Hyperparameter Search Code

Jupyter notebooks used to perform the hyperparameter search regime.

https://github.com/ShenZhouHong/radiography-ai-project/tree/master/python/hyperparam-search

A.1.3 Analysis Notebooks

Jupyter notebooks used to analyse the raw data, process for insights and visualisations, and output CSV files:

https://github.com/ShenZhouHong/radiography-ai-project/tree/master/python/analysis

A.1.4 Final Model Weights and Examples

Weights and instructions for the final model trained on the best set of hyperparameters, as well as example notebooks:

https://github.com/ShenZhouHong/radiography-ai-project/tree/master/python/final-model

A.2 Human Research Certification

Collaborative Institutional Training Initiative (CITI Program) certificate demonstrating compliance with human subject research. Verify authenticity at: www.citiprogram.org/verify/?w9f4d2ab4-4b26-446e-a73b-bf4308c881a8-51029320

CITI PROGRAM

Completion Date 01-Sep-2022
Expiration Date 31-Aug-2025
Record ID 51029320

This is to certify that:

Shen Hong

Not valid for renewal of certification through CME.

Has completed the following CITI Program course:

Human Subjects Research
(Curriculum Group)
Human Subject Research - Biomedical Research
(Course Learner Group)
1 - Stage 1
(Stage)

CITI
Collaborative Institutional Training Initiative

Under requirements set by:

Johns Hopkins Medical Institutions

Verify at www.citiprogram.org/verify/?w9f4d2ab4-4b26-446e-a73b-bf4308c881a8-51029320

A.3 Project Proposal Presentation

Project proposal presentation given on 2022-01-11, at the Johns Hopkins University, East Baltimore Campus.

Evaluating Fracture Healing with Artificial Intelligence

Using Transfer Learning to Predict RUST Scores from Radiographs.
A Major Extremity Trauma Research Consortium Project.

Shen Zhou Hong <shong@jhu.edu>

JOHNS HOPKINS BLOOMBERG SCHOOL of PUBLIC HEALTH

METRC Major Extremity Trauma and Rehabilitation Consortium

Goldsmiths UNIVERSITY OF LONDON

UNIVERSITY OF LONDON

Introduction and Table of Contents

- Table of Contents:
  - §1. Background Information
  - §2. Project Design, Methodology, & Endpoints
  - §3. Current Progress, Roadmap, & Challenges
  - Q&A Session.

2022-01-11 METRC, Johns Hopkins 2

Background Information

2022-01-11 METRC, Johns Hopkins 3

Using AI to Evaluate Fracture Healing

- We want to develop an AI that can infer RUST scores from radiographs.
- RUST: Radiographic Union Score for Tibial Fractures. Also used in other long bone fractures.
- RUST measures the progression of fracture healing via callus formation and bridging.

2022-01-11 METRC, Johns Hopkins 4

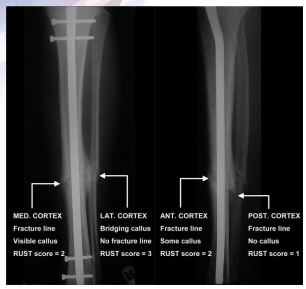


Image source: D. B. Whelan, M. Bhandari, D. Stephen, et al. 2010

2022-01-11 METRC, Johns Hopkins 5

Why is this project useful?

- The AI model can automate the analysis of archived study data.
- RUST scores have good biomechanical correlation, can serve as guidance for rehab.
- Project is novel: prior work in AI-radiography mainly focused on fracture detection.

Prior Research in AI-Radiography

- MURA: Anomaly detection. 40.5k radiographs, achieved AUROC of 0.929
- Lindsey et al: Anomaly detection & localization. 31.0k radiographs, achieved AUROC of 0.967
- Kim et MacKinnon: 1.3k radiographs, achieved AUROC of 0.954

Why has nobody done this before?

- Most individual institutions do not have access to datasets large enough to perform training.
- Most large radiography datasets do not have specialized, *evidence-based, adjudicated* labels.
- Most AI research organizations do not have access to medical research organizations.

METRC is uniquely positioned to conduct this research.

METRC
Major Extremity Trauma and Rehabilitation Consortium

Design, Methodology, & Endpoints

AI Models versus Statistical Models

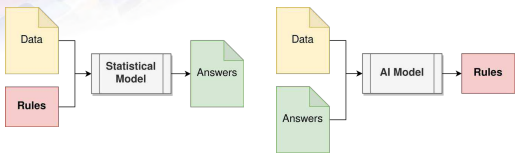


Image source: METRC 2022

METRC Radiography Datasets

- RetroDEFECT: 741 radiographs
- OUTLET: 707 radiographs
- PAIN: 370 radiographs
- PACS: 195 radiographs
- Total: ~2,013 radiographs
- Our study architecture is data-constrained.

Model Architecture

- *Transfer learning*: train an AI model on a large, general-purpose dataset. Then *fine-tune* on the smaller, task-specific dataset.
- Data Augmentation & K-Fold Validation
- We will use ImageNet and MURA as base models for transfer-learning.

Anatomy of a Deep Neural Network

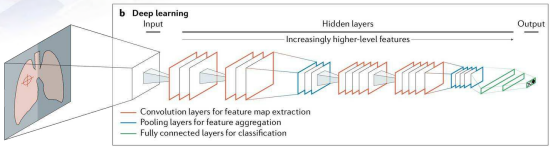


Image source: Hosny, Parmar, Quackenbush et al. 2018

Endpoints

- Model performance measured via AUROC: Area Under Curve (of) Receiver Operating Characteristic.
- Endpoint 1: Model AUROC > 0.50
- Endpoint 2: Model AUROC > "naive" ConvNet
- Endpoint 3: Model AUROC > 0.75

Progress, Roadmap, & Challenges

Roadmap and Current Progress

- 2022-11-15: Complete background research
- 2022-12-01: Study and DNN architecture design
- 2022-12-12: Initial exploration of METRC datasets
- 2023-01-02: Preprocessing of METRC radiographs. (**We are here)**
- 2023-01-31: Initial model development
- 2023-02-27: Fine-tuning, hyperparameter search.
- 2023-03-31: **Deadline: Impl. & Analysis**
- 2023-05-02: **Deadline: Poster Presentation at UoL Goldsmiths**
- 2023-05-12: **Deadline: Submission to UoL Goldsmiths**

Challenges

- Dataset validation and "cleaning."
- Combating model overfitting.
- Programmatically parsing data from REDCap.

REDCap Libraries and Tools

- REDCap Branch Parser: https://github.com/metrc/redcap-branch-parser
- REDCap Schema: https://github.com/metrc/redcapschema

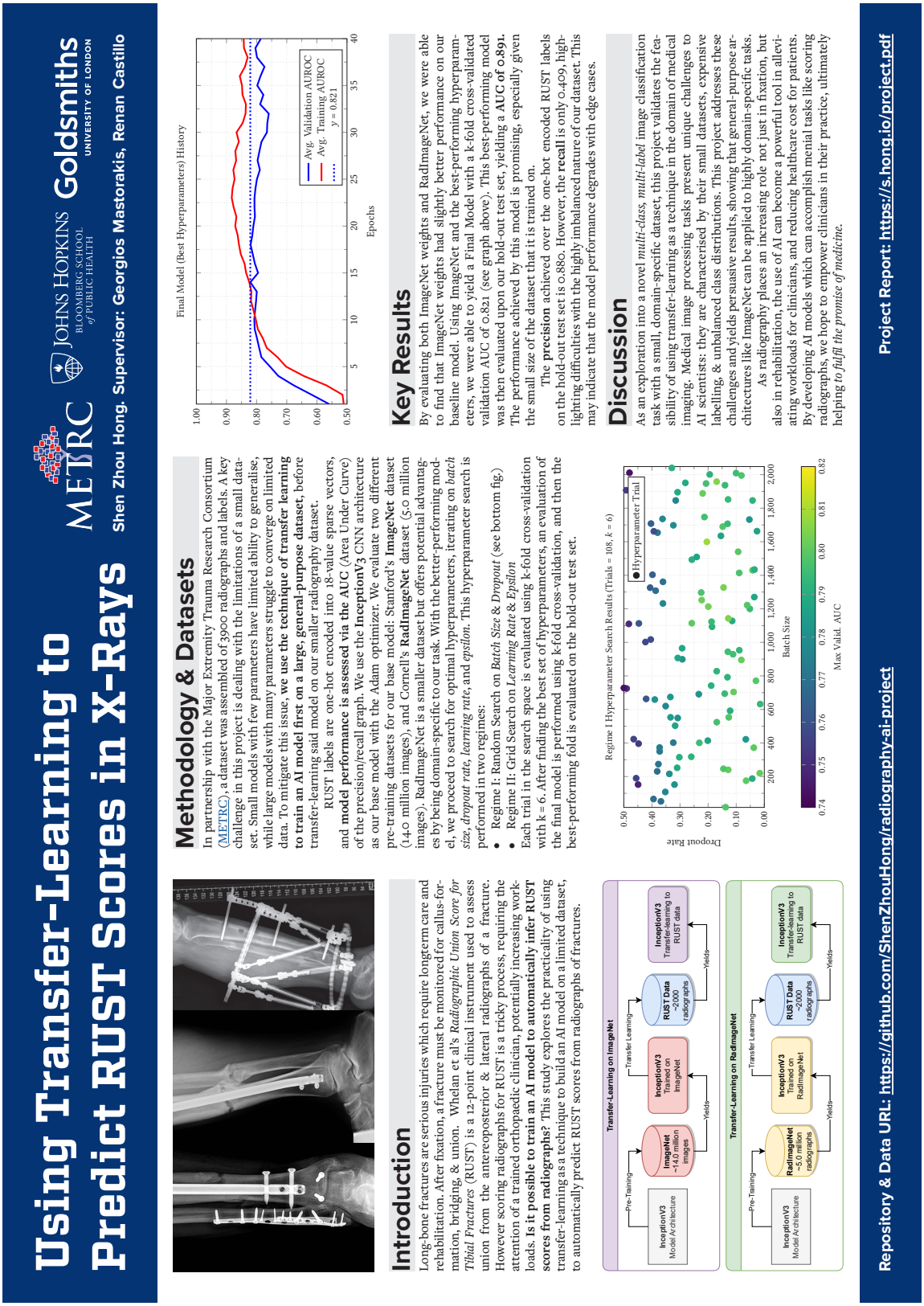
Future Pathways

- Heat-maps and "Explainable AI"
- Further assessment of model performance with human orthopedic specialists/surgeons
- Collect more RUST data. Train a more robust model, conduct serious evaluations of using AI as a diagnostic tool.

The End. Thank you for your attention!

A.4 Poster Presentation

Final project poster presentation, given on 2023-05-11 at Goldsmiths College, University of London.



Bibliography

[1] H. Stein, I. Weize, D. Hoerer, A. Lerner, N. Rozen, and G. Nierenberg, "Musculoskeletal trauma: High- and low-energy injuries," *Orthopedics*, vol. 22, no. 10, pp. 965–967, 1999. DOI: 10.3928/0147-7447-19991001-14. eprint: https://journals.healio.com/doi/pdf/10.3928/0147-7447-19991001-14. [Online]. Available: https://journals.healio.com/doi/abs/10.3928/0147-7447-19991001-14.

[2] M. S. Jones and B. Waterson, "Principles of management of long bone fractures and fracture healing," *Surgery (Oxford)*, vol. 38, no. 2, pp. 91–99, 2020, ISSN: 0263-9319. DOI: https://doi.org/10.1016/j.mpsur.2019.12.010. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0263931919302649.

[3] D. B. Whelan, M. Bhandari, D. Stephen, *et al.*, "Development of the radiographic union score for tibial fractures for the assessment of tibial fracture healing after intramedullary fixation," *Journal of Trauma and Acute Care Surgery*, vol. 68, no. 3, 2010, ISSN: 2163-0755. [Online]. Available: https://journals.lww.com/jtrauma/Fulltext/2010/03000/Development_of_the_Radiographic_Union_Score_for.24.aspx.

[4] J. Nicholson, N. Makaram, A. Simpson, and J. Keating, "Fracture nonunion in long bones: A literature review of risk factors and surgical management," *Injury*, vol. 52, S3–S11, 2021, ISSN: 0020-1383. DOI: https://doi.org/10.1016/j.injury.2020.11.029. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020138320309554.

[5] P. Panchoo, M. Laubscher, M. Held, *et al.*, "Radiographic union score for tibia (rust) scoring system in adult diaphyseal femoral fractures treated with intramedullary nailing: An assessment of interobserver and intraobserver reliability," *European Journal of Orthopaedic Surgery & Traumatology*, vol. 32, no. 8, pp. 1555–1559, Dec. 1, 2022, ISSN: 1432-1068. DOI: 10.1007/s00590-021-03134-6. [Online]. Available: https://doi.org/10.1007/s00590-021-03134-6.

[6] M. E. Cooke, A. I. Hussein, K. E. Lybrand, *et al.*, "Correlation between rust assessments of fracture healing to structural and biomechanical properties," *Journal of Orthopaedic Research*, vol. 36, no. 3, pp. 945–953, 2018. DOI: https://doi.org/10.1002/jor.23710. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/jor.23710. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/jor.23710.

[7] E. Debuka, N. S. Kushwaha, D. Kumar, A. Singh, and V. Sharma, "Rust score–an adequate rehabilitation guide for diaphyseal femur fractures managed by tens," *Journal of Clinical Orthopaedics and Trauma*, vol. 10, no. 5, pp. 922–927, Sep. 1, 2019, ISSN: 0976-5662. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0976566218302054.

[8] J. Irvin, P. Rajpurkar, M. Ko, *et al.*, *Chexpert: A large chest radiograph dataset with uncertainty labels and expert comparison*, 2019. DOI: 10.48550/ARXIV.1901.07031. [Online]. Available: https://arxiv.org/abs/1901.07031.

[9] S. University. "Lera — lower extremity radiographs." en-us, Center for Artificial Intelligence in Medicine & Imaging. (2014), [Online]. Available: https://aimi.stanford.edu/lera-lower-extremity-radiographs (visited on 11/13/2022).

[10] P. Rajpurkar, J. Irvin, A. Bagul, *et al.*, *Mura: Large dataset for abnormality detection in musculoskeletal radiographs*, 2017. DOI: 10.48550/ARXIV.1712.06957. [Online]. Available: https://arxiv.org/abs/1712.06957.

[11] S. J. Adams, R. D. E. Henderson, X. Yi, and P. Babyn, "Artificial intelligence solutions for analysis of x-ray images," *Canadian Association of Radiologists Journal*, vol. 72, no. 1, pp. 60–72, Feb. 1, 2021, ISSN: 0846-5371. DOI: 10.1177/0846537120941671. [Online]. Available: https://doi.org/10.1177/0846537120941671.

[12] L. Tanzi, E. Vezzetti, R. Moreno, and S. Moos, "X-ray bone fracture classification using deep learning: A baseline for designing a reliable approach," *Applied Sciences*, vol. 10, no. 4, 2020, ISSN: 2076-3417. DOI: 10.3390/app10041507. [Online]. Available: https://www.mdpi.com/2076-3417/10/4/1507.

[13] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," *CoRR*, vol. abs/1512.00567, 2015. arXiv: 1512.00567. [Online]. Available: http://arxiv.org/abs/1512.00567.

[14] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," pp. 248–255, 2009.

[15] X. Mei, Z. Liu, P. M. Robson, *et al.*, "Radimagenet: An open radiologic deep learning research dataset for effective transfer learning," *Radiology: Artificial Intelligence*, vol. 0, no. ja, e210315, 0. DOI: 10.1148/ryai.210315. eprint: https://doi.org/10.1148/ryai.210315. [Online]. Available: https://doi.org/10.1148/ryai.210315.

[16] A. Hosny, C. Parmar, J. Quackenbush, L. H. Schwartz, and H. J. W. L. Aerts, "Artificial intelligence in radiology," *Nature Reviews Cancer*, vol. 18, no. 8, pp. 500–510, Aug. 1, 2018, ISSN: 1474-1768. DOI: 10.1038/s41568-018-0016-5. [Online]. Available: https://doi.org/10.1038/s41568-018-0016-5.

[17] Y. Cao, H. Wang, M. Moradi, P. Prasanna, and T. F. Syeda-Mahmood, "Fracture detection in x-ray images through stacked random forests feature fusion," in *2015 IEEE 12th International Symposium on Biomedical Imaging (ISBI)*, 2015, pp. 801–805. DOI: 10.1109/ISBI.2015.7163993.

[18] K. Dimililer, "Ibfds: Intelligent bone fracture detection system," *Procedia Computer Science*, vol. 120, pp. 260–267, 2017, 9th International Conference on Theory and Application of Soft Computing, Computing with Words and Perception, ICSCCW 2017, 22-23 August 2017, Budapest, Hungary, ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2017.11.237. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050917324493.

[19] R. Lindsey, A. Daluiski, S. Chopra, *et al.*, "Deep neural network improves fracture detection by clinicians," *Proceedings of the National Academy of Sciences*, vol. 115, no. 45, pp. 11 591–11 596, 2018. DOI: 10.1073/pnas.1806905115. eprint: https://www.pnas.org/doi/pdf/10.1073/pnas.1806905115. [Online]. Available: https://www.pnas.org/doi/abs/10.1073/pnas.1806905115.

[20] D. H. Kim and T. MacKinnon, "Artificial intelligence in fracture detection: Transfer learning from deep convolutional neural networks," *Clinical Radiology*, vol. 73, no. 5, pp. 439–445, May 1, 2018, ISSN: 0009-9260. DOI: 10.1016/j.crad.2017.11.015. [Online]. Available: https://doi.org/10.1016/j.crad.2017.11.015.

[21] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 1, 2016.

[22] G. Litjens, T. Kooi, B. E. Bejnordi, *et al.*, "A survey on deep learning in medical image analysis," *Medical Image Analysis*, vol. 42, pp. 60–88, 2017, ISSN: 1361-8415. DOI: https://doi.org/10.1016/j.media.2017.07.005. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1361841517301135.

[23] P. Kora, C. P. Ooi, O. Faust, *et al.*, "Transfer learning techniques for medical image analysis: A review," *Biocybernetics and Biomedical Engineering*, vol. 42, no. 1, pp. 79–107, 2022, ISSN: 0208-5216. DOI: https://doi.org/10.1016/j.bbe.2021.11.004. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0208521621001297.

[24] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, May 1, 2017, ISSN: 0001-0782. DOI: 10.1145/3065386. [Online]. Available: https://doi.org/10.1145/3065386.

[25] F. Schneider, L. Balles, and P. Hennig, *Deepobs: A deep learning optimizer benchmark suite*, 2019. DOI: 10.48550/ARXIV.1903.05499. [Online]. Available: https://arxiv.org/abs/1903.05499.

[26] R. M. Schmidt, F. Schneider, and P. Hennig, "Descending through a crowded valley - benchmarking deep learning optimizers," in *Proceedings of the 38th International Conference on Machine Learning*, M. Meila and T. Zhang, Eds., ser. Proceedings of Machine Learning Research, vol. 139, PMLR, Jul. 18, 2021, pp. 9367–9376. [Online]. Available: https://proceedings.mlr.press/v139/schmidt21a.html.

[27] W. T. Obremskey, P. Tornetta 3rd, J. Luly, *et al.*, "Outcomes of patients with large versus small bone defects in open tibia fractures treated with an intramedullary nail: A descriptive analysis of a multicenter retrospective study," en, *J Orthop Trauma*, vol. 36, no. 8, pp. 388–393, Aug. 2022.

[28] R. C. Castillo, S. N. Raja, K. P. Frey, *et al.*, "Improving pain management and Long-Term outcomes following High-Energy orthopaedic trauma (pain study)," en, *J Orthop Trauma*, vol. 31 Suppl 1, S71–S77, Apr. 2017.

[29] A. Leroux, K. P. Frey, C. M. Crainiceanu, *et al.*, "Defining incidence of acute compartment syndrome in the research setting: A proposed method from the PACS study," en, *J Orthop Trauma*, vol. 36, no. Suppl 1, S26–S32, Jan. 2022.

[30] S. S. Burns, A. Browne, G. N. Davis, S. L. Rimrodt, and L. E. Cutting. "Pycap api package." (2023), [Online]. Available: https://github.com/redcap-tools/PyCap (visited on 01/18/2023).

[31] M. M. Bejani and M. Ghatee, "A systematic review on overfitting control in shallow and deep neural networks," *Artificial Intelligence Review*, vol. 54, no. 8, pp. 6391–6438, Dec. 1, 2021, ISSN: 1573-7462. DOI: 10.1007/s10462-021-09975-1. [Online]. Available: https://doi.org/10.1007/s10462-021-09975-1.

[32] L. Perez and J. Wang, *The effectiveness of data augmentation in image classification using deep learning*, Dec. 13, 2017. DOI: 10.48550/ARXIV.1712.04621. [Online]. Available: https://arxiv.org/abs/1712.04621.

[33] L. Taylor and G. Nitschke, "Improving deep learning with generic data augmentation," in *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*, Jan. 31, 2019, pp. 1542–1547. DOI: 10.1109/SSCI.2018.8628742.

[34] L. Yang and A. Shami, "On hyperparameter optimization of machine learning algorithms: Theory and practice," *Neurocomputing*, vol. 415, pp. 295–316, 2020, ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2020.07.061. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231220311693.

[35] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization.," *Journal of machine learning research*, vol. 13, no. 2, Dec. 2, 2012.

[36] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998. DOI: 10.1109/5.726791.

[37] M. V. Narkhede, P. P. Bartakke, and M. S. Sutaone, "A review on weight initialization strategies for neural networks," *Artificial Intelligence Review*, vol. 55, no. 1, pp. 291–322, Jan. 1, 2022, ISSN: 1573-7462. DOI: 10.1007/s10462-021-10033-z. [Online]. Available: https://doi.org/10.1007/s10462-021-10033-z.

List of Figures

List of Tables