Gal Almog - 806709
Agathe Benichou - 345428312

# AI For Healthcare

HW 3 - Representation Learning for Bone Fractures

## 1 Introduction

In this assignment, we explore medical imaging diagnostics, focusing specifically on the task of classifying radiographic images of bone fractures. We leverage the MURA dataset, which is a significant collection of musculoskeletal radiographs labeled as normal or abnormal by board-certified radiologists. Our primary goal is to leverage unsupervised machine learning methods to detect abnormalities effectively. We start with a naive baseline model, then implement a self-supervised learning approach, and explore how the performance varies with varying amounts of labeled data. By comparing these methods, we aim to investigate the effectiveness of self-supervised learning in situations with limited labeled data, a common challenge in the medical field.

We implemented Momentum Contrast (MoCo), a self-supervised learning method, to learn visual representations from the given dataset. Unlike traditional supervised learning methods, self-supervised learning approaches do not require manually labeled data. Instead, they formulate the learning problem in such a way that the data provides its own supervision. MoCo is a self-supervised learning algorithm with a constrastive loss. Contrastive learning can be thought of as training an encoder for a dictionary look-up task. MoCo provides its own supervision by creating positive and negative pairs of augmented versions of the same image. The model is trained to identify the positive pairs among a set of negative examples. A key part of MoCo is the use of a queue and a momentum-updated encoder to maintain a dynamic dictionary of negative samples, providing a large and consistent set of negatives for contrastive learning.

This method is quite different from the baseline approach, which is a traditional supervised learning approach. In the baseline method, we use labels provided in the dataset to train the model. The model learns to map inputs (images) to the corresponding labels (normal or abnormal classes). It learns from explicit annotations - an image and its corresponding label. This can be limiting in scenarios where labeled data is scarce or expensive to obtain, as is commonly the case in the medical field. On the other hand, the MoCo approach is able to leverage unlabeled data effectively by creating its own training signals. Therefore, self-supervised learning methods like MoCo can be especially beneficial when we have a large amount of unlabeled data.

Furthermore, the MoCo model is not directly trained to classify images. Instead, it learns useful representations that capture the semantics of the data. These representations can then be used for various downstream tasks, like classification or detection, by adding a small classification head on top of the pre-trained model and fine-tuning it on the labeled data. This can be more efficient and scalable compared to training a supervised model from scratch, particularly for large and complex datasets.

## 2 Methods

### 2.1 Data Exploration

The MURA dataset is a large collection of musculoskeletal radiographs; it comprises 14,863 studies from 12,173 patients, totaling 40,561 multi-view radiographic images. These images represent seven standard upper extremity radiographic study types: elbow, finger, forearm, hand, humerus, shoulder, and wrist. Each study has been meticulously labeled as either normal or abnormal by board-certified radiologists, making it a trustworthy source for training and evaluating machine learning models. The images were retrospectively reviewed and labeled as normal or abnormal. Each image is part of a patient's study, and for each patient, there are between 1-3 images for the same bone part.

Through data exploration, we are able to see that most of the images were labeled as normal (Figure 1). For our first experiments we limited our data to only the shoulder, elbow, and hand scans; we were able to see that the shoulder scans contain many images compared to elbow and hand (Image 2). Within the shoulder category, there is a comparable number of normal and abnormal images. Within the elbow category, there are more normal images than abnormal images. Within the hand category, there are significantly more normal images than abnormal images (Figure 3). The observed imbalance between normal and abnormal images, especially noticeable in categories like the hand, underscores a potential challenge in training our model. An imbalanced dataset might predispose the model to a bias towards predicting 'normal' results, a common challenge encountered in medical imaging datasets. Moreover, the varying number of images across different categories (shoulder, elbow, hand) might affect the model's ability to uniformly learn across all types of images. For example, the model might have a tendency to perform better on shoulder images due to the relative abundance of this category in the training data. We provide examples of scan images (both normal and abnormal) from the shoulder, hand, and elbow body part scans in Figure 4. To overcome the differences in the number of images per scan in each body part, we decided to treat each image as a single entity in our model. This required converting the black and white scans to three channel images (same values replicated three times to be 3-dimensional). This way, we were able to feed the images into our model, and not have to worry about the number of images from each scan.

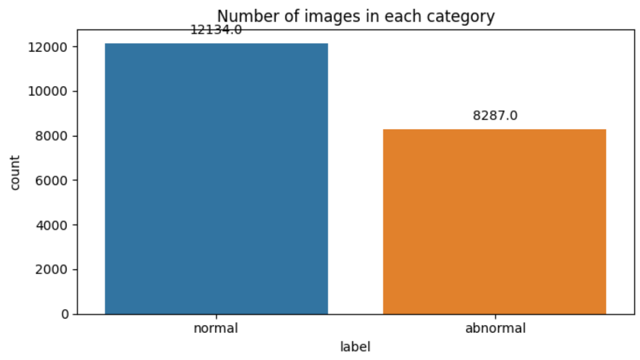

Figure 1: Number of images in each category (only shoulder, hand, elbow).

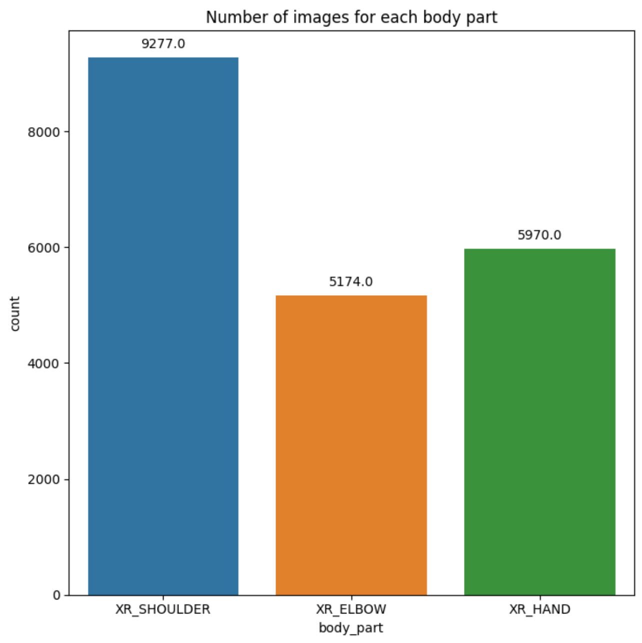

Figure 2: Number of images in each body part (only shoulder, hand, elbow).

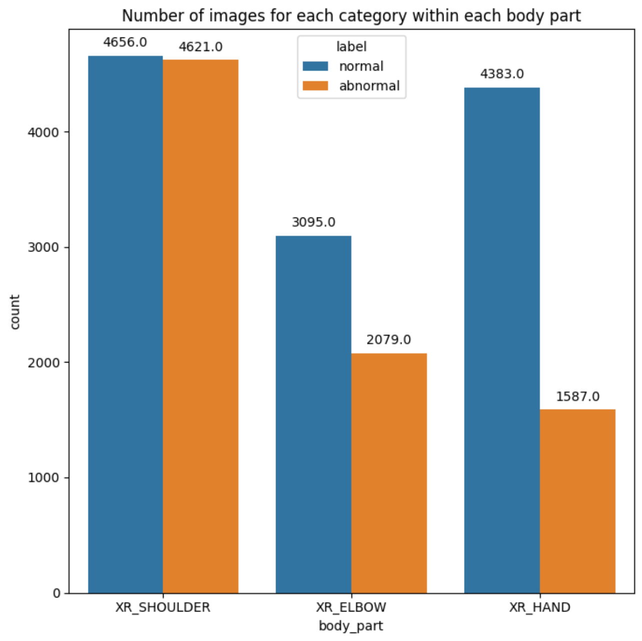

Figure 3: Number of images in each category within each body part(only shoulder, hand, elbow).

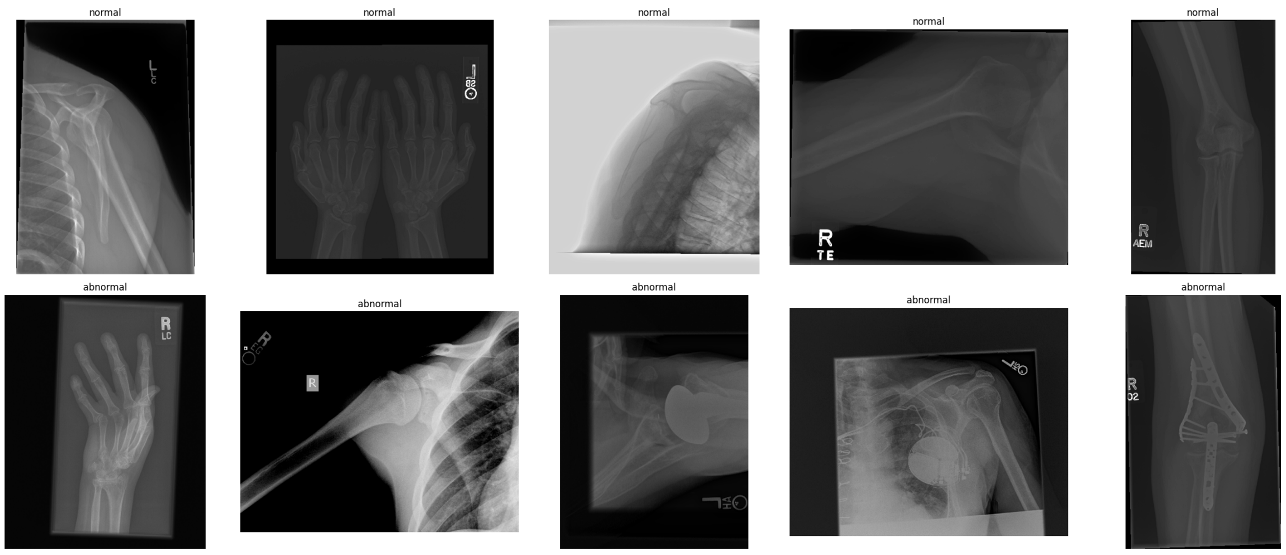

Figure 4: Sample images of both normal and abnormal images (only from shoulder, hand, elbow).

## 2.2 Naive Baseline

We implemented the naive baseline model for the MURA dataset by initializing a pre-trained ResNet-50 architecture as our base model and adapted it to suit the binary classification task at hand. This allowed us to leverage the power of transfer learning, as ResNet-50 is pre-trained on a much larger dataset (ImageNet) and can provide a robust feature extractor. This reduced the need for extensive data and computational resources.

To align with our specific task, we removed the last fully connected layer of the base model and added custom fully connected layers which finally output a single neuron with a sigmoid activation function to classify the images into 'normal' and 'abnormal' classes. For our training, we used binary cross-entropy as our loss function, with the Adam optimizer. We also included data augmentation techniques to increase the diversity of our training data and mitigate overfitting. Additionally, we split our training data further into a validation set, allowing us to monitor and evaluate the model's performance during training.

At the end of training, we evaluated the model on a separate test set, measuring the classification accuracy as our primary performance metric. The design decisions for our naive baseline were made to balance model performance, training efficiency, and simplicity, serving as a solid foundation for further optimization and complexity if required.

## 2.3 Implementing Momentum Contrast for Unsupervised Visual Representation Learning

We chose to implement Momentum Contrast (MoCo) self-supervised learning approach for visual representation learning. MoCo, a technique developed by researchers at Facebook AI, is an effective method to learn rich feature representations from unlabeled datasets. MoCo leverages the concept of a dynamic dictionary and a queue, along with a momentum encoder to create contrasting examples in the form of positive and negative keys, hence the name

'Momentum Contrast'. The primary advantage of MoCo is its ability to maintain a large and consistent dictionary, which is important for maintaining a high-quality negative sample.

The implementation of MoCo begins with training our self-supervised model on the available dataset, which does not require explicit labels. Throughout this process, the MoCo model essentially learns to distinguish between similar and dissimilar image features, thereby developing a robust and general understanding of the visual data. After training, the model produces learned representations or 'embeddings' of the input data.

We utilized the learned representations from the MoCo model and implemented the test loop with a k-NN (k-Nearest Neighbors) monitor. The k-NN monitor is used to evaluate the model's performance on a test set using a weighted k-NN search with features extracted from the model. The model was then evaluated based on its predictive accuracy, precision, recall, and other relevant performance metrics.

## 3 Results

### 3.1 Baseline Results

Table 1 shows the progression of the model's training over ten epochs. Throughout training, the training loss decreased from 0.476 in the first epoch to 0.083 in the final epoch. This suggests that the model successfully learned from the training data. The training accuracy increased from 78.03% to 96.98% over the ten epochs, indicating an improvement in the model's performance on the training set. This can also be seen in Figure 5.

| Epoch | Train Loss | Train Accuracy | Val Loss | Val Accuracy |
|---|---|---|---|---|
| 1 | 0.476 | 78.03 | 0.109 | 81.20 |
| 2 | 0.394 | 83.20 | 0.110 | 81.26 |
| 3 | 0.347 | 85.72 | 0.112 | 79.88 |
| 4 | 0.295 | 88.19 | 0.113 | 80.21 |
| 5 | 0.245 | 90.48 | 0.112 | 82.21 |
| 6 | 0.195 | 92.53 | 0.122 | 82.27 |
| 7 | 0.155 | 94.18 | 0.129 | 82.02 |
| 8 | 0.119 | 95.55 | 0.142 | 81.80 |
| 9 | 0.098 | 96.41 | 0.147 | 82.52 |
| 10 | 0.083 | 96.98 | 0.154 | 80.67 |

Table 1: Baseline model training and validation metrics.

The validation loss showed a general increasing trend, while the validation accuracy fluctuated with no clear pattern. This suggests that the model may have been overfitting to the training data, as its performance was deteriorating on the validation set, which it has not seen during training. The validation accuracy was mostly within the range of 80-82%, indicating that the model's ability to generalize to new data (validation set) is stable but doesn't improve greatly with the performance on the training data.

The baseline performance metrics on the test set are provided in Table 2. We achieved a high accuracy score of 78%. However, it is important to note that our baseline model is supervised,

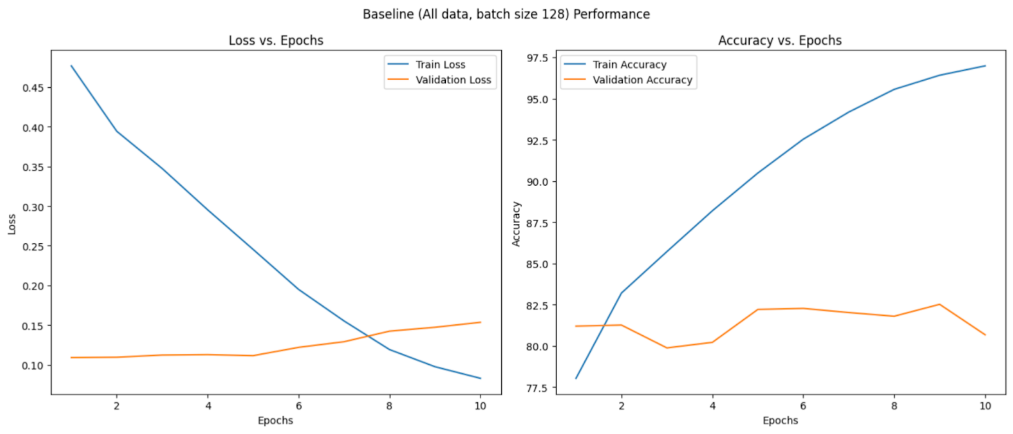

Figure 5: Baseline model training and validation metrics.

| Metric | Value |
|---|---|
| Accuracy | 0.7825 |
| Precision | 0.7708 |
| Recall | 0.7655 |
| F1 Score | 0.7682 |

Table 2: Baseline model test set performance metrics.

unlike our self-supervised MoCo model, and therefore it makes sense that the accuracy score would be higher here.

## 3.2 MoCo Results

Our best model was the MoCo model trained on the full dataset with a batch size of 256 and a batch size of 8. However, we first trained our model only on the shoulder, hand, and elbow data, to see the effect of having more data on the model's performance. In addition, we tried different batch sizes and batch normalizations. We include the results of three of these models here.

We first discuss the MoCo model trained on all of the data using a batch size of 8. The results are seen in Figure 6 and Table 3. We observe a consistent decrease in the training loss, indicating that the model successfully learned from the training dataset over time. In terms of test accuracy, the model was relatively stable, fluctuating around the 61-63% range. It started from 60.72% at epoch 0 and ended with a slight improvement at 62.56% in epoch 24. The highest test accuracy was achieved in epoch 13 at 63.13%. This moderate fluctuation might suggest the model's ability to generalize its learning to unseen data. However, the limited improvement in accuracy also implies that the model might be encountering difficulties to significantly enhance its predictive performance on the test set. It's also noteworthy that the decrease in train loss doesn't correlate directly with an increase in test accuracy, which could be an early sign of overfitting. To validate this, we would need to look into the validation loss or perform a more granular analysis over many more epochs. However, the original MoCo paper reported around a 60% accuracy on the ImageNet data; therefore, it seems like this is roughly what should be expected from this model. It is important to note that the MoCo model is self-supervised, so we expect a lwoer performance than with supervised models. In summary, while the model demonstrates steady learning, its effectiveness in terms of improved accuracy on the test set may require further optimization strategies or hyperparameter tuning.

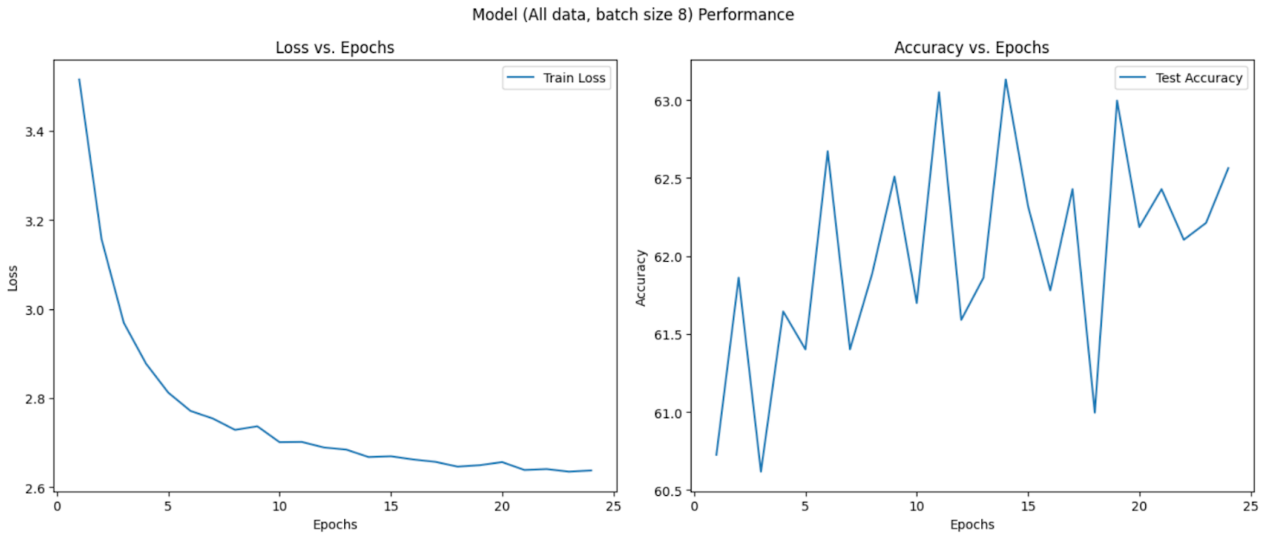

Figure 6: MoCo model test loss and accuracy over ten epochss.

| Epoch | Train Loss | Test Acc@1 |
|---|---|---|
| 1 | 5.5765 | 62.2126 |
| 2 | 5.2989 | 62.3208 |
| 3 | 5.1120 | 61.9151 |
| 4 | 4.9054 | 62.9970 |
| 5 | 4.6947 | 62.5642 |
| 6 | 4.5167 | 63.2405 |
| 7 | 4.3285 | 63.7544 |
| 8 | 4.1719 | 65.3773 |
| 9 | 4.0344 | 65.2962 |
| 10 | 3.9329 | 63.8626 |

Table 3: MoCo model test loss and accuracy over ten epochs.

The performance of the model with a batch size of 256 over 6 epochs is described in Figure 7. The training loss decreased steadily, and the test accuracy increased rather steadily. The model's performance on unseen data improved alongside the decrease in training loss, which is a positive sign of a well-generalizing model.

When we look at the test accuracy, the model starts with an accuracy of 61.48% at epoch 0, and it increases, reaching its highest point of 63.65% at the end of epoch 5. This indicates that the model's performance on unseen data is also improving alongside the decrease in training loss, which is a positive sign of a well-generalizing model. However, the accuracy does not improve consistently across epochs, experiencing a drop at epoch 2 and then a gradual increase. This fluctuation suggests that while the model is learning, its improvements are not entirely stable. The moderate number of epochs covered here may also mean that the model could benefit

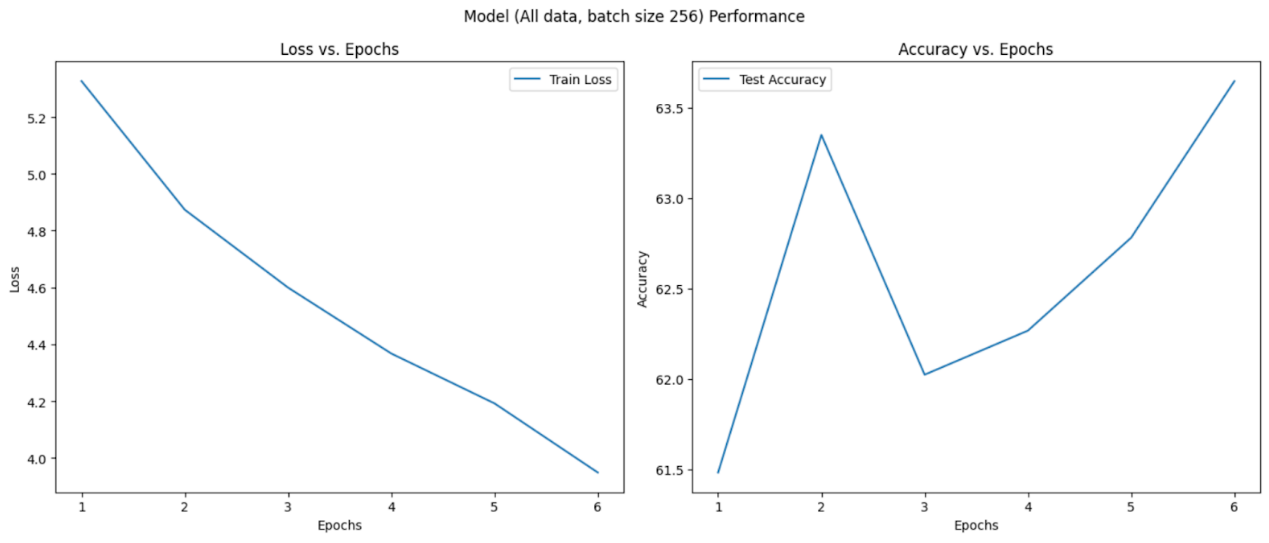

Figure 7: MoCo model test loss and accuracy over ten epochs.

from additional training to reach a higher, more stable accuracy. Overall, the model is showing promising signs of learning, but further training and possibly hyperparameter tuning might be necessary to achieve a more stable and higher test accuracy.

Our final model added the batch normalization of size 8, and was able to achieve the highest test accuracy of 65%. This makes sense, as batch normalization is a key component of the MoCo framework. The model's performance with a batch size of 256 and batch normalization of 8 over 10 epochs is reflected in Figure 8. From epoch 0 to epoch 10, we observe a steady decrease in the training loss, which indicates that the model progressively learned from the training data and improved its predictions.

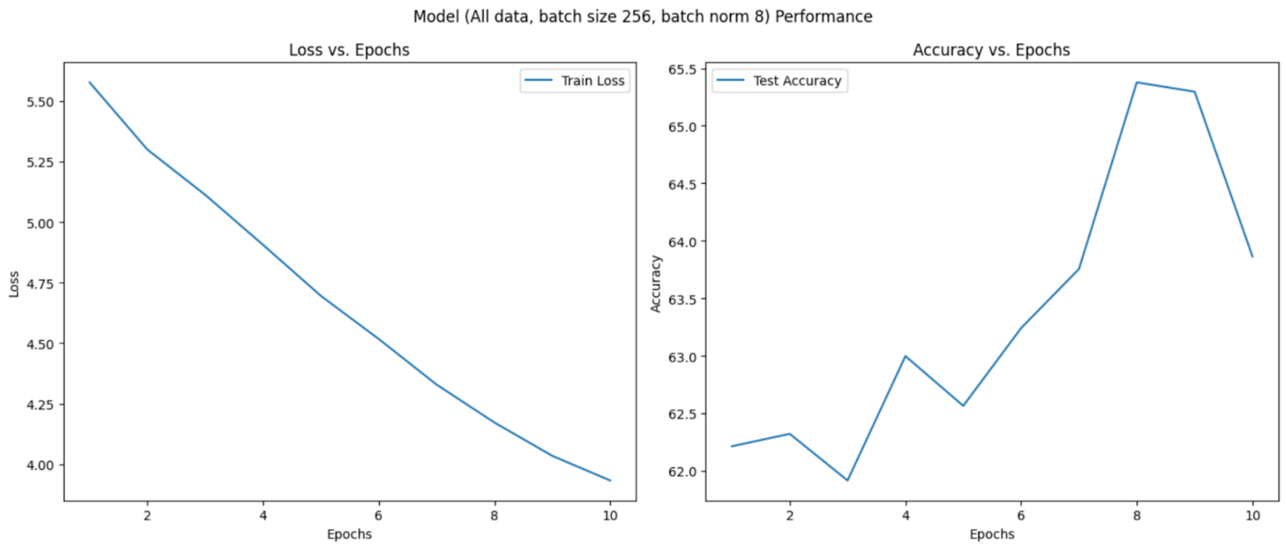

Figure 8: MoCo model test loss and accuracy over ten epochs.

When we consider the test accuracy, the model begins with a 62.21% accuracy at epoch 0.

It demonstrates a general increasing trend over the epochs, reaching the highest accuracy of 65.38% at epoch 8. However, the test accuracy does not increase uniformly across all epochs. It dips slightly at epoch 2 and again at epoch 10, indicating that the model's learning progress is not entirely linear and that it encounters some complexities in the data that are more challenging to learn. Nonetheless, the continuous overall improvement in both training loss and test accuracy indicates that the model is improving with further training. The application of batch normalization seems to have aided in stabilizing and improving the learning process, resulting in a high test accuracy of over 65% by epoch 8.

When comparing the three MoCo models trained with different parameters, some interesting patterns emerge. The first model, trained with a batch size of 8, has lower training loss values starting from the first epoch as compared to the other two models. This can be an indicator of faster convergence. However, the model's test accuracy does not seem to significantly surpass 63% during the 11 recorded epochs.

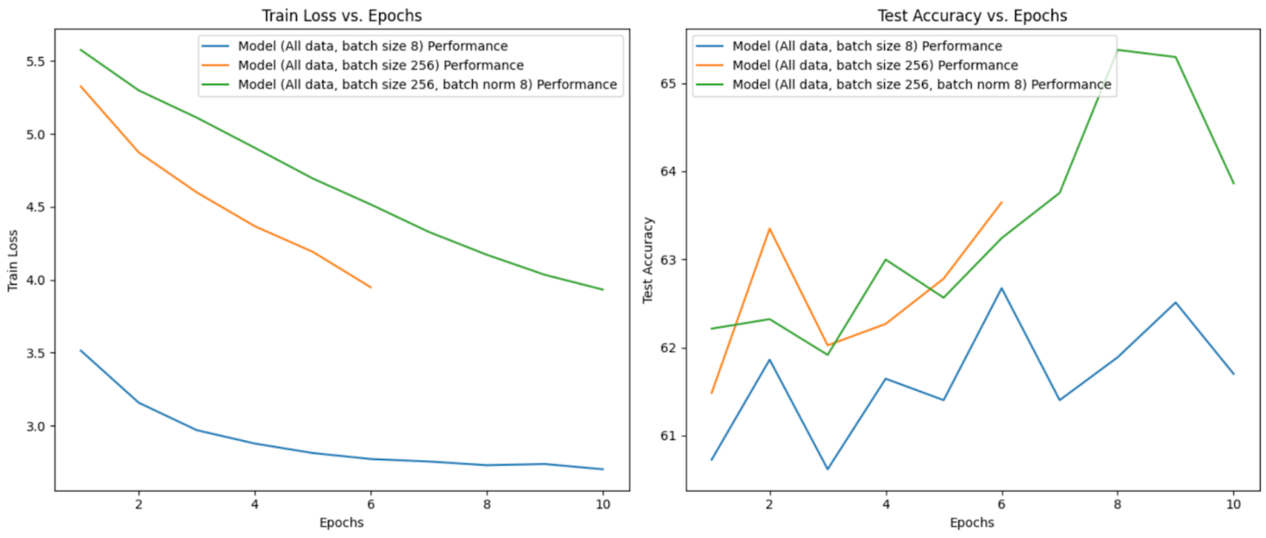

Figure 9: MoCo model test loss and accuracy over ten epochs.

The second model, using a larger batch size of 256, shows a higher training loss, which can be attributed to the increased complexity due to the larger batch size. Nevertheless, it achieved a comparable level of test accuracy, reaching about 63.65% in the sixth epoch, which suggests that the model could still generalize well to unseen data despite the higher training loss.

The third model, trained with a batch size of 256 and batch normalization of 8, starts with the highest initial training loss among the three models, which reduces considerably over the epochs, reflecting a promising learning rate. Most notably, this model achieves the highest test accuracy among the three models, reaching approximately 65.38% by the eighth epoch, indicating a superior generalization to the test data.

Therefore, the third model seems to have found a better balance between learning rate and generalization. The integration of batch normalization, despite the higher initial training loss, appears to enhance the model's performance on unseen data, which was the ultimate goal for our task.

## 4 Conclusion

In conclusion, through the implementation of ResNet-50 for our naive baseline model and the use of Momentum Contrast for unsupervised visual representation learning, we have explored an effective approach to binary classification of X-ray images as 'normal' or 'abnormal'. While the baseline model provided a solid foundation, the more complex MoCo technique offered a promising direction for learning rich feature representations from unlabeled data. Our findings suggest that applying appropriate data augmentation techniques, optimizers, and customized layers can indeed contribute to performance improvement. However, it was also observed that model improvements on the training data did not always translate to enhanced performance on the validation and test sets, potentially indicating overfitting. Furthermore, we found that adjusting parameters such as batch size and incorporating batch normalization can significantly influence the models' learning rate and generalization capability. Moving forward, these insights will serve as invaluable guidance to refine our models, explore further optimizations, and enhance our performance in detecting bone fractures in X-ray images.