# Pneumonia Detection on Chest X-Rays using Transfer Learning

William G Morgan
3/18/2019

## I. Definition

### Project Overview

Every year, in the United States alone, more than 1 million adults are hospitalized with pneumonia and roughly 5% or 50k die from the disease (CDC, 2017, https://www.cdc.gov/nchs/fastats/pneumonia.htm) . That's 15 to 1 per 100,000 population that end in death.

Currently, a chest X-Ray is one of the best available methods for diagnosing pneumonia (https://www.radiologyinfo.org/en/info.cfm?pg=pneumonia). However, identifying or detecting pneumonia in chest X-Ray is very challenging and relies on the expertise of a radiologists.

### Problem Statement

The workload of clinical radiologists continues to increase year over year, which places pressure on radiological services to become more efficient while maintaining and improving the quality of both patient care and radiology outcomes.

The purpose of this research is to develop a tool that can reasonably identify pneumonia by use of machine learning while taking advantage of Domain Adaptation thru Transfer learning.

We will demonstrate the performance of the models by statistical metrics like accuracy, recall, precision, and F1 score and comparing the metrics of these models to those of clinical radiologists.

The accuracy of most radiologists ranges from 88.5% to less than 85.2% depending on workloads (https://www.jacr.org/article/S1546-1440(10)00134-1/abstract).

Therefore, a machine learning algorithm that can accurately predict at 88.5% or better consistently, thus will out perform a human. There are documented studies as referencing the loss in accuracy as fatigue sets in (http://www.diagnosticimaging.com/practice-management/radiologists%E2%80%99-diagnostic-accuracy-drops-day-goes).

The below outline provides the workflow of the approach used in solving the identified problem:

1. Explore the Data
    a. Review training data
    b. Review validation data
    c. Review testing data
    d. Dimensionality of the data
    e. Statistical summary
2. Data Preprocessing/Augmentation
    a. Apply transformations
        i. Random scaling
        ii. Cropping
        iii. Rotation
    b. Normalize the means and standard deviations of the images to what the network expects
3. Build a new feed forward classifier
    a. ReLU
    b. Dropout
    c. Backpropagation
    d. Tune hyperparameters
    e. Features of pretrained model [Transfer Learning]
4. Model Tuning
5. Predict, summarize results

The expectations are that we will successfully be able to train a model using a convolutional neural network [CNN] with one of two architectures; Vgg13 or Alexnet, that will exceed the 77% metric as shown in the CheXnet model being used as our benchmark metric.

## Metrics

The evaluation metric(s) to be used are appropriate for this study given the context of the data, problem statement and the proposed solution. Thus, performance of the classification model will be evaluated using five statistical measures; accuracy, recall, precision, specificity and F1-score.

Accuracy is the number of correct predictions made as a ratio of all predictions made. Precision refers to the percentage of your results which are relevant.

Recall and specificity are statistical measures of performance commonly used in binary classification. Recall measures the proportion of actual positives that are correctly identified. Specificity measures the proportion of actual negatives that are correctly identified.

By using the number of true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN) we can formulate the following equations for the metric(s) used in this study:

Accuracy is defined as the ratio of the number of correctly classified cases and is equal to the sum of TP and TN divided by the total number of cases (TP+TN+FN+FP).

$$\text{Accuracy} = \frac{TP+TN}{(TP+TN+FN+FP)}$$

Precision is defined as the number of true positives (TP) over the number of true positives (TP) plus the number of false positives (FP).

$$\text{Precision} = \frac{TP}{TP+FP}$$

Recall is defined as the number of true positives (TP) over the number of true positives plus the number of false negatives (FN).

$$\text{Recall} = \frac{TP}{TP+FN}$$

Specificity is the rate of correctly classified negative and is equal to the ratio of TN to the sum of TN + FP.

$$\text{Specificity} = \frac{TN}{TN+FP}$$

F1-score is the weighted average of precision and recall. Therefore, this score takes both false positives and false negatives into account. F1-score is usually more useful than accuracy, especially if you have an uneven class distribution. **F1-score will be our benchmark metric for overall comparison**.

F1-score = (2 x (Precision x Recall)) / (Precision + Recall)

## II. Analysis

### Datasets and Inputs

The dataset(s) and/or input(s) used in this project/research where obtained through Kaggle (https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia/home).

The dataset was organized into 3 folders (train, valid, test) and contains subfolders (Normal, Pneumonia) which contain X-Ray images for each respective category. There is a total of 5863 images (JPEG).

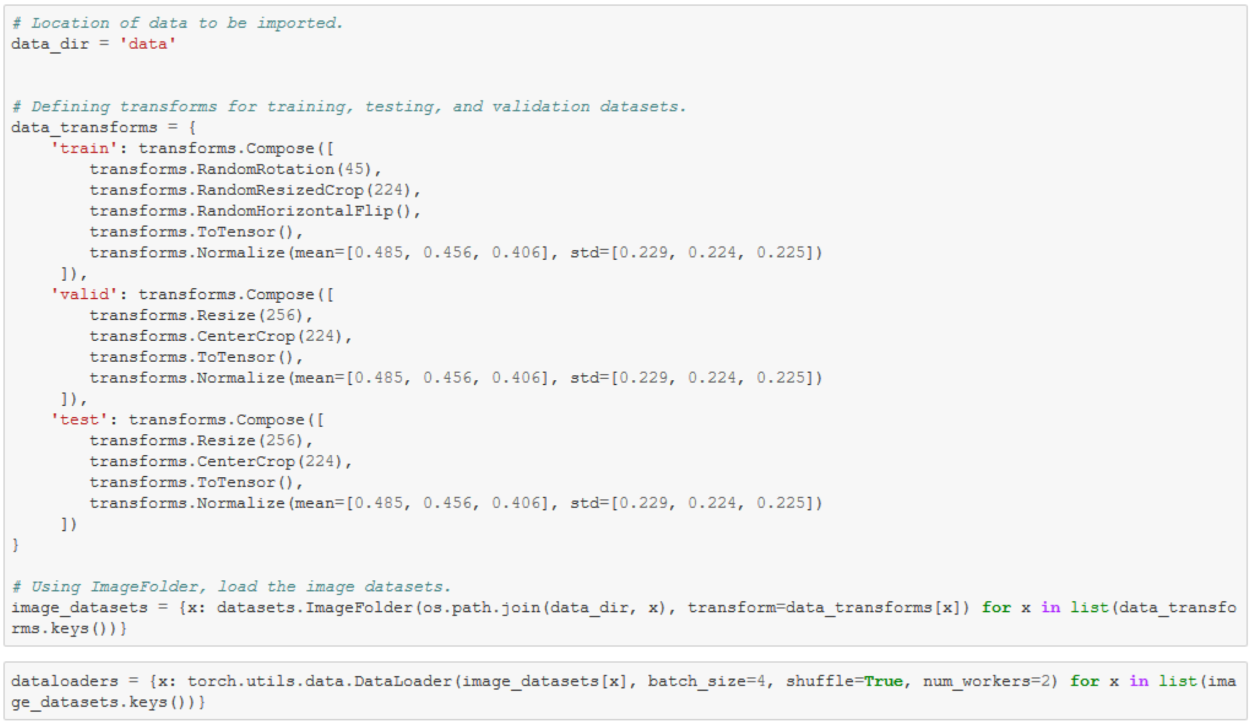
The chest X-Ray images are anterior-posterior in orientation and represent cohorts of pediatric patients of one to 5 years old from Guangzhou Women and Children's Medical Center, Guangzhou. All chest X-Ray images were performed as part of the patients' routine clinical care.

All chest radiographs were initially screened, and any low quality or unreadable scans were removed. Then, two expert physicians reviewed the resulting set of images prior to being cleared for training our AI system.

### Data Preprocessing

In figure 1, image data was loaded into each of the respective datasets; training, validation and testing into data transformation objects [pytorch]. This was done in order to resize images, crop images, perform random rotations [helps the network to generalize better] and normalize the means and standard deviations of the images to what the network expects.

For the means, [0.485, 0.456, 0.406] and for the standard deviations [0.229, 0.224, 0.225], calculated from the ImageNet images. This in effect causes each color channel to be centered at 0 and range from -1 to 1.



```python
# Location of data to be imported.
data_dir = 'data'


# Defining transforms for training, testing, and validation datasets.
data_transforms = {
    'train': transforms.Compose([
        transforms.RandomRotation(45),
        transforms.RandomResizedCrop(224),
        transforms.RandomHorizontalFlip(),
        transforms.ToTensor(),
        transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
     ]),
    'valid': transforms.Compose([
        transforms.Resize(256),
        transforms.CenterCrop(224),
        transforms.ToTensor(),
        transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
     ]),
    'test': transforms.Compose([
        transforms.Resize(256),
        transforms.CenterCrop(224),
        transforms.ToTensor(),
        transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
     ])
}

# Using ImageFolder, load the image datasets.
image_datasets = {x: datasets.ImageFolder(os.path.join(data_dir, x), transform=data_transforms[x]) for x in list(data_transfo
rms.keys())}
```

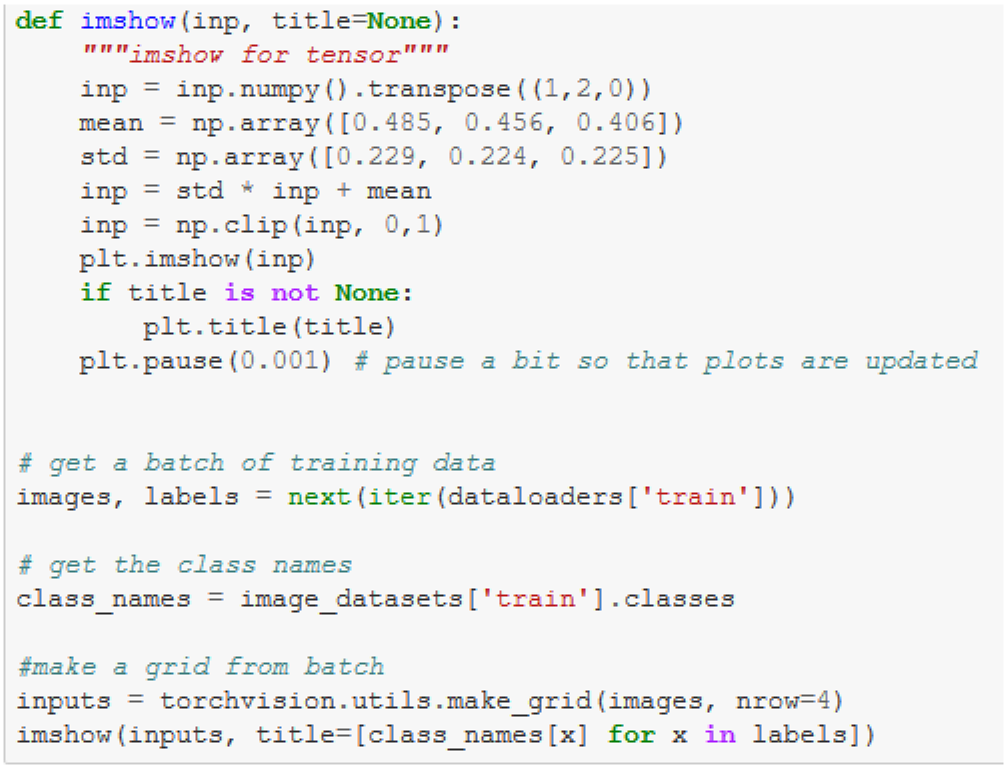
```python
dataloaders = {x: torch.utils.data.DataLoader(image_datasets[x], batch_size=4, shuffle=True, num_workers=2) for x in list(ima
ge_datasets.keys())}
```

**Figure 1.**

## Exploratory Visualization

We explore the data by first ensuring that the dataloaders for training, validation and testing datasets have loaded properly. Next, we load a batch of image data [training only] and create a grid with the class names and associated x-ray images.

Notice that the images may appear slightly rotated. This is due to the random rotation effect occurring in our training sample, which helps the network to generalize better (see Figure 2).



```
def imshow(inp, title=None):
    """imshow for tensor"""
    inp = inp.numpy().transpose((1,2,0))
    mean = np.array([0.485, 0.456, 0.406])
    std = np.array([0.229, 0.224, 0.225])
    inp = std * inp + mean
    inp = np.clip(inp, 0,1)
    plt.imshow(inp)
    if title is not None:
        plt.title(title)
    plt.pause(0.001) # pause a bit so that plots are updated


# get a batch of training data
images, labels = next(iter(dataloaders['train']))

# get the class names
class_names = image_datasets['train'].classes

#make a grid from batch
inputs = torchvision.utils.make_grid(images, nrow=4)
imshow(inputs, title=[class_names[x] for x in labels])
```

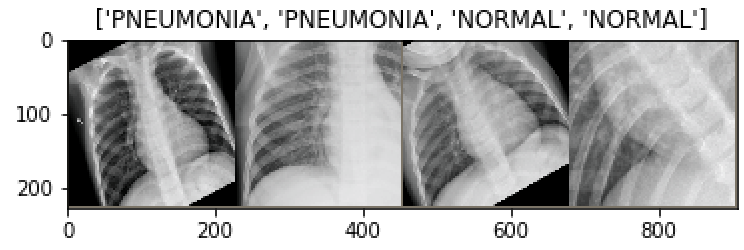


**Figure 2.**

Now we load each of the dataloaders (training, validation, testing) with corresponding class names, into bar charts to visually identify the number of images for each class being loaded into each dataloader [i.e., dataset, see Figure 3].

In the training dataset, there are a total of 5223 x-ray images comprising of 3875 x-ray images with known pneumonia and 1348 x-ray images with that are known to be normal.

Our validation dataset is comprised of 199 known x-ray images of pneumonia and 121 x-ray images known to be normal.

The testing dataset is comprised of 199 known x-ray images of pneumonia and 121 x-ray images known to be normal.

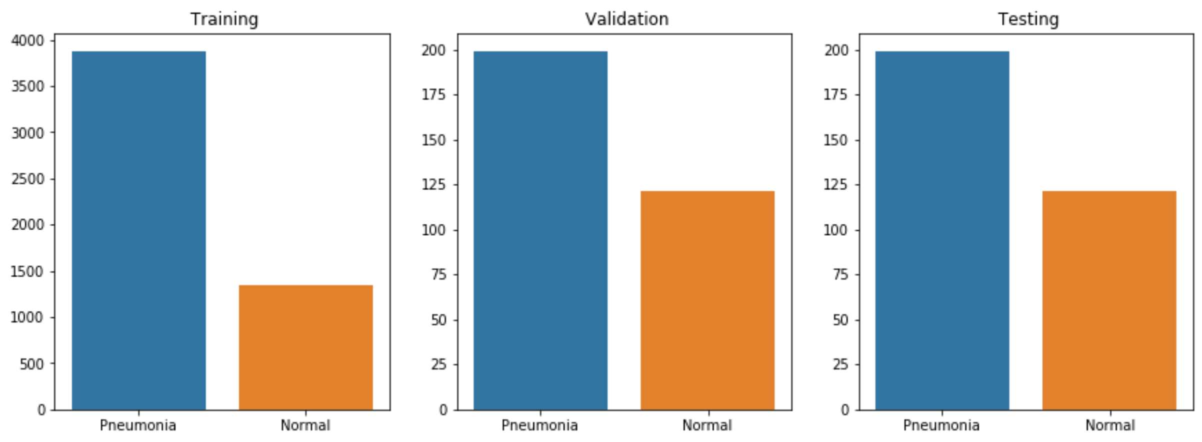


**Figure 3.**

## Algorithms and Techniques

Convolutional Neural Networks (CNN) are a special kind of multi-layered neural network. These networks are designed to recognize patterns directly from pixel images with minimal processing.

In our research we have taken advantage of CNNs in order to identify abnormalities within x-ray images. Our first approach involved looking at a pretrained model using a Vgg13 network. This CNN consists of 13 convolutional layers and is very appealing because of its uniform architecture (see Figure 4).

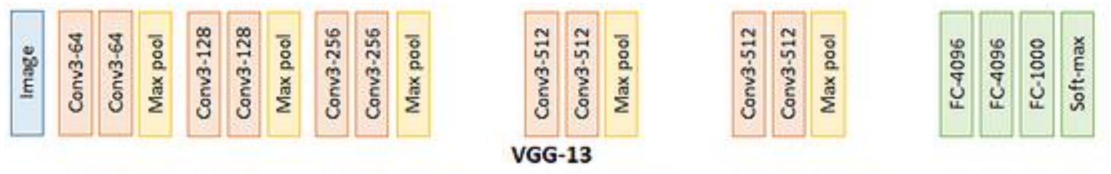


Figure 4.

One of the down sides to Vgg13 is that it is comprised of 133 million parameters, which can be quiet challenging to handle.

Our second approach was to exploit a pretrained AlexNet model. This CNN consists of 5 convolutional layers and 3 fully connected layers, total 8 layers in all. For a simple CNN this architecture has been trained on millions of ordinary images and provides rich features to be taken advantage of during transfer learning. In figure 5, an example of this architecture is shown.

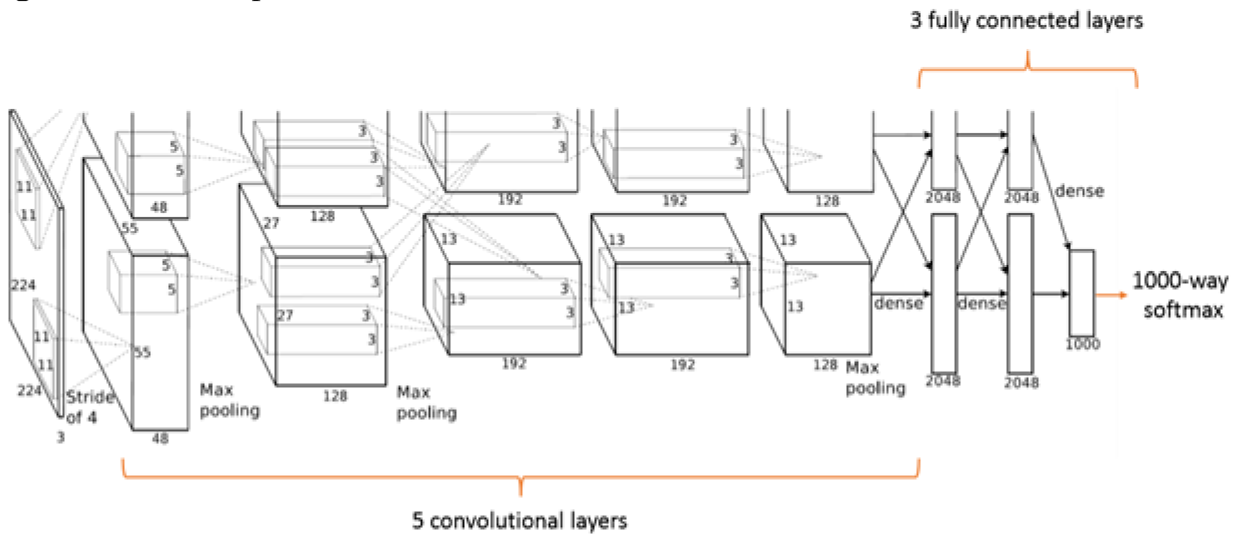


Figure 5.

In our training approach, a pre-trained CNN model [AlexNet] is loaded, then the parameters are frozen, and the last fully connected layer is removed.

Next the number of filters in the bottleneck layer are determined from features that were in the layer just removed.

The architecture of the model is then extended with new layers using;

```
features.extend([
        nn.Dropout(),
        nn.Linear(num_filters, hidden_units),
        nn.ReLU(True),
        nn.Dropout(),
        nn.Linear(hidden_units, hidden_units),
        nn.ReLU(True),
        nn.Linear(hidden_units, num_labels),

 ])
```

Also, the 1000 Way Softmax layer is reduced to a two [2] Way Softmax layer. For our purposes, we are only interested in predicting whether the x-ray image contains pneumonia or is normal.

Now we are ready to train our new architecture [model], while taking advantage of Transfer Learning. Transfer Learning comes from the weights that were acquired during the original model's training on ImageNet images, which we can now take advantage of for our model.

During a training cycle, a process takes place by where the extended model is loaded, the labels are loaded from our classes and image data is provided by our dataloaders.

Next our optimizer is defined using Stochastic Gradient Decent and Cross Entropy for our loss function. We set our variables for monitoring time, best model weights [best_model_wts] and best accuracy [best_acc] along with a few other metrics.

During training, each epoch has a training and validation phase, where we iterate over the data, track accuracy, loss for each epoch as well as optimize during the training phase only.

At the end of each epoch, if the current best model weights are better than the previous run, the model is saved, and another iteration of training is performed.

In our prediction approach we first load the saved model with the best weights, and then using 'test' data; the images and labels are loaded along with the images being passed to our model.

The true and predicted values are performed and then the below piece of logic is used to determine the count of True Positives, False Positives, True Negatives and False Negatives.

```
if labels[i]==predicted[i]==1:
        TP += 1
if predicted[i]==1 and labels[i]!=predicted[i]:
        FP += 1
if labels[i]==predicted[i]==0:
        TN += 1
if predicted[i]==0 and labels[i]!=predicted[i]:
        FN += 1
```

The Accuracy, Precision, Recall, Specificity and F1-score are then calculated based on the TP, FP, TN, FN counts.

**Benchmark**

The referenced dataset has been used in many different studies involving image recognition and one of the deep convolutional neural networks that we will be using as a benchmark measure of performance will be based on the results from the CheXnet model (https://arxiv.org/pdf/1711.05225.pdf). Using F1-score, we are attempting to outperform 77%.

## III. Methodology

The dataset will be split into three parts, training, validation, testing and contains subfolders (Normal, Pneumonia) which contain X-Ray images for each respective category. There is a total of 5863 images (JPEG).

For training, we will apply transformations such as random scaling, cropping, and flipping. This will help the network generalize leading to better performance.

We will also make sure the input data is resized to 256x256 pixels as required by the pre-trained networks.

The validation and testing sets are used to measure the model's performance on data it hasn't seen yet. Because of this, there won’t be any scaling or rotation transformations, but we will resize, then crop the images to the appropriate size.

The pre-trained network that we will use was trained on the ImageNet dataset where each color channel was normalized separately. For each of the datasets, we will normalize the means and standard deviations of the images to what the network expects. For the means, it's [0.485, 0.456, 0.406] and for the standard deviations [0.229, 0.224, 0.225], calculated from the ImageNet images. This in effect causes each color channel to be centered at 0 and range from -1 to 1.

On our first approach, the pre-trained network [VGG13] was loaded and the model parameters were frozen. Then the last fully connected layer was removed and then the model was extended using the existing architecture with new layers. This extension was done using ReLU, Dropout and convolutional layers.

Next, we trained the new model with the x-ray image data that had been preprocessed and stored in the dataloaders. Over a period of 30 epochs using 1 GPU, the necessary training was performed over 2 hours. A momentum of 0.9 and learning rate of 0.001 was used during the training session.

As the model trained over many epochs the loss and accuracy metrics were monitored to ensure that the model was learning and that our model was converging.

At the end of training the model was saved as, model_vgg13_xrays.pth, then reloaded into a test object, testmodelload, for use in validating the performance using the metrics derived from TP, TN, FP, FN.

In our second approach, the pre-trained network [AlexNet] was loaded and the model parameters were frozen. Then the last fully connected layer was removed and then the model was extended using the existing architecture with new layers. This extension was done using ReLU, Dropout and convolutional layers.

Next, we trained the new model with the x-ray image data that had been preprocessed and stored in the dataloaders. Over a period of 30 epochs using 1 GPU, the necessary training was performed over 2 hours. A momentum of 0.9 and learning rate of 0.001 was used during the training session.

As the model trained over many epochs the loss and accuracy metrics were monitored to ensure that the model was learning and that the model was converging.

At the end of training the model was saved as, model_alexnet_xrays.pth, then reloaded into a test object, testmodelload, for use in validating the performance using the metrics derived from TP, TN, FP, FN.

In both approaches the F1-score along with other metrics (accuracy, precision, recall, specificity) were recorded. F1-score is the weighted average of precision and recall. Therefore, this score takes both false positives and false negatives into account. F1-score is usually more useful than accuracy, especially if you have an uneven class distribution. F1-score is the benchmark metric for overall comparison used in this study.

## IV. Results

### Model Evaluation and Validation

The final model selection with metrics and our benchmark metric is listed in Table 1.

Using transfer learning from a Vgg13 pretrained model to create the new model, trained on the x-ray image data, over 30 epochs had an accuracy of 81.25% with a precision of 85.64%. The corresponding recall for this model was 83.91% with a specificity of 76.85% and an F1-score of 84.77%. Our benchmark metric which is described in CheXnet model (https://arxiv.org/pdf/1711.05225.pdf), is 76.8% [pneumonia] or roughly 77%.

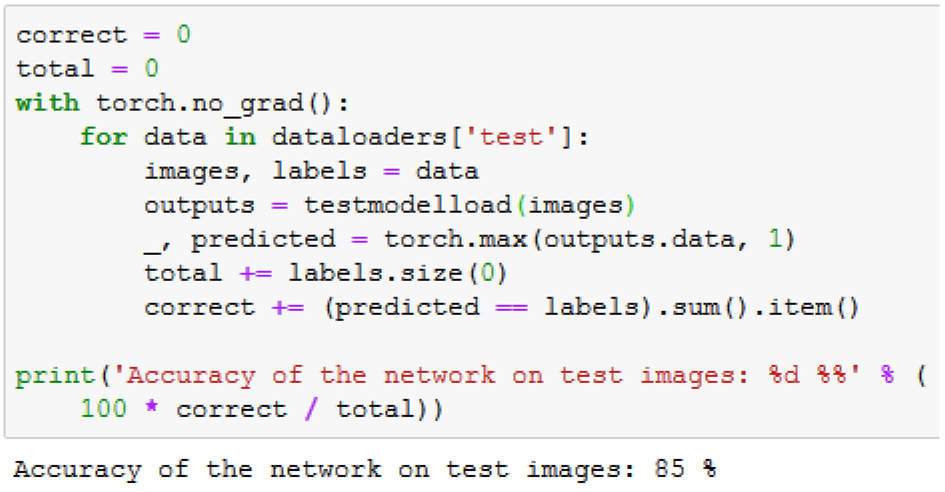
The new model outperformed the CheXnet model but the recall for this new model would be considered to low for any practical use in a clinic support system for radiologists.

In the second approach, using transfer learning from an AlexNet pretrained model, a new model was created and trained on the x-ray image data, over 30 epochs. The accuracy for this model as shown in Table 1 is 85.31% with a precision of 86.19% and the recall of 90.95%. The F1-score stands out at 88.51% because as sited in JACR (https://www.jacr.org/article/S1546-1440(10)00134-1/abstract), prior to the setting in of fatigue, most radiologists are roughly 88.5%. Also, the model’s F1-score outperforms the 77% benchmark.

| **Pre-Trained Model** | **Our model** | **Accuracy** | **Precision** | **Recall** | **Specificity** | **F1-Score** | **Metric to beat** |
|---|---|---|---|---|---|---|---|
| Vgg13 | Model_vgg13_xrays.pth | 81.25 % | 85.64 % | 83.91 % | 76.85 % | 84.77 % | 77% |
| AlexNet | Model_alexnet_xrays.pth | 85.31 % | 86.19 % | 90.95 % | 76.03 % | 88.51 % | 77% |

Table 1.

Now we investigate whether the model generalizes well to unseen data and essentially determine if the results of the model can be trusted. Looking at figure 6, we load a set of ‘unseen’ data to our model and then determine the accuracy of the network based on those test images. The accuracy of the network is 85%. This seems better than chance and looks like the network did learn.



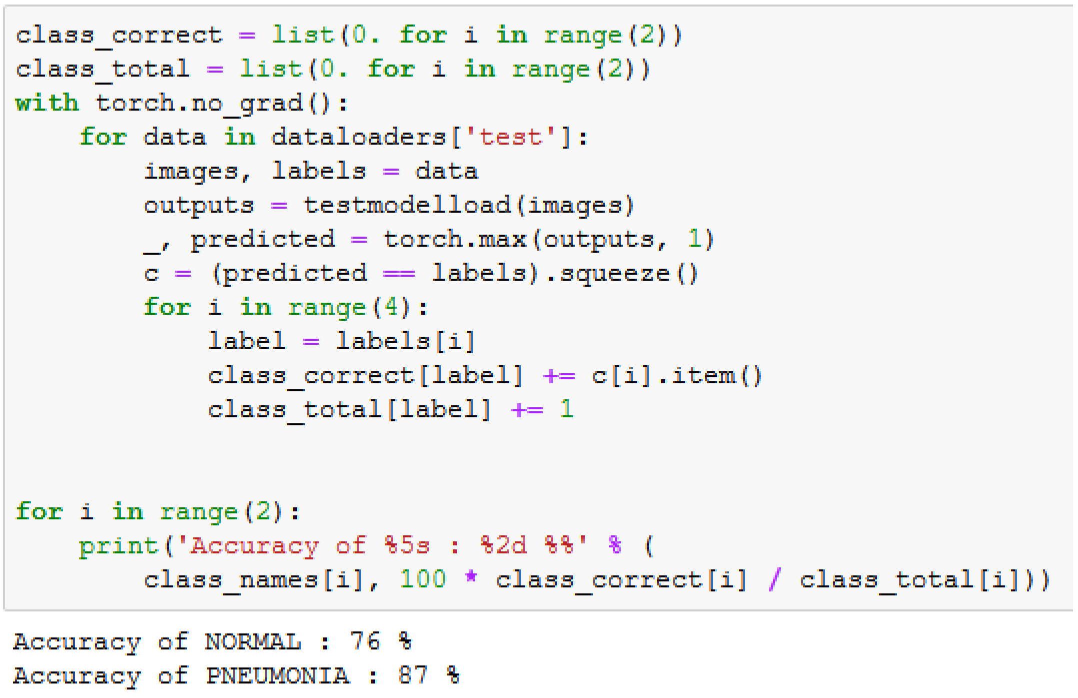
```
correct = 0
total = 0
with torch.no_grad():
    for data in dataloaders['test']:
        images, labels = data
        outputs = testmodelload(images)
        _, predicted = torch.max(outputs.data, 1)
        total += labels.size(0)
        correct += (predicted == labels).sum().item()

print('Accuracy of the network on test images: %d %%' % (
    100 * correct / total))
```

```
Accuracy of the network on test images: 85 %
```

Figure 6.

We go a step further and see how the individual classes performed. In figure 7, the unseen test data is loaded into our model and then the accuracy of each corresponding class is shown. The two classes: NORMAL at 76% and PNEUMONIA at 87% seems to fully support that the model is robust, generalizes well and can be trusted.



```
class_correct = list(0. for i in range(2))
class_total = list(0. for i in range(2))
with torch.no_grad():
    for data in dataloaders['test']:
        images, labels = data
        outputs = testmodelload(images)
        _, predicted = torch.max(outputs, 1)
        c = (predicted == labels).squeeze()
        for i in range(4):
            label = labels[i]
            class_correct[label] += c[i].item()
            class_total[label] += 1


for i in range(2):
    print('Accuracy of %5s : %2d %%' % (
        class_names[i], 100 * class_correct[i] / class_total[i]))
```

```
Accuracy of NORMAL : 76 %
Accuracy of PNEUMONIA : 87 %
```

Figure 7

### Justification

Clearly, the model using transfer learning, based on AlexNet[pretrained], and trained on the x-ray image data outperforms the benchmark set, as well as, provides a model that does not suffer from fatigue, which can be used as a tool for clinical support systems for radiologists.

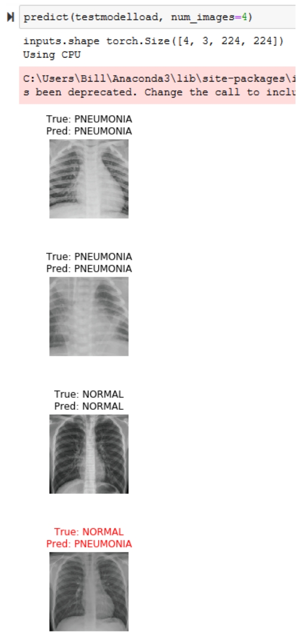
## V. Conclusion

In figure 8, the predicted and true labels are shown, which could be incorporated into a possible clinical support system tool.

Use as a tool, once a batch of images is processed, the true labels are shown as well as the predicted labels are presented along with a snapshot of the x-ray images associated.

This implementation as shown helps to add credibility to the overall performance of the tool which allows for visual verification.

The tool would also provide prediction with associated x-ray image for actual real-time use.



```
predict(testmodelload, num_images=4)

inputs.shape torch.Size([4, 3, 224, 224])
Using CPU

C:\Users\Bill\Anaconda3\lib\site-packages\i
s been deprecated. Change the call to inclu
```

Figure 8.

In this research we identified a potential area of use for machine learning, in which, the workload of Radiologists could be improved using a tool that can accurately predict Pneumonia from standard anterior-posterior x-rays.

The approach made use of a convolution neural network specifically AlexNet architecture, where the results of the F1-score outperformed our benchmark metric as defined in the CheXnet model (https://arxiv.org/pdf/1711.05225.pdf).

A challenge that I faced was removing the last fully connected layer and then adding back in the new layer in order to take advantage of transfer learning.

Further research should be done in the investigation using ResNet models, which allow for a skipping of a convolution layer, which might help in improving performance due to the negative artifacts are not transferred to the next convolutional layer, thus possible improve during training.

## References

1. Tataru, C., Shenoyas, A., Yi, D., Ma, A., Deep Learning for abnormality detection in Chest X-Ray images (2017), http://cs231n.stanford.edu/reports/2017/pdfs/527.pdf .

2. CheXNet: Radiologist-Level Pneumonia Detection on Chest X-Rays with Deep Learning (2017) https://arxiv.org/pdf/1711.05225.pdf .

3. Akcay, S., Kundegorski1, M., Devereux, M., Breckon, T., Transfer Learning Using Convolutional Neural Networks for Object Classification Within X-Ray Baggage Security Imagery (2016), IEEE International Conference on Image Processing (ICIP), (http://breckon.eu/toby/publications/papers/akcay16transfer.pdf).

4. Smith, L., Cyclical Learning Rates for Training Neural Networks (2017), US Naval Research Laboratory, (https://arxiv.org/pdf/1506.01186.pdf).