Introduction

In the realm of diagnostic healthcare, chest X-rays (CXR) play an instrumental role. They offer non-invasive and detailed insights into an individual's thoracic health, helping detect a wide spectrum of conditions ranging from benign abnormalities to critical illnesses like pneumonia. However, the full potential of CXR diagnostics is yet to be tapped, particularly in resource-constrained regions. The interpretation of these images remains an intricate task, burdened by several challenges and marked inaccuracies.

One of the often-overlooked issues during X-ray imaging is the presence of foreign objects, such as jewelry or coins, that can hinder accurate diagnosis. These objects, though seemingly insignificant, can obstruct critical anatomical structures in the imaging field, leading to suboptimal images, possibly resulting in misdiagnosis or a missed diagnosis. This makes a foreign **object detection** an important task. CXR, in particular, serves as a vital tool for diagnosing pneumonia, an infection that inflames the air sacs in one or both lungs, leading to a range of symptoms like cough with phlegm or pus, fever, chills, and difficulty breathing. While viral pneumonia tends to evolve more slowly and is often less severe, bacterial pneumonia can develop quickly and lead to serious or life-threatening illness. This means that each type necessitates a different treatment protocol. Yet, distinguishing between these types through chest X-rays is challenging, given their subtle differences, making it crucial to develop an efficient **classification** mechanism as well.

Our project introduces a comprehensive deep learning solution that targets these two critical aspects of CXR image analysis. Firstly, we employ advanced object detection models - Region-based Convolutional Neural Networks (R-CNN) and the high-speed You Only Look Once (YOLO) model - to reliably identify and alert about foreign objects in the image. This significantly reduces the risk of diagnostic inaccuracies due to image obstruction. Secondly, our solution harnesses the power of Convolutional Neural Networks (CNN) and the renowned Visual Geometry Group (VGG) model through transfer learning to differentiate between viral pneumonia, bacterial pneumonia, and normal lungs based on CXR images. This classification is vital for accurate diagnosis and appropriate treatment administration, leading to better patient outcomes. Our deep learning solution offers significant potential to transform healthcare provision, particularly in resource-limited settings. By enabling efficient foreign object detection and pneumonia type classification, we aim to accelerate diagnostic processes, reduce the pressure on medical resources, and most importantly, ensure that patients receive the right treatment promptly.

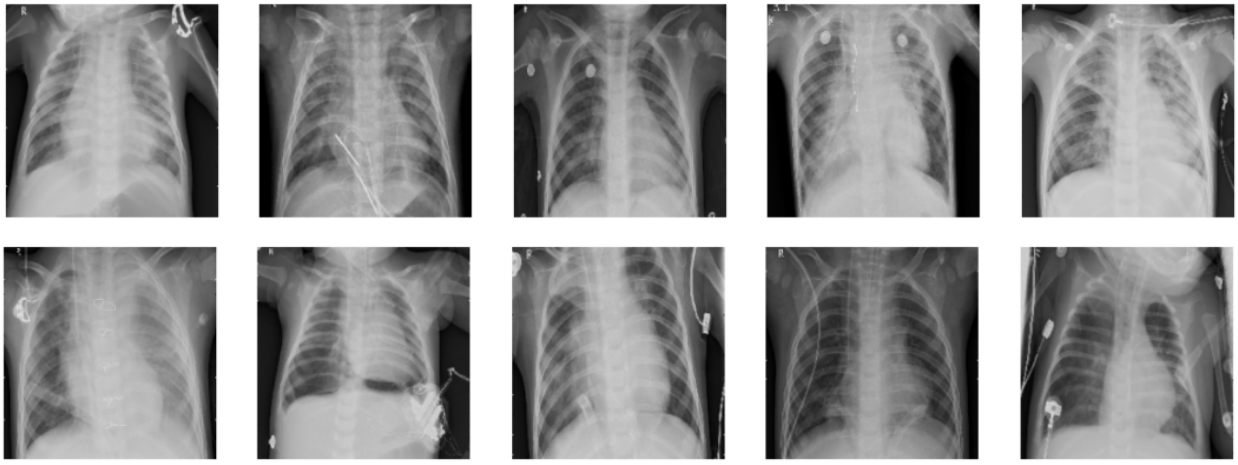
Through a series of extensive experiments, we've designed a final solution that employs custom bounding box regression for object detection, complemented by the VGG16 transfer learning model for multi-classification tasks. Our top-performing model for object detection achieves an average IoU (IoU) score of 0.11. Meanwhile, the best model for multi-classification delivers robust results, boasting a test accuracy of 0.78, precision of 0.79, and recall of 0.77, which

collectively signify the efficacy of our approach. We discuss the specifics of our approach and its broader implications in subsequent sections. In essence, our project epitomizes how deep learning can revolutionize healthcare, particularly chest X-ray diagnostics, contributing to our pursuit of universal healthcare accessibility.

Exploratory Data Analysis

Object Detection

Our motivation for object detection actually arose from our classification dataset[1]. We noticed some foreign objects from our classification dataset. This can be a problem particularly in regions where technicians who took these X-rays are miles away from the physicians who inspect them, where the cost of communications are high.



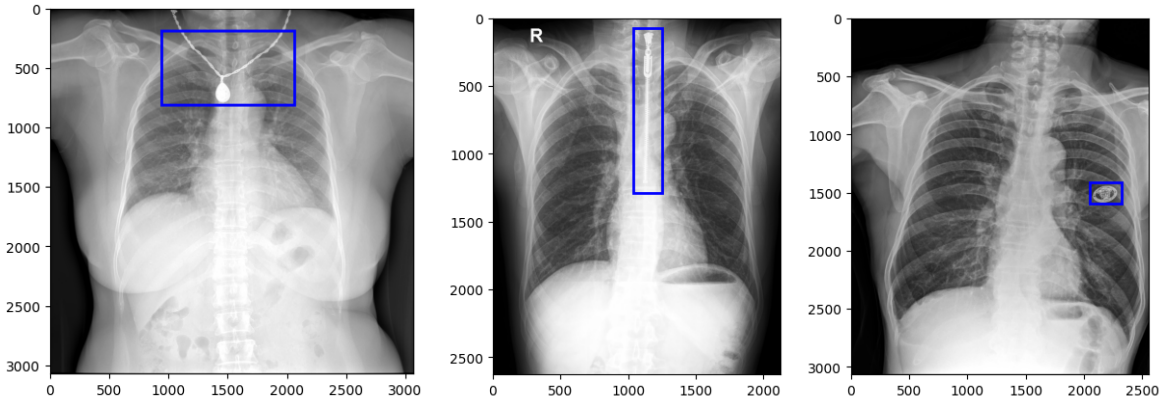
To make sure we are minimizing the bias in terms of ethnicity, we sourced a set of CXRs[2] that were filmed and collected from about 300 township hospitals in China. The original dataset contains 5,000 frontal CXR images with foreign objects presented and 5,000 without them. 12 medically-trained radiologists with 1 to 3 years of experience annotated all the images. The annotation is in the following format - where we have the file name, the type (0 for rectangle, 1 for circle, and 2 for polygon) and the coordinates (x1, y1, x2, y2, and for polygon it's a sequence)

```
image_path,annotation
/path/#####.jpg,ANNO_TYPE_IDX x1 y1 x2 y2;ANNO_TYPE_IDX x1 y1 x2 y2 ... xn yn;...
/path/#####.jpg,
/path/#####.jpg,ANNO_TYPE_IDX x1 y1 x2 y2
...
```

For simplicity and computational cost, we did:

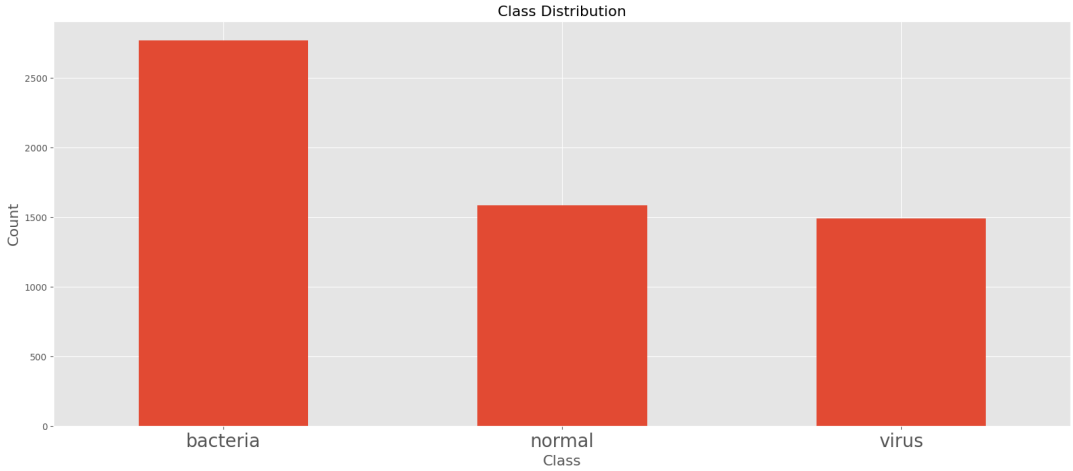
- 10% sub-sampling of the training data (4,000 with objects and 4,000 without objects), which yielded a balance training set (400 with objects and 400 without objects)
- Filter for objects only, since we are already showing classification in the next part
- Filter for rectangle only, dropping 3 images of polygon for simplicity

That left us with 397 images with rectangles annotated. Foreign objects were annotated with bounding boxes. Sample frontal CXR image with foreign objects (necklace, zipper and pin) annotated looks like this:

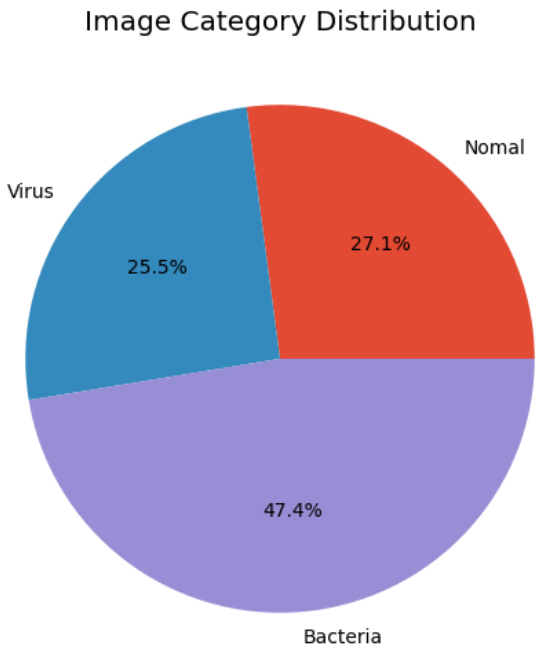


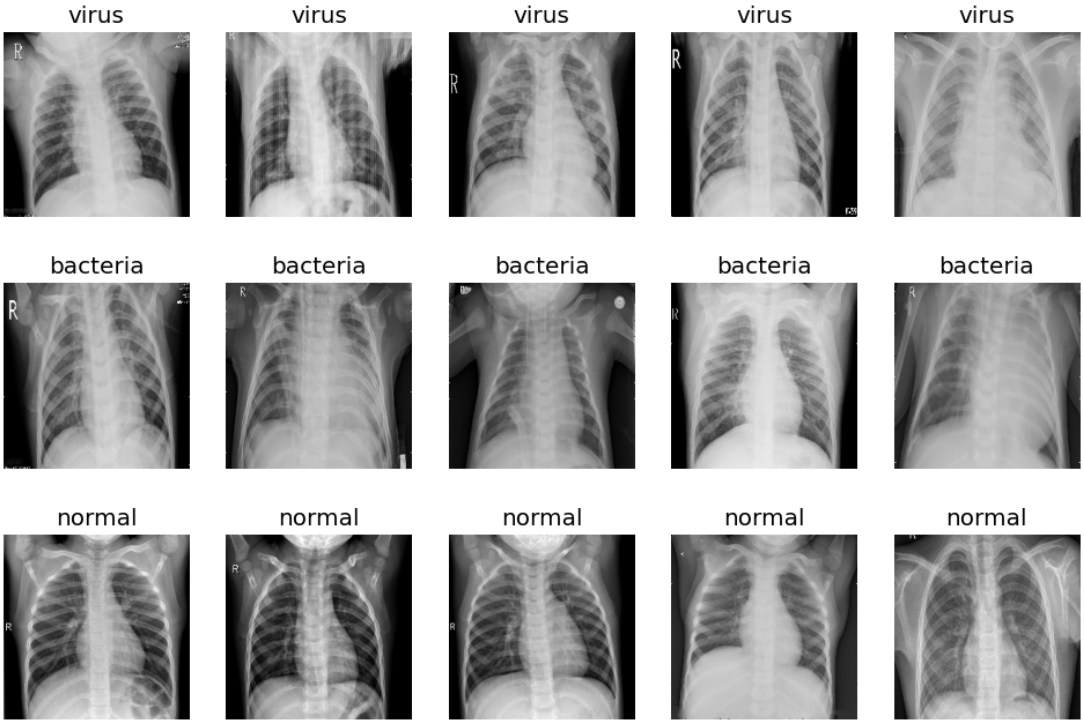
Classification

For the classification problem, three different classes are distributed as follows: **normal** (1,583 images, 27.1%), **virus** (1,490 images, 25.5%), and **bacteria** (2,769 images, 47.4%). No clear class imbalance is present in the dataset. Sample images from all three classes are shown below.



Class Distribution





The nature of X-ray images requires more tailored image augmentation choices. For both problems, after careful examination of image-level differences, image augmentation was designed to mainly cover variation and differences in image orientation (horizontal vs. vertical), centering, size, and saturation level. To be exact, width shift, height shift, channel shift, and zoom ranges are being considered, along with a custom defined orthogonal rotation of +/- 90 degrees. All images are also being adjusted to 180*180.

Model Training and Evaluation for

Object Detection

Challenger: YOLO Transfer Learning

YOLO (You Only Look Once) is a popular object detection algorithm in computer vision. It's known for its real-time object detection capabilities and has been widely used for various applications, including autonomous driving, surveillance systems, and image analysis.

YOLO approaches object detection as a regression problem, where it predicts bounding boxes and class probabilities directly from the input image. For us, **the cost of false positives** (i.e., identifying a foreign object even when there's none) **is trivial**. Thus, we are approaching this problem as purely a detection problem (i.e., bounding box regression), which is also why we only fed the model images with objects only.

Knowing what we will use YOLO for also narrows down the version choice - for this task, we picked YOLOv5[3]. It's the first in the YOLO family that is easy to use and deploy on the edge device. Even though YOLOv8 proves to be faster and more accurate, the architectural overhead is mostly devoted to real-time prediction, which is not the major concern for the problem at hand. In other words, YOLO is already somewhat of an overkill, we don't need a bigger gun.

During the data pre-processing step, we did a 60/15/25 train/val/test split. We used one of the built-in `no-augmentation.yaml` as our training configuration. As the name suggests, it comes from no image augmentation. YOLOv5 was trained on the COCO[4] dataset, with colored photos and very clear labels and boundaries from the real-world. CXR images, on the other hand, by passing ionizing radiation through the body so that it can collapse relative positions onto a 2D representation, not to mention they are black and white. Lack of texture and color information, combined with complex backgrounds and overlapping structures, added a level of complexity to the problem. As a result, we choose to preserve image integrity.

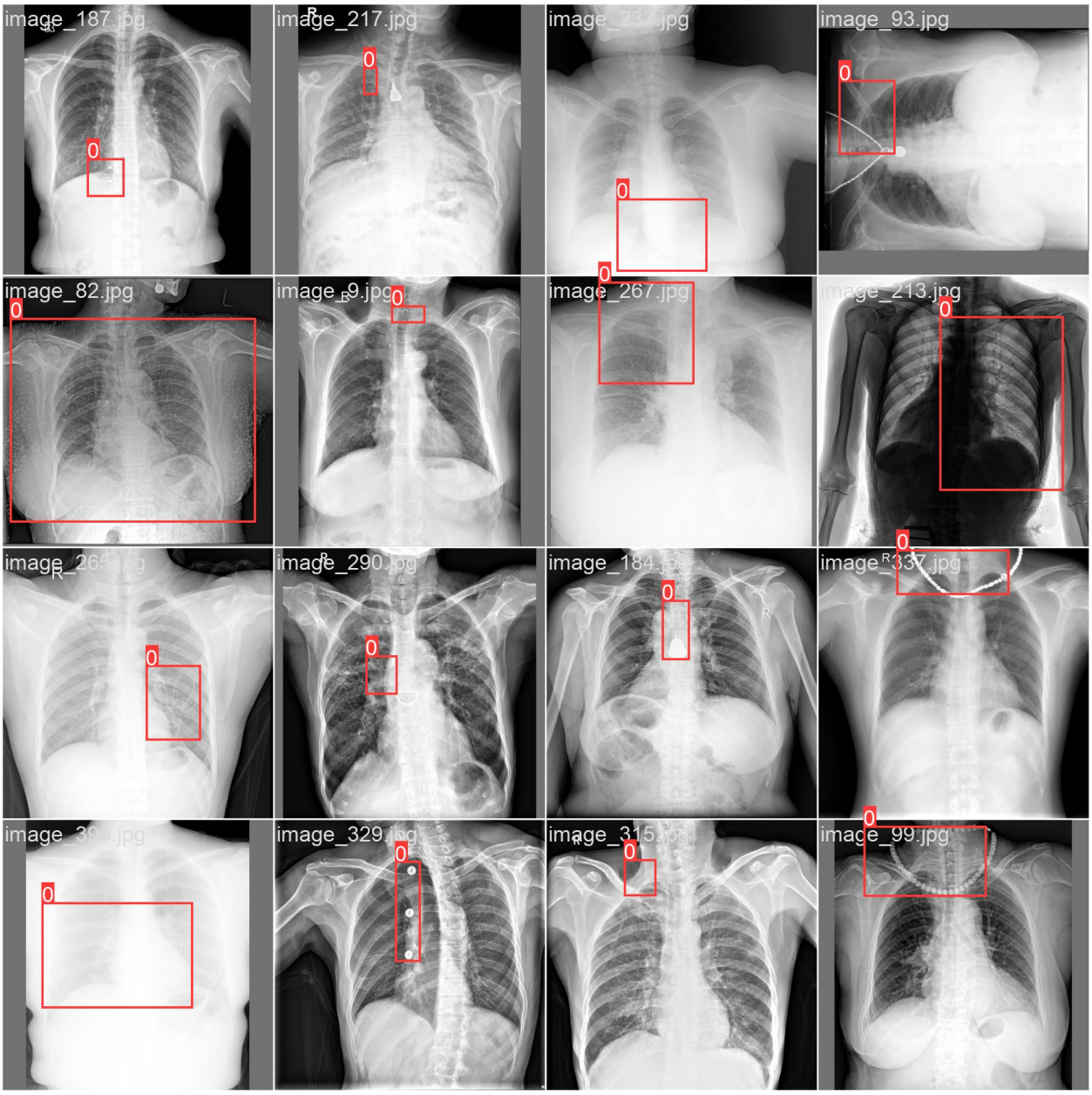
We started our transfer learning with YOLOv5s, and gradually move our way to YOLOv5x.The underlying architecture is the same - the only difference is combination of `depth_multiple` (used to increase the number of layers in the network, 0.33 in 5s to 1.33 in 5x) and `width_multiple` (used to increase the number of channels in each layer, 0.50 in 5s to 1.25 in 5x).

Model	size (pixels)	mAP^{val} 50-95	mAP^{val} 50	Speed CPU b1 (ms)	Speed V100 b1 (ms)
YOLOv5n	640	28.0	45.7	**45**	**6.3**
YOLOv5s	640	37.4	56.8	98	6.4
YOLOv5m	640	45.4	64.1	224	8.2
YOLOv5l	640	49.0	67.3	430	10.1
YOLOv5x	640	50.7	68.9	766	12.1

YOLO v5 consists of [6]

- `Backbone`: `New CSP-Darknet53`
- `Neck`: `SPPF, New CSP-PAN`
- `Head`: `YOLOv3 Head`

We started by freezing all layers, except for the final convolution layers (--freeze 24) to freezing the head, and finally to freezing the backbone (--freeze 10, containing all modules with 'model.0.' - 'model.9.' in their names). Unfortunately, no satisfactory outcome was achieved. So we ended up also fine-tuning the weights of the pre-trained network. Here, we trained with a batch size of 32 and 32 epochs using all 4 NVIDIA RTX A6000 GPUs we have on the deepdish server. The training took about 20-30 minutes.



We then run detection using our best weights - since we have one and only one object per image, we set both our confidence threshold and Non-Maximum Suppression Intersection over Union (NMS IoU) threshold to 0.1% and cap the maximum detections per image to 1.

Champion: Custom BBR

Our second approach utilized a convolutional neural network with a head of dense layers, the final of which contains four nodes, solving a regression problem. This model can be referred to as bounding box regression, where the inputs are images paired with a vector of four values.

These four values represent the upper left and lower right corners of the bounding box surrounding a foreign object in the images of X-rays.

Some processing was needed to make this modeling possible. Images were initially of varying sizes, and the aforementioned annotations were in terms of absolute coordinates. In order to properly scale against these differences when it comes to regression, the x values were converted to relative position in terms of image width, and the y values were converted to relative position in terms of image height.

The model architecture consists of several blocks of convolutional and pooling layers followed by a series of fully connected layers for bounding box regression (BBR). The input to the model is a 224x224 RGB image. In the first block, there are two convolutional layers with 32 filters each, using a 3x3 kernel and the ReLU activation function. The padding is set to 'same' to preserve the spatial dimensions of the input. The output of each convolutional layer is then passed through a max pooling layer with a 2x2 window size and a stride of 2. The second block follows a similar structure, with two convolutional layers with 64 filters each and another max pooling layer. The third and fourth blocks continue to stack convolutional layers, each with 128 filters, and a max pooling layer after the second convolution. In the fifth block, there are three convolutional layers with 128 filters each and a final max pooling layer. The sixth and final block consists of three convolutional layers with 256 filters each and a max pooling layer. After the last max pooling layer, the output is flattened to a 1D vector. The BBR part begins with a fully connected layer with 256 units and ReLU activation, followed by three more fully connected layers with 128, 64, and 32 units, respectively, all using ReLU activation. The final layer has four units, representing the four coordinates of the bounding box, and uses a linear activation function to provide continuous regression output. This output should be coordinates of opposite corners, proportional in relation to image dimensions. Overall, this architecture gradually reduces the spatial dimensions of the input image while increasing the number of filters, allowing the model to learn hierarchical features. The fully connected layers at the end process the extracted features for bounding box regression.

In compiling the model, the RMSprop optimizer is used, with the learning rate set to 0.0001 after tuning. The loss function chosen for this model compilation is the Mean Absolute Error (MAE). MAE. It provides a measure of the average magnitude of the errors, in this case, the proportional inaccuracy of coordinates relative to the dimensions of the image. In addition to the loss function, two evaluation metrics are specified: Mean Squared Error (MSE) and MAE again. These metrics provide additional information (perhaps slightly redundant) about the performance of the model during training and evaluation.

The model was then trained for 32 epochs, to allow for room to reduce loss, and at a batch size of 16, to balance performance and allow for an adequate sample of images in each step. Each epoch, when trained on a single mobile GPU, took under 10 seconds.

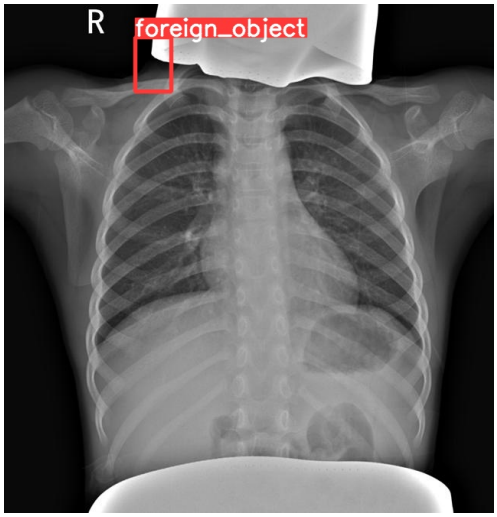
Model Comparison

Model performance and comparison was done with an intersection over union (IoU) metric. In order to determine the efficacy of the boxes, the area of overlap (intersection) is divided by the combined unique area (union). An image demonstrating this metric is below.

When the two models are assessed, the following performance is achieved:

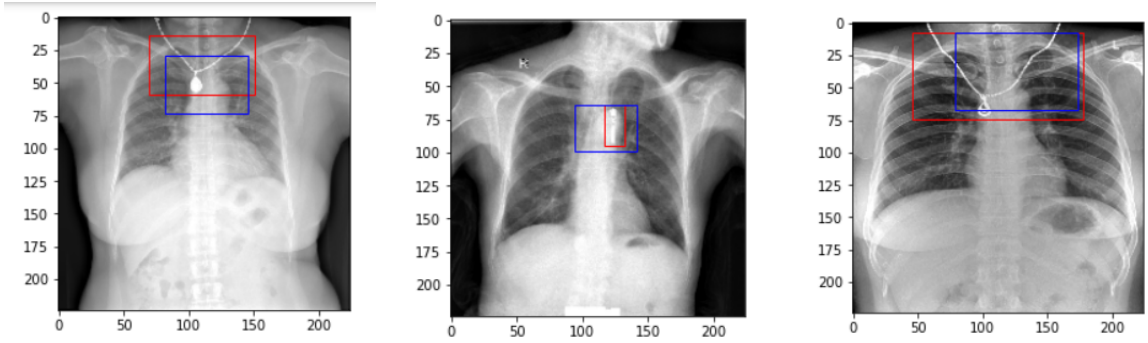
Average IoU from transfer learning with YOLOv5: 0.006351515828372584
Average IoU from in house bounding box regression: 0.11399357833664814

Sample image from YOLOv5 detection:

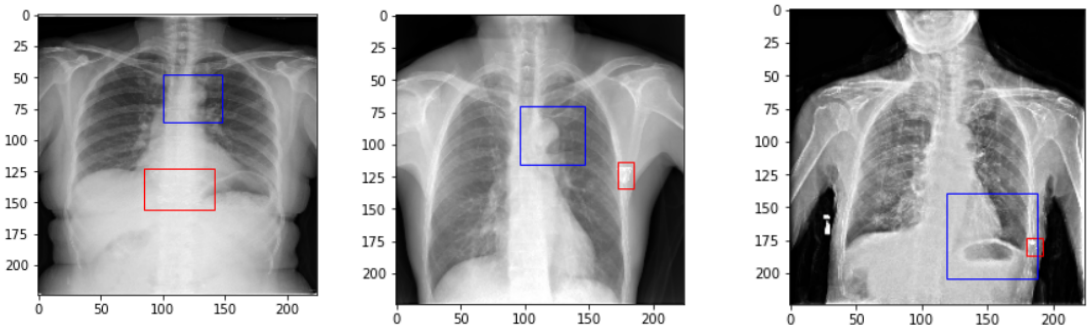


This particular sample is of the few images that have non-zero IoU. As we can see, this non-zero value is achieved as the YOLO box just happens to slightly overlap the corner of the true bounding box. A hypothesis for why YOLO is performing so poorly on this particular dataset is that the classification labels, in this case “foreign_object”, are identical across all annotations, while the actual foreign objects are of different shapes and sizes. In other words, the model is unable to determine what a foreign object should look like, due to a combination of insufficient data and excessive variation in samples. One particular solution would be to obtain higher quality annotations, labeling objects more specifically (with classification labels such as “necklace” or “zipper”) to improve fit with the YOLO architecture.

Sample images from bounding box regression (red is ground truth, blue is predicted):



The three examples above are of where the bounding box regression excels. One hypothesis is that the model is most competent in detecting objects that are significantly brighter than the rest of the image, perhaps through an edge detection layer. Another is that the boxes are being mainly drawn around the upper chest region, perhaps because the model is able to identify relative positions of the figures in the X-rays.



The three examples above are of cases where BBR falters. Note that, aligning with the first hypothesis, objects that have high contrast against their surroundings are being incorrectly boxed. Additionally, boxes are being drawn closer to the upper chest in most examples, providing evidence for the second hypothesis.

Classification

After merging all images from the original kaggle folders, train, test, and validation sets are being created using stratified 60/20/20 split. The training set thus contains 3,504 images, and both testing and validation sets contain 1169 images. A custom CNN model and a pre-trained Visual Geometry Group with 16 layers (VGG16) model are compared and contrasted based on their corresponding pros and cons in image classification as well as for medical images specifically. Custom models are more flexible in both structure depth and width, which allows for customization based on the specific problem at hand. The extent of freedom, however, needs to be coupled with detailed tuning, adjustments, and experimentation to achieve optimal

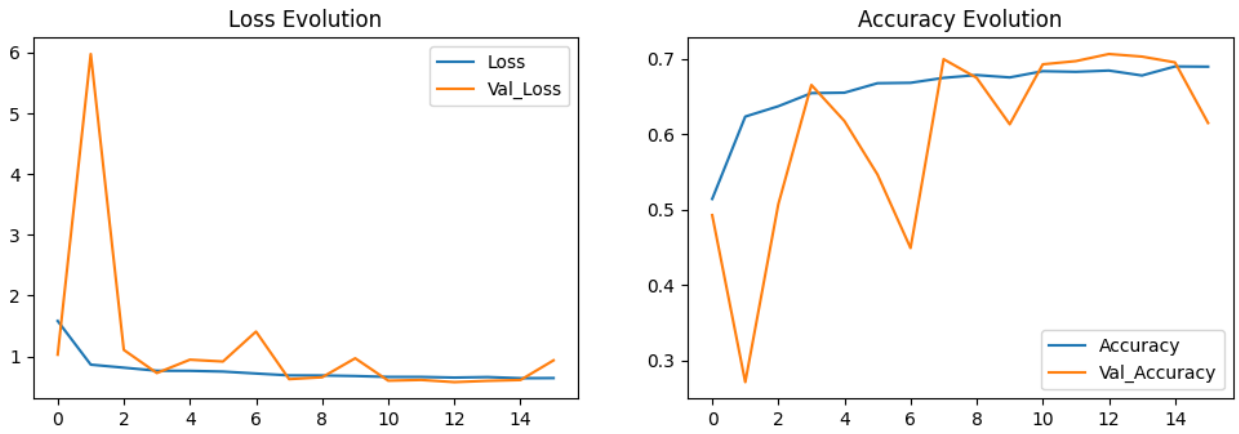
performance. Pretrained VGG16 model, on the other hand, is known to be able to learn and represent intricate image features. Nevertheless, its fixed architecture could limit its flexibility for more specific, and in this case much smaller tasks. Solutions such as freezing most of the early layers which capture more generic features will need to be incorporated in order for transfer learning to be truly adaptable and suitable.

Challenger: Custom CNN

Convolutional Neural Networks (CNNs) are a class of deep learning models that are specifically designed for processing structured grid data such as images. CNNs take their name from their use of convolutional layers, which perform a mathematical operation called convolution. This operation essentially involves filtering an input, such as an image, to highlight certain features or patterns. The unique strength of CNNs lies in their ability to automatically learn these filters, effectively teaching themselves to recognize complex patterns by building up from simpler ones. This makes them extremely powerful tools for tasks such as image recognition, where they have achieved state-of-the-art results. By using multiple layers of these convolutions, CNNs can capture hierarchical patterns within the data, with lower layers learning simple features such as edges and colors, and higher layers learning more complex, abstract features.

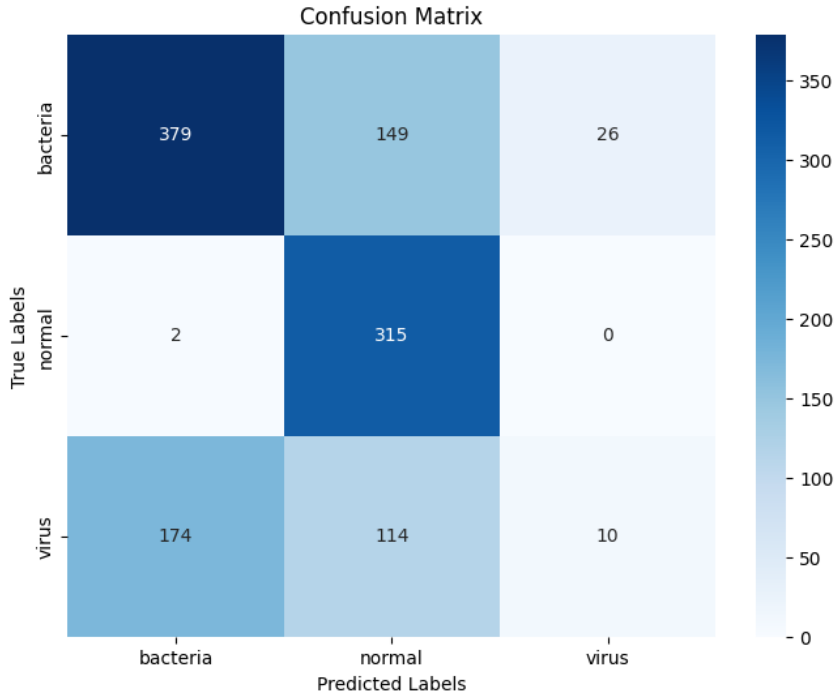
Our custom CNN model consists of three convolutional layers, each followed by a max pooling layer, batch normalization, and dropout regularization. The first layer has 32 filters, while the second and third layers have 64 and 128 filters respectively. After the last convolutional layer, the feature maps are flattened, followed by two dese layers with 64 units each. ReLU activation is used throughout. The output layer consists of three units with softmax activation for multi-class classification. Learning rate is scheduled using ExponentialDecay, with a starting rate of 0.001, a decay rate of 0.85, and a decay step of 100. Adam optimizer and cross-entropy loss are being used. Accuracy, precision, and recall are all being considered for the model evaluation to be more holistic. Dropout rates are being tuned using GridSearchCV, with 0.4 being the best hyperparameter. Due to computational constraints, we have chosen to conduct a grid search for the optimal hyperparameters using 32 epochs and 3-fold cross-validation.

The best model was fitted with 16 epochs. Training and validation loss and accuracy progression were plotted to track model performance. The accuracy evolution indicates a steadily increasing training accuracy while a fluctuating validation accuracy, underscoring the possibility of overfitting. Given the rather simple model structure and the implementation of scheduled learning rate, the lack of regularization might be one of the leading causes to such a trend.



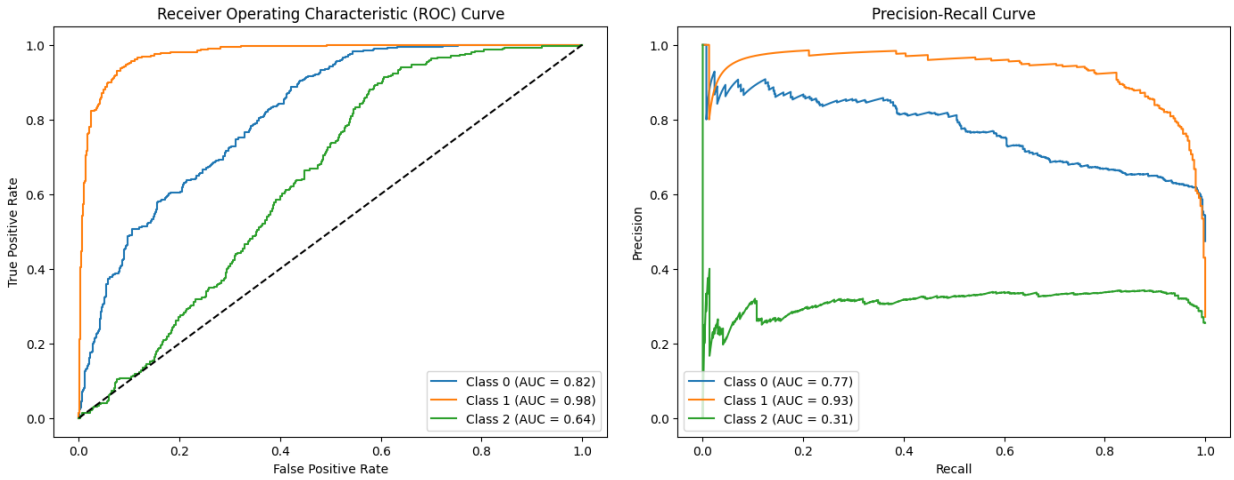
Custom CNN Model Loss and Accuracy Evolution

When evaluated against the test set, the CNN model has test accuracy of 0.60, precision of 0.65, and recall of 0.55. Given the context of the problem, **the false negative cost is higher than the false positive cost**, thereby recall is of larger interest in terms of evaluating model performance. The detailed confusion matrix is shown below.



When model performance is broken down by classes, the results indicate that the CNN model is really good at predicting normal cases, less so with bacteria cases, and the model almost failed to learn for virus cases. The recall for normal class is 0.99, bacteria class is 0.68, and virus is 0.03.

Both ROC and precision-recall plots illustrate similar results: the model is superior in learning and predicting normal cases, but less so for bacteria or virus cases.



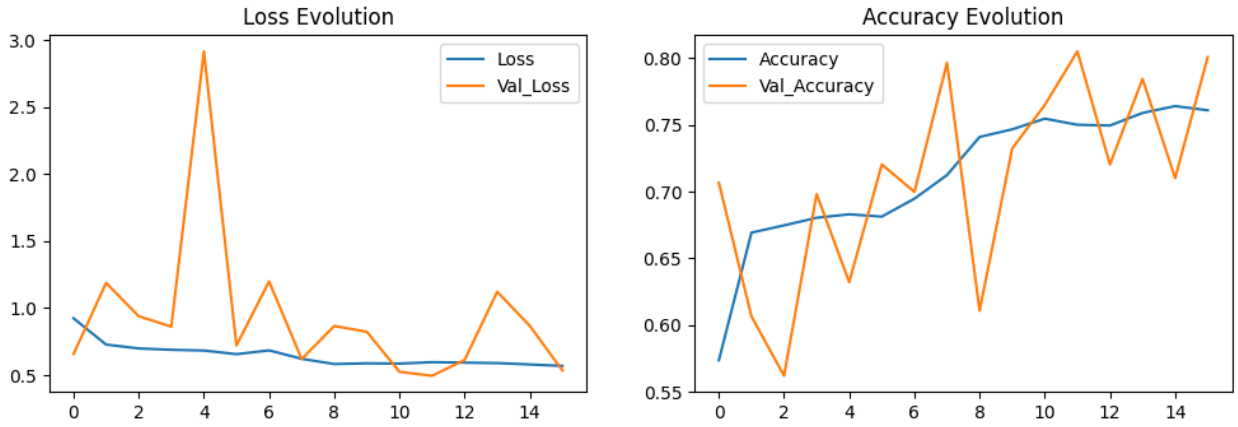
Champion: VGG16 Transfer Learning

The VGG16 model is a convolutional neural network model that stands out due to its simplicity and performance, specifically in the context of image recognition tasks. Developed by the Visual Geometry Group (VGG) at Oxford University, VGG16 refers to a model with 16 layers that include weights. These layers consist of 13 convolutional layers, followed by three fully-connected layers. VGG16, known for its deep yet simple to understand architecture, was a milestone in demonstrating the importance of depth in neural networks. In the context of transfer learning, VGG16 has often been employed. Transfer learning is a machine learning technique where a pre-trained model, such as VGG16, is used as a starting point for a related task. The idea is to leverage the feature-hierarchy learned by VGG16 on a large dataset (like ImageNet), rather than starting the learning process from scratch. This is especially useful when the new task has limited data. In such cases, the lower layers of VGG16 (which have learned more generic features from the initial large dataset) are frozen during training, and only the final few layers are trained to adapt to the specific task. By doing so, the transfer learning approach with VGG16 not only boosts the performance on tasks with limited data but also significantly reduces the computational cost and time.

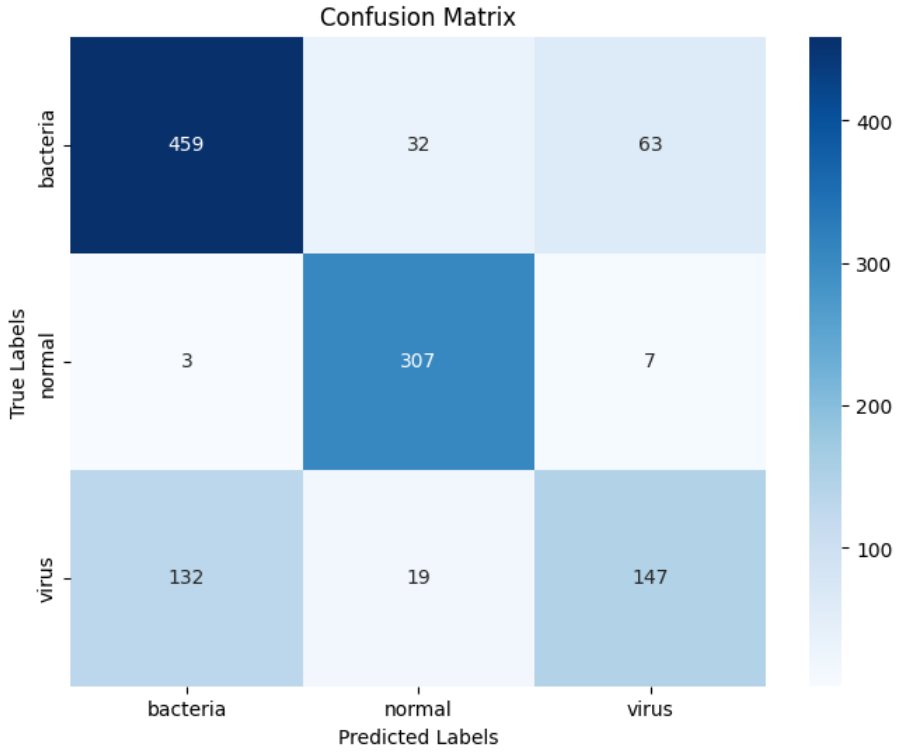
In terms of our approach for VGG16 transfer learning, all layers of the VGG16 based model are freezed except the last two. The dynamic model is constructed by adding global average pooling, followed by fully connected layers with batch normalization, dropout regularization, and ReLU activation. The model has a softmax output layer with three units for classification. The learning rate, optimizer, loss function, and evaluation metrics are the same as the custom CNN model for the purpose of comparability. Dropout rates are also being tuned using GridSearchCV, with 0.2 being the best hyperparameter.

The model was fitted with 16 epochs as well. Training and validation loss and accuracy progression were plotted to track model performance. Both loss and accuracy evolution are similar to those of the custom CNN model. The validation accuracy specifically is just as

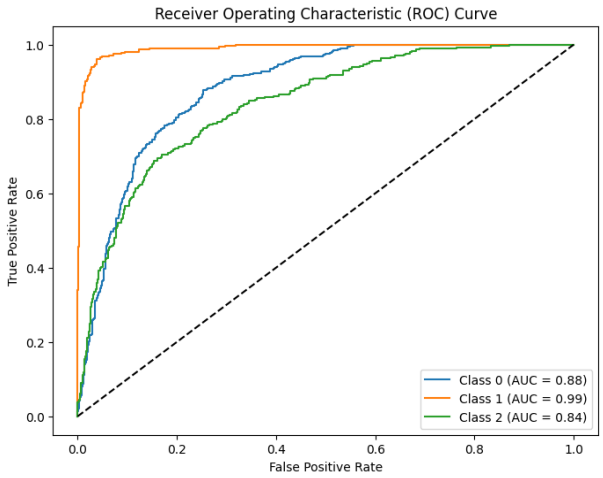
fluctuating.

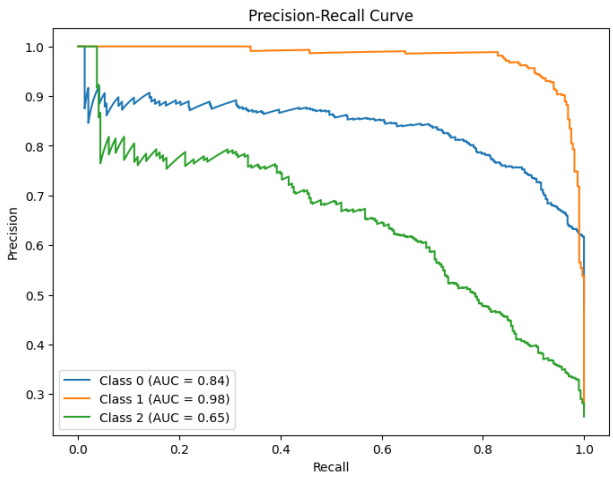


When evaluated against the test set, the VGG16 model has test accuracy of 0.78, precision of 0.79, and recall of 0.77. The detailed confusion matrix is shown below.



The VGG16 model is noticeably better at predicting both bacteria and virus classes than the custom CNN model. The bacteria class recall is 0.83 and the virus class recall is 0.49. The ROC and precision-recall plots show similar results.

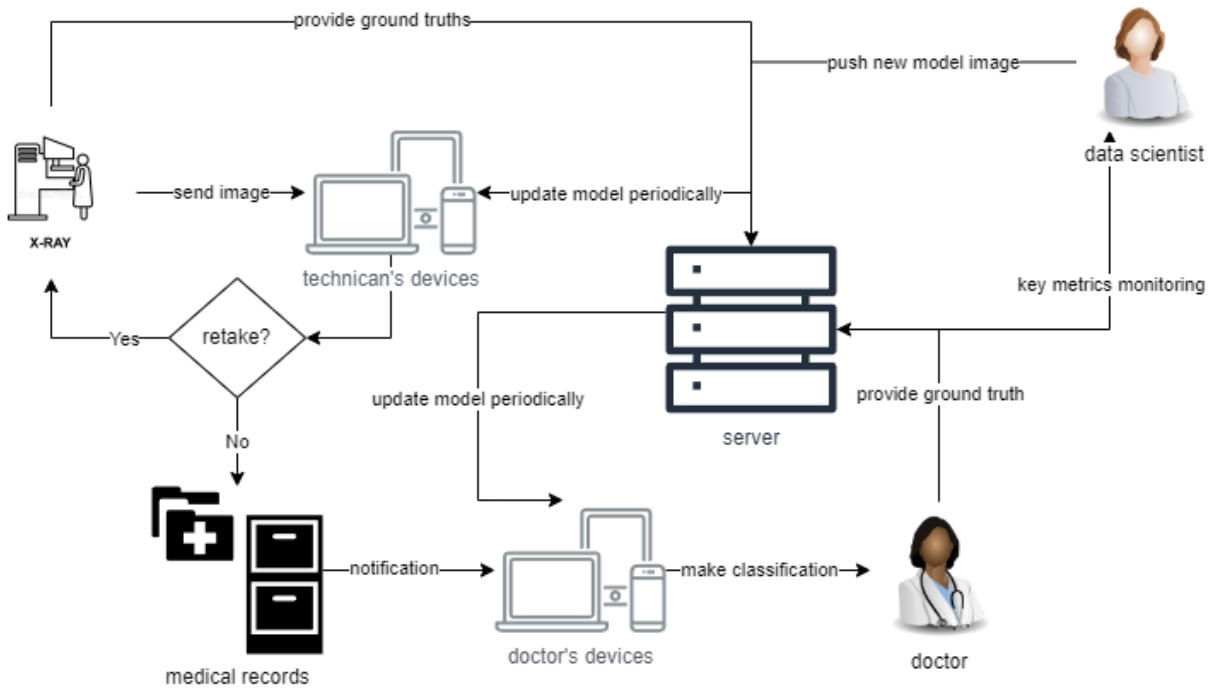




Model Comparison

Both CNN and VGG16 models are able to rather accurately predict normal cases. In fact, the CNN model is even better at predicting the normal class. The CNN model, however, appears not to be able to differentiate between normal and bacteria cases nor between normal and virus cases. It has the tendency of categorizing all classes as normal. The VGG16 model, aside from being nearly as good as predicting normal class, is better at telling bacteria class apart from normal class, yet it appears to be incapable of telling bacteria and virus classes apart. It highlights the fact that the real challenge lies in capturing the minute differences between virus and bacteria cases.

Model Operation



Deployment Mechanism

One of the inspirations for this project was to help out in regions that are constrained on trained professionals, which is why we want to achieve a high level of decoupling. We aim to minimize the need for direct communication between the technician operating the X-ray machine and the doctors inspecting the charts. Even today, organizations like Doctors Without Borders have over 45,000 dedicated employees worldwide, tirelessly working to provide medical services to underprivileged regions. Our goal is to support them in reaching their objectives efficiently and effectively.

The process starts with the technician operating the X-ray machine. After each CXR taken, the edge device will provide the person with a “retake” alert, drawing boxes around potential foreign objects. If the technician chooses to retake it, he/she would also provide the ground truth by drawing the boxes manually, which would be registered with the server. If not, the CXR image is uploaded.

Once uploaded, the doctor’s device would get a notification and the machine would make an inference on the type of pneumonia. The doctor would use it as a suggestion and make his/her own judgment. Once the diagnosis is done, it’s also sent to the server.

Model Maintenance

In the background, the server monitors the model performance quietly, keeping track of key metrics data scientists set forth. If a data drift is detected, the data scientist would be alerted and can retrain the model. He/she would identify suitable datasets, paying close attention to recent cases and potential edge cases. Depending on the severity of the situation, the data scientist can update parameters or even architecture.

After retraining, the model would be pushed to the server, and the server would broadcast it to end devices. The end users would be alerted with patch notes, ensuring transparency and understanding of the updated model. Older models would be versioned and preserved.

Conclusion

The challenges faced by healthcare imaging cannot be entirely overcome by a single solution. However, deep learning approaches have potential to become an inexpensive solution to expedite routine processes and prevent human errors.

With object detection, larger and better annotated datasets can greatly enhance predictive potential. While a more simple modeling approach might be sufficient in identifying X-rays that have abnormalities, being able to localize these abnormalities can improve X-ray quality while reducing time and training overhead.

Our classification model on pediatric chest X-rays shows promising results for identifying between normal and abnormal images, which already allows for quick identification of pneumonia in patients. However, future improvements on distinguishing between bacterial and viral pneumonia can be beneficial in personalizing treatment and prioritizing the more severe bacterial cases.

Reference and Resources:

1. The original dataset for classification derived from a selected cohort of pediatric patients of one to five years old from Guangzhou Women and Children's Medical Center in China. Kaggle
2. The data for foreign object detection can be found on Kaggle, more details of the dataset can be found here
3. YOLOv5 git repo
4. COCO dataset
5. YOLOv5 architecture summary