Machine Learning Engineer Nanodegree

Capstone Project

Nory Diankov
June 21, 2018

I. Definition

Project Overview

Much recent interest has been garnered in the application of AI in healthcare, particularly in the diagnosis of diseases. Lung conditions, a category that includes both communicable and non-infectious diseases, have remained the second leading cause of death globally in the last 15 years, just after heart disease, according to the World Health Organization. In 2016 alone, diseases such as chronic obstructive pulmonary disease, tuberculosis, lung cancer, and lower respiratory infections accounted for 9 million deaths (The World Health Organization , 2018).

Despite the critical need for early screening and detection, it is estimated that two-thirds of countries do not have sufficient access to basic radiology services, such as a simple x-ray or ultrasounds. In particular, low-income countries are handicapped by an insufficient infrastructure and a considerable burden of disease, compounded by the need to allocate scarce resources to basic necessities such as clean water and nutrition. As a result, common limitation to the high mortality rate is a lack of staff and the cost of hiring radiologists (Silverstein, 2016).

In response, this study contributes to the growing body of literature by attempting to utilize deep learning to provide a model to aid computer-aided detection and diagnosis (CAD) in the field of pulmonary diseases.

Problem Statement

A lack of access to well-trained radiologists could delay diagnosis and the prevention and identification of deadly lung diseases. It is estimated that in a country of 43 million, Kenya has only 200 radiologists, whereas, one Boston hospital, Massachusetts General, alone has 126. Although there have been experiments in telemedicine—the practice of available experts in the US and Canada reading and diagnosing electronic medicine records of patients in countries of high-need—there are numerous challenges to overcoming delays associated with different time zones and the speediness of response (Wamala, 2013).

Therefore, the problem is to develop software that can 1) distinguish between normal and abnormal x-ray images, and 2) perform “diagnosis”, which in the case of radiological evidence usually entails implicating several possible conditions of roughly equal

probability. This study aims to begin the development of such a tool; for the purposes of the Udacity nanodegree, my goal is to show an algorithm that is capable of these tasks, even if it is not fully optimized.

Metrics

To measure the effectiveness of the proposed model's performance, I will calculate the accuracy of classification on a test dataset of images not previously "seen" by the CNN. Further in the report I provide a detailed definition of this metric.

Classification Accuracy is the ratio of number of correct predictions to the total number of input samples.

$$Accuracy = \frac{Number\ of\ Correct\ predictions}{Total\ number\ of\ predictions\ made}$$

This metric is optimal in evaluating the efficacy of the selected Machine Learning algorithm because the number of samples is generally well-distributed over the three classes, which renders it particularly effective in calculating the true accuracy of the model's predictions without bias from skewed data.

II. Analysis

Data Exploration and Exploratory Visualization

I used a dataset of 5,232 chest X-ray images taken from 5,856 pediatric patients, 1 to 5 years old, from the Gaungzhou Women's and Children's Hospital. Academic physicians have classified 3,883 of these as depicting pneumonia, within which 2,538 bacterial and 1,345 viral pneumonia cases, and 1,349 images as normal. The dataset is publicly available (Kermany, Zhang, & Goldbaum) and was used in a published study (Kermany, 2018). The images are of varying sizes and are grayscale.

The dataset images are labeled with the labels "Normal," "Bacterial" and "Viral" to refer to the clinical findings of healthy lungs, bacterial pneumonia and viral pneumonia. The dataset presents several challenges. The first challenge is intrinsic to the nature of medical radiographs: they are grayscale and low in contrast, making distinctions between different clinical findings difficult to make. The second challenge in this dataset is unique to this data: the images are of children between the ages of one and five, making the acquisition of stable radiographs more difficult and resulting in contortions in body positions. Fig. 1 shows several representative images of the dataset with their labels.

I randomly divided the data into "train", "validation" and "test" folders. Within each of these folders, the images were divided into sub-folders with the names of the three

classes. There were 5856 total X-ray images, with 4765 training images within which 1342 "normal" images, 1202 images with viral pneumonia and 2223 images with bacterial pneumonia. Fig. 2 shows the distribution of image labels in the training and test datasets.

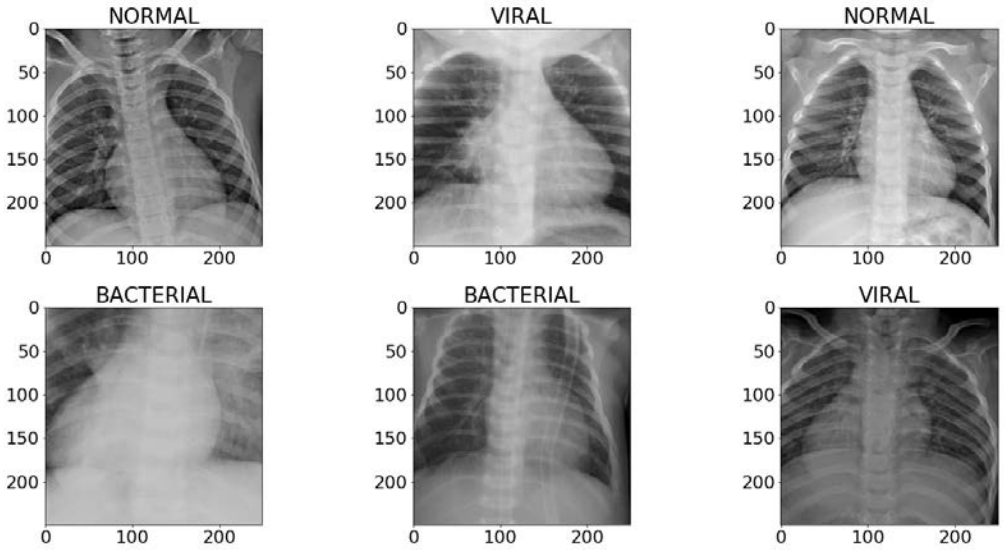

Fig. 1 Six representative, labeled images from the database after center-cropping to 250 pixels.

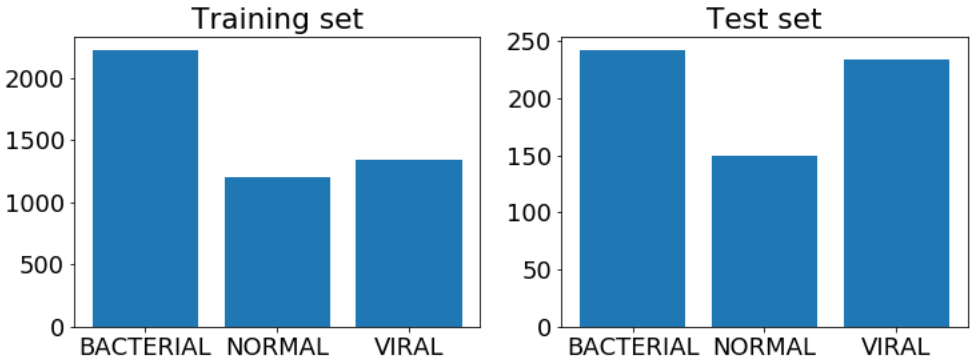

Fig. 2 Categories of images in the training and test data sets.

Expected Algorithms and Techniques

From the outset, I intended to use an image classifying algorithm that builds upon a well-known convolutional neural network (CNN), such as VGG-16, ResNet or DenseNet variants. CNNs trained on the ImageNet database contain multiple convolutional blocks each consisting of two or more conv2d layers of size 32 – 128 pixels, with kernels of sizes such as 3x3 or 7x7, nonlinear activation functions, 2d-pooling layers and dropout layers. The attraction of such CNNs is their ability to extract, map and recognize multiple levels of features. Finally, flattening and global pooling or averaging layers are applied, followed by a sequence of one or more fully-connected layers with the final one often applying the LogSoftmax function, which allows the direct interpretation of the resulting output weights in the output vector as probabilities. A "topk" method can finally retrieve the top-k most likely labels.

My plan was to use PyTorch rather than Tensorflow due to the ease with which I can integrate Torchvision with my GPU and the attractive nature of the modules and attributes available in Torch modules.

The theoretical gist behind the use of CNNs for image classification is the use of kernels to extract first "high-level" features and then, with increase in depth and number of layers, to extract some of the finer details. To prevent overfitting, dropout layers are used. The ability of the model to actually classify depends in large part on the use of specific non-linear "activation" functions, such as LogSoftmax.

Benchmark Model

Recent forays into CAD regarding pulmonary diseases have been made by researchers, particularly at Stanford University. A study published in 2017 utilized an algorithm, CheXNet, of a 121-layer convolutional neural network that takes X-ray image and returns an output of the probability of pathology. In this specific case, CheXNet focuses on identifying pneumonia, with a F1 performance of 0.435 that exceeds the average radiologist performance of 0.387 (Rajpurkar, et al., 2017).

The model architecture of CheXNet utilizes DenseNets to the improve flow of information and gradients through the network, with the final fully connected layer replaced with one that has a single output. A sigmoid nonlinearity was applied thereafter. Network weights were initialized with those from a model pretrained on ImageNet, using Adam with standard parameters ($\beta1 = 0.9$ and $\beta2 = 0.999$) and minibatches of size 16. The authors used an initial learning rate of 0.001 that is decayed by a factor of 10 each time the validation loss plateaus after an epoch, and pick the model with the lowest validation loss. A study published in the journal Cell in 2018 also uses a standard, ImageNet-pretrained network that was directly applied to the images with very minimal changes to account for the number of labels in the classifier (Kermany, 2018)

For my benchmark, I use the basic metric of accuracy in predicting the image labels from images in a test dataset. More specifically, accuracy is calculated as in the following code:

```
for ii, (images, labels) in enumerate(testloader):
   model.eval()
   inputs = Variable(images, requires_grad = False).cuda()
   targets = Variable(labels, requires_grad = False).cuda()
   output = model.forward(inputs).cuda()
   test_loss += float(criterion(output, targets))
   ps = torch.exp(output).data
   equality = (targets.data == ps.max(1)[1])
   accuracy += equality.type_as(torch.FloatTensor()).mean()
   accuracy = float(accuracy)
print("Validation Loss: {:.3f}.. ".format(test_loss),
    "Validation Accuracy: {:.3f}".format(accuracy/len(testloader)))
```

After the images torch tensors have been subjected to the forward pass of the model (with gradients frozen for inference mode), the test_loss is calculated from the class NLLloss in the torchvision modules (here shown as “criterion”). The output is then exponentiated and a Boolean comparison is made between predictions and targets. This serves as the foundation of the “accuracy” calculation.

III. Methodology

Data Preprocessing

I performed standard image transformations using the Torchvision-implemented PIL transformations (torchvision.transforms). The images in the training dataset underwent resizing, center-cropping to ImageNet standards, random rotations by 30 degrees and random horizontal flips. The transformations are as follows:

```
data_train_transforms = transforms.Compose([
   transforms.Resize(256),
   transforms.CenterCrop(250),
   transforms.RandomHorizontalFlip(),
   transforms.ToTensor(),
   transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])])
```

```
data_test_transforms = transforms.Compose([
   transforms.Resize(256),
   transforms.CenterCrop(250),
   transforms.ToTensor(),
   transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])])
```

Implementation

To classify the images into the three classes of "normal, bacterial and viral", I used a DenseNet variants from Torchvision, the CNN densenet201 consisting of 201 layers connected to a classifier I built made of fully-connected layers.

DenseNet consists of several large "DenseBlocks", each of which is comprised of multiple convolutional (conv2d) layers of varying output and kernel sizes, nonlinear activation 'ReLu", appropriate pooling and averaging layers, as well as dropout layers.

The architecture of the network is shown below.

```
DenseNet(
  (features): Sequential(
    (conv0): Conv2d(3, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3), bias=False)
    (norm0): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (relu0): ReLU(inplace)
    (pool0): MaxPool2d(kernel_size=3, stride=2, padding=1, dilation=1, ceil_mode=False)

  (denseblock1): _DenseBlock(
      (denselayer1): _DenseLayer(
        (norm1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (relu1): ReLU(inplace)
        (conv1): Conv2d(64, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (relu2): ReLU(inplace)
        (conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      )
   .
.
.
      (denselayer6): _DenseLayer(…
 .
.
.
    (transition1): _Transition(
      (norm): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (relu): ReLU(inplace)
      (conv): Conv2d(256, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (pool): AvgPool2d(kernel_size=2, stride=2, padding=0)
    )

    (denseblock2): _DenseBlock(
.
.
.
      (denselayer32): _DenseLayer(
        (norm1): BatchNorm2d(1888, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (relu1): ReLU(inplace)
        (conv1): Conv2d(1888, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (relu2): ReLU(inplace)
```

```
        (conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      )
    )
    (norm5): BatchNorm2d(1920, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  )
  (classifier): Sequential(
    (0): Linear(in_features=1920, out_features=512, bias=True)
    (1): ReLU(inplace)
    (2): Dropout(p=0.5)
    (3): Linear(in_features=512, out_features=256, bias=True)
    (4): ReLU(inplace)
    (5): Dropout(p=0.5)
    (6): Linear(in_features=256, out_features=128, bias=True)
    (7): ReLU(inplace)
    (8): Dropout(p=0.5)
    (9): Linear(in_features=128, out_features=64, bias=True)
    (10): ReLU(inplace)
    (11): Dropout(p=0.5)
    (12): Linear(in_features=64, out_features=2, bias=True)
    (13): ReLU(inplace)
    (14): Dropout(p=0.5)
    (15): LogSoftmax()
  )
)
```

I added a classifier comprised of three fully-connected layers, ReLU activation and dropout in order to allow the classification of the images into three categories. I used multiple dropout layers to avoid overfitting. A Logsoftmax function is applied at the end and the negative-log loss function is used as criterion (NLLloss). Adam optimizer was used with initial learning rate of 0.05-0.1. Up to 10 epochs were used for training, although in practice it was observed that loss sharply declined and did not further improve within 1-2 epochs. A learning-rate scheduler (lr_scheduler) was used between epochs.

Transfer learning was used such that the model was pre-trained on the ImageNet database, and gradients were only calculated for the parameters of the classifier. The CNN was implemented within the PyTorch/Torchvision 0.4.0 framework.

One of the complications I encountered was in initially not matching the output size of the feature-extracting (convolutional) layers of the model with the input size of the classifier that I custom-built. Since I did not modify the final convolutional layers, the classifier expected to accept as input an exact number – 1920 in this case. I realized this after carefully examining the model structure, after which the model was able to implement both forward passes and back-propagation with gradient calculations (the latter for the classifier only).

A second complication to keep in mind for the future was the need to carefully keep track of what form the images are in at different steps of the process. There are at least three different data structures that are used for the images. Initially, the images are loaded as PIL images and they take the form of PIL Image classes. This is necessary for the application of suitable image transforms. Next, the images are converted into “Torch” tensors, 4-dimensional classes that contain the image size tuple, the color channels, and the number of images in the batch. Finally, following the use of the CNN model, it is useful to visualize results by feeding individual images to the trained model and performing inference. At this point, the images are to be converted to numpy arrays and

visualized with Matplotlib (PIL can also be used though it is not as flexible as numpy). This last conversion can be tricky: the torch tensor has to be transposed due to the different ordering of channels in the tensor and the appropriate normalization has to be performed.

Refinement

I attempted numerous strategies to improve the accuracy of the classifier:

- Not being satisfied with the accuracy of the model when it encountered previously "unseen" data, I first wondered whether the pre-trained features from ImageNet might be unsuitable for medical images such as grayscale X-ray images of lungs. Using the fully-untrained DenseNet network, I initialized the weights with random numbers and fully trained all 201 layers. Even though I was able to train the entire network within a reasonable amount of time, doing so did not result in a noticeable improvement in the accuracy.
- Based on this result, I next considered whether appropriate image augmentation might help. I therefore performed additional image pre-processing steps in the training data such as rotations and flips. Again, no significant improvement resulted. This result led me to believe that even more sophisticated data augmentation may have to be performed. One way to do this, given more time, is to carefully examine the actual extracted features from the convolutional layers by mapping them spatially onto the image. At what point does the network recognize what features? Which of those features are, from a medical point of view, critical to the accurate classification of the image? Can we either augment the image or finetune those specific layers in the network so that model "comprehension" becomes better?
- I also modified the nature of the classifier – different numbers of fully-connected layers. The classifier takes as input 1920 features and essentially needs to output only 3. I found that the use of several fully-connected layers that more gradually go from 1920 to 3 leads to higher accuracy. Therefore, the final version uses five fully-connected layers: 1920 → 512, 512 → 256, 256 → 128, 128 → 64 and 64 → 3, with reLU activation and dropout with probability 0.5 after the first four of these layers. Finally, a logSoftmax function is applied.

IV. Results

Model Evaluation and Validation

I achieved classification accuracy of ~ 0.6 on the test image dataset (with minor variations based on learning rate). The change in the loss rate with steps during the first

epoch of the training process is shown in fig. 5. The loss rate exhibits a fast drop in the early phases of the training.

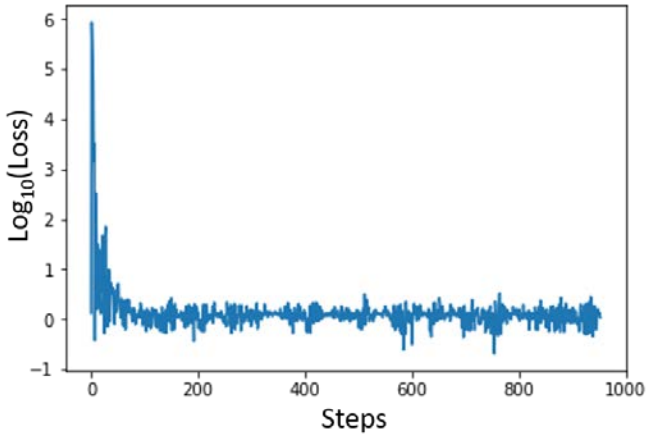

Fig. 4 Loss (a result of the NLLloss criterion function) as a function of training steps

To evaluate robustness, I used the model to make inference on two separate test datasets. I found that the model is robust in the sense that accuracy was rather stable between the two datasets. However, accuracy remains lower than desired. My conclusion is that the training process does "learn" some of the important features and the model can then apply those to classify images up to an extent. However, the model misses some additional important features that require greater sophistication specifically for the purposes of classifying difficult and subtle images such as X-ray radiographs.

Justification and Comparison

Therefore, I do not obtain a satisfactorily high accuracy rate on the test dataset. In comparison, using a customized version of an ImageNet-trained CNN and training with 100 epochs, (Kermany, 2018) reports a 0.9 accuracy on a test dataset of lung X-ray images. I attribute the lower accuracy in my work to the inadequate time available for proper image pre-processing and augmentation. Given the subtle differences in classes, even a highly-trained and very deep CNN cannot ordinarily attain very high accuracy.

V. Conclusion

Free Form Visualization

To enable better visualization of the grayscale images, I used the fact that RGB values of grayscale images (dtype = uint8) are largely uniform and extracted one of the channels, which was then plotted with a color map as shown in Fig. 5

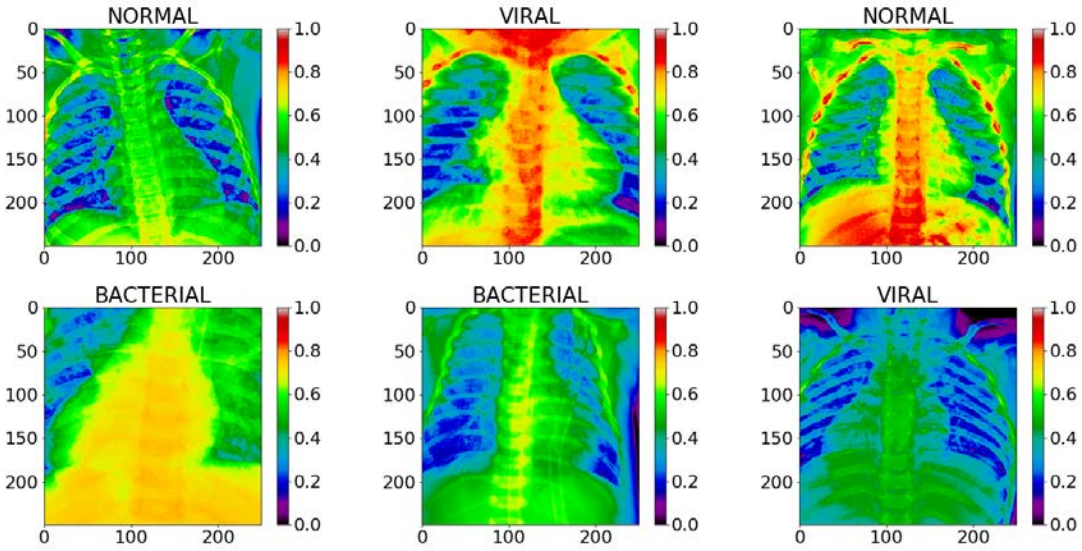

Fig. 5 The images shown in fig. 2 false-colored to emphasize differences in contrast

The normalized false-color images suggest that intensity values of pixels in the lung area are correlated with clinical findings. Focusing on the more transparent to X-rays rib-cage areas away from the spine and abdominal area, healthy images display a preponderance of pixel intensities in the range of ~ [0, 0.4], bacterial images in the range of [0.5-0.7] and viral images appear to have a diffuse, extended range throughout the intensity range. Moreover, viral pneumonia images appear with lung features that are spatially diffuse, whereas bacterial pneumonia images display features that are more localized.

Fig. 6 displays the histograms of a healthy image and a viral-pneumonia image. For the purposes of this plot, intensity counts (y-axis) were plotted as a function of 256 pixel "bins" presented in the normalized range (0,1) The histogram of the viral image shows a relatively uniform intensity distribution, slightly skewed toward the top half of the intensity distribution. The normal image also shows a somewhat uniform distribution but with less spectral weight around the middle of the intensity distribution than that of the pneumonia images.

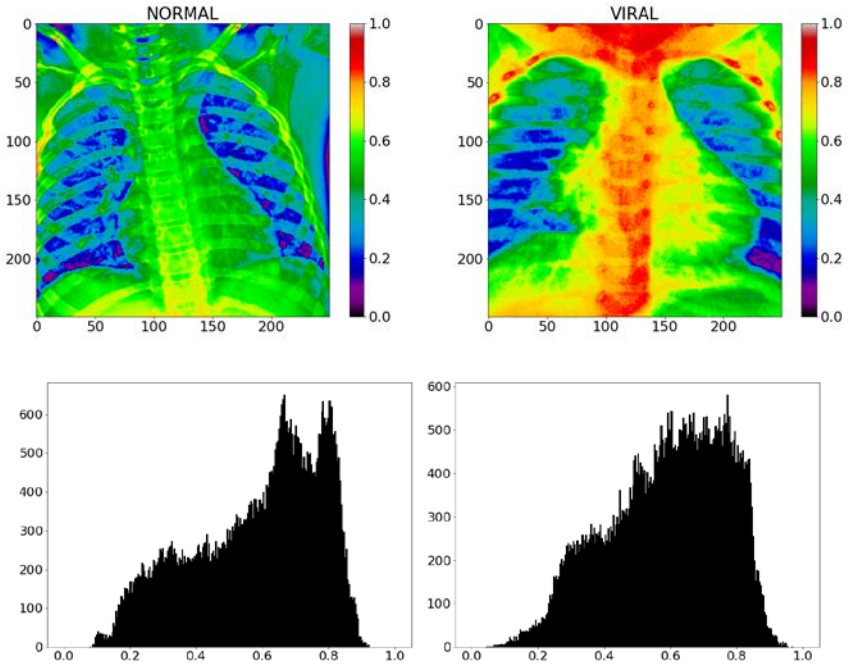

Fig. 6 Histograms of several representative images of the three classes

Given sufficient time, a quantitative analysis of the intensity distributions based on class and their usage in the convolutional neural network to localize and extract specific spatial features would be highly informative. Here I take the first step in calculating the mean intensity distribution for 40 randomly selected images in each class – normal, viral and bacterial (Fig. 7).

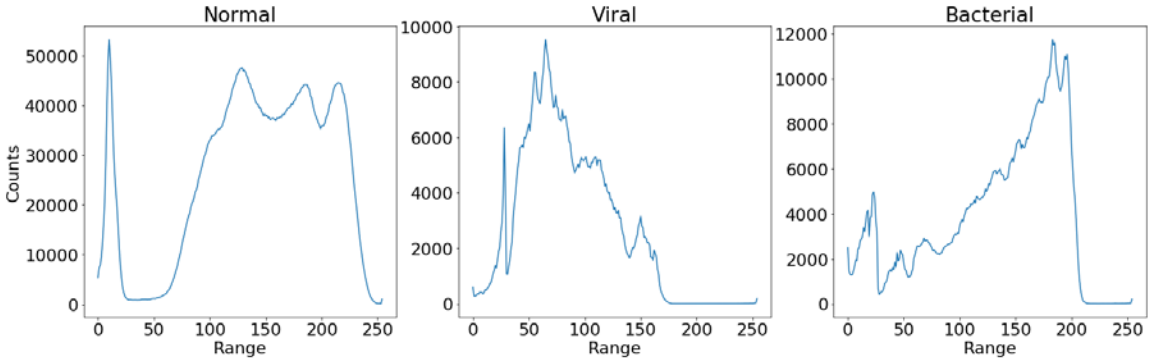

Fig. 7 Averaged histograms of 40 images per label, all grayscale.

Fig. 7 shows that there are significant differences in intensity distributions between classes. Viral pneumonia radiographs tend to be “darker” in showing much more intensity around pixel levels in the lower half of the distribution (where black is 0 and white is 255). Bacterial pneumonia images, in contrast, display intensity distribution

shifted to “brighter” pixel bins. With sufficient time, this intriguing finding could be extended and coupled with CNN image classification.

Reflection

In this study, a CNN algorithm utilizing DenseNet201 with flattening and global pooling or averaging layers was applied, followed by a sequence of one or more fully-connected layers with the final one often applying the LogSoftmax function. From the results obtained through the training dataset, it is clear that reliable classification of images can be obtained with this network.

However, further investigation of the problematic areas that contribute to a fast drop rate in the early stages of training could prove to beneficial in highlighting areas for improvement. Targeting inefficient areas of the algorithm and reducing any potential over fitting can lead to a higher accuracy score.

An extension of this work would require additional image preprocessing as well as careful feature extraction and mapping. Image preprocessing could include the removal of irrelevant areas of the image. Feature extraction could tell us which specific spatial features our model is able to recognize and which of those are important for accurate classification. Furthermore, a different neural network, such as the recently published demonstration of generative adversarial networks, which have been shown to be particularly suitable for recognizing a hierarchy of features (Radford, 2015).

I found this project illuminating in helping me realize that image classification with AI will require substantial further improvements to take on difficult tasks that currently require highly-trained professionals. This is even more important in the medical field. The current state of AI and machine learning, with the presence of numerous ready-made frameworks is a good start, but qualitative improvements will require of AI designers deep understanding of the mathematics, the algorithmic thinking and the physics of real-world sensory inputs.

Reference

F1
Score. (n.d.). Retrieved from https://www.slideshare.net/ThomasPloetz/bridging-the-gap-machine-learning-for-ubiquitous-computing-evaluation

Kermany. (2018). Identifying Medical Diagnoses and Treatable Diseases by Image-Based Deep Learning. *Cell, 172*(5), pp. 1122-1131.

Kermany, D., Zhang, K., & Goldbaum, M. (n.d.). Labeled Optical Coherence Tomography (OCT) and Chest X-Ray Images for Classification. Retrieved from https://data.mendeley.com/datasets/rscbjbr9sj/2

Powers, D. (2011). Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness & Correlation. *Journal of Machine Learning Technologies*, 37-63.

Radford, A. M. (2015). Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. *CoRR*. Retrieved from http://arxiv.org/abs/1511.06434

Rajpurkar, P., Irvin, J., Zhu, K., Yang, B., Lungren, M., & Ng, A. (2017). CheXNet: Radiologist-Level Pneumonia Detection on Chest X-Rays with Deep Learning. *arXiv:1711.05225v3*.

Raschka, S. (n.d.). Macro-averaging. Retrieved from https://sebastianraschka.com/faq/docs/multiclass-metric.html

Silverstein, J. (2016, September 27). Most of the World Doesn't Have Access to X-Rays. *The Atlantic*.

The World Health Organization . (2018). *Global Health Estimates 2016: Deaths by Cause, Age, Sex, by Country and by Region, 2000-2016.* Geneva.

Wamala, D. S. (2013). A meta-analysis of telemedicine success in Africa. .*Journal of Pathology Informatics, 4*(6). doi:http://doi.org/10.4103/2153-3539.112686

Wang, X., Peng, Y., L, L., Z, L., M, B., & RM, S. (2017). ChestX-ray: Hospital-scale Chest X-ray Database and Benchmarks on Weakly-Supervised Classification and Localization of Common Thorax Diseases. *IEEE CVPR*.

Classification accuracy
https://towardsdatascience.com/metrics-to-evaluate-your-machine-learning-algorithm-f10ba6e38234