Cluster Neural Networks for Edge Intelligence in Medical Imaging

Cluster Neural Networks for Edge Intelligence in Medical Imaging

Kordel K. France, Zachary A. Newman

Seekar Technologies

Abstract

Cluster convolutional neural networks (CCNNs) have many advantages over single, one-shot neural networks that may be capitalized on in edge devices. Some industries investigating the use of machine learning into their products entertain applications where the consequences of low sensitivity or specificity in a model may be catastrophic. In some of these scenarios, processing capability is minimal and there is no availability for cloud transactions. In these instances, conventional convolutional neural networks (CNNs) may not fit the bill. Here, we propose a machine learning model architecture that produces increased accuracy at lower computational cost than conventional CNNs. We establish how this architecture is inherently smaller in size by design without compromising accuracy, opening artificial intelligence models up to a much larger population of edge devices and solutions in medicine. We illustrate our proof of concept of this architecture through the identification of seven respiratory conditions in chest X-ray images by utilizing the machine learning model in a mobile app. Additionally, we elaborate on how the model better protects data privacy, may be trained on much less data than typical neural networks, and allows for explainability of results. Finally, we evaluate the efficacy our model through a clinical trial and compare its results to a conventional CNN on the same data.

Introduction

Research into the use of neural networks has been increasing in popularity throughout the better part of the last two decades, particularly in that of computer vision and image classification. In consumer-based products, we have seen social media successfully integrate convolutional neural networks (CNNs) into playful image filters with the aid of augmented reality and we have seen our posted photos automatically tagged with labels of probable landmarks and people presented in the photo. These applications have done a fantastic job at proving the proficiency of machine learning in computer vision while also proving the market appeal of its capability. However, the consequences of these models underperforming are not near the degree to that of applications in industries such as medicine. Additionally, in the event that these consumer social media models provide an incorrect response, the end-user is always there to supervise and provide the correction with the convenience of time. Billions of people engage in social media worldwide which provides a substantial amount of training data and supervised learning on a newly launched algorithm.

The healthcare industry, in contrast, faces much less luxurious circumstances. Risk of significant financial compromise and even loss of life are consequences of a machine learning model underperforming in medicine. Here, we evaluate the efficacy of our model, a hierarchical cluster of convolutional neural networks (CCNN), to address five of the biggest issues with artificial intelligence in medicine and healthcare products – *explainability, lack of training data, patient privacy, efficiency,* and *quality of experience.*

A. Explainability

The use of machine learning models in healthcare mandates explainability in artificial intelligence (XAI). Neural networks are often labeled as being "black boxes" since the response received from a neural network can be difficult to trace. This is partially due to the fact that millions of possible paths are possible for an input to travel through as a result of a certain conclusion achieved. Although its accuracy may be high and the performance repeatable, the trace of logic that the network performs is especially difficult to extract. Many institutions[13,14] have proposed notable techniques to extract a process from a network decision. As one can obviously interpret, this "unexplainability" becomes a problem in scenarios where a life-threatening medical diagnostic must be performed. When a model provides a diagnosis that does not agree with the physician, the model needs to elaborate on its reasoning just as the physician can. Most products incorporating artificial intelligence are rejected by the Food & Drug Administration (FDA) for market use because they lack the ability to explain how a result is achieved[1]. Products demonstrated for the Department of Defense also encounter a similar issue, but contributors to this[2] are outside the scope of this paper.

B. *Lack of Training Data*

Many suggested artificial intelligence applications in healthcare are centered around scenarios where little training data exists. In light of this, the deployment of a new machine learning algorithm into the wild is extremely sensitive and the results of it being wrong are catastrophic. In the most recent event of the COVID-19 pandemic, many cutting-edge machine learning solutions have been proposed to aid healthcare responders in the overwhelming demand they face with screening for the virus. However, the timeline to when an accurate solution was provided (at least in the early stages of the pandemic) was less than ideal due to the fact that the virus was new and little information was available to sufficiently train a model. In many cases with cancers, there are an abundance of edge cases (highly improbable scenarios) in which the cancer may be identified, which makes acquiring sufficient data to train a model that much harder.

C. *Patient Privacy*

The merits maintained in the Healthcare Insurance Portability and Accountability Act (HIPAA) and the Digital Imaging and Communications in Medicine (DICOM) standards demotivate substantial data collection to protect patient privacy, and for good reasons. Machine learning models requiring cloud processing make it difficult to uphold these standards. Less network traffic correlates with less probability of signal compromise and data loss. Additionally, there are scenarios where the physician may not have the luxuries of time to wait for processing on a cloud network to interpret a result, which contributes another point calling for on-device machine learning models, or *edge intelligence*.

D. *Efficiency*

Edge intelligence is a popular research field. The appeal for edge computing, or performing processing on the device itself versus sending the computation to a remote server for processing, has been driven by a large demand for 5G communication and development in the Internet of Things (IoT)[19]. Neural networks and deep learning models in general are known to take a lot of computational time and energy to train. A deep learning model also results in large storage size which makes it difficult to incorporate into storage-limited edge devices. Various model compression methods exist and are continuously improving to allow deep learning models to live on these edge devices, but it becomes increasingly important to ensure this compression does not compromise accuracy. Offloading the processing power of millions of devices to a cloud will congest the network and poses several issues with latency. Additionally, performing transfer learning (further training or "fine-tuning" a

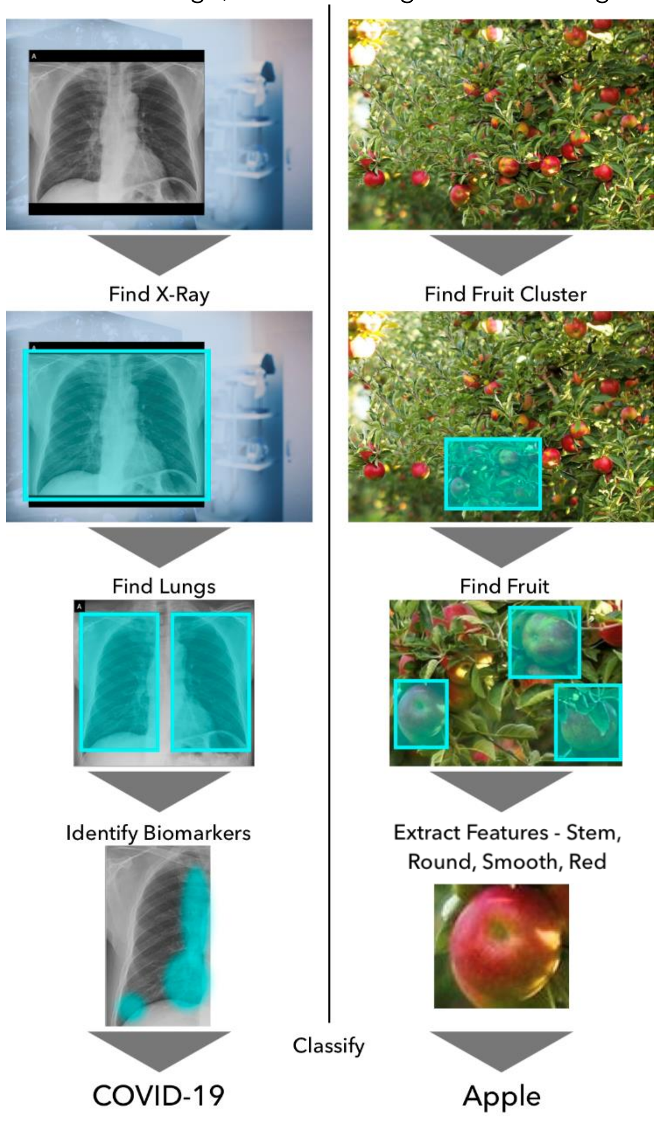


Figure 1 - Concept of how our model breaks a typical classification problem down into smaller subproblems.

model after it has been deployed) must be managed in such a way that calls for computational energy are kept to a minimum to make edge computing tractable.

E. Quality of Experience

New algorithms in artificial intelligence and machine learning are eloquently presented in research papers every month. These algorithms all show increased accuracy and reliability, but most simply remain at a conceptual level or are demonstrated on expensive laboratory computer systems not designed to favor user experience and real-world robustness. Additionally, these algorithms are often given clean data that doesn't entertain noise typically encountered with product use. In order for artificial intelligence systems to be more widely accepted into healthcare, development focus on the user experience becomes just as important as the previous four factors. Physicians need to have the comfort of knowing that they can rely on a product and intuitively navigate its functions through high-stress scenarios.

These factors give appeal to a different angle of thinking for engineers in designing models. The goal of this paper is to outline an approach to model design that could aid edge devices in gaining better classification performance and providing more forgiveness in use by addressing the five points above. We show that by breaking a difficult computer vision problem down into smaller tasks, a cluster of many small CNNs can outperform a single large CNN in many ways. We evaluate the efficacy and accuracy of this model in its ability to detect 7 different respiratory conditions in chest X-ray images. To achieve accurate representation of using an edge device in a practical clinical setting, an iOS software application was developed to both import and capture pictures of the X-ray images for evaluation by the mentioned model. We show how our algorithm helps rectify the problems listed above and, as a consequence of the model design, how a "train of thought" may be extracted from the algorithm to interpret how a certain conclusion resulted from the input image. Additionally, we illustrate how the machine learning model allows for more forgiveness on less training data as well as a method for increasing sensitivity on a young, freshly launched algorithm. As a benchmark, we compare the properties and performance of this algorithm against a standard VGG-style CNN on the same dataset. The approach

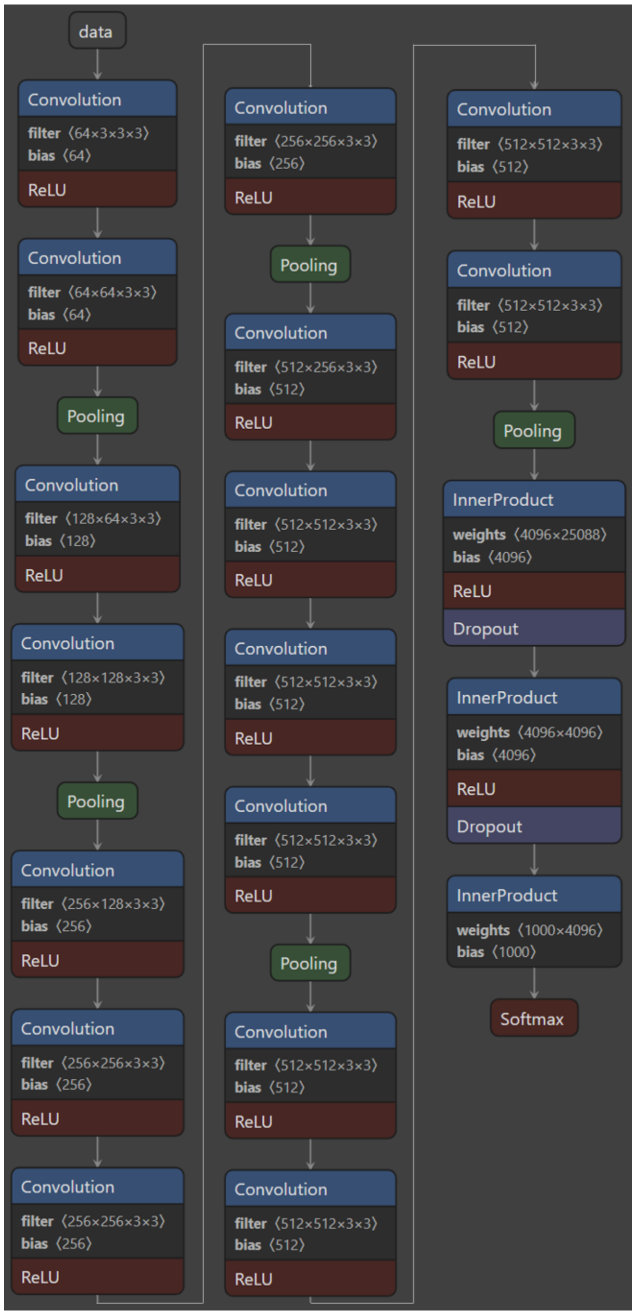


Figure 2 - Architecture of the 19-layer benchmark CNN[18].

we take is particularly favorable towards computer vision applications for computationally-starved edge devices and the model was evaluated on such a device one that is commercially available. Apple iPad's are commonly used in clinics so the entire evaluation procedure for the clinical trial was evaluated using a 2020 iPad Pro. Though a device of this type possess much higher data storage capacities than typical IoT devices, we simply use it as the vehicle to illustrate our proof of concept while maintaining all other restrictions regarding processing power, sensor availability, computational complexity, and final application storage size of the model that would be maintained by a typical edge device.

Dataset

For the purpose of the experiment, several open-sourced datasets of X-ray and CT scans were used to train and test the model. Datasets of radiographic images from patients with confirmed COVID-19 infections, Severe Acute Respiratory Syndrome (SARS), Common Bacterial Pneumonia, Emphysema, Atelectasis, Pneumothorax, and normal healthy subjects were utilized from open-sourced datasets on Github[4,6] and Kaggle[5]. The radiographic images were obtained from three open-sourced datasets including the National Institute of Health via Kaggle[7] and two GitHub datasets[15,9]. All evaluation data, however, was accumulated from a clinical partner through the process of a clinical trial specifically organized to determine the efficacy of this machine learning model.

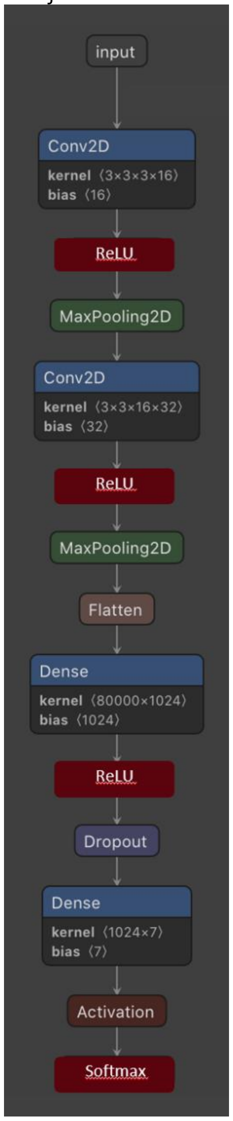


Figure 3 - Network architecture of each of the 8 node networks[18].

Inputs into each neural network were first normalized since image dimensions of the data were not all identical. To do this, the radiographic images were all rescaled to the mentioned size of 600 pixels in width by 600 pixels in height in 3 dimensions along the RGB channel. In order to achieve some independence of rotation, the training images were each mirrored and rotated 5, 10, and 15 degrees both clockwise and counterclockwise about the central axis effectively making an additional 6 copies of each image to potentially pool from during the training dataset.

Background

In a typical computer vision application, a single convolutional neural network

Table 1 - Image count for each node network.

HNN Name	Image Count
Node 1 (Trunk)	4,209
Node 2	9,286
Node 3	9,892
Node 4	2,990
Node 5	6,186
Node 6	3,340
Node 7	5,216
Node 8	5,216
Total Images	46,335
Benchmark	16,441

Table 2 - Final storage size of each node network.

HNN Name	Size (KB)
Node 1 (Trunk)	99
Node 2	34
Node 3	34
Node 4	17
Node 5	17
Node 6	17
Node 7	17
Node 8	17
Total HNN Size	**252**
Benchmark	1,260,000

may be deployed to identify patterns in an image. Once trained on thousands of images, the accuracy achieved from these networks in correctly identifying these patterns is substantial. Consequences of using this accurate, trained network are a large storage size, large parameter count, and the requirement of a multitude of readily available training data. For small edge devices, megabytes or gigabytes of storage are not available to host such a large network. One solution to rectifying this issue is to host the network on a cloud and allow the device to transmit the data there for processing on a large neural network, providing the edge device was built with the capability of hosting an internet connection. However, in some applications, the time it may take for cloud processing to complete and return a response may not be a luxury available to the patient and physician. Additionally, internet connectivity may not be available to facilitate such a transaction. Another solution is to remove some learning layers from the network to decrease its storage size, but this comes with a compromise in algorithm accuracy and robustness. Since the number of parameters and connections (storage size) increases exponentially with more classification inputs and learning layers, accurate networks that contain several layers are inherently large. In this experiment, we use a VGG-style CNN reflecting the above restrictions as the benchmark model to validate against our new model. This new model incorporates a cluster of eight smaller CNNs organized in a hierarchical fashion. This effectively creates a network of CNNs with each CNN specifically trained to identify a different pattern. The training properties and performance of these two models will be analyzed and compared.

Fundamentally, our proposed model works by breaking a typical classification problem down into smaller subproblems. As an example for how our model better classifies respiratory conditions in X-ray images, consider the concept presented in Figure 1. We are given an image of a fruit tree where the goal is to determine whether or not the image contains apples. A conventional neural network would attempt to classify whether or not the entire image contains an apple. Our model breaks the image down by identifying all of the regions in the image that contain objects that look like clusters of apples, then identifying each object within that cluster that looks like an individual apple, and then finally looking at each of those objects to see if, indeed, it is an apple. Our approach allows the model to have sub-models that specialize in each task with a final model that stiches all of the sub-model data together for a general understanding. One may think of this as a form of scene segmentation, similar to that of what some self-driving vehicle models use. This approach shares many similarities in how image classification works in the cortex of the human brain[20].

Nearly all deep learning algorithms emulate a common recipe by combining a specification of a dataset, a cost function, an optimization protocol, and a model. Our model remained true to this process in its overall design. A multi-step procedure was used to teach the model how to not only identify the seven respiratory conditions but also alleviate significant optical noise from the X-ray images. This three-phase process involved (1) deep learning on eight independent CNNs, (2) hierarchical learning among those eight CNNs together, and (3) coupling an image preprocessor into the final data pipeline for denoising.

Phase I: Deep Learning

Eight individual CNNs were designed for this trial in order to adequately distinguish the seven respiratory

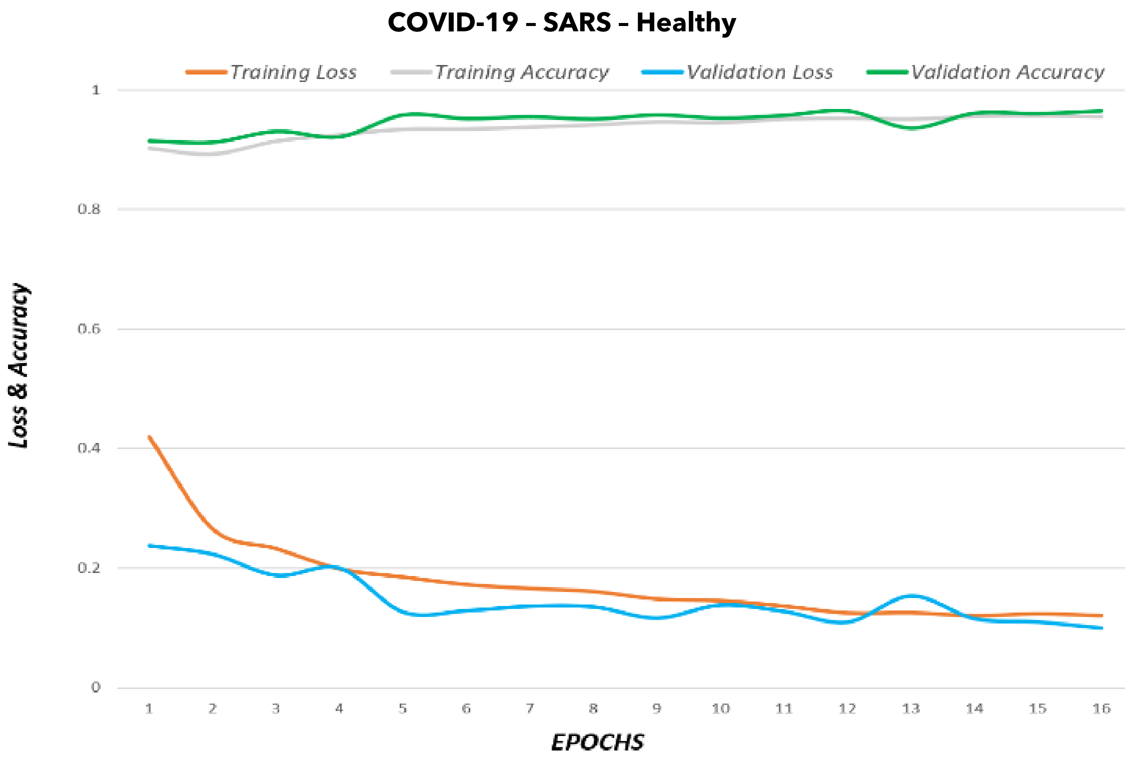


Figure 4 - Training metrics for final Node 7 network.

conditions, so each individual CNN initially needed to be trained and evaluated separately. These networks were trained on a desktop computer with an Intel Core i9 processor and an AMD Radeon RX 560 GPU. The hardware used is typical off-the-shelf computing hardware that is non-custom. It was important to the authors to illustrate the achievability of the results of this architecture on commonly available hardware. These models all followed a VGG-style architecture with 3x3 convolutional filters used for each of their 4 layers; the model architecture is better shown in Figure 3. As we will later discuss, each of these CNNs is a "network node" that belongs to a larger cluster network so each was trained and evaluated on a different permutation of the 7 possible respiratory conditions. The exact count of images used to train each node network is illustrated in Table 1. Note that the total number of images used to train the node networks is high but that each node network itself required many less images to train. This is to demonstrate the capability of this architecture to learn and achieve accuracy on less data.

As an example, the results are shown in Figure 4 for the final evaluated Node 7 network before it was compressed for evaluation in edge-computing. These results illustrate the convergence to a sensitivity of 98.1%. The node networks were all trained with similar hyperparameters with the exception of epoch count. The number of epochs, or number of times the model passes through the entire dataset, used for training each network differed slightly due to the fact that the training losses on some models converged sooner than others.

The architecture of the 19-layer benchmark model, by contrast, may be viewed in more detail in Figure 2. The model took just over 6 hours to train on the mentioned hardware with 8 epochs, a batch size of 32, and a learning rate of 0.001.

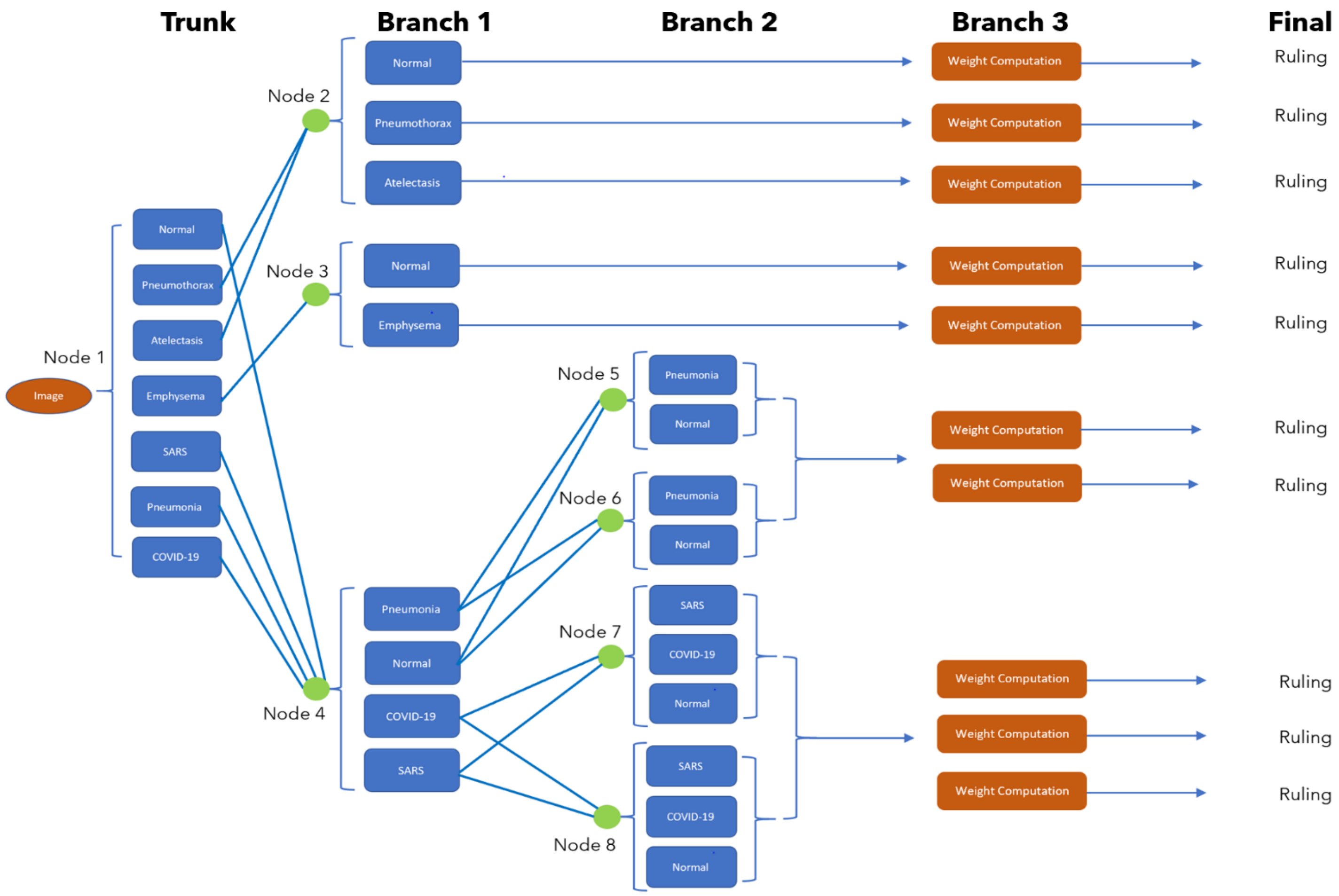


Figure 5 - Architecture for the cluster network. Each green dot represents a separate trained convolutional neural network. The blue boxes near it are the training labels for that network.

Once trained, each network was then compressed to fit on an Apple iPhone or iPad (the edge device). The benchmark model was also compressed. In total, the entire CCNN comprising the eight smaller models totaled 251 KB versus 1.26 GB for the benchmark VGG model. Table 2 illustrates the size of each CNN trained in the model.

Phase II: Hierarchical Learning

Consider the mathematical model in Equation 1 of an individual neuron of a simple convolutional neural network:

$$f(\mathbf{x}) = b + \sum_{j=1}^{N_H} v_j h(\mathbf{x}; \mathbf{u}_j) \qquad \text{Equation 1}$$

The network takes an input ***x*** and contains one hidden layer with N_H units, where v_j represents the output weights and $h(\boldsymbol{x};\boldsymbol{u})$ is the transfer function dependent on those weights. A linear combination of the outputs of the hidden units with a bias *b* is used to obtain *f(x)*[11].

In a similar manner to how the weights, ***u***, and biases, ***b***, are used to influence the final output of the neuron in a standard CNN, our proposed model calculates weights and biases for individual networks in a cluster of CNNs. These individual CNNs are called "network nodes" and the weight and bias calculations are used to analyze how each network node influences the final outcome in a hierarchy of network nodes. Effectively, the relationship between network nodes is similar to that between individual neurons.

In Figure 5**,** each green dot represents a node that denotes a separate CNN; the blue boxes next to that node indicate the possible classification labels from that network. These eight network nodes are organized in such a way that each subsequent network has fewer output possibilities, effectively decreasing loss with each successive network. As an image is first classified in the CCNN, it enters a CNN containing output labels of all 7 different conditions; this first CNN is denoted as the *Trunk* in Figure 5. Contingent on which classification is determined from this trunk network, it is then fed into a subsequent node network in Branch 1. Similarly, based on the classification from the first branch network, it enters into either another CNN in a subsequent branch or moves to a final weight comparison that is used to determine the final ruling. This final weight comparison is denoted in *Branch 3* in Figure 5. At this point, scores taken from each of the CNNs in the experienced nodes are evaluated for a final classification result.

One will notice that some nodes have identical output labels as their neighbors. This is intentional—the model was constructed to significantly reduce the potential of the patient receiving a false-negative result on their X-ray image. Pneumonia, SARS, and COVID-19 are similar in their appearance in an X-ray image and, in experimentation, it was particularly easy to confuse the model between these three when evaluating an X-ray image[7]. Due to this, multiple models with different emphases were trained on data of the same output labels, and the processing path was forked after the determination in the first node. Each model looks for slightly different patterns over the same labels when the image is passed through.

Each node network of the overall network outputs a confidence in classification that is fed into subsequent node networks for use in the final prediction. As an example, let's consider an image that truly shows symptoms of COVID-19. In order for an image to gain a true positive result of COVID-19, the image must go through *at least* 4 nodes of CNNs and a final statistical weighted analysis. Upon completion of this weighted analysis, if an adequate ruling with a confidence level greater than 65% cannot be determined, the image is rotated about its central axis by an arbitrary angle between -5 degrees and +5 degrees and sent through these those 4 nodes a second time. This adds an element of "stochastic activation" to the model. This process is recursively repeated until the 65% confidence threshold is achieved. Only after this process is complete is the final result presented. In the event that it is never achieved after 5 cycles through these 4 nodes, then an "indeterminant" classification is returned from the CCNN. This process helps maximize recall and minimize false negative results due to the fact that there is a consistent "check" against multiple networks that look for different biomarkers on the same image. For example, if the network in Node 4 outputs a 65% confidence in the image representing COVID-19 and the second network of Node 7 outputs a 98% confidence in the

$$p(\mathbf{y}_A|\mathbf{y}_B) = \frac{p(\mathbf{y}_A)p(\mathbf{y}_B|\mathbf{y}_A)}{p(\mathbf{y}_B)} \qquad \text{Equation 2}$$

$$p(y|\mathbf{x}) = \frac{p(y)p(\mathbf{x}|y)}{\sum_{c=1}^{C} p(\mathcal{C}_c)p(\mathbf{x}|\mathcal{C}_c)} \qquad \text{Equation 3}$$

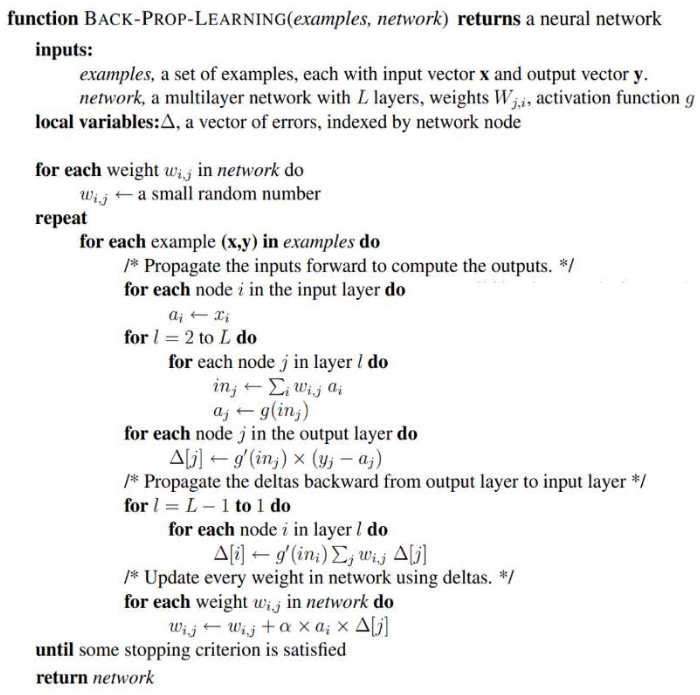
image representing a healthy individual, the image is not automatically deemed as "healthy." The confidence numbers and weights from the networks of all previous nodes are compared against these new numbers to contribute to the final ruling; if a ruling still receives an overall low-confidence, it is fed back into the network for reclassification until a ruling is achieved above the mentioned threshold. Since each node network is weighted, those weights are used in determining how much total contribution the result of a node network should have towards the final ruling. The architecture is designed to slightly favor sensitivity over specificity due to the severity of a false-negative occurrence in this particular application.

Bayes' theorem is a foundational element to every statistical learning algorithm. Concisely, it decomposes the probability of a specific event being achieved by basing it on prior knowledge of the contributing conditions of that event. It is expressed mathematically in Equation 2[11].

One can see the importance of this in evaluating the prior and posterior probability distributions of an image classification as the image is fed from node to node. More explicitly, at the conclusion of each network node, class-conditional distributions *p(x|y)* for $y = C_1, \dots, C_C$ with the prior probabilities of each class were modeled. These values were then used to compute the posterior probability for each class using Equation 3.

For example, referring to the probability of an X-ray image being classified as COVID-19 at the outcome of *Branch 1*, given that it has been classified as COVID-19 through the trunk of the network can be expressed in Equation .

A similar process can be followed for each classification-node permutation encountered by the image as it travels through the network classification sequence. Let it be re-stated that weights are attributed to these numbers and compared at the end of *Branch 2* of the network; as transfer learning occurs, these weights may shift the resulting posterior probabilities up or down.



```
function BACK-PROP-LEARNING(examples, network) returns a neural network
  inputs:
      examples, a set of examples, each with input vector x and output vector y.
      network, a multilayer network with $L$ layers, weights $W_{j,i}$, activation function $g$
  local variables: $\Delta$, a vector of errors, indexed by network node

  for each weight $w_{i,j}$ in network do
      $w_{i,j} \leftarrow$ a small random number
  repeat
      for each example (x,y) in examples do
          /* Propagate the inputs forward to compute the outputs. */
          for each node $i$ in the input layer do
              $a_i \leftarrow x_i$
          for $l = 2$ to $L$ do
              for each node $j$ in layer $l$ do
                  $in_j \leftarrow \sum_i w_{i,j}\, a_i$
                  $a_j \leftarrow g(in_j)$
          for each node $j$ in the output layer do
              $\Delta[j] \leftarrow g'(in_j) \times (y_j - a_j)$
          /* Propagate the deltas backward from output layer to input layer */
          for $l = L - 1$ to $1$ do
              for each node $i$ in layer $l$ do
                  $\Delta[i] \leftarrow g'(in_i) \sum_j w_{i,j}\, \Delta[j]$
          /* Update every weight in network using deltas. */
          for each weight $w_{i,j}$ in network do
              $w_{i,j} \leftarrow w_{i,j} + \alpha \times a_i \times \Delta[j]$
  until some stopping criterion is satisfied
  return network
```

Figure 6 – Shown for reference[3], the pseudocode for the Backpropagation algorithm in a neural network from Russell's and Norvig's "Artificial Intelligence: A Modern Approach." The CCNN uses an additional version of "backprop" between node networks.

Gradient-based optimization is a cornerstone for most machine learning models. Our model stays true to this by incorporating a derivative of this principle into the hierarchical learning between node networks. During hierarchical learning, the model optimizes the gradient between node networks just as it does between layers during the training of each node network individually. Backpropagation played a similar role between node networks in achieving model cohesion as backpropagation between layers in a conventional CNN. The weights for each node network are updated after each classification during hierarchical training through backpropagation. As we will show later, this also occurs during transfer learning after model deployment. Specific attention had to be attributed to optimizing the gradient of the

$$P(\mathit{COVID\text{-}19})_{\text{Nodes 1 and 4 Posterior}} = \frac{[P(\mathit{COVID\text{-}19})_{\text{Node 1 CNN Prior}} \times P(\mathit{COVID\text{-}19})_{\text{Node 4 | Node 1}}]}{P(\mathit{COVID\text{-}19})_{\text{Node 4 CNN Prior}}}$$

Equation 4

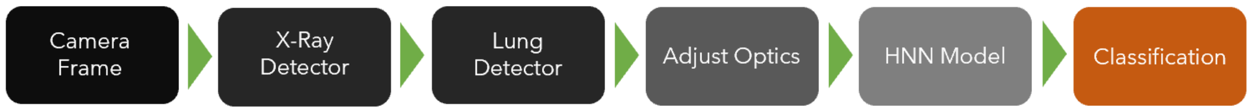


Figure 7 - Processing pipeline of each X-ray image after it is captured by the camera or imported.

hierarchical training loss as a consequence. The backpropagation algorithm is shown in Figure 6 for reference[3].

Phase III: Preprocessing and Denoising

For this project, it was important to the authors to design the experiment to closely resemble use in a practical setting. After all, the primary appeal for this model is for use in edge devices. Many new machine learning models proposed are proven accurate on a stationary computer with clean data. As any computer scientist knows, these models often behave in arbitrary ways when coupled with environmental noise, optical fallacies, and user input error. Even a perfected model in a controlled setting is not independent of minor shifts in translation or rotation as Goodfellow and others have shown[11]. In a clinical setting, these consequences are devastating and can determine the fate of patients. To these points, the model demonstrated in Phases I and II was coupled with an image pre-processor and optical denoiser to allow for some forgiveness in the integrity of the captured or imported image, to provide a better user experience, and to promote overall confidence in the model architecture. Extra attention was paid to the design of the user interface in the app to ensure the user experience was intuitive and simple for a physician to use. Actual insight from real physicians was sought during the design process to deduce explicit user needs.

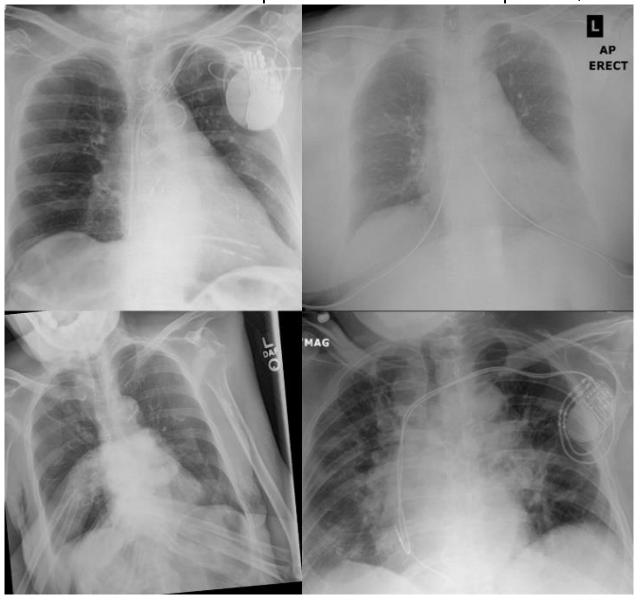
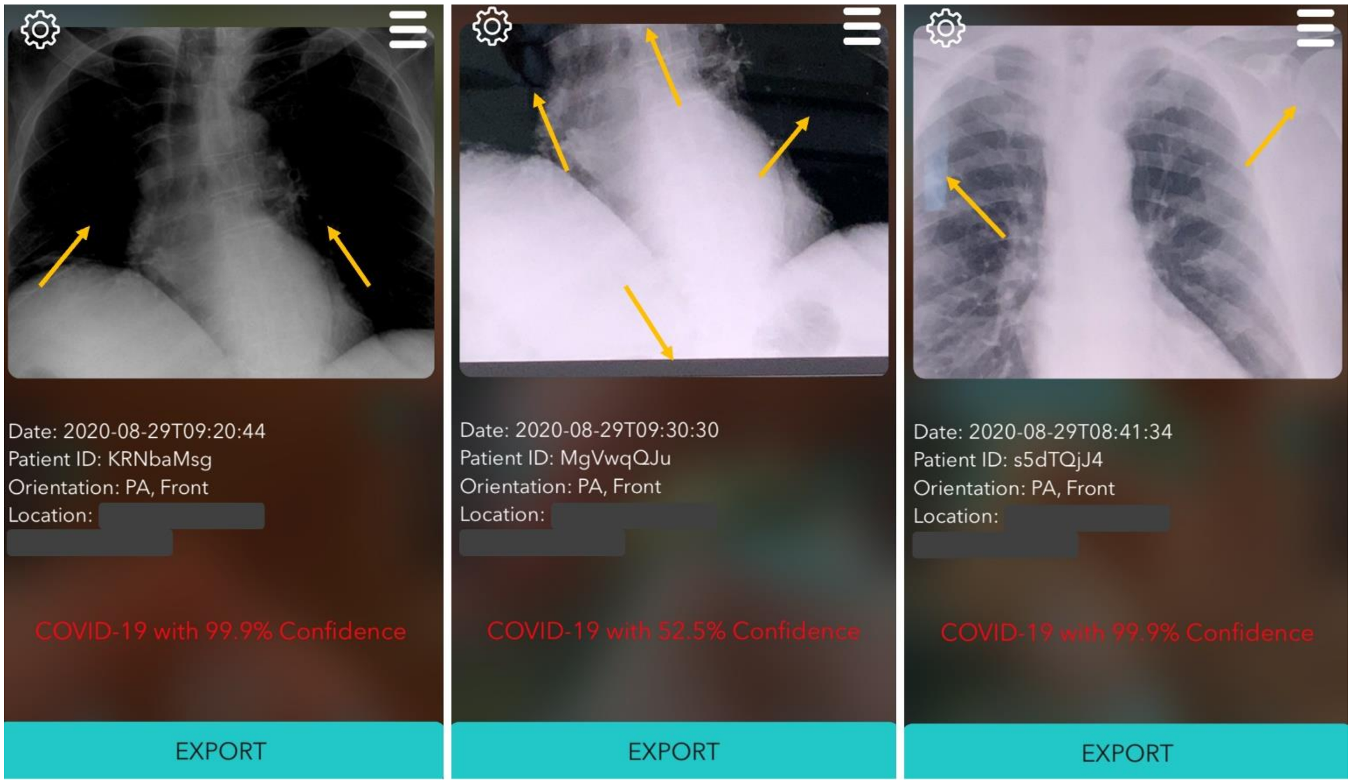


Figure 8 - Four images showing the model's ability to generalize and look past atypical artifacts present in the X-ray. (Top Left) A pacemaker is present in the patient's chest. (Top Right) Text is present in the top right of the image and cables are visible towards the bottom. (Bottom Left) The image is not vertically aligned and poorly cropped. (Bottom Right) A pacemaker is present in the patient's chest and more text is visible on the image.

The hierarchical layout of network nodes, in part, contributes significantly to the denoising effort by segmenting classification. In addition, the app makes use of a live object detector (the X-ray detector) that identifies a chest X-ray image within a live camera feed with an inference speed of 60 frames per second. This object detector automatically locates the X-ray image and highlights it with a visual blue box. This box provides the bounding dimensions for the input image that is allowed in the network. Since our model receives an input image of 600 x 600 dimensionality, the bounding box crops the image to a 600 x 600 resolution. Once the radiographic image is identified and cropped, a second object detector (the lung detector) is initialized to find and segregate the lungs within the cropped X-ray image. Since most of the biomarkers that identify the seven possible respiratory conditions are visible in the lungs, it is important to distinguish these in the image to avoid having the CCNN classify unintended patterns in other parts of the X-ray image. Next, light properties are then adjusted in an optical denoiser module to realize more image resolution; these properties include exposure, saturation, vignetting, and focus. Consequently, the resulting image is rid of much of the glares, ghosts, and other optical noise that may be present in a radiologist technician room. The image may now be fed into the CCNN for classification. Figure 7 shows the entire pipeline incorporating this pre-processor.

Finally, we arrive at the point of model explainability. The model was built to log the telemetry of each classification during training. Each time an image was classified, a report was generated that showed the


Date: 2020-08-29T09:20:44
Patient ID: KRNbaMsg
Orientation: PA, Front
Location: [redacted]

COVID-19 with 99.9% Confidence

Date: 2020-08-29T09:30:30
Patient ID: MgVwqQJu
Orientation: PA, Front
Location: [redacted]

COVID-19 with 52.5% Confidence

Date: 2020-08-29T08:41:34
Patient ID: s5dTQjJ4
Orientation: PA, Front
Location: [redacted]

COVID-19 with 99.9% Confidence

Figure 9 – 3 screenshots from the app of images showing COVID-19 captured via the device app. Arrows indicate optical fallacies and general noise. (Left) A report showing the model's ability to auto-adjust contrast to reveal critical biomarkers. (Middle) an image showing the model's ability to re-crop an X-ray image that was originally poorly cropped and rid it of optical noise. (Right) Image showing the model's ability to remove optical glares and contrasts.

probabilities, weights, and labels associated with each node network passed through as well as the backpropagation metrics. The model returned a score on the quality of each image after it was fed through the preprocessor to determine how much noise was still present. This report showed the model's "train of thought" by illustrating exactly how the image was transferred between node networks. This exposed any faults within the model and showed engineers specific weights that needed to be manually adjusted in order for the model to continue increasing accuracy and escape a local minimum in loss. Continual incorrect classifications could expose entire nodes at fault and indicate the need for their retraining or replacement. In transfer learning, this was specifically helpful since each individual node network could be adjusted itself instead of having to adjust the whole network. Additionally, in one specific case where the model continuously returned false positives on *Emphysema,* node networks 4, 5, 6, 7, and 8 were "turned off" to focus generalization on only a few labels. The engineers did not encounter any scenario during transfer learning where adjusting a specific node network negatively impacted the entire cluster network. This represents a huge advantage to doctors and physicians in being able to determine where weak points in the model are and receive the model's final results with appropriate regard. The ability for a machine learning model to explain itself also contributes to one of the most important topics of AI in general; in order for machine learning to truly gain acceptance in medicine, a model must be able to explain its rationale just as a physician can.

Results

Overall performance for the model evaluated during the clinical trial showed very promising in the final analysis. Results for the CCNN showed to acquire 94.55%, 86.54%, and 90.65% in sensitivity, specificity, and accuracy, respectively, in correctly distinguishing COVID-19 between the 7 respiratory conditions. On the other hand, the larger benchmark CNN showed more meager results with 58.93%, 83.33%, and 70.91% respectively in sensitivity, specificity, and accuracy. 110 images were used for the evaluation of both the benchmark and CCNN models. 55 images were assessed using the "import" function of the mobile application developed, and 55 images were assessed through capturing a photograph of the X-ray through the camera on the edge device. Both models were evaluated on the same dataset of 110 images.

A major contributor to the successful results was the preprocessor. The segmentation of the X-ray image in the camera field of view and subsequent segmentation of the lungs allowed the model to work as intended and only analyze the critical part of the X-ray report. To demonstrate how well the preprocessor performed, Figure 8 shows examples of rather noisy images that were correctly classified as their ground truth labels. One can see how the model generalizes well enough to filter out the pacemaker, text in the corners, and other artifacts that may appear as unintentional patterns for the model to interpret as biomarkers and negatively affect model sensitivity. Additionally, Figure 9 shows actual screenshots taken from the mobile app used for the clinical trial. These particular screenshots further demonstrate the model's ability to reconcile bad images.

Discussion

Some readers may be skeptical about whether or not a neural network is applicable and even capable of accurately classifying respiratory conditions in X-ray images. The universal approximation theorem[12] states that a neural network, given enough hidden units and layers, may approximate any function mapping from any finite dimensional discrete space to another[8]. Interpreted differently, the theorem validates that a mathematical model is capable of learning to approximate a close solution to this classification problem (a nonlinear function), but that the size and specific architecture of this network is not explicitly determined. This was exemplified in the above study since the benchmark model was a single neural network with significant learning depth and significant size, but the model still was unable to learn enough to approximate this closed solution to the desired accuracy. Equivalently, a derivative of the universal approximation theorem may be used with our cluster network model in that the cluster network is capable of *learning* to approximate a close solution, but the number of node networks and hierarchical architecture is not explicitly determined. In this case, that architecture was determined through hierarchical learning while the number of node networks was determined through experimentation.

Probably the most noteworthy success of this model after accuracy is its decision explainability. Simplifying the task by segmenting the image allows each model component to become more specific in the patterns it recognizes. It also allows one the ability to uncover erroneous behavior without having expertise in the overall system[14]. Constant mis-classifications of Emphysema, for example, point to the CNN at Node 3 and the engineer knows to focus on the adjustment of those weights or swap the CNN out entirely. The topic of segmentation arises often within the self-driving car industry. Some scientists side with the approach of scene segmentation since its facilitation is very similar to how the human brain interprets its visual field through the optical nerve in the eyes[17]. Others back an end-to-end approach indicating that computers can make better sense of the information than a human without breaking down the scene and that segmentation actually slows the model down by having so many tasks running in parallel. While there is not (yet) a correct answer to which is most robust, our model favors the former in that best results were achieved parallel processing many smaller tasks.

Throughout the experiment, we also found transfer learning simpler. Transfer learning typically involves updating the knowledge of a single CNN *after* it has been trained. The CNN is trained to classify additional labels and its weights are adjusted to accommodate this new label without compromise in the accuracy of the other existing labels. Our model takes an additional angle with transfer learning by giving the engineer the ability to adjust or entirely replace node CNNs within the CCNN. We carried out this experiment by transfer learning a prior CCNN without Pneumonia to include Pneumonia. That model eventually evolved to the exact model used in this paper. The success of this gives hope to being able to design AI products that can increase their scope of capability while maintaining the trust of the physician.

Conclusion

This paper introduced five main fields the proposed model achieves significant advantages in:

- *Explainability*—We showed how integrating a report that logs the classification process of every image input into the model is invaluable to artificial intelligence engineers and physicians. Breaking a hard problem usually tackled by a single CNN down into several smaller subproblems allowed our model to achieve much higher accuracy since each node network was trained to recognize a very specific pattern. This further facilitates explainability since it can be shown how an image flows through the cluster network and how model sensitivity is affected by design decisions.

- *Privacy*—The smaller node networks allowed for a much smaller overall model negating the need for an internet connection which, in turn, protects patient privacy. This is further achieved through data encryption on the device.
- *Lack of Training Data*—We showed how each node network was much smaller in scope compared to the benchmark model and required less training data to achieve good accuracy.
- *Efficiency*—We explained how incorporating a preprocessor to clean the image before classification was necessary to achieve high model sensitivity. Each model also generalized better on their respective datasets and correctly classified more edge cases than the benchmark model. It was proven that multiple smaller CNNs can match the level of learning of one large CNN[16]. All of these points lead to a model that consumes less power and computational bandwidth from a processor while achieving *better* accuracy than the alternative benchmark design. This complements the demanding needs for edge computing and shows a feasible solution for obtaining high accuracy with neural networks through local distributed processing.
- *Quality of Experience*—Finally, we demonstrated the high accuracy of our model in a real product through a clinical trial showing how the model compensated for nuances found in the wild along the way.

We hope the advantages presented with this model architecture will be particularly leveraged by readers designing machine learning products in the medical industry. Much of the fundamental knowledge employed during the design of a single neural network is leveraged throughout our design of the CCNN. As a consequence, we hope that this paper opens up regulatory consideration of artificial intelligence products in FDA product screening by exposing the invaluable benefits AI can provide in image recognition problems. Through better model design and a focus on usability, artificial intelligence can significantly augment physicians' capabilities within the clinic. The surging demand in 5G communications complements the significant appeal of the above techniques in realizing true edge intelligence and the authors hope this helps pave the path toward more sophisticated edge devices. The authors plan to continue development of the CCNN model by broadening its scope to accommodate additional respiratory conditions. Additionally, work is underway to apply the same architecture towards the analysis of eye images to detect biomarkers for blindness, to recognition of neurological disorders in brain MRI images, and to audio recordings to detect biomarkers of potential neuropsychological conditions.

As an effort to lighten physician demand for the COVID-19 pandemic, the authors have published the application with the model used in this paper to the Apple App Store free for download under the name "COVID-AI".

References

1. Benjamens, S.; Dhunnoo, P.; Mesko, B. (2020). The state of artificial intelligence-based FDA-approved medical devices and algorithms: An online database [Abstract]. Nature. doi:https://www.nature.com/articles/s41746-020-00324-0

2. Scharre, P. (2019). Army of None. New York, NY: WW Norton & Co.

3. *Artificial Intelligence: A Modern Approach. Third Edition*. Russel, Stuart J.; Norvig, Peter. 2015, Pearson India Education Services Pvt. Ltd. p 961-962.

4. *COVID-19 Image Data Collection.* Cohen, Joseph Paul; Morrison, Paul; Dao, Lan. https://github.com/ieee8023/covid-chestxray-dataset. 12 April 2020.

5. *NIH Chest X-ray Dataset.* National Institute of Health. https://www.kaggle.com/nih-chest-xrays/data/version/1. 8 October 2019.

6. *COVID-CT Dataset: A CT Scan Dataset about COVID-19*. Zhaou, Jinyu and Zhang; He, Xuehai; Xie, Pengtao. https://github.com/UCSD-AI4H/COVID-CT. 12 April 2020.

7. Sanchez-Oro, R.; Nuez, J. T.; Martinez-Sans, G. (2020). Radiological findings for diagnosis of SARS-CoV-2 pneumonia (COVID-19). Elsevier Medicina Clinica, 155(1), 36-40. doi:https://www.sciencedirect.com/science/article/pii/S2387020620301959?via%3Dihub

8. *Deep Learning.* Goodfellow, Ian; Bengio, Yoshua; Courville, Aaron. 2016, Massachusetts Institute of Technology. p 147 -149, 525 – 527.

9. *Gaussian Processes for Machine Learning.* Rasmussen, Carl Edward; Williams, Christopher K. I. 2006, Massachusetts Institute of Technology. p 34 - 35, 90, 199 - 200.

10. *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning.* Samek, Wojciech; Montavon, Gregoire; ssVedaldi, Andrea; Hansen, Lars Kai; Muller, Klaus-Robert. 2019, Springer Nature Switzerland AG.

11. Goodfellow, I.; Shlens, J.; Szegedy, Christian. (2015). Explaining and Harnessing Adversarial Examples. Print. sArXiv 1412.6572. doi: https://arxiv.org/abs/1412.6572.

12. Hornik, K.; Stinchcombe, M.; White, H. (n.d.). Multilayer Feedforward Networks are Universal Approximators. Neural Networks, 2, 359-366. doi:https://deeplearning.cs.cmu.edu/F20/document/readings/Hornik_Stinchcombe_White.pdf

13. Hofmarcher, M.; Unterthiner, T.; Arjona-Medina, J.; Klambauer, G.; Hochreiter, S.; Nessler, B. (2019). Visual Scene Understanding for Autonomous Driving Using Semantic Segmentation. Explainable AI, 285-296. doi:https://doi.org/10.1007/978-3-030-28954-6_15

14. Samek, W.; Muller, K. (2019). Towards Explainable Artificial Intelligence. Explainable AI, 5-22. doi:https://doi.org/10.1007/978-3-030-28954-6_1

15. Apostolopoulos, I., &amp; Mpesiana, T. (2020). Covid-19: Automatic Detection from X-Ray Images Utilizing Transfer Learning with Convolutional Neural Networks. Physical and Engineering Sciences in Medicine. doi:https://doi.org/10.1007/s13246-020-00865-4

16. Ba, J.; Caruana, R.: Do deep nets really need to be deep? In: Advances in Neural Information Processing Systems, NIPS (2014).

17. McClelland, J. L.; Rumelhart, D. E. (1999). Parallel distributed processing: Explorations in the microstructure of cognition, Volume 1. Cambridge, MA: MIT Press.

18. Roeder, L. (2020, June 25). Netron. Last retrieved October 24, 2020, from https://github.com/lutzroeder/netron

19. S. Deng, H. Zhao, W. Fang, J. Yin, S. Dustdar and A. Y. Zomaya, "Edge Intelligence: The Confluence of Edge Computing and Artificial Intelligence," in IEEE Internet of Things Journal, vol. 7, no. 8, pp. 7457-7469, Aug. 2020, doi: 10.1109/JIOT.2020.2984887.

20. Kurzweil, R. (2014). *How to create a mind: The secret of human thought revealed*. London: Duckworth.

21. McClelland, J. L.; Rumelhart, D. E. (1999). Parallel distributed processing: Explorations in the microstructure of cognition, Volume 2. Cambridge, MA: MIT Press.