Abstract

The aim of this work is to develop a neural network training framework for continual acquisition of small amounts of medical imaging data and create heuristics to assess training in the absence of a validation or test set. Though many machine learning (ML) techniques have achieved impressive performances after being trained on a large dataset, issues persist with sustainability as more data is acquired and technology advances. With medical imaging data, subtleties can vary across different imaging modalities and even the same modality but from different vendors. Perhaps the most essential component would be the precise classification of diagnosed diseases in medical images. This issue is further heightened when radiologists discover new and interesting cases that arrive in real-time in a clinical environment. Because most modern ML techniques take use of a large repository of data for training purposes, the aforementioned scenario would consider mini-batches of data arriving sequentially over time. We formulated a sequential learning approach that would train and consistently update a model on a mini-batch of medical images over time. This approach would lead to better recognition of existing class information while also adapting newer distributions of training data, either as entirely newer classes or newer observations of existing classes. We address problems that impede sequential learning such as overfitting, catastrophic forgetting, and concept drift through experimentation that utilizes PyTorch convolutional neural networks (CNN) and publicly available medical imaging datasets. We begin by comparing experiments that consider a sequential trained CNN with base training on a large batch of the dataset to a sequentially trained CNN without base training. We then transition to estimating an appropriate amount of training iterations for a mini-batch of data for full information extraction without overfitting. This was accomplished through delineated experiments with unique training and validation data recruitment. We also consider examples of real-life data
that better distinguish how our approach would be implemented as a mainstream research tool, and lastly establish the future directions of our work.

Readers and Advisors:

Dr. Haris I. Sair
Dr. Craig Jones
Dr. Brian Caffo
Acknowledgement

This research project was made possible by the guidance of Dr. Haris Iqbal Sair, Associate Professor of Radiology and Director of Neuroradiology at Johns Hopkins Medicine, and Dr. Craig Jones, Research Scientist at Johns Hopkins Malone Center for Engineering in Healthcare. I especially want to thank Dr. Jones for his continual support, from building neural network architectures prior to the project’s conceptual early stages during the Radiology for Engineers course, to analyzing results all the way through the project experimentation phase. My background and interest in computer vision for medical imaging data has grown tremendously throughout my time in the Radiology AI Lab (RAIL) and has led to my pursuit of a PhD in Medical Physics. I would like to thank Dr. Sair for allowing me to attend lab meetings which promote discussion over ongoing topics and future directions in the radiological sciences. It is through these learning opportunities that I discovered sequential learning and took an interest in further exploring this project for thesis completion. I would also like to thank Dr. Brian Caffo, Biostatistician and Professor at Johns Hopkins Bloomberg School of Public Health, for his time in advising me on exploratory data analyses and interpretations. Finally, I would like to extend thanks to the Biomedical Engineering faculty at Johns Hopkins Whiting School of Engineering for a rewarding graduate-level education that cultivated my skills for biomedical imaging research.
# Table of Contents

Abstract .......................................................................................................................... ii
Acknowledgement ......................................................................................................... iv
Table of Contents ......................................................................................................... v
List of Figures ............................................................................................................... vi

Chapter 1: Introduction ................................................................................................. 1

Chapter 2: Pre-training vs No Pre-training Experiments ........................................... 9
  2.1 Background............................................................................................................ 9
  2.2 Data and Model.................................................................................................... 10
  2.3 Experiments........................................................................................................ 14
  2.4 Discussion.......................................................................................................... 36

Chapter 3: Sequential Learning Experimentation ....................................................... 37
  3.1 Background.......................................................................................................... 37
  3.2 Data and Model.................................................................................................... 39
  3.3 Experiments........................................................................................................ 40
  3.4 Discussion.......................................................................................................... 60

Chapter 4: Real-Life Classification Tasks ..................................................................... 62
  4.1 Background.......................................................................................................... 62
  4.2 Data and Model.................................................................................................... 63
  4.3 Experiments........................................................................................................ 68
    4.3.1 Modified CXR Experiments ..................................................................... 70
    4.3.2 NIH Chest X-Ray Experiments ............................................................... 85
  4.4 Discussion.......................................................................................................... 93

Chapter 5: Summary & Future Directions .................................................................. 95

References ..................................................................................................................... 100
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Chapter</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1</td>
<td>Chapter 2</td>
<td>Sample cases for three Medical MNIST classes</td>
<td>11</td>
</tr>
<tr>
<td>Figure 2</td>
<td>Chapter 2</td>
<td>Sample case for fourth Medical MNIST class</td>
<td>11</td>
</tr>
<tr>
<td>Figure 3</td>
<td>Chapter 2</td>
<td>Overview of experimental frameworks that include pre-training</td>
<td>15</td>
</tr>
<tr>
<td>Figure 4</td>
<td>Chapter 2</td>
<td>Overview of experimental frameworks without pre-training</td>
<td>16</td>
</tr>
<tr>
<td>Figure 5</td>
<td>Chapter 2</td>
<td>Experiment 1a: Pre-training results</td>
<td>19</td>
</tr>
<tr>
<td>Figure 6</td>
<td>Chapter 2</td>
<td>Experiment 1a: Sequential training results</td>
<td>20</td>
</tr>
<tr>
<td>Figure 7</td>
<td>Chapter 2</td>
<td>Experiment 1b: Sequential training results</td>
<td>22</td>
</tr>
<tr>
<td>Figure 8</td>
<td>Chapter 2</td>
<td>Experiment 2a: Pre-training results</td>
<td>26</td>
</tr>
<tr>
<td>Figure 9</td>
<td>Chapter 2</td>
<td>Experiment 2a: Sequential training results</td>
<td>28</td>
</tr>
<tr>
<td>Figure 10</td>
<td>Chapter 2</td>
<td>Experiment 2b: Sequential training results</td>
<td>30</td>
</tr>
<tr>
<td>Figure 11</td>
<td>Chapter 2</td>
<td>Experiment 3a: Pre-training results</td>
<td>33</td>
</tr>
<tr>
<td>Figure 12</td>
<td>Chapter 2</td>
<td>Experiment 3a: Sequential training results</td>
<td>34</td>
</tr>
<tr>
<td>Figure 13</td>
<td>Chapter 2</td>
<td>Experiment 3b: Sequential training results</td>
<td>35</td>
</tr>
<tr>
<td>Figure 14</td>
<td>Chapter 3</td>
<td>Overview of individual data recruitment experiments</td>
<td>41</td>
</tr>
<tr>
<td>Figure 15</td>
<td>Chapter 3</td>
<td>Experiment 0: Traditional training results</td>
<td>43</td>
</tr>
<tr>
<td>Figure 16</td>
<td>Chapter 3</td>
<td>Experiment 1: Sequential training results</td>
<td>46</td>
</tr>
<tr>
<td>Figure 17</td>
<td>Chapter 3</td>
<td>Experiment 1a: Single iteration sequential training results</td>
<td>50</td>
</tr>
<tr>
<td>Figure 18</td>
<td>Chapter 3</td>
<td>Experiment 1a: Averaged sequential training results</td>
<td>52</td>
</tr>
<tr>
<td>Figure 19</td>
<td>Chapter 3</td>
<td>Experiment 1b: Single iteration sequential training results</td>
<td>56</td>
</tr>
<tr>
<td>Figure 20</td>
<td>Chapter 3</td>
<td>Experiment 1b: Averaged sequential training results</td>
<td>58</td>
</tr>
<tr>
<td>Figure 21</td>
<td>Chapter 3</td>
<td>Experiment 1b: Renewed model sequential training results</td>
<td>59</td>
</tr>
<tr>
<td>Figure 22</td>
<td>Chapter 4</td>
<td>Sample cases for three modified CXR classes</td>
<td>65</td>
</tr>
<tr>
<td>Figure 23</td>
<td>Chapter 4</td>
<td>Sample cases for three NIH Chest X-Ray classes</td>
<td>66</td>
</tr>
<tr>
<td>Figure 24</td>
<td>Chapter 4</td>
<td>Modified CXR Experiment 0: Traditional training results</td>
<td>71</td>
</tr>
<tr>
<td>Figure 25</td>
<td>Chapter 4</td>
<td>Modified CXR Experiment 1: Sequential training results</td>
<td>73</td>
</tr>
<tr>
<td>Figure 26</td>
<td>Chapter 4</td>
<td>Modified CXR Experiment 1a: Sequential training results</td>
<td>76</td>
</tr>
<tr>
<td>Figure 27</td>
<td>Chapter 4</td>
<td>Modified CXR Experiment 1b: Sequential training results</td>
<td>78</td>
</tr>
<tr>
<td>Figure 28</td>
<td>Chapter 4</td>
<td>Modified CXR Experiment 2a: Pre-training results</td>
<td>81</td>
</tr>
<tr>
<td>Figure 29</td>
<td>Chapter 4</td>
<td>Modified CXR Experiment 2a: Sequential training results</td>
<td>83</td>
</tr>
<tr>
<td>Figure 30</td>
<td>Chapter 4</td>
<td>Modified CXR Experiment 2b: Sequential training results</td>
<td>84</td>
</tr>
<tr>
<td>Figure 31</td>
<td>Chapter 4</td>
<td>NIH Experiment 0: Traditional training results</td>
<td>86</td>
</tr>
<tr>
<td>Figure 32</td>
<td>Chapter 4</td>
<td>NIH Experiment 1: Pre-training results</td>
<td>89</td>
</tr>
<tr>
<td>Figure 33</td>
<td>Chapter 4</td>
<td>NIH Experiment 1: Sequential training results</td>
<td>90</td>
</tr>
<tr>
<td>Figure 34</td>
<td>Chapter 4</td>
<td>NIH Experiment 2: Sequential training results</td>
<td>92</td>
</tr>
</tbody>
</table>
Chapter 1: Introduction

Identifying injury and disease extent in human beings and animals through medical imaging technology emphasizes the importance of radiology in both the medical and research domain. In addition to numerous improvements to both hardware and software aspects of various imaging modalities, in recent years radiology has seen tremendous growth through computer vision (CV) and artificial intelligence (AI) applications. This is due to advanced computing power, techniques, and algorithms that are being recruited to solve different complex problems. One particular subset of AI that has seen involvement within radiology research is the use of ML to make predictions on medical imaging data. In this project we focused on the visual recognition problem of image classification. Though modern ML approaches have seen major success, there are a few challenges that lead to its limited usage in real-life scenarios. A major challenge that has been discussed is the ability to learn new class data incrementally over time. In traditional ML approaches for image classification, a large repository of data is readily available and typically contains 1000s of datasets for batch-learning. These datasets are typically split into training, validation, and testing sets. As a neural network learns characteristics of the training data and validates on those characteristics through the validation data, accuracy and loss metrics of the trained model can be assessed on the testing data [1]. A key limitation of a traditional neural network is the need to be trained across many epochs, long run-times, and considerable computational power. In real-life scenarios often what occurs is that new data, which could contain new observations of old classes along with newer classes, arrives sequentially over time and would need to be processed in a similar way. Radiologists in particular see a large variety of medical images during regular clinical work. Should a new and interesting diagnosis be observed, a team of radiologists may seek to train a neural network to accurately identify the unique case. However, the
availability of training data is unlikely in the early stages of a new clinical study. Often what occurs is continuous labeling of medical images in real-time by radiologists who receive a certain number of new findings each day over a period of days. As a result, translating traditional ML techniques to real-life medical imaging cases is challenging due to limited datasets and uncertain training approaches.

Sequential learning provides an ML approach that satisfies training a neural network with few datasets each time. Also referred to as incremental learning, the technique provides the ability to maintain life-long learning for multiple classes’ data, maintaining good performance on both present and future data streams, and shorter run-times required to update the network, which is advantageous if resources are constrained [2]. While incremental learning is advantageous for real-life scenarios, there are two major obstacles that prevent its major implementation: catastrophic forgetting and concept drift. Steadily introducing datasets through a data stream of both existing and newer classes’ information presents a sufficient strategy during the earlier stages of training a neural network. As the metric for time continues however, newer classes can cause a trained network to forget the learned characteristics of initial class information, a phenomenon known as catastrophic forgetting [2]. Although a neural network can be trained to accurately classify initial class information, after enough time passes the initial datasets are capable of being forgotten. Newer datasets of the initial classes are also capable of impacting the distribution of all the training data the network has been learning from. This phenomenon, referred to as concept drift, occurs when enough key differences between initial and future datasets can negatively degrade the performance of the classification network [2]. Our goal is to also explore the deficiencies caused by both catastrophic forgetting and concept drift in our experimentation. Lastly, because the training data will be introduced to the network in limited quantities, it is important to establish the hyperparameters for the model to not overfit on the data. Overfitting is another
phenomenon that can occur when a model is being trained. Often too much training can cause subtle oscillations in the training data, such as the details and noise, to be misinterpreted as important features to learn on and severely hinder a model’s ability to generalize on future oncoming datasets.

As a separate technique, sequential learning has also been referred under different terminologies such as incremental learning, progressive learning, online learning, and lifelong learning. The subtleties for each of these techniques are mostly based on differences in opinions from experts within the research spectrum. However, what distinguishes such algorithms from traditional ML techniques is their ability to adapt to changes in data distribution for continuous training in real-time. The most common nomenclature in literature for said technique is often incremental learning, and the term is not always consistent. Incremental learning refers to data streaming strategies which work under limited memory resources [3]. As a result, this strategy rules out learning in batch modes where samples are stored up to a certain time step in memory, and instead relies on a compact representation of observed signals [3]. Sequential learning is the strategy we used for our experimentation and similarly varies under different terminology. It differs from incremental learning in that it learns a sequence of tasks without having access to training data from previous or future tasks [4]. Additionally, this strategy is also used for datasets arriving in a certain order over time, meaning a group of datasets would always arrive on iteration one followed by another group of datasets on iteration two and so on. This would be established for all runs of the sequential learning architecture to ensure cohesion across the CNN experiments. Progressive learning is an approach similar to other continuous training techniques, however the information of number of classes is fixed as the model initializes and ultimately restricts learning newer classes on the run [5].

Online learning algorithms are preferable for large scale datasets and real-time problems. All online learning techniques can adapt to newer datasets, but it is possible they can also
forget previously learned information due to catastrophic forgetting. There are many possibilities here which range from mini-batch techniques that accumulate a small number of samples to batch learning approaches where all samples are stored internally [3,6]. Lifelong learning is a form of continual learning defined by an adaptive algorithm that can learn streams of information, where the information is steadily made available over time and the number of learning tasks are not predefined [7]. Having the model trained in this approach also mitigates catastrophic forgetting by seeing the same pseudo-randomly shuffled training examples over and over, allowing it to recover any lost knowledge [7].

Although the aforementioned continual learning techniques vary to their own degree, different methodologies have been proposed to address their limitations while also being compared to similar state-of-the-art methods. Michieli et al. provided an approach for a semantic segmentation task through incremental learning while also tackling performance drop due to catastrophic forgetting [8]. The methodology included training a network architecture to recognize training data and new class information, having incremental learning steps, updating the model with a linear combination of cross-entropy and distillation loss, and after k-th incremental step the model is saved and the procedure repeats for each new instance of new class learning [8]. Although our evaluation is for image classification with different datasets, our sequential learning pipeline draws heavily from this architecture. Michieli et al. trained their network with a Stochastic Gradient Descent that considers an initial training stage followed by an incremental one [8]. After five hours of training, a single class was added to the network and the performance was evaluated. A more challenging task was the addition of five classes, incrementally added one-by-one, as the accuracy drop was far more significant due to overestimation of the new classes. Freezing the encoder was determined as the best approach because a single class addition does not alter the responses of the whole network [8]. In our case, only a single class was ever added during our sequential learning
experimentation and as such this approach was not implemented. Perkonigg et al. provide an approach for continual learning with medical imaging data similar to Michieli et al. [9]. Although traditional ML models yield impressive results after being trained once on image datasets, as technology improves it will grow increasingly difficult to utilize said models due to difficulties adapting to newer datasets and classes. To reach implementation in clinical practice, the ML method should be able to deal with new data sources, i.e. different scanners, in a continuous stream of medical images. This method is known as dynamic memory (DM), and it keeps a small but diverse subset of the data within memory to tackle issues of catastrophic forgetting [9]. Pekonigg et al. demonstrated their method with two different imaging modalities and tasks: cardiac magnetic resonance imaging (MRI) segmentation and computer tomography (CT) lung nodule detection. The cardiac MRI data was acquired from four different vendors and split into training, validation, and testing sets, whereas the CT data was acquired from the Lung Image Database Consortium database. Their method was compared to and outperformed other state-of-the-art strategies for average dice score. For the cardiac MRI data, results had shown that data from subsequent scanners can improve the final model performance [9]. Our experimentation also considered the DM strategy as we keep a running list of all “used” data our sequential learning network has trained on.

Performance degradation afflicts most modern ML models due to data distribution shifts over time, and as a result many serviceable methods can be rendered outdated. The main challenge implementing such tools in the medical imaging domain are the striking costs that associate with retraining models from scratch. In addition to this, potential computing power and low infrastructure limitations make it infeasible to streamline such tools, especially when considering the regulatory constraints dealing with patient data in clinical settings [10]. This served as added motivation when preparing our experiments as continual learning minimizes resource constraints while leveraging data
to mitigate catastrophic forgetting. Srivastava et al. formulated a domain incremental learning approach with large-scale chest x-ray datasets with clear domain shifts [10]. Such shifts can be attributed to changes in acquisition hardware or even the physical properties of the same hardware, all of which can severely impact the performance of any modern ML model. Srivastava et al. implemented their method on the data stream using constraints such that no distinct boundary existed between the domains [10]. Similar to other methods, the diagnostic model is trained on an initial base domain and then adapts to oncoming data, in this case mini-batches, which it must reliably predict on. Class imbalances were mitigated by each label weighted in the cross-entropy loss according to the global frequency observed in the data stream [10]. Experimentation was done on three public medical imaging datasets, one of which being the National Institutes of Health (NIH) Chest X-Ray Dataset. We discuss potential applications of our sequential learning on the same NIH dataset as an example of real-life data in our future directions. The team’s method showed steady performance following base training, although the NIH dataset domain shifts contributed to performance drops. Compared to other popular continual learning approaches in medical imaging however, the team’s method utilized minimal memory space while also proving to be the most resilient against catastrophic forgetting.

Perhaps the most important objective in our experimentation was to identify an appropriate number of training iterations a batch of data should undergo to maximize the amount of information extracted by the network. A single training iteration is without a doubt insufficient during the sequential learning process. However, it is important that we do not overestimate the training iterations as too many can cause the network to overfit on the sequentially added data. Zhu et al. addresses the problem of overfitting through incremental learning with new classes but with fewer samples [11]. Existing incremental learning methods have an inability to expand the representation space after the initial representation space is established for existing classes. Additionally, smaller samples
being introduced in an incremental fashion is insufficient for recognizing new classes while also retaining old class knowledge. Zhu et al. addresses these problems through a random episode selection strategy and self-promoted prototype refinement mechanism on three public imaging datasets [11]. The defined approach of few-shot class-incremental learning focuses on quickly learning new class information with few reference images. Zhu et al. based their methodology around a standard learning paradigm which randomly samples images from the entire dataset while minimizing the loss function [11]. They build upon this through their incremental prototype learning scheme which contains two important components. The first is random episode selection which forces the gradients to adapt to randomly simulated incremental processes, and the second is dynamic relation projection which maintains old class information while enhancing discrimination of newer classes [11]. Analysis showed that an increased number of iterations showed a decrease in corresponding accuracy likely due to a direct relationship with training difficulty and iteration count. Zhu et al. also demonstrated that one-shot learning is slightly worse than other conditions, and that if the number of shots exceeds five the performance decreases [11]. These results agree with our hypotheses and substantiate our objective to find an optimal number of training iterations for a sequentially trained network.

The following chapters encompass our sequential learning implementation through experiments conducted with publicly available medical imaging datasets. Chapter 2 details our work comparing two different CNNs: the first CNN is trained in a batch-learning scenario before undergoing sequential learning that would further refine the performance of the model, whereas the second CNN undergoes sequential learning exclusively. Chapter 3 examines the hyperparameters that would enable a CNN to fully extract information from data that is obtained sequentially. The training performance of the sequentially trained network was quantified through two different methods of organizing training and validation data. Chapter 4 explores more challenging classification tasks
which make use of our sequential learning architecture with examples of real-life medical image datasets. Lastly, we will define future directions for our sequential learning pipeline.
Chapter 2: Pre-training vs No Pre-training Experiments

2.1 Background

The goal of sequential learning is to train a neural network on a defined number of oncoming datasets over separate “days”. The network begins with an untrained model that steadily improves its ability to classify medical imaging data as it acquires more datasets to train on. Within this chapter we explore experiments that incorporate a traditional batch training on a pre-allocated set of training data before sequential learning begins. We identify the traditional training as pre-training once the model gets slightly familiarized with data structures before strongly improving its classification accuracy over the period of sequential training “days”. Traditionally, a classification neural network is trained with pre-allocated imaging datasets. These datasets contain a large quantity of imaging data which can effectively train a network to a high accuracy threshold. Depending on the model and learning parameters (epoch count, batch size, learning rate, etc.), we should expect that recruiting data in a sequential learning manner to train a model would attain comparable results to a traditionally trained neural network. This will be determined using the Medical MNIST dataset for 3-way classification of three different imaging modalities [12].

As mentioned, we will experiment with different methods for sequential learning with and without pre-training in this chapter. For each experiment, we will compare the sequential learning findings to show how it remains a viable option for neural network classification if pre-training with batch training data is not a viable option. Section 2.3.1 describes the sequential learning base experiment comparing the approach that incorporates a pre-training loop to an experiment without a pre-training loop. What follows in Section 2.3.2 describes a pre-training vs. no pre-training experiment on a 3-way
classifier that introduces a new Medical MNIST data class during the sequential learning loop, effectively transforming the model into a 4-way classifier. Section 2.3.3 includes experiments that run in an identical manner to Section 2.3.2, except the amount of training datasets recruited each “day” in the sequential learning loop is randomized. This is more representative of a real-life scenario where the amount of recruited data each “day” lacks uniformity and is always changing.

2.2 Data and Model

We utilized the publicly available Medical MNIST imaging dataset on Kaggle, contributed by Arturo Polanco Lozano, for our traditional and sequential learning experiments [12]. This dataset consists of six unique classes for a total of 58,954 medical images, all of which were taken from other datasets and processed into JPEGs with 64x64 dimensions. For our experiments we wanted to create a 3-way classifier with the following modalities: CXR, Hand, and HeadCT. The combined sum of medical images across the three modalities was 30,000 images (10,000 images for each modality) that were split into training, validation, and testing subsets. A sample medical image from each modality can be seen in Figure 1. For certain experiments the model would be trained as a 3-way classifier before being introduced to a fourth class during sequential training. When introduced, the AbdomenCT modality updated the model into a 4-way classifier, a sample medical image of which can be seen in Figure 2.
2.2.1 Medical MNIST Dataset

![Figure 1: Sample cases for each of the three classes (from left to right: CXR, Hand, HeadCT)](image)

The neural network is using the three classes (CXR, Hand, HeadCT) for classification with the following splits: 21,000 global training datasets, 3,000 global validation datasets, and 6,000 global testing datasets. A subsample of the global datasets will be selected for our experiments as follows: 20,000 datasets for training, 1,000 datasets for validation, and 2,000 datasets for testing. This global split of the datasets remained consistent for every experiment listed in Chapters 2 and 3. The goal here was to maintain consistent findings across all sequential learning experiments. Such findings could be significantly varied if the training, validation, or testing data is changing with each run. Additionally, most of the data is allocated for training purposes to reduce variance among the parameter estimates.

![Figure 2: Sample case for AbdomenCT class](image)
For Experiments 2 and 3 listed below the sequential learning loop would train as a 3-way classifier until the halfway mark, defined by half the total number of “days”, where AbdomenCT data is mixed in with the training and testing data. Because the global testing data consists of 2000 datasets with an approximate \( \frac{1}{3} \) split between the three primary classes, 670 AbdomenCT datasets would be added to the global testing data, and all remaining AbdomenCT datasets would be added to global training data.

2.2.2 Training, validation, and testing data splits

We created a Python notebook using Google Colab to load 30,000 Medical MNIST datasets into one list. First, the order of the 30,000 datasets were randomized. From the randomized list, the filenames for the first 21,000 datasets were written out to the text file “filenames_train.txt”. This would assure the same training data was used for each of the CNN experiments. From the same list, the filenames for the subsequent 3,000 datasets were written out to the text file “filenames_val.txt”, which would assure the same validation data was being used for each experiment. Lastly, the filenames for the remaining 6,000 datasets were written out to the text file “filenames_test.txt” for the same reason. When loading in all 30,000 datasets, the text files would be used to pre-set the training, validation, and testing splits to maintain consistent findings across the individual CNN experiments.

2.2.3 Data Augmentation

Every image within the training, testing, and validation datasets that are utilized in the sequential learning pipeline undergoes some form of data augmentation. We have defined a few key transformations that are consistent across all experiments. A Medical MNIST image has a 50% probability of undergoing a random horizontal flip, will be rotated somewhere within the range of 5 degrees clockwise or counterclockwise, will be translated horizontally in the left or right direction by 0.05x the image width, will be translated
vertically up or down by 0.05x the image height, and a 5% chance of an applied color jitter effect. Although all images within the utilized Medical MNIST dataset are provided as 64x64 dimensions, we resize each image into 64x64 dimensions anyway as an assurance. We also convert all images in the range 0-255 to a tensor in the range 0-1 for computing purposes. Lastly, we normalize all images with the mean and standard deviation values provided by ImageNet.

2.2.4 Neural Network Algorithm

The PyTorch ResNet50 model was used for our image classification algorithm with data augmentation applied to the incoming data. This neural network is one of the more prominently used vision model architectures for image classification research. During early sequential learning tests, multiple models were compared to ResNet50 but ultimately the chosen model generated the best results. However, the findings in Section 2.3 should generalize to any architecture and therefore we only show the results for ResNet50. The cost function computed the cross-entropy loss between the input and the target, and the optimization algorithm used was Adam. The number of N datasets recruited per “day” varied for each experiment along with the total “days”, total day-epochs, and model learning rate (LR). This base algorithm was used for every experiment to maintain consistent findings. The preliminary datasets (training, validation, testing) were also kept consistent to reduce variability with the neural networks. For the experiments that add a fourth class to the training and testing cohort, we train on N datasets of a randomized sample of the four classes per “day” to estimate the amount of time the network needs to properly recover.
2.3 Experiments

We explored three experimental frameworks to compare sequential training with pre-training vs only sequential training. Experiments 1a and 1b represent the baseline experiments, where 1a includes the model undergoing pre-training on an allocated set of training data until a desirable accuracy threshold is reached (~65-70%). Once pre-training is completed, the model continues to be trained in a sequential learning format for a predefined number of “days” and datasets per “day”. Sequential learning continues until the model accuracy stabilizes to 90+%%. Experiment 1b uses the same hyperparameters as 1a, but the model does not undergo pre-training and instead trains through sequential learning exclusively. The sequential learning results from both experiments will be compared to each other to ensure cohesion between frameworks. Experiments 2a and 2b represent sequential learning with and without pre-training but with a fourth class added during sequential training. Experiment 2a undergoes traditional pre-training as a 3-way classifier and continues to train as such initially during sequential training. Halfway through sequential training however, datasets from the fourth class are added to the global training and testing data and the model must retrain to 90+% accuracy with the newly added class. This is similarly done in Experiment 2b with the same hyperparameters and no pre-training performed. These two experiments will also be compared from the sequential learning results for consistency between frameworks. Whereas Experiments 3a and 3b utilize the same architecture as Experiments 2a and 2b, they differ in that a random number of datasets are being recruited for “daily” training. The sequential learning results for these experiments will also be compared to ensure consistency between frameworks.
Figure 3: Three experimental frameworks for CNN learning that includes batch pre-training followed by sequential learning. Experiment 1a includes a traditional training method with 3-way classification on an allocated set of training data before shifting to sequential learning for a defined number of datasets acquired per “day”. Experiment 2a involves the same traditional training method as Experiment 1a. Sequential training is performed similarly as well until the halfway mark, defined by half the total number of “days”, where a fourth class is introduced which reduces the accuracy and requires retraining over the second half of training “days”. Experiment 3a is identical to Experiment 2a but involves a changing number of daily datasets acquired per “day”.

![Figure 3: Three experimental frameworks for CNN learning.](image-url)
Figure 4: Three experimental frameworks for neural network learning that only involves sequential learning. Experiment 1b is performed with 3-way classification in a sequential learning format for a defined number of datasets acquired per “day”. Experiment 2b operates similarly as Experiment 1b until the halfway mark where a fourth class is introduced which reduces the accuracy and requires retraining over the second half of training “days”. Experiment 3b is identical to Experiment 2b but involves a changing number of daily datasets acquired per “day”.
2.3.1 Baseline Sequential Learning (Experiment 1a & 1b)

Goal

The goal of this experiment is to train a 3-way classifier via two methods. The first method incorporates a traditional batch-training loop on a small subset of training data. Because the model obtains an initial understanding of the data via this small subset before being sequentially trained, we refer to this traditional training loop as pre-training. The sequential training continues improving overall accuracy and loss metrics by training on N datasets per “day” for a predefined number of “days”. The second method has the model begin learning from scratch on N datasets per “day”. The results from these base frameworks with and without pre-training are meant to be comparable to confirm that sequential training can be performed even without an initial sample subset of data.

Methods

The CXR, Hand, and HeadCT classes from the Medical MNIST dataset were used for training, validation, and testing for a multi-classifier network. There were 30,000 CXR, Hand, and HeadCT datasets pre-set from each associated “filenames” text file with the following splits: 70% training data, 10% validation data, and 20% testing data.

Batch pre-training: The PyTorch ResNet50 CNN was trained with the parameters batch size of 16, five epochs, and LR of 1e-6. For the baseline traditional training, we used a subset of the global training data, which consists of 500 images, and validated on the pre-training results against the global validation data. Each of these splits are defined in greater detail in Data Section 2.2.2 above. After training on a limited number of epochs, we expect the pre-training to reach a desirable accuracy threshold (~70%) before the model shifts to training in a sequential learning format.
Sequential training: The CNN was further trained each “day” on N = 20 datasets recruited from the global training data for one day-epoch over a period of 300 “days”. The test set of data is not traditionally evaluated until after the neural network is completely trained and no decisions on training are applied to the test data set. In this case we are using the test dataset as a held-out sample of the population which is evaluated after the training each “day” as a way to understand the training. As the model continues to train over the “days”, we expect it to be fully trained (90+%%) with stabilized training data oscillations as it nears completion.

Sequential training, no pre-training: Similar to before, the PyTorch ResNet50 CNN was trained with the parameters batch size of 16 and a LR of 1e-6. Because there is no pre-training with this method, we sequentially trained over a period of 500 “days” while also recruiting N = 20 datasets per day. The increased number of days is meant to show a comparable number of datasets being used between methods. After training on a single day-epoch, the results should show a fully trained (90+% accuracy) model that has stabilized training data oscillations as it nears completion.

Results

Batch pre-training: The results in Figure 5 are shown from batch pre-training for five epochs with a LR of 1e-6. We can see the accuracy and loss curves of the traditional multi-classifier network as a function of epochs (accuracy) and cross-entropy loss. We can observe how the accuracy can reach ~70% even with a limited sample size, which we deemed a good starting point for the CNN before it begins sequential learning. The loss metrics also show a steady decrease but do not appear anywhere near stabilizing (extracting as much information as possible from the training datasets).
Figure 5: The pre-training accuracy of the 3-way classifier (CXR vs. Hand vs. HeadCT) as a function of epochs (top). The pre-training loss of the 3-way classifier as a function of cross-entropy loss (bottom) per epoch. This is shown for training and validation metrics.

Sequential training: The results in Figure 6 represent the sequentially trained network for one day-epoch over 300 “days”. The pre-trained model from Figure 5 is carried into sequential training and trains on 10 datasets recruited each “day” from the global training data. We can observe how the starting point of testing accuracy begins near where validation accuracy ends during pre-training. This indicates proper continuation of training between learning loops. The model oscillates heavily for training accuracy over roughly the first 100 “days” as can be seen in the oscillation between 60-95% accuracy. Following “day” 100 the training curve oscillates mostly between 80-100% until “day” 250 which indicates that it is beginning to stabilize. In fact, following “day” 250 the model mostly
oscillates between 90-100% accuracy, though not without a few precipitous declines that can be observed around “days” 190 and 275. Shortly after these declines however the model quickly self-corrects and restabilizes. The testing accuracy steadily increases over the period of 300 “days” and mostly stabilizes at ~95% for the final 100 “days”. The loss metrics show a steady decrease, though the training loss shows wild oscillations over the period of learning “days”. The testing loss steadily decreases all the way until “day” 200 where it appears to stabilize around 0.4. This means the network has mostly extracted enough information from the datasets and would not need further training past “day” 300.

**Figure 6:** The sequentially trained CNN following pre-training for one day-epoch. The accuracy of the 3-way classifier (CXR vs. Hand vs. HeadCT) as a function of “days” (top). The loss of the 3-way classifier as a function of cross-entropy loss (bottom) per “day”. The training metrics are for the 20 datasets evaluated by the CNN each “day”. The testing metrics are evaluated by the CNN at the end of each “day” on the set aside testing data.
Sequential training, no pre-training: The results in Figure 7 represent the sequentially trained network that begins learning without any pre-training for one day-epoch over a period of 500 “days”. The lack of pre-training can be distinguished by the testing accuracy that begins at ~30%. Initially, the training accuracy oscillates heavily between 20-50% and is quite unstable. This changes over the duration of sequential learning however as the oscillations shift higher and fluctuate far less, eventually reaching 70-90% by “day” 250. Over the final 250 “days” the training accuracy mostly oscillates between 80-100%. The testing accuracy metric, which begins slow due to very few datasets and learning information, steadily increases for the first 300 “days” of training before reaching ~90% accuracy. As more information is collected over the following 200 “days”, the testing accuracy reaches ~95% and stabilizes over the final 100 “days”. The loss metrics start off very high due to little information extraction but steadily decreases through sequential learning. The training loss oscillates wildly over the final 200 “days” and can be presumed to fluctuate past “day” 500 should more training days be considered. It can also be inferred that the testing loss would continue to decrease as there is no indication of the curve flattening as the network reaches “day” 500.
Figure 7: The sequentially trained CNN without pre-training for one day-epoch. The accuracy of the 3-way classifier (CXR vs. Hand vs. HeadCT) as a function of “days” (top). The loss of the 3-way classifier as a function of cross-entropy loss (bottom) per “day”. The training metrics are for the 20 datasets evaluated by the CNN each “day”. The testing metrics are evaluated by the CNN at the end of each “day” on the set aside testing data.

As we can observe from Figures 6 & 7, the testing accuracy for sequential learning with and without pre-training both stabilize to ~95%, and the testing loss also stabilizes to ~0.4. The comparative metrics show how sequential learning remains a viable option for CNN classification if there is no initial data subset to train on. The CNN would still benefit from initial training data though as a sequentially trained model would require fewer timepoints to reach a desirable accuracy threshold.
2.3.2 Sequential Learning, Added Class (Experiment 2a & 2b)

Goal

The goal of this experiment is to perform sequential training on a 3-way classifier with and without pre-training. For both methods during sequential training, a new class of Medical MNIST data, AbdomenCT, will be introduced to the CNN. Datasets from the AbdomenCT class will be added to both the global training and global testing data and the model will continue to train on N datasets per “day”, effectively transforming the model into a 4-way classifier. The results from this experiment are meant to show how introducing a previously unseen class can significantly impact a network that is reaching desirable accuracy and loss metrics. We also hope to explore the number of datasets and sequential learning “days” it would take for the network to recapture the previously reached accuracy and loss metrics with the newly added class.

Methods

The CXR, Hand, and HeadCT classes from the Medical MNIST dataset were used for training, validation, and testing for a multi-classifier network. There were 30,000 CXR, Hand, and HeadCT datasets pre-set from each associated “filenames” text file with the following splits: 70% training data, 10% validation data, and 20% testing data.

**Batch pre-training:** The PyTorch ResNet50 CNN was trained with the parameters batch size of 16, five epochs, and LR of 1e-6. Similar to the baseline traditional training in Section 2.3.1, we used a subset of the global training data, which consists of 500 images, and validated on the pre-training results against the global validation data. Each of these splits are defined in greater detail in Data Section 2.2.2 above. After training on a limited number
of epochs, we expect the pre-training to reach a desirable accuracy threshold (~70%) before the model shifts to training in a sequential learning format.

**Sequential training:** The CNN was further trained each “day” on N = 20 datasets recruited from the global training data for one day-epoch over a period of 400 “days”. We again used a test dataset as a held-out sample of the population to be evaluated against the model after training each “day”. As the model sequentially trains as a 3-way classifier over the first 200 “days” of learning, we expect it to reach close to fully trained (90+%) metrics. On “day” 201, the AbdomenCT class and its datasets will be introduced to the network. The global testing data will acquire 670 AbdomenCT datasets, and all remaining AbdomenCT datasets will be added to the unused global training data and then shuffled. Unlike before where sequential learning would recruit 20 datasets per “day”, after the new class is added we would start recruiting N = 50 datasets per “day” in an effort to retrain the network at a faster rate.

**Sequential training, no pre-training:** Similar to before, the PyTorch ResNet50 CNN was trained with the parameters batch size of 16 and a LR of 1e-6. Because there is no pre-training with this method, we sequentially trained over a period of 500 “days” while also recruiting N = 20 datasets per “day” from the global training data. The increased number of “days” is meant to show a comparable number of datasets being used between methods. Without pre-training, the model would learn from the training data as a 3-way classifier for the first 250 “days”. The AbdomenCT class would be introduced on “day” 251, and the training and testing splits would be created in the same way as before. Here we similarly increase to N = 50 datasets recruited per “day” for the remaining training “days” in an effort to retrain the network at a faster rate.
Results

**Batch pre-training:** The results in Figure 8 are shown from batch pre-training for five epochs with a LR of 1e-6. We can see the accuracy and loss curves of the traditional multi-classifier network as a function of epochs (accuracy) and cross-entropy loss. Like the pre-training in Section 2.3.1, we can observe how the accuracy can reach ~70% even with a limited sample size. In fact, the near-identical pre-training metrics to Section 2.3.1 indicate a consistency being met across experiments. The final accuracy metrics during pre-training were similarly deemed a good starting point for the CNN before it begins sequential learning. The loss metrics also show a steady decrease but do not appear anywhere near stabilizing (extracting as much information as possible from the training datasets).
Sequential training: The results in Figure 9 represent the sequentially trained network for one day-epoch over 400 “days”. The pre-trained model from Figure 8 is carried into sequential training and trains on 20 datasets recruited each “day” from the global training data. Below we can see how the starting point of testing accuracy appears around the ending point of validation accuracy during pre-training. The accuracy metrics oscillate for the first 200 “days” similar to Section 2.3.1. In fact, the testing accuracy trends in an identical manner to the baseline sequential learning experiment, once again indicating consistency with the global training data being met across experiments. On “day” 201, we
can observe a precipitous dip in both accuracy metrics due to the appearance of the newly added class. The model is unfamiliar with this class, and similar to the first epoch of pre-training requires several datasets with the new class to improve its confidence in making accurate classifications. Increasing the recruited datasets to 50 each “day” from the previous 20 is meant to increase this retraining rate. For the first 50 “days” of retraining, the model appears stagnant and no observable progress is made as the testing accuracy is stable at \( \sim 70\% \). Slowly however it appears to retrain over the subsequent 100 “days” as the testing accuracy rises from \( \sim 70\% \) to \( \sim 90\% \). The final 50 “days” of retraining shows the training accuracy oscillations stabilizing between 80-90\% accuracy, and the testing accuracy is maintained at \( \sim 93\% \). The loss metrics show a steady decrease and trend in an identical manner as Section 2.3.1. The spike that appears on “day” 201 for both loss metrics indicates the lack of familiarity with new class data. The loss steadily decreases over the final 200 “days” of retraining although it does not slow down as it approaches “day” 400. This current rate of decrease indicates that the network can continue extracting information past “day” 400 from the 4-class datasets even though it can confidently classify on them.
Figure 9: The sequentially trained CNN following pre-training for one day-epoch. The accuracy of the 3-way classifier (CXR vs. Hand vs. HeadCT) as a function of the first 200 “days” (top). The loss of the 3-way classifier as a function of cross-entropy loss (bottom) per “day” for the first 200 “days”. Following “day” 200, the accuracy and loss is for a 4-way classifier (CXR vs. Hand vs. HeadCT vs. AbdomenCT). The training metrics are for the 20 datasets evaluated by the CNN each “day” for the first 200 “days” followed by 50 datasets evaluated each “day” for the final 200 “days”. The testing metrics are evaluated by the CNN at the end of each “day” on the set aside testing data.

**Sequential training, no pre-training:** The results in Figure 10 represent the sequential trained network that begins learning without any pre-training for one day-epoch over a period of 500 “days”. The lack of pre-training can be distinguished by the testing accuracy that begins at ~30%. Additionally, the testing accuracy trends in an identical manner to Section 2.3.1, where around “day” 200 both appear to stabilize at ~80%. On “day” 251 the same precipitous dip we had seen previously appears in the training accuracy curve due to the newly added class. Because the model requires several datasets with the new class to improve its classification accuracy, we again increased the recruited datasets to 50 each “day” from the previous 20 to increase the retraining rate. For the first 50 “days” of
retraining, the training oscillations are quite low but stable between 40-70%. The testing accuracy also appears unchanged for this period. The subsequent 100 “days” however both accuracy metrics steadily increase, where the testing accuracy reaches ~80% accuracy once more and the training accuracy oscillates between 60-80% as the model better identifies the AbdomenCT data. Over the final 100 “days” of retraining, we can observe how the training oscillations stabilize between 80-95% accuracy and the testing accuracy is maintained at ~90%. The loss metrics also decrease at a steady yet similar rate over the first 200 “days” compared to Section 2.3.1. There is a spike which appears on “day” 251 due to a lack of familiarity with the newly introduced class and steadily decreases over the final 250 “days” of retraining. As the loss curves approach “day” 500, the rate at which they decrease does not appear to slow down, indicating that the network could continue extracting information past “day” 500 from the 4-class datasets even though it can confidently classify on them.
Figure 10: The sequentially trained CNN without pre-training for one day-epoch. The accuracy of the 3-way classifier (CXR vs. Hand vs. HeadCT) as a function of the first 250 “days” (top). The loss of the 3-way classifier as a function of cross-entropy loss (bottom) per “day” for the first 250 “days”. Following “day” 250, the accuracy and loss is for a 4-way classifier (CXR vs. Hand vs. HeadCT vs. AbdomenCT). The training metrics are for the 20 datasets evaluated by the CNN each “day” for the first 250 “days” followed by 50 datasets evaluated each “day” for the final 250 “days”. The testing metrics are evaluated by the CNN at the end of each “day” on the set aside testing data.
2.3.3 Sequential Learning, Added Class, Varying Daily Datasets (Experiment 3a & 3b)

Goal

The goal of this experiment is to perform sequential training on a 3-way classifier with and without pre-training in a manner identical to Section 2.3.2. For both methods during sequential training, a new class of Medical MNIST data, AbdomenCT, will be introduced to the CNN. Datasets from the AbdomenCT class will be added to both the global training and global testing data and the model will continue to train on N datasets per “day”, effectively transforming the model into a 4-way classifier. In this experiment, the number of N datasets are randomized between 15-30 to mimic a real-life learning scenario where the number of acquired class datasets are variable.

Methods

The CXR, Hand, and HeadCT classes from the Medical MNIST dataset were used for training, validation, and testing for a multi-classifier network. There were 30,000 CXR, Hand, and HeadCT datasets pre-set from each associated “filenames” text file with the following splits: 70% training data, 10% validation data, and 20% testing data.

Batch pre-training: The PyTorch ResNet50 CNN was trained with the parameters batch size of 16, five epochs, and LR of 1e-6. Similar to the baseline traditional training in Section 2.3.1 and 2.3.2, we used a subset of the global training data, which consists of 500 images, and validated on the pre-training results against the global validation data. Each of these splits are defined in greater detail in Data Section 2.2.2 above. After training on a limited number of epochs, we expect the pre-training to reach a desirable accuracy threshold (~70%) before the model shifts to training in a sequential learning format.
Sequential training: The CNN was further trained each “day” on N datasets recruited from the global training data for one day-epoch over a period of 400 “days”. Here, N is picked between 15-30 at random and trains on an ever-changing random number of datasets each “day”. Further details on the sequential training performed here can be referred to in Section 2.3.2. On “day” 201 where the new AbdomenCT class is introduced to the network, the model starts recruiting 45-60 datasets per day. The variable number of daily datasets simulates real-life scenarios where there is no fixed number of datasets acquired each “day”.

Sequential training, no pre-training: Similar to before, the PyTorch ResNet50 CNN was trained with the parameters batch size of 16 and a LR of 1e-6. Further details on sequential training without pre-training can be referred to in Section 2.3.2. The number of daily datasets again fluctuates between 15-30 for the first 250 “days”, and 45-60 datasets for the final 250 “days”.

Results

Batch pre-training: The results in Figure 11 are shown from batch pre-training for five epochs with a LR of 1e-6. We can see the accuracy and loss curves of the traditional multi-classifier network as a function of epochs (accuracy) and cross-entropy loss. Like the pre-training in Sections 2.3.1 and 2.3.2, we can observe how the accuracy can reach ~70% even with a limited sample size. We again treated the final accuracy metrics during pre-training as a starting point for the CNN before it begins sequential learning. The loss metrics also show a steady decrease but do not appear anywhere near stabilizing (extracting as much information as possible from the training datasets).
Figure 11: The pre-training accuracy of the 3-way classifier (CXR vs. Hand vs. HeadCT) as a function of epochs (top). The pre-training loss of the 3-way classifier as a function of cross-entropy loss (bottom) per epoch. This is shown for training and validation metrics.

Sequential training: The results in Figure 12 represent the sequentially trained network for one day-epoch over 400 “days”. The pre-trained model from Figure 11 is carried into sequential training and trains on 15-30 datasets recruited each “day” from the global training data. Below we can see how the starting point of testing accuracy appears around the ending point of validation accuracy during pre-training. The accuracy metrics show similar trends for the first 200 “days” to Section 2.3.2 that differ very slightly due to a variable amount of training datasets acquired each passing “day”. On “day” 201 there is a similar precipitous dip in both accuracy metrics as in 2.3.2 due to the newly added class,
and the network would need to recruit an increased number of datasets with the new class to make accurate classifications. Though we are continuing to add a variable number of N datasets past “day” 201, we would increase the acquisition to 45-60 datasets each “day” to speed up the retraining rate. We can observe how the accuracy and loss trends for the final 200 “days” also show strong correlations with Section 2.3.2.

**Figure 12:** The sequentially trained CNN following pre-training for one day-epoch. The accuracy of the 3-way classifier (CXR vs. Hand vs. HeadCT) as a function of the first 200 “days” (top). The loss of the 3-way classifier as a function of cross-entropy loss (bottom) per “day” for the first 150 “days”. Following “day” 200, the accuracy and loss is for a 4-way classifier (CXR vs. Hand vs. HeadCT vs. AbdomenCT). The training metrics are for the 15-30 datasets evaluated by the CNN each “day” for the first 200 “days” followed by 45-60 datasets evaluated each “day” for the final 200 “days”. The testing metrics are evaluated by the CNN at the end of each “day” on the set aside testing data.

**Sequential training, no pre-training:** The results in Figure 13 represent the sequential trained network that begins learning without any pre-training for one day-epoch over a period of 500 “days”. The testing accuracy which begins at ~25% is nearly identical to that
in Section 2.3.2 and trends very similarly for the first 250 “days”. The testing accuracy is stabilized at ~85% until “day” 251 where a precipitous dip occurs due to the newly added class. Similar to before, we are continuing to recruit 15-30 datasets per “day” right until “day” 251 where we would increase the number to 45-60 datasets recruited each “day” to speed up the retraining rate. We can observe how the accuracy and loss trends for the final 250 “days” also show strong correlations with Section 2.3.2.

![Sequential Accuracy and Loss](image)

**Figure 13:** The sequentially trained CNN without pre-training for one day-epoch. The accuracy of the 3-way classifier (CXR vs. Hand vs. HeadCT) as a function of the first 250 “days” (top). The loss of the 3-way classifier as a function of cross-entropy loss (bottom) per “day” for the first 250 “days”. Following “day” 250, the accuracy and loss is for a 4-way classifier (CXR vs. Hand vs. HeadCT vs. AbdomenCT). The training metrics are for the 15-30 datasets evaluated by the CNN each “day” for the first 250 “days” followed by 45-60 datasets evaluated each “day” for the final 200 “days”. The testing metrics are evaluated by the CNN at the end of each “day” on the set aside testing data.
2.4 Discussion

Although sequentially trained CNNs for image classification have been increasingly implemented in recent years, there are limited studies which explore medical imaging data for such networks. As such, our preliminary task was to explore baseline experiments that would establish sequentially trained CNNs as viable alternatives to traditional neural networks with the Medical MNIST dataset. In this chapter we have described methods that would train a multi-classification model from scratch until it can confidently classify different medical imaging modalities 90+% of the time. We have shown we can improve the classification accuracy for an untrained model by recruiting and training on datasets in a sequential format over an appropriate number of training “days”. Additionally, if a research team had access to a subset of the training data population, then they can pre-train a model to a desirable starting point (~70%) and then continue refining it with datasets acquired sequentially post hoc. We have shown that no matter the beginning approach, whether your CNN will or will not incorporate pre-training, the end result will be the same. By adding in a previously unseen class to the network during sequential training we have also shown that the model can adapt to new information. Although, the model maintains difficulty retaining information from previously seen datasets once a new class is observed. Future experiments can explore the effect of sequential training on new datasets acquired per “day” in addition to training on used data to avoid concept drift. However, in the current framework the sequentially acquired datasets each “day” only see a single pass of training. We hypothesize that a single pass, or epoch, is not enough for a model to fully extract the information on data recruited each “day”. In Chapter 3, we aim to further explore sequential learning CNNs through experiments that evaluate multiple day-epochs. We will also explore different ways to recruit validation data that is most appropriate for a sequential learning pipeline.
Chapter 3: Sequential Learning Experimentation

3.1 Background

As datasets arrive in the sequential learning pipeline, one question remains: how to extract the full information on the data for each “day”? There would ideally be a one-size-fits-all scenario with an equal number of datasets arriving sequentially. However, in real-world scenarios the number of relevant cases could vary on a day-to-day basis. There is also added difficulty with certain medical imaging modalities that include contrast resolution, noise, or even organ complexities that can make it challenging for a classification model. With certainty we can assume that one training loop (epoch) of the sequential learning data is insufficient to extract maximum information from the datasets in a given “day”. Because of this, it is important to determine an appropriate epoch count to maximize the information extraction of sequentially obtained data without overtraining a model to the point of overfitting on the data. Usually in a traditional training model there is a validation dataset set aside to validate on the training performance of the neural network. Often with sequential learning though, there is only a small set of data which cannot be set aside for validation purposes. From this we estimated two methods of quantifying validation data most appropriate for a sequentially trained neural network.

Work done by other researchers have shown the ability to deal with data shifts that can arise due to new emergent data sources at unknown time points. Though there are various styles observed in sequential learning methodologies the goal is to ensure the full information is extracted from medical images in a continuous data stream. Perkonigg et al. proposed their DM strategy to continuously update parameters of a task model with novel data characteristics while sustaining diversity of the entire cohort [9]. At each step of the defined task, data from the samples stored in memory would be used to train the model along with new examples from the oncoming data stream. This approach allows
the model to consistently extract information from older datasets by reintroducing them during continual training, ensuring the memory is diverse and representative of all visual variations [9]. Another approach proposed by Ebrahimi et al., coined as adversarial continual learning (ACL), learns a private latent space for a defined task and a shared feature space to enhance knowledge transfer and better recall previously learned tasks [13]. This approach makes the shared features less prone to forgetting due to the replay buffer mechanism, a system which takes information from samples used in prior tasks to help with better factorization and improve model performance [13]. Mittal et al. proposed a class-incremental learning technique that identifies the poor quality of learned representation due to effects of overfitting and loss of secondary class information [14]. They had shown this in an initial base task that would be followed by an incremental learning task. The base network was trained for 500 epochs and would see the validation loss overfit after roughly 100 epochs. The results had shown the increasing validation loss caused a drop in the performance of the corresponding incremental learning model. This was addressed through regularization techniques, such as data augmentation and self-distillation, that would boost the average incremental accuracy while mitigating the risk for overtraining on the data, allowing the network to retain high amounts of secondary information [14].

In this chapter, we experimented with two methods of establishing a training and validation dataset which could be used each “day” in the sequential learning loop. For both experiments, the validation dataset would be used as a surrogate for a true batch-style validation dataset. Section 3.3.1 describes a standard image classification network using a traditional batch-style process to determine baseline accuracy and loss metrics that would be compared to the sequential learning metrics. What follows in Section 3.3.2 describes a sequential learning network trained with N datasets used for training each “day” without replacement and one epoch per “day”. In Section 3.3.3 we then train the
same sequential learning network using multiple day-epochs with the two different methods of generating training and validation data. For each experiment, we trained a sequential learning network to obtain accuracy and loss metrics for a single iteration. We then trained 10 iterations of the network and took an average of the resulting accuracy and loss metrics. This allowed us to focus on the trends of the data by refining subtle training differences that would follow single runs of the network.

3.2 Data and Model

We utilized the publicly available Medical MNIST imaging dataset for our traditional and sequential learning experiments [12]. Further information on the dataset can be found in Data Section 2.2.1. Unlike Chapter 2, we will be using only the three classes of CXR, Hand, and HeadCT in the following experiments. The training, validation, and testing splits are as they were in Chapter 2 and remain consistent for every experiment listed below. The previously generated text files described in Section 2.2.2 would similarly be used in the following experiments to ensure the same data splits and to maintain consistent findings across the individual CNN experiments. Each CNN utilizes the PyTorch ResNet50 model as the image classification algorithm. Data augmentation, described in further detail in Section 2.2.3, is then applied on the training, testing, and validation datasets across all experiments. The cost function and optimizer are described in Section 2.2.4, and the number of N datasets recruited per “day” varied with each experiment along with the total “days”, total day-epochs, and model LR.
3.3 Experiments

Three experimental frameworks were explored to compare sequential learning to a traditionally trained neural network. Experiment 0 represents the baseline traditional training method for a 3-way image classifier. The results from this experiment will be compared to the final results obtained from the sequential learning experiments to ensure cohesion between frameworks. The sequential learning will be done across three experiments: Experiments 1, 1a, and 1b. Experiment 1 is the sequential learning model with 3-way classification that uses the default approach where training is accomplished on N datasets arriving each “day” with a single day-epoch and no validation data. After training each “day”, the model is evaluated on the testing data. Experiment 1a is the sequential learning model with 3-way classification that will use N datasets arriving in the “present day” as validation data, and N datasets from the “previous day” as training data. Experiment 1b is a similar sequential learning model with a 3-way classification that will combine N/2 datasets from the “present day” and N/2 datasets from the “previous day” as training data, and the remaining N/2 datasets from the “present day” will be used as validation data. For both Experiments 1a and 1b, we will adjust the hyperparameters from the default approach in Experiment 1 where training per day is done with a single day-epoch by training with fewer overall days and multiple day-epochs. We will also run multiple iterations of Experiments 1a and 1b and evaluate the average accuracy and loss metrics against the single run experiments. Lastly, we will evaluate a version of Experiment 1b that renews the PyTorch model weights after each “day”. We expect the model to fully train (90+% accuracy) on the daily recruited datasets and have the full information extracted.
Figure 14: Breakdown of experiments that establish training and validation datasets used in a given day. For the first day, only the present day’s data is used for training and validation splits. For day N, which represents any day that follows the first day, both the previous day and present day’s combined data is used for the defined training and validation splits.
3.3.1 Baseline Traditional Batch Training (Experiment 0)

Goal

The goal of this experiment is to establish a baseline 3-way classifier to determine the overall accuracy and loss expected from a traditionally trained network. The model would be trained until it starts to overfit based on the validation loss. These metrics would be expected from sequential learning on the same 3 class data (CXR, Hand, HeadCT).

Methods

The CXR, Hand, and HeadCT classes from the Medical MNIST dataset were used for training, validation, and testing for a multi-classifier network. There were 30,000 CXR, Hand, and HeadCT datasets pre-set from each associated “filenames” text file with the following splits: 70% training data, 10% validation data, and 20% testing data. Each of these splits are defined in greater detail in Data Section 2.2.2 above. The PyTorch ResNet50 CNN was trained with the parameters batch size of 16, 30 epochs, and LR of 1e-6. After training for the defined number of epochs, we expect the model to reach a fully trained accuracy threshold (90+%).

Results

The results in Figure 15 are shown from traditional training across 30 epochs with a LR of 1e-6. We can see the accuracy and loss curves of the traditional multi-classifier network as a function of epochs (accuracy) and cross-entropy loss. We can also observe that the number of epochs needed to reach 99+% accuracy across all metrics is roughly five epochs. This is shown for the training, validation, and testing metrics.
Figure 15: The accuracy of the 3-way classifier (CXR vs. Hand vs. HeadCT) as a function of epochs (top). The loss of the 3-way classifier as a function of cross-entropy loss (bottom) per epoch. This is shown for training, validation, and testing metrics.

As we can see above, the traditional training network has shown successful results for both accuracy and loss metrics. By the end of training on the fifth epoch, we approach ~99% accuracy across all metrics and the loss curves mostly bottom out around this time as well.
3.3.2 Baseline Sequential Learning - Single Epoch Per Day (Experiment 1)

Goal

The goal of this experiment is to evaluate and compare the baseline sequential learning network to the traditional training network. Because sequential learning with few $N$ datasets arriving per day does not necessarily allow for a separate validation dataset to track overfitting, it is important to evaluate ways to extract as much information as possible for a particular day's datasets. First, we will train a sequential learning network with one day-epoch to establish a baseline for the 3-way classifier. This base framework will confirm the results that sequential learning attains comparable results to the traditional training neural network.

Methods

The CXR, Hand, and HeadCT classes from the Medical MNIST dataset were used for training, validation, and testing for this sequential learning network. From the global training data, $N = 50$ datasets would be recruited each “day” without replacement for sequential training over a period of 200 “days”. The PyTorch ResNet50 CNN was trained with the parameters batch size of 16, one day-epoch, and LR of 1e-6. Following the conclusion of each “day’s” training for this experiment, the CNN was tested against the global testing data defined in Data Section 2.2.2 above. The test set of data is not traditionally evaluated until after the neural network is completely trained and no decisions on training are applied to the test data set. In this case we are using the test dataset as a held-out sample of the population which is evaluated after the training each “day” to understand the training. Normally a test set may not be available in a sequential learning scheme.
Results

The results shown in Figure 16 represent the sequentially trained network out to 200 “days” with one day-epoch per “day” and a LR of 1e-6. Each “day” the network was trained on the 50 datasets recruited in the “present day” without replacement. These datasets were selected from the global training datasets that have been pre-set. We can observe the accuracy and loss curves for the training and testing data of the multi-classifier network as a function of “days” (accuracy) and cross-entropy loss. We can also observe how the sequential trained model requires the full 200 “days” to reach 90+% accuracy across both training and validation metrics. Additionally, the training accuracy oscillates wildly over the first 50 “days” between ~20-60% accuracy, but mostly stabilizes to 80-90% over the final 50 “days”. Similarly, both loss metrics continue to drastically decrease until “day” 200, although they do not appear to reach stabilization. This would indicate that the sequential learning network can continue to train on data past “day” 200 as the information has not been fully extracted from the recruited datasets.
Figure 16: The baseline sequentially trained CNN for one epoch per “day”. The accuracy of the 3-way classifier (CXR vs. Hand vs. HeadCT) as a function of “days” (top). The loss of the 3-way classifier as a function of cross-entropy loss (bottom) per “day”. The training metrics are for the 50 datasets evaluated by the CNN each “day”. The testing metrics are evaluated by the CNN at the end of each “day” on the set aside testing data.
3.3.3 Sequential Learning - Multiple Epochs Per Day

3.3.3.1 Train on previous day’s data & validate on present day’s data (Experiment 1a)

Goal

The goal of this experiment is to evaluate how consistently updated validation data would affect the number of day-epochs in the sequential learning neural network. Before we had used a single day-epoch as an established baseline for the 3-way classifier. However, we expect multiple day-epochs would be needed to extract as much information as possible for a particular “day’s” datasets. We will begin by training a sequential learning network with multiple day-epochs and compare these results with that of the sequential learning network trained on one day-epoch. We will also be comparing results from two runs of the model to ensure consistent findings.

Further evaluation will be done on the hyperparameters of Experiment 1a with multiple day-epochs as we expect different findings after two separate runs. To reduce the variation and obtain comparable results, we want to run several iterations of that same experiment and average the resulting metrics to focus on trends. We will train a sequential learning network like before with multiple day-epochs and compare these results with that of the sequential learning network trained on one day-epoch.

Methods

The CXR, Hand, and HeadCT classes from the Medical MNIST dataset were used for training, validation, and testing for a sequential-learning network. From the global training data, N = 20 datasets would be recruited each “day” without replacement for sequential
training over a period of five “days”. The PyTorch ResNet50 CNN was trained with the parameters batch size of 16, 150 day-epochs, and LR of 1e-6.

The validation data was not pre-set in this experiment. Instead, the N datasets arriving in the “present day” would be used as validation data, and the N datasets from the “previous day” would be used to train the model. Because there is no “previous day” on “day” one, there is no training performed on “day” one and instead the CNN begins with validation. It is important to note that the validation data defined here is different from the global validation data used in the traditional training experiment. This experiment did not use the global validation data as it would not traditionally be available in a sequential learning paradigm. Following the conclusion of each “day’s” training for this experiment, the CNN was tested against the global testing data. Normally a test set may not be available in a sequential learning scheme.

We will further evaluate the network by running 10 iterations in succession with the same hyperparameters, where following the conclusion of one iteration the resulting metrics will be appended to a running list. After all 10 iterations are completed, an average of the iterations will be taken and displayed for the accuracy and loss curves.

Results

The results shown in Figure 17 represent the sequentially trained network out to five “days” with 150 day-epochs per “day” and a LR of 1e-6. Each “day” the network was trained on 20 datasets from the “previous day” and validated on the 20 datasets arriving in the “present day” without replacement. These datasets were selected from the global training data that has been pre-set. Because there is no “day” prior to “day” one, no training was performed on “day” one. We can observe the accuracy and loss curves for training,
validation, and testing data of the multi-classifier network as a function of epochs (accuracy) and cross-entropy loss. To ensure consistency, we ran the CNN twice with the same hyperparameters and compared the findings. We can observe slight differences between the two runs. The accuracy curve oscillates more wildly during “day” two in the first run and reaches a 90+% accuracy threshold sooner compared to the second run. This would indicate that the model trains slightly better in the first run and is confirmed by the testing accuracy which is ~70% as opposed to the ~60% in the second run, both being on day two. For “day” four, the model in the second run maintains a stronger understanding of the 3-class datasets and oscillates primarily between 80-100%. This performance is slightly better compared to the first run where the first half of “day” four training oscillates between 60-90%. For the final “day”, the accuracy metrics across both runs show no visual distinction. These differences over the duration of sequential training would indicate variations between each run which we hope to reduce by running multiple iterations of this experiment and averaging the resulting metrics. The training loss curves for both runs on the other hand decrease at a near identical rate and appear to similarly stabilize towards the end of “day” five. Evaluating the validation metrics for both runs show no distinct differences except during “day” four where the validation accuracy in the first run oscillates slightly more inconsistent compared to the second run. Interestingly, on “day” four the validation loss for both runs appear to exclusively show signs of catastrophic forgetting that is likely attributed to the datasets which are recruited on that “day”. On “day” five, the validation loss decreases and appropriately stabilizes as the full information of the recruited data is extracted.
Figure 17: The sequentially trained CNN for multiple day-epochs. On top is the first run of this experiment, and on bottom is the second run. For each run, the accuracy of the 3-way classifier (CXR vs. Hand vs. HeadCT) as a function of “days” and day-epochs (top). The loss of the 3-way classifier as a function of cross-entropy loss (bottom) per day-epoch. The training metrics are for the 20 datasets evaluated by the CNN from each “previous day”. We can observe how the training curves begin on the second “day”. The validation metrics are evaluated by the CNN for the 20 datasets acquired in the “present day”. The testing metrics are evaluated by the CNN at the end of each “day” on the set aside testing data.
The results shown in Figure 18 represent the averaged sequentially trained network out to five “days” with 150 day-epochs per “day” and a LR of 1e-6 across 10 iterations. Because the curves represent an average of 10 iterations of the network, more often than not these results are what is expected for the CNN with the aforementioned hyperparameters. The accuracy curves steadily increase during training on “days” two and three before reaching a 90+% threshold roughly ¾ into training on “day” three. On “day” four the model experiences some catastrophic forgetting and dips initially to <80% training accuracy and ~60% validation accuracy before being sequentially trained back to ~90% accuracy for both metrics. On “day” five the model shows strong understanding of the data with little oscillation at a 90+% accuracy threshold. The loss curves reflect these same trends as they decrease steadily on “days” two and three before stabilizing towards the end of “day” five training. The same occurrence of what appears to be catastrophic forgetting on “day” four is likely attributed to the datasets which are recruited on that “day”.
Figure 18: The sequentially trained CNN for multiple day-epochs averaged across 10 noise iterations. The accuracy of the 3-way classifier (CXR vs. Hand vs. HeadCT) as a function of “days” and day-epochs (top). The loss of the 3-way classifier as a function of cross-entropy loss (bottom) per day-epoch. The training metrics are for the 20 datasets evaluated by the CNN from each “previous day”. We can observe how the training curves begin on the second “day”. The validation metrics are evaluated by the CNN for the 20 datasets acquired in the “present day”. The testing metrics are evaluated by the CNN at the end of each “day” on the set aside testing data.
Goal

The goal of this experiment is to evaluate how consistently updated validation data would affect the number of day-epochs in the sequential learning neural network. Before we had used a single day-epoch as an established baseline for the 3-way classifier. However, we expect multiple day-epochs would be needed to extract as much information as possible for a particular “day’s” datasets. Similar to experiment 1a, we will train a sequential learning network with multiple day-epochs and compare these results with that of the sequential learning network trained on one day-epoch. We will also be comparing results from two runs of the model to ensure consistent findings.

Like before, we will further evaluate the hyperparameters of Experiment 1b with multiple day-epochs as we expect different findings after two separate runs. To reduce the variation and obtain comparable results, we want to run several iterations of that same experiment and average the resulting metrics to focus on trends. We will train a sequential learning network with multiple day-epochs and compare these results with that of the sequential learning network trained on one day-epoch. Additionally, we will evaluate a version of this experiment that renews the model weights after each “day” over several “days” to determine if in fact the model can reach 90+% accuracy for each isolated training “day” on the recruited datasets.

Methods

The CXR, Hand, and HeadCT classes from the Medical MNIST dataset were used for training, validation, and testing for a sequential-learning network. From the global training data, N = 20 datasets would be recruited each “day” without replacement for sequential
training over a period of five “days”. The PyTorch ResNet50 CNN was trained with the parameters batch size of 16, 150 day-epochs, and LR of 1e-6.

Unlike the previous experiment, N/2 of the datasets arriving in the “present day” combined with N/2 of the datasets from the “previous day” would be used as training data for the model. The validation data was not pre-set in this experiment either, and the remaining N/2 datasets from the “present day” would be used as the validation data. Like before, it is important to note that the validation data defined here is different from the global validation data used in the traditional training experiment. This experiment did not use the global validation data as it would not traditionally be available in a sequential learning paradigm. On “day” one, because there is no “previous day” only 10 datasets would be used for training. The CNN was trained each “day” with N = 20 datasets (without replacement). Following the conclusion of each “day’s” training for this experiment, the CNN was tested against the global testing data. Normally a test set may not be available in a sequential learning scheme.

We will further evaluate the network by running 10 iterations in succession with the same hyperparameters, where following the conclusion of one iteration the resulting metrics will be appended to a running list. After all 10 iterations are completed, an average of the iterations will be taken and displayed for the accuracy and loss curves. Finally, we will run a variant of this CNN that renews the model weights after each “day” over a 10 “day” period, where we will only adjust the number of N datasets recruited daily to N = 50.

Results

The results shown in Figure 19 represent the sequentially trained network out to five “days” with 150 day-epochs per “day” and a LR of 1e-6. Each “day” the network was trained on
20 datasets (N/2 from the “present day” and N/2 from the “previous day”) and validated on 10 datasets (remaining N/2 from the “present day”) without replacement. These datasets were selected from the global training data that has been pre-set. Because there is no “day” prior to “day” one, only N/2 datasets were used for training on “day” one. We can observe the accuracy and loss curves for training, validation, and testing data of the multi-classifier network as a function of epochs (accuracy) and cross-entropy loss. To ensure consistency, we ran the CNN twice with the same hyperparameters and compared the findings. We can observe slight differences between the two runs for this experiment along with a few similarities. The training accuracy for both runs, though it oscillates greatly over the first two “days”, steadily increases and fluctuates between 80-100% accuracy roughly halfway through “day” two training. In fact, over the final three “days” all accuracy metrics for the first run appear stable and oscillate between 90-100% accuracy. The second run performs slightly worse over “days” three and four as shown by the oscillations between 80-100% accuracy before reaching 90-100% on “day” five. Additionally, the first half of “day” four training during the second run displays signs of catastrophic forgetting as had appeared in experiment 1a before the network self-corrects over the second half of “day” four training. This similarly occurs for the first run but to a lesser degree. Another key difference can be seen during “day” two training, where the second run has its validation accuracy stabilized to oscillate between ~70-100% compared to the first run that is more unstable and oscillates between 40-90%. This alone would indicate variability across runs that would be reduced by running multiple iterations of this experiment and averaging the resulting metrics. The loss curves steadily decrease over the five “days” for both runs as well and appear to stabilize towards the end of “day” five. There are additional differences which can be seen between the two runs, particularly on “days” three and four.
Figure 19: The sequentially trained CNN for multiple day-epochs. On top is the first run of this experiment, and on bottom is the second run. For each run, the accuracy of the 3-way classifier (CXR vs. Hand vs. HeadCT) as a function of "days" and day-epochs (top). The loss of the 3-way classifier as a function of cross-entropy loss (bottom) per day-epoch. The training metrics are for the 20 datasets evaluated by the CNN (N/2 from the “present day” and N/2 from the “previous day”). The validation metrics are evaluated by the CNN for the remaining N/2 datasets acquired in the “present day”. The testing metrics are evaluated by the CNN at the end of each “day” on the set aside testing data.
The results shown in Figure 20 represent the averaged sequentially trained network out to five “days” with 150 day-epochs per “day” and a LR of 1e-6 across 10 iterations. Because the curves represent an average of 10 iterations of the network, more often than not these results are what is expected for the CNN with the aforementioned hyperparameters. The accuracy curves steadily increase and ultimately reach a 90+% threshold towards the end of “day” one training. Though the training accuracy initially sees some catastrophic forgetting on “day” two, the model quickly self corrects to 90+% accuracy halfway through “day” two training. Over the final three “days”, the model shows strong understanding of the data with little oscillation between 80-100% accuracy. The loss curves reflect these same trends as they steadily decrease over the first two “days” before stabilizing early on during “day” five training. The same occurrence of what appears to be catastrophic forgetting can be seen on “day” four that is likely attributed to the datasets which are recruited on that “day”.


Figure 20: The sequentially trained CNN for multiple day-epochs averaged across 10 noise iterations. The accuracy of the 3-way classifier (CXR vs. Hand vs. HeadCT) as a function of “days” and day-epochs (top). The loss of the 3-way classifier as a function of cross-entropy loss (bottom) per day-epoch. The training metrics are for the 20 datasets evaluated by the CNN (N/2 from the “present day” and N/2 from the “previous day”). The validation metrics are evaluated by the CNN for the remaining N/2 datasets acquired in the “present day”. The testing metrics are evaluated by the CNN at the end of each “day” on the set aside testing data.

The results shown in Figure 21 represent the sequentially trained network out to 10 “days” with 150 day-epochs per “day” and a LR of 1e-6. These curves represent isolated training scenarios due to the model weights being reset after the conclusion of each “day’s” training. We can observe how the accuracy metrics at the beginning of each training “day” are reset as they all begin at <40%. The CNN trains on newly recruited datasets over the day-epochs and sees all accuracy metrics reach 90+% before the subsequent “day” begins. Because this trend is shown for the entire period of training “days”, we speculate that the multi-classification task on the chosen Medical MNIST dataset is not challenging enough for our sequential learning network.
Figure 21: The sequentially trained CNN for multiple day-epochs with the model weights renewed each "day" over 10 "days". The accuracy of the 3-way classifier (CXR vs. Hand vs. HeadCT) as a function of "days" and day-epochs (top). The loss of the 3-way classifier as a function of cross-entropy loss (bottom) per day-epoch. The training metrics are for the 50 datasets evaluated by the CNN (N/2 from the "present day" and N/2 from the "previous day"). The validation metrics are evaluated by the CNN for the remaining N/2 datasets acquired in the "present day". The testing metrics are evaluated by the CNN at the end of each "day" on the set aside testing data.
3.4 Discussion

Our sequential training experiments that train on multiple day-epochs each “day” have shown the benefit of full information extraction within a data stream. This can be seen across Figures 17-21 that show strong likelihood of training metrics reaching 90+% after each single “day’s” worth of training. Compared to the baseline sequential training experiment in Section 3.3.2, it is far more advantageous to train over multiple day-epochs to allow the network more time to learn on the recruited datasets. This provides an inverse relationship between day-epochs and “days”, where more day-epochs would mean fewer “days” necessary to reach fully trained accuracy metrics and vice-versa. It is necessary to incorporate the generated text files described in Section 2.2.2 to better evaluate the findings across the individual CNN experiments that undergo distinctive validation data recruitment. This can be seen across the five sequential training “days” for both experiments in Section 3.3.3, where the trends remain identical during any reruns of the sequential learning networks. This is due to the same training and validation data subsets being recruited each “day”. The averaged metrics image for both experiments would also not be possible if the trends were inconsistent over the 10 noise iterations, making the argument more plausible on how the isolated catastrophic forgetting that occurs on “day” four for each experiment is due to the datasets recruited on that “day”. Currently, we are recruiting N = 20 datasets without replacement from the global training data over the five “day” period beginning with the first file. We can potentially explore outcomes where we would skip over the first 100 datasets in the global training data repository and observe the effect that would have on the network. If the observed forgetting is indeed dataset-driven, then updating the data recruited on “day” four should completely change the accuracy and loss metrics.
It is important however to note the downsides to sequential learning over multiple day-epochs particularly in the form of overfitting. We intentionally trained the network on a larger number of day-epochs to not only fully train on the daily recruited training datasets, but also attempt to force overtraining on the data. However, our attempt failed as the model’s performance was unchanged and would retain information weights of the recruited datasets. Overfitting typically occurs when a model learns the details of training data so well that it can negatively impact the model’s performance on newer data. We hypothesize the multi-classification task using the Medical MNIST datasets is perhaps too easy for our traditional and sequential learning frameworks, resulting in ideal accuracy and loss metrics across all experiments. We evaluated this theory with our sequential learning experiment that renewed the model weights after each “day” over the training period. As can be seen in Figure 21, the accuracy metrics reach 90+% each “day” even though the network recruits different N datasets each “day”, which is an unrealistic expectation when working with real-life medical imaging datasets. We were then subsequently motivated to explore other medical imaging datasets that would better challenge our traditional and sequential learning experimentation.
Chapter 4: Real-Life Classification Tasks

4.1 Background

Though the Medical MNIST dataset used in our earlier experiments provided exceptional results, real-life medical imaging datasets would be far more challenging to perform classification tasks on. Because of this, we sought out medical imaging datasets that would exemplify more challenging classes to train on using our sequential learning network. Our first example took advantage of our existing Medical MNIST dataset by isolating and performing simple data augmentation on the CXR class. By doing so, we split the CXR class into three subclasses which correspond to the applied augmentation. The three subclasses are as follows: CXR image rotated 90 degrees to the left, original CXR image, CXR image rotated 90 degrees to the right. We utilized the same NIH Chest X-Ray dataset used by Srivastava et al. as our second dataset, which is a better reflection of our motivation to perform these sequential learning experiments [15]. This dataset consists of chest x-ray exams which have been clinically diagnosed with disease labels from 30,805 unique patients [15]. These labels were created by the authors who used natural language processing to determine the appropriate disease class from the associated radiologist reports, which are expected to be 90+% accurate [15]. This dataset consists of 14 unique class diagnoses in addition to a “no findings” class that contains healthy scans. In total there are 15 unique class labels, but for our multi-classification sequential learning experiments we will only be using the following three classes: effusion, mass, no finding.

In this chapter, both the modified CXR and NIH Chest X-Ray datasets would be used in experiments that have been implemented in previous chapters. Section 4.3.1 details the experiments that will be performed using modified CXR dataset. There are many experiments in this section which detail the findings involving different comparisons.
Section 4.3.1.1 details the first comparison made for a traditional batch-style trained network against a network trained exclusively under sequential learning with N datasets each “day” for one day-epoch without replacement over a defined period of training “days”. Section 4.3.1.2 details the second comparison which uses the two different methods of generating training and validation data, as described in Section 3.3.3, using multiple day-epochs trained under the same sequential learning network. Section 4.3.1.3 details the final comparison, similarly performed in Section 2.3.1, that compares a sequential learning experiment with a pre-training loop against an experiment without a pre-training loop. In Section 4.3.2, the three experiments that use the NIH Chest X-Ray dataset are detailed. The first experiment described in Section 4.3.2.1 is the traditional batch-style trained network that determines baseline accuracy and loss metrics. This would be compared to the experiment described in Section 4.3.2.2 that utilizes the batch-style training loop as pre-training before the network undergoes sequential training. Lastly, both experiments will be compared to the experiment in Section 4.3.2.3 which describes a network that undergoes sequential learning exclusively. All sequential learning experiments in Section 4.3.2 will be done with five day-epochs and an equal number of N datasets used for training each “day”.

4.2 Data and Model

As mentioned previously, we took advantage of the CXR class contained within the Medical MNIST imaging dataset and applied data augmentation to the images. This data augmentation split the class into three subclasses that include the original image, the image rotated left, and the image rotated right. The split into the three subclasses would allow them to each contain a comparable number of datasets. There are 3,309 images within the left rotated class, 3,316 images within the original class, and 3,375 images within the right rotated class. Because the original CXR class contained 10,000 images, we
created the modified CXR dataset such that the combined sum across the three subclasses would be 10,000 as well. Each image is processed into a JPEG with 64x64 dimensions, and the training, validation, and testing subsets are established differently for each experiment listed below. A sample image of each CXR subclass can be seen in Figure 22.

The second dataset we used was the publicly available NIH Chest X-Ray imaging dataset on Kaggle for our traditional and sequential learning experiments. Of the 15 unique class labels split amongst 112,120 medical images, all of which are processed as PNGs with 1024x1024 dimensions, we have selected three classes to be used for our experiments. Within these classes, effusion consisted of 3,955 images, mass consisted of 2,139 images, and no finding consisted of 60,361 images. A sample medical image from each class can be seen in Figure 23.

Both datasets utilize the PyTorch ResNet50 model as the image classification algorithm. The same data augmentation, described in further detail in Section 2.2.3, is also applied on the training, testing, and validation datasets across all experiments. The cost function and optimizer are described in Section 2.2.4, and the number of N datasets recruited per “day” varied with each experiment along with the total “days”, total day-epochs, and model LR.
4.2.1 Modified CXR Dataset

This global split of the datasets differed with each experiment comparison listed below. Modifying the original CXR images was performed using a shell script via the command line. The images would be selected at random to be rotated in either direction before being sorted into the appropriate subclass folder. We then filtered the total images generated such that there are a comparable number for each subclass. Additionally, most of the data is allocated for training purposes to reduce variance among the parameter estimates.
The global split of the datasets was consistent with each experiment comparison listed below, though there were additional steps required before sorting the three defined classes into training, validation, and testing cohorts. All 112,120 images obtained from the Kaggle dataset were de-identified and were not pre-sorted into the 15 defined class labels. Instead, the dataset was accompanied by a data entry spreadsheet that identifies the image index # with its corresponding diagnosis. Taking advantage of this spreadsheet, we wrote a MATLAB program that automatically sorts all 15 class datasets into self-named subfolders. We loaded the spreadsheet in the MATLAB workspace and through use of “for” loops exhausted the spreadsheet of every entry for a particular class label. Each entry would have accompanied index values that were made into a list which contained all instances of the defined class label. The list then took each index value to find the corresponding file name for the defined class label, and the program proceeded to move the selected file to its corresponding file folder. It is important to note that many images in this dataset were classified under multiple diagnoses. For the sake of training our classification CNN, we only sorted images that were identified under a single class label. The decision-making that went into choosing the three defined classes of effusion, mass, and no finding was to pick distinctive diagnoses for a less challenging multi-classification task, and to select class labels that contained a substantial amount of data for training a
CNN. Of the three, the mass class label contained the fewest images at 2,139. To ensure no training bias for our experiments, we measured equal data splits to end up with 2,100 images per class. The images were then sorted into training, validation, and testing cohorts that are described in further detail below.

4.2.3 Training, validation, and testing data splits

We created a Python notebook using Google Colab to load 10,000 Modified CXR datasets into one list. First, the order of the 10,000 datasets were randomized. From the randomized list, the filenames for the first 7,000 datasets were written out to the text file “filenames_train_CXR.txt”. This would assure the same training data was used for each of the CNN experiments. From the same list, the filenames for the subsequent 1,000 datasets were written out to the text file “filenames_val_CXR.txt”, which would assure the same validation data was being used for each experiment. Lastly, the filenames for the remaining 2,000 datasets were written out to the text file “filenames_test_CXR.txt” for the same reason. When loading in all 10,000 datasets, the text files would be used to pre-set the training, validation, and testing splits to maintain consistent findings across the individual CNN experiments.

We did not create text files that would pre-set the training, validation, and testing splits of the NIH Chest X-Ray dataset. Instead, we manually created the splits for the 2,100 images allocated for each class. As was similarly done for the Medical MNIST dataset, 70% of the data for each class is allocated for training, 10% is allocated for validation, and 20% is allocated for testing. The same training, validation, and testing data remains consistent across the individual CNN experiments with the only difference being randomized N datasets being used for the NIH sequential training experiments.
4.3 Experiments

There were different experimental frameworks used between the two datasets. Beginning with the modified CXR dataset, Experiment 0 represents the baseline traditional training method for a multi-classification network. The results from this experiment will be compared to the results from Experiment 1, which is a sequential learning network that performs multi-classification on N datasets arriving each “day” with a single day-epoch and no validation data. After training each “day”, the model is evaluated on the testing data. Experiments 1a and 1b represent the sequential learning networks with distinctive validation data recruitment that perform multi-classification over multiple day-epochs with fewer overall days. Experiment 1a uses N datasets arriving in the “present day” as validation data, and N datasets from the “previous day” as training data. Experiment 1b combines N/2 datasets from the “present day” and N/2 datasets from the “previous day” as training data while the remaining N/2 datasets from the “present day” is used as validation data. Both experiments will be compared to each other to evaluate how different approaches to validation data recruitment will affect a sequential learning model’s performance. Experiments 2a and 2b represent the baseline experiments evaluating a sequential learning network with a pre-training component against a network without any pre-training. The results from these two experiments will be compared to ensure consistency between frameworks.

The NIH Chest X-Ray dataset uses fewer but equally important experiment frameworks. Experiment 0 represents the baseline traditional training method for a multi-classification network. The results from this experiment will be compared to the final results from two sequential learning networks to ensure cohesion between frameworks. The first framework is Experiment 1 which uses the traditional training method in Experiment 0 as a form of pre-training until a desirable validation accuracy threshold is
obtained (>50%) before continuing training in a sequential learning format for a single day-epoch over a defined period of “days” with N datasets per “day”. The second framework is Experiment 2 which is the sequential learning model that trains exclusively on N datasets arriving each “day” with one day-epoch over a defined period of “days”. This experiment trains on more “days” than Experiment 1 to ensure a comparable amount of data is recruited by the sequential learning network.
4.3.1 Modified CXR Experiments

4.3.1.1 Baseline - Traditional Batch Training vs. Sequential Training (Experiment 0 & 1)

Goal

The goal of this baseline experiment is to not only establish a CNN multi-classifier for the modified CXR datasets, but to compare the accuracy and loss curves between a traditionally trained network against a sequentially trained network. We expect comparable results across the two experiments even though they do not share the same training passes (epochs vs “days”). For traditional batch training, the model would learn on all the training data over a defined number of epochs until it starts to overfit based on the validation loss. For sequential learning, a defined number of N datasets would be recruited from the global training data to be trained with one day-epoch over a defined period of “days”.

Methods

Data for the three classes from the modified CXR dataset were used for training, validation, and testing for a traditional batch training and sequential learning network. There were 10,000 total datasets pre-set from each associated CXR “filenames” text file with the following splits: 70% training data, 10% validation data, and 20% testing data. Each of these splits are defined in greater detail in Data Section 4.2.3 above. Both PyTorch ResNet50 CNNs were trained with parameters batch size of 16, though they differed regarding training passes and LR. We expect the traditional batch training network, with a LR of 1e-6, to be fully trained (90+% accuracy) after 30 epochs. The sequential learning network, with a LR of 1e-4, should reach a similar accuracy threshold after 100 days of training recruiting $N = 50$ datasets each “day” without replacement. After each “day’s”
training, the CNN was tested against the global testing data defined in Data Section 4.2.3 above, though normally testing data may not typically be available in a sequential learning scheme.

Results

The results shown in Figure 24 are shown from traditional training across 30 epochs with a LR of 1e-6. All accuracy and loss metrics steadily increase and decrease, respectively, throughout the duration of training. We can observe that the number of epochs needed to reach 90+% accuracy is roughly 20 epochs. The loss metrics appear to stabilize just after 25 epochs.

![Epoch Accuracy](image)

![Epoch Loss](image)

**Figure 24:** The accuracy of the 3-way classifier on the modified CXR data (rotated left vs. original vs. rotated right) as a function of epochs (top). The loss of the 3-way classifier as a function of cross-entropy loss (bottom) per epoch. This is shown for training, validation, and testing metrics.
The results shown in Figure 25 represent the sequentially trained network out to 100 “days” with one day-epoch per “day” and a LR of 1e-4. Each “day” the network was trained on the 50 datasets recruited in the “present day” without replacement. These datasets were selected from the global training datasets that have been pre-set. We can observe the accuracy and loss curves for the training and testing data of the multi-classifier network as a function of “days” (accuracy) and cross-entropy loss. This sequentially trained model appears to reach 90+% validation accuracy roughly halfway through the period of “training days”. However, the training accuracy continues to oscillate wildly around this same time between ~80-95% accuracy before finally stabilizing between ~90-95% over the final 20 “days”. Both loss metrics steadily decrease over the 100 training “days” before stabilizing over the final 20 “days” as well. This would indicate that the sequential learning network has been almost fully trained and would not benefit from training past “day” 100 as the full information has been extracted from the training data.
Figure 25: The baseline sequentially trained CNN for one epoch per day. The accuracy of the 3-way classifier on the modified CXR data (rotated left vs. original vs. rotated right) as a function of “days” (top). The loss of the 3-way classifier as a function of cross-entropy loss (bottom) per “day”. The training metrics are for the 50 datasets evaluated by the CNN each “day”. The testing metrics are evaluated by the CNN at the end of each “day” on the set aside testing data.
4.3.1.2 Sequential Learning - Multiple Epochs Per Day (Experiment 1a & 1b)

Goal

The goal of this sequential learning experiment is to evaluate a sequential learning network that trains over multiple day-epochs for a defined period of “days”. This will be accomplished using the same approach described in Section 3.3.3 that consistently updates the validation data through two unique methods. By training over multiple day-epochs, we can hopefully extract the full information from a particular “day’s” datasets. The results from these experiments will inform us of how information extraction differs between the two methods, whether any overfitting occurs, and the advantages this approach provides compared to sequential learning networks that train with only a single day-epoch.

Methods

The three classes from the modified CXR dataset were used for training, validation, and testing for this multi-classifier network. There were 10,000 total datasets pre-set from each associated CXR “filenames” text file with the following splits: 70% training data, 10% validation data, and 20% testing data. From the global training data, N = 50 datasets would be recruited each “day” without replacement over a sequential training period of 10 “days”. The PyTorch ResNet50 CNNs were trained with the parameters batch size of 16, 150 day-epochs, and LR of 1e-5.

The validation data was not pre-set in this experiment and reflects the same approaches described in Section 3.3.3. The first approach recruits N datasets that arrived in the “present day” to be used as validation data, and the N datasets from the “previous day” are used to train the model. The second approach recruits N/2 datasets arriving in the...
“present day” along with N/2 of the datasets from the “previous day” as training data for the model, and the remaining N/2 datasets from the “present day” to be used as validation data. The breakdown of these experiments is described in further detail in Sections 3.3.3.1 and 3.3.3.2. Following the conclusion of each “day’s” training for both experiments, the CNNs were tested against the global testing data. Normally test sets may not be available in a sequential learning scheme.

Results

The results shown in Figure 26 represent the sequentially trained network out to 10 “days” with 150 day-epochs per “day” and a LR of 1e-5. Each “day” the network was trained on 50 datasets from the “previous day” and validated on the 50 datasets arriving in the “present day” without replacement. These datasets were selected from the global training data that has been pre-set. Because there is no “day” prior to “day” one, no training was performed on “day” one. We can observe the accuracy and loss curves for training, validation, and testing data of the multi-classifier network as a function of epochs (accuracy) and cross-entropy loss. Though the training and validation accuracy curves fluctuate wildly over the first six “days” of sequential training, the oscillations are kept compressed between a ~20% range each “day”. The training accuracy stabilizes between 90-100% roughly halfway through “day” five training, whereas the validation accuracy stabilizes between 80-90% around this same time. The validation accuracy would eventually need to reach “day” 10 to oscillate between 90-100% accuracy. Over the duration of sequential training the network steadily increases and only exhibits instances of catastrophic forgetting for the validation curve on “days” six and eight. Compared to the results in Section 3.3.3.1, the modified CXR data requires twice the number of “days” to reach 90+% accuracy metrics which would verify the increased difficulty this dataset poses for the classification CNN. The training and validation loss curves reflect their accuracy
curves, with training loss stabilizing halfway through “day” five as well. Overall, the validation loss shows a decrease over the subsequent “days”, though it mostly oscillates each “day” without an observable trend through the period of sequential training.

**Figure 26**: The sequentially trained CNN for multiple day-epochs. The accuracy of the 3-way classifier on the modified CXR data (rotated left vs. original vs. rotated right) as a function of “days” and day-epochs (top). The loss of the 3-way classifier as a function of cross-entropy loss (bottom) per day-epoch. The training metrics are for the 50 datasets evaluated by the CNN from each “previous day”. We can observe how the training curves begin on the second “day”. The validation metrics are evaluated by the CNN for the 50 datasets acquired in the “present day”. The testing metrics are evaluated by the CNN at the end of each “day” on the set aside testing data.
The results shown in Figure 27 represent the sequentially trained network out to 10 “days” with 150 day-epochs per “day” and a LR of 1e-5. Each “day” the network was trained on 50 datasets (N/2 from the “present day” and N/2 from the “previous day”) and validated on 25 datasets (remaining N/2 from the “present day”) without replacement. These datasets were selected from the global training data that has been pre-set. Because there is no “day” prior to “day” one, only N/2 datasets were used for training on “day” one. We can observe the accuracy and loss curves for training, validation, and testing data of the multi-classifier network as a function of epochs (accuracy) and cross-entropy loss. The training and validation accuracy curves oscillate similar to the previous experiment over the first five “days” of sequential training. Here the validation curve oscillations are less compressed between ~30% range each “day”. The training accuracy stabilizes much sooner than the previous experiment between 90-100% roughly halfway through training on “day” one. Over the final five “days”, the training accuracy further improves to oscillate between 95-100% accuracy. The validation accuracy manages to stabilize on “day” six between 90-100%, however the training performance is isolated as the model destabilizes slightly over the following four “days” between 80-100%. Over the duration of sequential training the network steadily increases and only exhibits instances of catastrophic forgetting for the validation curve on “days” five and eight. Compared to the results in Section 3.3.3.2, the modified CXR data requires nearly three times the number of “days” to reach 90+% accuracy metrics which would verify the increased difficulty this dataset poses for the classification CNN. The training and validation loss curves reflect their accuracy curves, with training loss stabilizing halfway through “day” one as well. Overall, the validation loss shows a decrease over the subsequent “days”, though it mostly oscillates each “day” without an observable trend through the period of sequential training.
Figure 27: The sequentially trained CNN for multiple day-epochs. The accuracy of the 3-way classifier on the modified CXR data (rotated left vs. original vs. rotated right) as a function of “days” and day-epochs (top). The loss of the 3-way classifier as a function of cross-entropy loss (bottom) per day-epoch. The training metrics are for the 50 datasets evaluated by the CNN (N/2 from the “present day” and N/2 from the “previous day”). The validation metrics are evaluated by the CNN for the remaining N/2 datasets acquired in the “present day”. The testing metrics are evaluated by the CNN at the end of each “day” on the set aside testing data.
4.3.1.3 Baseline Sequential Learning (Experiment 2a & 2b)

Goal

The goal of this experiment is to train a 3-way classifier for the modified CXR datasets via two methods. The first method incorporates a traditional batch-training loop on a small subset of the training data. Because the model obtains an initial understanding of the classes via this small subset of data before being sequentially trained, we refer to this traditional training loop as pre-training. The sequential training continues improving overall accuracy and loss metrics by training on N datasets per “day” for a predefined number of “days”. The second method consists of sequential training exclusively. For this method, the model starts learning from scratch on N datasets per “day”. The results from these base frameworks with and without pre-training are meant to be comparable to confirm that sequential training can be performed even without an initial sample subset of data.

Methods

The three classes from the modified CXR dataset were used for training, validation, and testing for this multi-classifier network. There were 10,000 total datasets pre-set from each associated CXR “filenames” text file with the following splits: 70% training data, 10% validation data, and 20% testing data.

**Batch pre-training:** The PyTorch ResNet50 CNN was trained with the parameters batch size of 16, five epochs, and LR of 1e-5. For the baseline traditional training, we used a subset of the global training data, which consists of 500 images, and validated on the pre-training results against the global validation data. Each of these splits are defined in greater detail in Data Section 4.2.3 above. After training on a limited number of epochs, we expect the pre-training to reach a desirable accuracy threshold (~70%) before the model shifts to training in a sequential learning format.
Sequential training: The CNN was further trained each “day” on N = 30 datasets recruited from the global training data for one day-epoch over a period of 200 “days”. Before sequential training began, we updated the model LR so it may classify better on the chosen dataset. The test set of data is not traditionally evaluated until after the neural network is completely trained and no decisions on training are applied to the test data set. In this case we are using the test dataset as a held-out sample of the population which is evaluated after the training each “day” as a way to understand the training. As the model continues to train over the “days”, we expect it to be fully trained (90+% accuracy) with stabilized training data oscillations as it nears completion.

Sequential training, no pre-training: Similar to before, the PyTorch ResNet50 CNN was trained with the parameters batch size of 16 and a LR of 1e-4. Because there is no pre-training with this method, we sequentially trained over a period of 230 “days” while also recruiting N = 30 datasets per day. The increased number of days is meant to show a comparable number of datasets being used between methods. After training on a single day-epoch, the results should show a fully trained (90+% accuracy) model that has stabilized training data oscillations as it nears completion.

Results

Batch pre-training: The results in Figure 28 are shown from batch pre-training for five epochs with a LR of 1e-5. We can see the accuracy and loss curves of the traditional multi-classifier network as a function of epochs (accuracy) and cross-entropy loss. We can observe how the accuracy can reach ~70% even with a limited sample size, which we deemed a good starting point for the CNN before it begins sequential learning. The loss
metrics also show a steady decrease but do not appear anywhere near stabilizing (extracting as much information as possible from the training datasets).

![Graphs showing pre-training accuracy and loss](image)

**Figure 28:** The pre-training accuracy of the 3-way classifier on the modified CXR data (rotated left vs. original vs. rotated right) as a function of epochs (top). The pre-training loss of the 3-way classifier as a function of cross-entropy loss (bottom) per epoch. This is shown for training and validation metrics.

**Sequential training:** The results in Figure 29 represent the sequentially trained network for one day-epoch over 200 “days”. The pre-trained model from Figure 28 is carried into sequential training and trains on 30 datasets recruited each “day” from the global training data. We can observe how the starting point of testing accuracy begins near where validation accuracy ends during pre-training. This indicates proper continuation of training between learning loops. The model fluctuates heavily for training accuracy over roughly
the first 25 “days” as can be seen in the oscillation between 55-85% accuracy. Although the training accuracy does improve over the subsequent 50 “days”, it continues to oscillate wildly between 70-100%. Following “day” 75 however and throughout the remaining days the training curve mostly oscillates between 90-100%, indicating that it has mostly stabilized even though there are a few declines which can be observed around “days” 110 and 152. Shortly after these declines however the model quickly self-correction and restabilizes. The testing accuracy steadily increases over the period of 200 “days” and mostly stabilizes at ~95% for the final 125 “days”. The loss metrics show a steady decrease, though the training loss shows wild oscillations over the period of learning “days”. The testing loss steadily decreases all the way until “day” 100 where it appears to stabilize around 0.1. This means the network has mostly extracted enough information from the datasets halfway through the sequential training period and would not need further training past “day” 100. We included the additional training “days” to observe the trend and potential overfitting that may or not have occurred.
Sequential training, no pre-training: The results in Figure 30 represent the sequentially trained network that begins learning without any pre-training for one day-epoch over a period of 230 “days”. The lack of pre-training can be distinguished by the testing accuracy that begins at ~40%. Over the first 50 “days”, the training accuracy oscillates between 60-95% and is generally stable as it learns from the recruited datasets. Over the duration of sequential learning however, the oscillations shift higher and fluctuate far less, eventually reaching ~90-100% by “day” 100 and is maintained at said accuracy over the final 130 “days”. The testing accuracy metric, which begins slow due to very few datasets and learning information, steadily increases for the first 50 “days” of training before reaching ~90% accuracy. As more information is collected over the subsequent 50 “days”, the
testing accuracy reaches ~95% and stabilizes over the final 130 “days”. In fact, over the final 80 “days” the testing accuracy increases a bit further to ~98%. The loss metrics start off very high due to little information extraction but steadily decreases throughout the sequential learning period. The training loss is mostly stabilized over the final 130 “days”, though there are a few jumps around “days” 160 and 210. The testing loss appears to stabilize around 0.1 over the same period as the training loss, meaning that the network has mostly extracted the full information over the first 100 “days” and would not need further training past “day” 100. Like before, we included the additional training “days” to observe potential overfitting that may or may not have occurred.

Figure 30: The sequentially trained CNN without pre-training for one day-epoch. The accuracy of the 3-way classifier on the modified CXR data (rotated left vs. original vs. rotated right) as a function of “days” (top). The loss of the 3-way classifier as a function of cross-entropy loss (bottom) per “day”. The training metrics are for the 30 datasets evaluated by the CNN each “day”. The testing metrics are evaluated by the CNN at the end of each “day” on the set aside testing data.
4.3.2 NIH Chest X-Ray Experiments

4.3.2.1 Baseline Traditional Batch Training (Experiment 0)

Goal

The goal of this baseline experiment is to establish a baseline multi-classifier for the three classes of effusion, mass, and no finding. We will determine overall accuracy and loss metrics expected from a traditionally trained network, and the model will be trained until we observe overfitting based on the validation loss. The results from this traditional training experiment are meant to produce comparable results to a sequential learning network with and without pre-training.

Methods

The effusion, mass, and no finding classes from the NIH Chest X-Ray dataset were used for training, validation, and testing for a multi-classifier network. There 6,300 total datasets across the three classes which have been equally pre-set into the following splits: 70% training data, 10% validation data, and 20% testing data. Each of these splits are defined in greater detail in Data Section 4.2.3 above. The PyTorch ResNet50 CNN was trained with the parameters batch size of 16, 10 epochs, and LR of 1e-4. Other user-created classification models with the NIH Chest X-Ray dataset have shown little success at generating high accuracy thresholds (90+%). In fact, the consensus performance that most models achieve is ~65-75% classification accuracy [10,16-19]. For our baseline traditional training experiment, we expect to reach similar accuracy metrics on our validation and testing data.

Results

The results shown in Figure 31 are from a traditionally trained network across 10 epochs and a LR of 1e-4. We can see the accuracy and loss curves of the traditional multi-
classifier network as a function of epochs (accuracy) and cross-entropy loss. The training accuracy steadily increases, and the training loss steadily decreases over the 10 epochs. By the end of training, the training loss does not appear to stabilize which indicates the model would benefit from further training epochs. Meanwhile the validation and testing accuracy metrics both appear to reach their threshold after the seventh epoch at 68% and 70%, respectively, indicating our model achieves comparable results to what other users have shown. The subsequent three epochs are shown to overfit on the validation and testing datasets which would indicate little benefit to any additional training past this point.

![Graph showing accuracy and loss over epochs](image)

**Figure 31:** The accuracy of the 3-way classifier on the NIH Chest X-Ray data (effusion vs. mass vs. no finding) as a function of epochs (top). The loss of the 3-way classifier as a function of cross-entropy loss (bottom) per epoch. This is shown for training, validation, and testing metrics.
4.3.2.2 Baseline Sequential Learning, Pre-Training (Experiment 1)

Goal

The goal of this experiment is to establish a baseline sequential learning multi-classifier for the three classes of effusion, mass, and no finding. We will incorporate a traditional batch-training loop on a small subset of the training data to obtain an initial understanding of the three classes. This will be referred to as pre-training before the model continues improving overall accuracy and loss metrics through sequential training on N datasets per “day” for a predefined number of “days”. The result from this sequentially trained network will be compared to the results from the traditionally trained network and the sequentially trained network without a pre-training component.

Methods

Beginning with pre-training, the PyTorch ResNet50 CNN was trained with the parameters batch size of 16, 10 epochs, and a LR of 1e-5. This initial traditional training loop used a subset of the global training data, which consists of 1000 images, and validated on the pre-training results against the global validation data. Each of these splits are defined in greater detail in Data Section 4.2.3 above. After training on a limited number of epochs and data subset, we expect the pre-training to reach a desirable validation accuracy threshold (~50%) before the model shifts to training in a sequential learning format. The CNN further trained each “day” on N = 20 datasets recruited from the global training data for five day-epochs over a period of 170 “days”. We evaluated the model after training each “day” on the global testing set which is typically unavailable for sequential training. As the model trains over the defined period of “days”, we expect the testing accuracy to reach ~65-75% with stabilized oscillations as it nears completion [10,16-19].
Results

The results in Figure 32 are shown from batch pre-training for 10 epochs with a LR of $10^{-5}$. We can observe the accuracy and loss curves of the traditional multi-classifier network as a function of epochs (accuracy) and cross-entropy loss. The pre-training accuracy steadily increases over the 10 epochs of traditional training whereas the pre-training loss steadily decreases. Because the pre-training loss does not stabilize by the end 10 epochs, we can infer the model would benefit from additional training to fully extract information from the class datasets. The validation accuracy remains stable with little distinguishable improvement. However, it does reach a desirable accuracy threshold (~50%) on epochs seven and eight, which we deemed a good starting point before sequential learning begins.
Figure 32: The pre-training accuracy of the 3-way classifier on the NIH Chest X-Ray data (effusion vs. mass vs. no finding) as a function of epochs (top). The pre-training loss of the 3-way classifier as a function of cross-entropy loss (bottom) per epoch. This is shown for training and validation metrics.

The results in Figure 33 represent the sequentially trained network for five day-epochs over 170 “days”. The pre-trained model from Figure 32 is carried into sequential training and trains on 20 datasets recruited each “day” without replacement from the global training data. We can observe how the starting point of testing accuracy begins near where validation accuracy ends during pre-training. This indicates proper continuation of training between learning loops. Over the entirety of the sequential training loop, there is little to no improvement with unstable training accuracy that oscillates between 40-90%. The testing accuracy improves very little over the first 75 “days” with a stable oscillation.
between 50-60%. Over the following 25 “days” the CNN appears to briefly overfit on the
data before restabilizing around “day” 100. Over the final 75 “days”, the testing accuracy
steadily improves and mostly stabilizes between 60-68%. The ~68% testing accuracy was
achieved around “day” 120, indicating that sequential learning with pre-training achieved
comparable results to both our traditional training experiment and what other users have
shown.

Figure 33: The sequentially trained CNN following pre-training for five day-epochs. The accuracy
of the 3-way classifier on the NIH Chest X-Ray data (effusion vs. mass vs. no finding) as a function
of “days” (top). The loss of the 3-way classifier as a function of cross-entropy loss (bottom) per
“day”. The training metrics are for the 20 datasets evaluated by the CNN each “day”. The testing
metrics are evaluated by the CNN at the end of each “day” on the set aside testing data.
4.3.2.3 Baseline Sequential Learning, No Pre-Training (Experiment 2)

Goal

The goal of this experiment is to evaluate the performance of the baseline sequential learning network without pre-training and compare the results to the previous two experiments. This model will begin sequential training from scratch to improve overall accuracy and loss metrics on N datasets per “day” for a predefined number of “days”.

Methods

The effusion, mass, and no finding classes from the NIH Chest X-Ray dataset were used for training, validation, and testing for a multi-classifier network. Beginning from scratch, N = 20 datasets would be recruited each “day” from the global training data without replacement for sequential training over a period of 220 “days”. The PyTorch ResNet50 CNN was trained with the parameters batch size of 16, five day-epochs, and a LR of 1e-5. Following the conclusion of each “day”s” training for this experiment, we evaluated the model against the global testing set which is typically unavailable for sequential training. As the model trains over the defined period of “days”, we expect the testing accuracy to reach ~65-75% with stabilized oscillations as it nears completion [10,16-19].

Results

The results in Figure 34 represent the sequentially trained network for five day-epochs over 220 “days” and a LR of 1e-5. The model begins training from scratch on the 20 datasets recruited each “day” without replacement from the global training data. Unlike the previous experiment, testing accuracy starts much lower at ~35% before steadily increasing over the period of “days”. The training accuracy oscillates wildly over the first 50 “days” between 20-70% before seeing slight improvements to oscillate between 40-95% over the final 50 “days”. The testing accuracy increases nearly 30% over the first 150
“days” of sequential training before stabilizing over the final 70 “days” to oscillate between 60-68%. The ~68% testing accuracy was achieved near the very end of the period of training “days”, indicating that sequential learning without pre-training achieved comparable results to the previous two experiments and what other users have shown.

Figure 34: The sequentially trained CNN without pre-training for five day-epochs. The accuracy of the 3-way classifier on the NIH Chest X-Ray data (effusion vs. mass vs. no finding) as a function of “days” (top). The loss of the 3-way classifier as a function of cross-entropy loss (bottom) per “day”. The training metrics are for the 20 datasets evaluated by the CNN each “day”. The testing metrics are evaluated by the CNN at the end of each “day” on the set aside testing data.
4.4 Discussion

When comparing the results from the Medical MNIST dataset to the results from the Modified CXR and NIH Chest X-Ray datasets, the latter have proven to be more challenging for our multi-classification CNN experiments. This can be observed across Figures 24-34 which display results from experiments that require an increased number of training passes (epochs or “days”) to reach similar performances as their Medical MNIST counterparts. The NIH Chest X-Ray dataset in particular is the more interesting of the two as we can observe overfitting occur for all three experiments. As such it is imperative to fine tune the network hyperparameters to optimize the model’s training approach. Additionally, the time duration required to train each experiment in Section 4.3.2 was significantly longer than any experiment using the other two datasets, likely due to NIH images having larger dimensions with a higher resolution, and correspondingly more features to extract by the network. This presents the added benefit of sequential learning experimentation because of limitations that afflict traditional batch training processes. Such limitations include GPU speed, RAM, and disk space. The results from our sequential learning experiments have shown a comparable performance against the traditional batch training experiments, enabling it as a viable alternative approach for CNN multi-classification. Should sequential learning be adopted with wider research audiences, they would discover how saving the model weights after being run once each “day” with fewer datasets is more advantageous because of computation time. Should there be any set-aside data for a particular classification task, it would be valuable to conduct pre-training to extract maximal information from the data without overfitting and then proceed to acquire more cases on a day-by-day basis. Though our results were satisfactory compared to what other users have shown, we can still consider changes that would improve our model’s performance. In fact, because other users have explored a variety of
pre-trained models which differ from ResNet50, we could similarly evaluate other models’ performances on our dataset to determine if another model performs better than the one we adopted for our experiments.
Chapter 5: Summary & Future Directions

To summarize, we have shown sequential learning as a viable ML multi-classification technique capable of producing results on par with traditionally trained CNNs. In Chapter 2, we evaluated sequential learning CNNs that trained, validated, and tested on data obtained from the Medical MNIST dataset. By recruiting N datasets over an appropriate number of training “days”, we can improve the classification accuracy for an untrained model until it reaches 90+%%. Additionally, if a research team seeks to perform sequential training for a new clinical study, they would benefit from having any small amount of training data to pre-train a model to a desirable starting point. We have implemented this method by pre-training on a small subset of the training data before continuing to improve the model’s performance through sequential training. Although a CNN which incorporates a pre-training component will reach desirable results faster than a CNN without one, the end result for both approaches will remain the same. This baseline experiment considered only three classes from the Medical MNIST dataset: CXR, Hand, and HeadCT. To further challenge our sequential learning CNN, we added in a fourth class during sequential training to show how our model can adapt to newer information. This new class data was added at the halfway point of sequential training, causing a significant decline in all accuracy metrics as the model initially struggled to adapt to the new class data. However, with continued training with N datasets from four classes over the second half of training “days”, the CNN was shown to train up to a desirable accuracy (90+%).

In Chapter 3, we further evaluated our sequential learning experimentation with the same Medical MNIST dataset. We began by comparing the results obtained from a traditionally trained network and a sequentially trained network. Ensuring cohesion between these two frameworks would prove that sequential learning is a viable alternative ML strategy. Up to this point however we only trained on a single day-epoch during the
sequential learning loop. Because a single “day’s” worth of training is not viable for full information extraction, we increased the number of day-epochs to allow the network more time to learn on the recruited datasets. This was done for two methods of establishing a validation dataset used each “day” during sequential learning that differed from the global validation dataset. This approach would provide insight for an appropriate number of day-epochs used for full information extraction without overtraining the network. Both methods generated trends that showed strong understanding of the data recruited over the five training “days”. In fact, for both methods the network reached high accuracy thresholds without any signs of overfitting. This led us to assume that this dataset was not challenging enough for our sequential learning experimentation, and as such we evaluated an experiment that would sequentially train across 10 “days” with the model weights renewed each “day”. The results from this experiment had shown the network reaching 90+% across all metrics by the end of each “day” over the 10 “days”, which is an unrealistic scenario when working with real-life medical imaging datasets.

In Chapter 4, we sought to perform similar experiments done previously but on datasets that would better challenge our classification CNNs. The first dataset took advantage of the CXR class from the Medical MNIST dataset which we applied data augmentation to and split into three subclasses: rotated left, original, rotated right. The second dataset used three classes from the NIH Chest X-Ray dataset: effusion, mass, no finding. The modified CXR dataset was evaluated across different experiments done previously. We had shown that comparable results were obtained between a sequential learning network and traditional training network, though this dataset did take longer to reach a desirable accuracy threshold than was shown in Chapter 3. We also performed the multiple day-epoch experiment to compare two methods of training and validation data recruitment, and have shown that all accuracy metrics require five to six training “days” before stabilizing at a desirable accuracy threshold, roughly twice the number of “days”
compared to the results shown in Section 3.3.3. Lastly, we performed the pre-training vs
no pre-training experiment with this dataset, the results of which achieved comparable
results. This would indicate that even with a more challenging multi-classification task,
sequential learning can be performed without an initial subset of data and achieve
sufficient results. The NIH Chest X-Ray dataset was evaluated against three experiments:
a traditionally trained network, a sequentially trained network with pre-training, a
sequentially trained network without pre-training. For all three experiments we observed
overfitting, making it imperative to fine tune the hyperparameters to optimize the training
approach. Although the testing accuracy was not great across the three experiments (~65-
70%), the oscillations would mostly stabilize as training neared completion.

Our sequential learning experimentation was not without some key limitations. To
begin, all of our Python notebooks were executed using Google Colab, a hosted Jupyter
notebook service with its own GPUs. Because standard Colab users receive slower GPUs,
we upgraded to Collab Pro to access the faster GPUs and other resource upgrades. While
these GPUs were sufficient for our work, they are not the fastest or have the highest
computational power. This was more apparent with our experiments using the NIH Chest
X-Ray dataset that would train for several hours at a time. In future work we hope to recruit
even stronger GPUs to reduce training time and run more notebooks in parallel. Another
limitation we faced was a reliance on public imaging datasets. While useful for our proof-
of-concept experimentation, there exist some uncertainties with the more challenging
datasets. The NIH Chest X-Ray dataset is a good example of a single imaging modality
with different diagnoses. This dataset consists of captured x-ray images acquired over a
period of 20 years. This dataset is also limited in total images per class for training a CNN,
making it challenging to reach desirable accuracy thresholds due to limited data. The
origin of said data also makes it equally if not more difficult for a classification algorithm to
differentiate between scans with subtle differences especially if they were acquired over
20 years apart from each other. In addition to exploring a greater variety of PyTorch models that may perform better than ResNet50 for classification experiments, we hope to recruit datasets from a more direct resource where we can better recognize and filter images on scanners, scan parameters, and diagnoses. Although we achieved optimal results for every experiment evaluating the Medical MNIST dataset, there was more trial and error involved with adjusting the hyperparameters for the two real-life datasets. The precise tuning of model parameters required with these two datasets was due to a classification difficulty increase and simultaneously a higher likelihood of overfitting or concept drift. Here we were limited with our fine-tuning approach and moving forward we hope to implement optimization strategies that can find the best combination of hyperparameters.

The performance of our model was evaluated through different experiments which took advantage of large publicly available medical imaging datasets. Particularly the NIH Chest X-Ray dataset posed the greatest challenge due to its complex class labels which reflect that of real-life data. In the future we hope to recruit real medical imaging data directly from research and clinical institutions, beginning with our very own, to perform pilot studies with sequential learning that would eventually see the ML technique be integrated as a research tool. Another future direction would be to perform power analyses based on our selected model and hyperparameters. One key objective would be to evaluate the total number of datasets and sequential learning “days” needed to reach a certain accuracy threshold. Sequential learning is meant to provide an alternative approach for research protocols that acquire imaging samples in real-time. By recruiting N samples to a sequential learning CNN over M training days, we hope to generate a trend that would predict the number of “days” needed to fully train the network. This of course assumes that the few datasets on hand are representative of the entire population. We have shown recurrent instances of overfitting in our sequential learning experiments.
evaluating the NIH Chest X-Ray dataset. There is a delicate balance when evaluating the number of day-epochs required per “day” training because too few of them would not allow the network enough time to learn on the data and ultimately prolong the training duration of the entire network. On the other hand, too many day-epochs can negatively impact the network’s performance to the point of breaking it. A similar power analysis would be to evaluate an optimal number of day-epochs needed for full information extraction on N datasets in a sequential learning format. We hope to develop a dynamic algorithm that adjusts the day-epoch count depending on the number of N datasets recruited per “day”. Lastly, our multi-classification was capped at four classes at any point during sequential learning to show proof-of-concept, and in future experimentation we will expand that number to establish a stronger classification network.
References


diseases from chest x-rays using MobileNet. International Journal of Analytical
and Experimental Modal Analysis, 12(3), 665-672.

abnormality localization and diagnosis. IEEE transactions on medical imaging,
40(10), 2698-2710.

transfer learning for pneumothorax classification in chest X-ray images. Medical
Physics, 49(1), 231-243.