Transfer Learning: Classifying Chest Medical Conditions From Chest X-Rays

Adelaide Chambers
MIT
Cambridge, MA
chambers@mit.edu

Ayush Sharma
MIT
Cambridge, MA
ayushs@mit.edu

Yuxuan Liu
MIT
Cambridge, MA
yuxuan@mit.edu

Abstract

Neural Networks have been trained to correctly classify images for many tasks, including classification of the ImageNet dataset using ResNet models. Learning from medical images, however, is often a much harder task. We selected a dataset of chest x-rays spanning 15 different disease labels and diseased area segmentations, and created multiple transfer learning models to compare their performance to each other and to a baseline model trained fully on the larger, labeled dataset we used. Overall we found that the baseline model for the classification problem dramatically outperformed the transfer classification model, which converged to a majority classifier. Our object-detection transfer model gives very few prediction results due to low confidence in the disease area detection task.

1. Introduction

Convolutional Neural Networks have brought about enormous gains in learning complex representations, yet applying their methods to the medical domain remains challenging. One of the key reasons for that is the limited availability of labeled data that Neural Networks mandate. In this paper, we explore Transfer Learning as an approach to bypass the problem of smaller datasets.

For our purposes, we chose to work with the **NIH Chest X-Ray Dataset** [4]. This dataset, extracted from the clinical PACS database at the National Institutes of Health Clinical Center, consists of 110K frontal x-ray images from 30K unique patients. The images are 1024x1024 pixels and are labeled with 14 common chest conditions: `atelectasis`, `cardiomegaly`, `effusion`, `infiltration`, `mass`, `nodule`, `pneumonia`, `pneumothorax`, `consolidation`, `edema`, `emphysema`, `fibrosis`, `pleural thickening`, `hernia`. There was also a `no_finding` label. This dataset is sufficiently large to train a Convolutional Neural Network from scratch, therefore allowing us to compare baseline model vs transfer learning model. Only 893 images were labeled with a bounding box denoting the diseased area, so our object-detection transfer model struggled with insufficient data.

The rest of this paper is organized as follows - Section 1.1 describes our data pre-processing techniques. Section 2 goes through our methods in detail. Section 3 summarizes evaluation metrics and presents our results. Finally, Section 4 concludes our findings.

1.1. Data Pre-Processing

1.1.1 Image Datagenerator

We processed information about each image from an input CSV file to create training, validation, and testing sets of different sizes. The size of the available dataset is passed in as a parameter to allow for differing amounts of data to be provided to our transfer learning model. Each set is a random, mutually exclusive group of image IDs, which are mapped to their corresponding labels. Under the default parameters, the training set uses 80% of the available dataset, and the validation and testing sets each use 10% of the available dataset.

Once the training, validation, and testing sets are constructed, we use the `ImageDataGenerator` from Keras, including introducing some random variation to the images themselves, such as randomly flipping some images vertically, slight random rotations, and other small manipulations to the input data.

1.1.2 Bounding Box Datagenerator

We modify the bounding box information from the CSV file to contain the image IDs, their labels, and the corresponding bounding box coordinates. The total dataset is then split into a training,validation and testing set, distributed uniformly across the different disease labels. The training, validation, and testing sets are additionally convert to an XML file for the Yolo off-the-shelf and object-detection transfer models.

2. Methods

2.1. Baseline Resnet

ResNet [1] is one of the most successful image recognition Convolutional Neural Networks. As CNNs started becoming deeper, researchers faced the problem of degradation. That is, the deeper layers of the CNN actually ended up degrading the accuracy instead of modeling more complex data. The Deep Residual Network solves this problem, and it does so by introducing Residual Blocks. Figure 1 shows a residual block, which forms the basis of ResNet. Eq. 1 describes the transformation performed by that residual block within a Deep Residual Network.

$$\mathbf{y} = \mathcal{F}(\mathbf{x}, W_i) + \mathbf{x} \tag{1}$$

For this project, we decided to use the ResNet50 model for our baseline classification training on the chest x-ray dataset. ResNet50 is the 50-layer version of ResNet that performs quite well on the standard ImageNet dataset, with a top-5 classification accuracy of 92.9%.

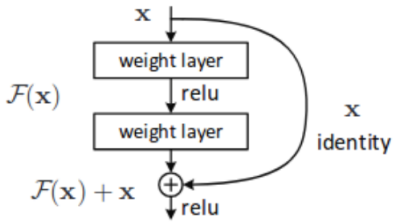

Figure 1. Residual Block used in ResNet architecture

We trained a ResNet50 from scratch on the NIH chest x-ray dataset. For the first baseline, we trained the model out-of-the-box with a `softmax` activation layer and `categorical_crossentropy` loss function. We used a Google Cloud Deep Learning VM instance for the GPU accelerated training of our models throughout this project.

Next, we trained another ResNet50 modified for the multi-label multi-class classification problem as described in the NIH chest x-ray dataset. The class labels for a given chest x-ray are not mutually exclusive and vary independently. Therefore, the modifications to ResNet included choosing the `sigmoid` function as the activation for the output layer, and using `binary_crossentropy` as the loss function. Choosing the sigmoid, $\sigma(z) = \frac{1}{1+\exp(z)}$, gives us the desired flexibility as the multiple labels are now independent a priori.

2.2. Transfer Model

Widely available medical datasets are often insufficiently large to completely train new models. To address this general problem, we used a ResNet50 model that was pre-trained on the ImageNet dataset, provided by Keras. This model could then exploit all of the information on understanding images that it learned from its ImageNet experience and use it to help compensate for the relative scarcity of x-ray data. One final, fully-connected layer was added after the ResNet50 model layers to connect the 1000 outputs of the ResNet's final layer to 15 final output nodes, representing the 15 possible binary labels associated with our chest x-ray dataset. As in the baseline model, we used a `sigmoid` activation function for the final layer and `binary_crossentropy` for our loss function. We also configured the model to report its `accuracy` metric.

We allowed only the final c layers of the ResNet50 model to be trained, and fixed the weights in all proceeding layers. We did not consider layers with no parameters, which could therefore never be trained, to be one of the c layers. Given the relatively small dataset provided to our transfer learning model, we chose $c \in [0, 19]$. It should be noted, though, that $c + 1$ layers were always trained, since the fully-connected layer at the end was always initialized with weights randomly drawn from a uniform distribution and then trained on the data.

We also varied the size of our input dataset. Medical datasets often don't exceed 5000 images [2], so the transfer model's dataset size s was chosen such that

$$s \in \{100, 200, 300, 400, 500, 750, 1000, \\ 1500, 2000, 2500, 3000, 4000, 5000\}$$

The optimal combination of these two hyperparameters, $\hat{c}$ and $\hat{s}$, was determined using a grid search over all possible combinations of the two, in a training run lasting five epochs with one hundred steps in each epoch.

The hyperparameters were tuned according to the maximum accuracy they obtained on the validation set.

$$\hat{c}, \hat{s} = \text{argmax}(\texttt{accuracy}_{\texttt{validation}}(c, s))$$

This optimal combination was then used to train the model for one hundred epochs with one hundred steps each, and evaluated on a test set, as described in Section 2.2.

2.3. Object-Detection Model

As the NIH dataset contains 881 images with the disease label and diseased area segmentation, we also implemented an object detection model. We implemented 3 methods for detecting the disease area: 1) Yolo OTS (off-the-shelf) model; 2) a transfer-learning-based Yolov2 pre-trained model; 3) an 18-layer simplified Deep Convolutional Neural Network inspired by the ResNet structure and Yolo algorithm.

2.3.1 Methodology of Evaluation

Although the input label is $y \in \{0, 1\}$, the prediction label is a confidence level $\hat{y} \in [0, 1]$. Therefore we provide two methods of evaluation: 1) $\hat{y}$ as a `One-Hot Encoding`; 2) evaluating with the predicted confidence level. The prediction is evaluated by the `f1` measure, which is calculated using `Precision` and `Recall`.

$$\texttt{Precision} = \frac{1}{n}\sum_{i=1}^{n} \frac{Area(y) \cap Area(\hat{y})}{Area(\hat{y})}$$

$$\texttt{Recall} = \frac{1}{n}\sum_{i=1}^{n} \frac{Area(y) \cap Area(\hat{y})}{Area(y)}$$

The final result for evaluation is given by the `f1` measure.

$$F_1 = \frac{2PR}{P+R}$$

To measure our model's ability to predict larger diseased areas, we also evaluate its performance on bounding boxes larger than 384x256 pixels.

2.3.2 YOLO(You Only Look Once) OTS

YOLO is a commonly used object detection model for detecting daily objects [3]. It uses a single Convolutional Neural Network to predict the bounding box location. Yolov2 is originally designed to detect 80 daily objects. We first test the detection accuracy of Yolov2 OTS and describe the results in Section 3.3.1.

2.3.3 YOLOv2 Transfer Learning

We then used the dataset labeled with the bounding box locations to train a custom Yolov2 to detect disease areas. The filter in last convolutional layer is decreased from 425 to 65 as our dataset only contains 8 labels as compared to the original Yolov2's 80 labels. The region layer is also modified for our specific task. We set the batch size to be 16 and used the Yolov2 pre-trained weights. Section 3.3.2 gives the results of this model after training for 40 epochs.

2.3.4 Custom 18-Layer CNN

As neither the Yolo OTS model nor the Yolov2 transfer model converge in an acceptable training time, we built our own simplified version of an 18-layer Deep Convolutional Neural Network inspired by the Yolo algorithm. We first subdivided each input image into a 16×16 grid, where each cell is 64×64 pixels as shown in Figure 2. We also generated a `One-Hot Encoding` vector, where the index corresponding to each cell was 1 if more than 50% of that cell was covered by the bounding box.

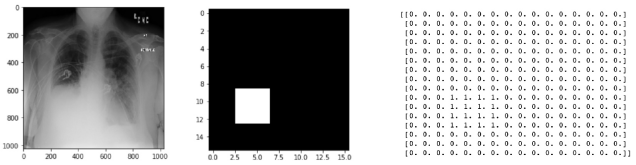

Figure 2. **Left**: Grayscale chest x-ray **Middle** and **Right**: output grid and visualization

Inspired by U-Net [1] and ResNet structure, our model contains 18 convolutional layers followed by 1 fully connected layer. Batch normalization layers are introduced to accelerate training and stability by reducing internal co-variate shift. We chose the `sigmoid` function as the activation for the output layer. `cosine_proximity`, `mean_squared_error`, and `L1_loss` were evaluated as loss functions. Since the model trained with L1 loss tends to predict much less confidently than the model based on mean squared error, mean squared error was used to train the final model.

We used Adam as our optimizer with a learning rate of 0.005. We used a batch size of 1 to train the model for 5 epochs with 764 steps each, and evaluated on a test set. The results are described in Section 3.3.3.

3. Results

3.1. Baseline Resnet Performance

For the baseline ResNet50, we trained both a multi-class single label and a multi-class multi-label model. In our case, we noticed a significantly better out-of-the-box performance when we switched to the multi-label setting by using the sigmoid $\sigma(z)$ activation on the output layer and using `binary_crossentropy` for the loss. Our loss improved from **2.43** to **0.24**, and the accuracy, as measured on the test set of 10,000 images, went up to **92%** from a meager **55%**. The figures below summarize our result with the two baseline ResNet50 models.

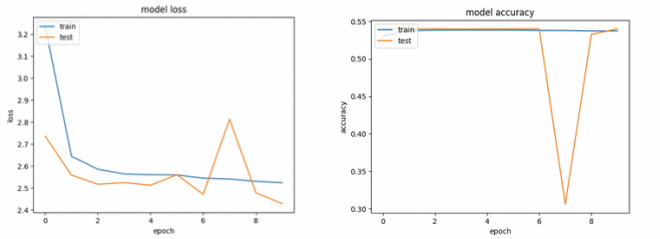

Figure 3. **Left**: ResNet50 baseline single label loss and **Right**: Single label Accuracy

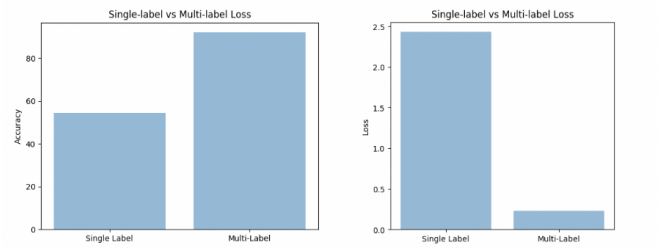

Figure 4. **Left**: Accuracy comparison and **Right**: Loss comparison

3.2. Transfer Model

The Transfer Model ended up converging to a majority classifier. When choosing the hyperparameters, we found significant variation in performance on our validation set as shown in Figure 5. As we were only training for five epochs with 100 steps each, this is likely explained by the wider variation of data labels provided to our training and validation sets.

Due to the high level of variability in our results, we actually chose to use two combinations of hyperparameters to perform a longer testing run with. We chose $(\hat{c}, \hat{s}) = (2, 5000)$ and $(\hat{c}, \hat{s}) = (19, 3000)$ as the two hyperparameter combinations to test further. On our validation set, both of these combinations slightly outperformed a simple majority classifier (by about 6.59% and 6.33% respectively. Another reason we felt more confident in these options was that, fixing the dataset size, each performed similarly well for different values of c, suggesting some robustness to this selection.

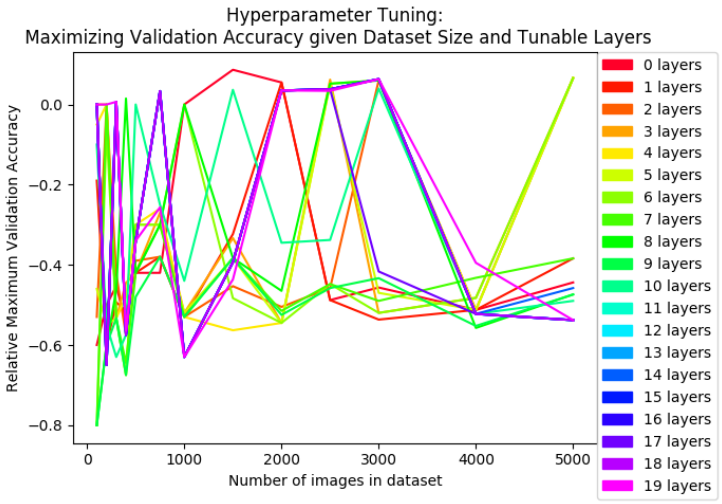

Figure 5. Validation performance of different combinations of restricted dataset sizes and trainable layers. The validation accuracy is shown relative to the performance of a majority classifier on the validation data to prevent the randomness in the validation set from corrupting our results.

We then performed a longer training run, with 100 epochs of 100 steps each, to test the model. The results of these iterations are given in Figure 6 and Figure 7.

In both cases, the performance of the model on the test set converged to almost exactly that of the simple majority classifier (within 0.01, with this variation likely being caused by exactly which 500 images were sampled to perform a testing run).

Upon closer inspection, the model does in fact act as a majority classifier. It correctly classifies 100% of the `no_finding` labels and incorrectly classifies all other labels. `no_finding` makes

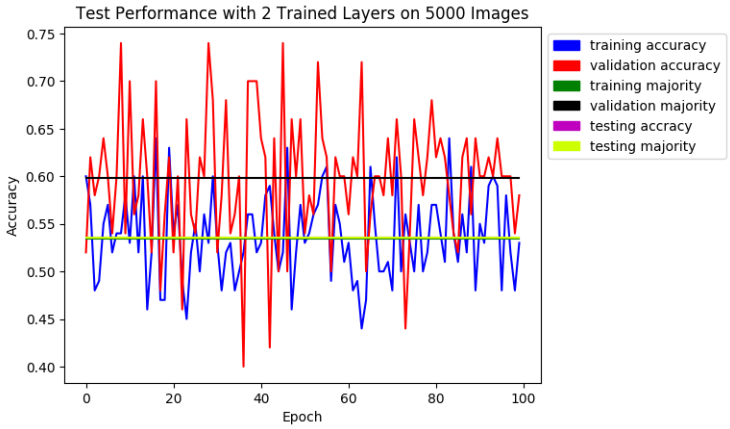

Figure 6. Performance during a training run using 5000 images and 2 trainable layers, followed by a testing run on the resulting model. The comparison of the performance of the majority classifier is also given.

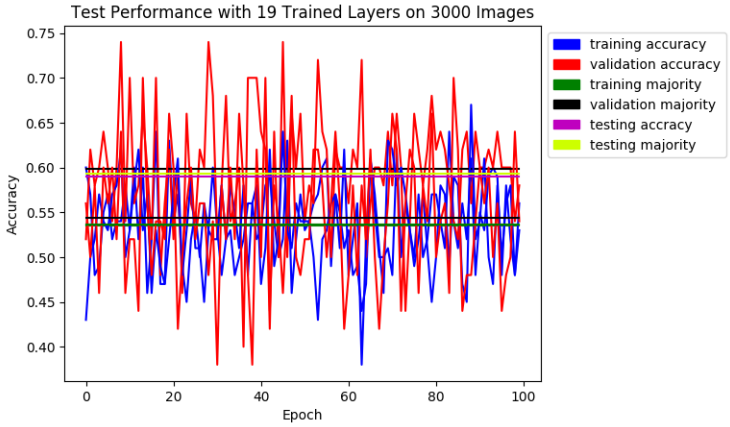

Figure 7. Performance during a training run using 3000 images and 19 trainable layers, followed by a testing run on the resulting model. The comparison of the performance of the majority classifier is also given.

up the majority of the data, so this behaves exactly as a majority classifier in both $(\hat{c}, \hat{s})$ values we considered.

3.3. Object Detection Model

3.3.1 Yolo OTS Model

The Yolo OTS model achieves an f1 measure of approximately 15% on the bounding box location.

3.3.2 Yolov2 Transfer Model

The Yolov2 transfer model results were poor. Predictions of the bounding box were visible in the first few epochs, but disappear after continuing to train for 20 epochs due to a deterioration in confidence level. It re-appears after another 50 epochs, but only 24 out of 129 test images had any predicted diseased locations. This might be because we were only adjusting the final layers of the Yolo model, and since a chest x-ray is drastically different from other daily objects, the model tends to predict nothing because of the high confidence threshold.

3.3.3 Custom 18-Layer CNN

Figures 8 and 9 show the prediction results of our model trained on 670 and 760 images respectively.

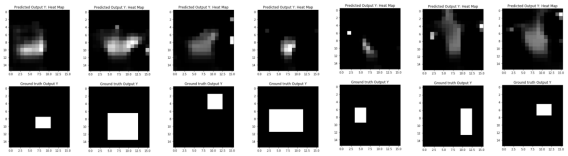

Figure 8. **Top**: Prediction results from a 670 image training set **Bottom**: Ground Truth

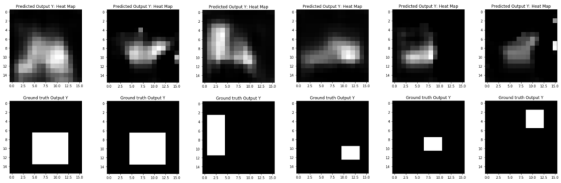

Figure 9. **Top**: Prediction results from a 764 image training set **Bottom**: Ground Truth

For the model trained on 764 images for 5 epochs, the training and validation loss drop from 0.24 to 0.06, as shown in Figure 10. The `recall` one-hot encoding $\hat{y}$ is 100%, indicating that the model correctly outputs a positive prediction to the diseased area.

The `precision` and `recall` tested on all disease sizes based on the predicted confidence level is 17.27% and 21.32% respectively. The f1 measure for all diseases sizes is 19.09%.

The `precision` and `recall` tested on larger disease areas based on the predicted confidence level is 25.42% and 24.04% respectively. The f1 measure for larger diseased area is 24.71%. We observed an increase in the f1 measure by 5% for a larger

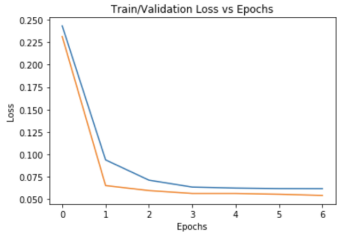

Figure 10. Train/validation loss during 5 epochs of train

diseased area. We also observed an increase in the f1 measure by 9% with only 90 more training images. Given that our model was trained on only 764 training images, we believe a more accurate prediction would be possible with an greater number of training images.

4. Conclusion

Both our classification and object detection tasks reported better results with their corresponding baseline models. Since for both transfer models, we used a model that was pre-trained on ImageNet and then adjusted hyperparameters, it is likely that early convolutional layers are important to capture key features necessary for our task.

5. Contributions

- **Ayush** : Paper = {Introduction section, 2.1 Methods, 3.1 Results }, Code = {Setting up and training ResNet50 baseline model + multi-label model; Evaluating ResNet50 baseline + multi-label model on test data set; setting up Google Cloud Deep Learning VM environment with GPU support, General Proof-reading}

- **Addie** : Paper = {Abstract, 1.1.1 Data Pre-Processing, 2.2 Methods, 3.2 Results}, Code = {Building datagenerator for simple image data and creating training, validation, and testing sets; Creating transfer model based off of the ResNet50 model; Tuning hyperparameter values and evaluating the resulting model; Identifying the dataset and overall direction for the project}

- **Yuxuan**: Paper = {1.1.2 Data Pre-Processing, 2.3 Methods, 3.3 Results, 4 Conclusion}, Code = {Data preprocessing and generator for disease area detection; Experiment on Yolo OTS model; Train and Evaluate on Transfer model based off of the Yolov2 model; Setting up and training a disease area detection custom model inspired by ResNet structure and Yolo algorithm, develop evaluation metrics and evaluation on test data set; setting up AWS VM environment}

References

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[2] Marc D. Kohli, Ronald M. Summers, and J. Raymond Geis. Medical image data and datasets in the era of machine learning—whitepaper from the 2016 c-mimi meeting dataset session. *Journal of Digital Imaging*, 30(4):392–399, Aug 2017.

[3] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015.

[4] Xiaosong Wang, Yifan Peng, Le Lu, Zhiyong Lu, Mohammadhadi Bagheri, and Ronald Summers. Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases. In *2017 IEEE Conference on Computer Vision and Pattern Recognition(CVPR)*, pages 3462–3471, 2017.