# Detecting Pneumonia utilizing Convolutions and Dynamic Capsule Routing for Chest X-Ray Images

Ansh Mittal[1], Deepika Kumar[2]

[1,2] Bharati Vidyapeeth's College of Engineering, New Delhi, India

anshm18111996@gmail.com, deepika.kumar@bharatividyapeeth.edu

**Abstract:** A group of neurons can be called a capsule, in which a specific type of entity existing in an image can normally be depicted by the activity instantiation vector. Recently, Multi-layered capsules, called CapsNet, have been one of the state-of-the-art for classification purposes. This research extrapolates the prowess of these algorithm to detect Pneumonia from Chest X-Rays (CXR) images. Here, the entity (as aforementioned) can help determine if CXR is normal or has pneumonia. This simple model of capsules (simple CapsNet) has been provided results comparable to best Deep Learning models used earlier. So, an integration of convolutions and capsule is performed to get two models which outperform the previous models. These models – ICC (i.e. Integration of convolutions with capsules), and ECC (i.e. Ensemble of convolutions with capsules) outperform previous models, detecting pneumonia with test accuracy of 95.33% and 95.90%, respectively. All these models had been trained, validated and tested on 5857 images from Mendeley data website.

***Keywords: Pneumonia, Chest X-Ray (CXR), Simple CapsNet, Deep Learning, ECC, ICC, L-OCT***

## 1. INTRODUCTION

Pneumonia has been one of the most pre-eminent medical disease in today's world. A total population of about 450 million (i.e. about 7 % of the world's population) gets diagnosed with pneumonia each year. Of those, a fatality rate of about 4 million annually had been observed for every year [1]. Hence, pneumonia remains a prominent cause of death of old and young in developing countries. It can be described as an inflammatory condition of lung, which affects the small air sacs (more commonly known as Alveoli) [2]. The most common clinical method for diagnosis of pneumonia have been through Chest X-Ray (CXR) images. Although this method may sound plausible and help detect pneumonia better, sometimes even expert radiologists may be peeved with the conundrum of detecting pneumonia using CXR images (such as shown in figure 1). This can lead to certain aberrations in detection of pneumonia which can result in furthering fatalities rather than ameliorating the situation. Hence, there had been a paradigm shift from manual methods to computational methods.

**Figure 1:** Chest X-Ray (CXR) image depicting Pneumonia (as can be seen by inflammatory condition of Lung).

Some of the earlier works for lung diseases used Computer-aided Design (CAD) systems for chest radiographs. Earlier, this had been accomplished through textual analysis [3-6]. This can be inferred from the CAD system of the Kun Rossman Lab, located in Chicago [7]. This CAD system divided the lung images into multiple Region of Interest (ROIs) to determine and analyse if there were any abnormalities. These further led to integration of CAD systems with Deep Neural Network (DNNs) algorithms. This had finally led to usage of pretrained Convolutional Neural Networks (CNNs) and pretrained networks such as U-Net, CardiacNet and SegNet for Pneumonia detection [8-10]. Apart from this, there have been many approaches for Reinforcement Learning and Evolutionary Algorithms to

optimize hyper parameters of DNNs. Nevertheless, these approaches have been time consuming while also requiring a high computation power, which leads to quandary of finding better algorithms for classification.

Capsule Network (more commonly known as CapsNet) [11] has been one of the algorithms that have been shaping the world in terms generative and deterministic networks. These have been much more sensitive to images than CNNs. These networks, quite literally, squash multiple convolutional layers in capsules which have been then subjected to non-linearity, quite similar to what had been done in CNNs for single layers of neurons. CapsNet have also been used in tasks such as brain tumor classification and segmentation [12]. CapsNet have also been used in other researches such as lung cancer detection [13] and blood cell classification using image segmentation pre-processing [14].

As would be clear later onwards, this research's contribution has been three-fold for performing justice to this paper's title, which have been summarized as follows.

1. A comparative analysis of machine learning and deep learning algorithms has been conducted in Table 1. This analysis depicts that among the commonly used model, the Okeke Stephan method provides one of the best accuracies.
2. A simple CapsNet has been used to check the performance for CXR images if it gets accuracy comparable to earlier best accuracy.
3. Two other models have been defined for detection of pneumonia which perform even better than simple CapsNet.

This paper is organized as follows. The next section, i.e. section 2 discusses the works done earlier for detecting pneumonia while also comparing these works. Section 3 elucidates the model architectures in detail along with the dataset and the mathematics involved in the research. Next, the section 4 enlists results obtained by the three models that have been described in section 3. Finally, conclusions have been drawn from the results obtained and briefly discussed along with the limitations in the research conducted.

# 2. RELATED WORKS

Earlier works of pneumonia detection mainly dealt with CAD systems (as aforementioned). These gave a considerably low accuracy, in comparison to DNNs and CNNs at the time. One of the CAD which helped with detection of Pneumonia had been named Pneumo-CAD [15]. This system had been tasked with similar task (as in this paper) of classifying if someone had a presence of pneumonia or had no presence of it. These were given 2 classes-PP (Pneumonia Present) or PA (Pneumonia Absent), in accordance with the CAD. Its CAD design utilized k-Nearest Neighbour (kNN) scheme and/or Haar wavelet transforms. These methods were unable to achieve any considerable results and had a high misclassification accuracy of about 20% for the method with kNN and 10% with KNN and Haar Wavelet despite the area-under-curve (AUC) value being comparatively less for receiver operating characteristic (ROC) curve. Later on, a comparative analysis of the algorithm had been conducted using Pneumo-CAD with Sequential Forward Elimination (SFE), Pneumo-CAD without SFE, and Support Vector Machine (which used SFE) gave accuracies of 66%, 70% and 77% [16]. This research also tested Naïve Bayes' algorithm and got an accuracy of 68% with it. All these works were performed and analysed for chest radiographs which have essentially been the CXR images.

A comparative study performed in [17] and works done in [18-19] (that used Self Organizing Maps (SOM) and Fuzzy C-Means) had been performed for finding the optimal model for detecting pneumonia. The algorithms-both having pretrained weights (from datasets such as ImageNet [20]) and user defined, have also been extensively used for detecting pneumonia. This can be inferred from the contributions of researchers such as those by Pranav Rajpura et al [21] and Can J Saul et al [22]. CheXNet which had been introduced by the former researchers utilized 121 layer CNN that had been trained on ChestX –ray14. Despite, the algorithm providing a relatively low accuracy of approximately 76.80% for predicting pneumonia, it had been able to classify 13 other chest diseases which could have been detected from CXR images. The latter researchers proposed a CNN architecture for early diagnosis of pneumonia and achieving an accuracy of 78.73% while outperforming earlier best accuracy of 76.8% with a much simpler architecture. Apart from these studies, there have been works [23-24] which utilized Xception, VGG16, VGG19 models [25-26] that were pretrained on ImageNet dataset. These works managed to achieve an accuracy of 82% for Xception pretrained model, 87% for VGG16 pretrained model and 92% for VGG19 model. The work done in [27] compared the proposed CNN model architecture with algorithms such as Backpropagation Neural Networks (BPNN), Competitive Neural Networks (CpNN)

and pretrained and fine-tuned networks (such as VGG16 and VGG19). Later on, Okeke Stephan et al [28] succeeded in achieving a validation accuracy of 93.73% for similar classification with the usage of their defined CNN architecture.

Other than these attempts at detecting pneumonia using CNNs, there have been many more attempts to detect pneumonia accurately [29-31]. Nonetheless, all these attempts haven't been able to outperform algorithms described in work done by Okeke Stephan et al. Here, the first attempt had been made using ResNet-152, which had been yet another pre-trained model. This model gave an accuracy of approximately 88%. The second attempt had been made using yet another pretrained model-VGG16, which achieved an accuracy of approximately 88.89% while the third attempt had been made using CNN architecture which had been batch normalized after every convolutional layer. This model also simultaneously reduced the learning rate of the model if there had been no decrease in validation loss and hence, achieved a validation accuracy of 91.55% and test accuracy of 91.02%. All these attempts were made on a very famous and commonly used pneumonia dataset CXR image dataset known as Labelled optical coherence tomography (OCT) and Chest X-Ray (from hereon, L-OCT & CXR) images [32] which has been posted on Mendeley data website. Other attempts have also been made [33-34] which were able to utilize ResNet34 to get an accuracy of approximately 91% and the next attempt by J. Erthal got an accuracy of approximately 80%.

Recently, there has been an increasing shift from CNN models to the more compact and hierarchical capsule networks (h-CapsNet). This had been made possible due to the early research pertaining to the fields of capsules which started with work done in [11] and has been advancing lately [35-38]. There have been many fields which have utilized CapsNet to get better results that were earlier not possible using ANNs, CNNs or any other detection or classification algorithm. These have been mentioned earlier and have hence been inclusive of brain tumor segmentation, lung cancer classification and blood cell classification. Apart from this, CapsNet had been recently used for detecting pneumonia along VGG16 blocks to create an integrated model [39]. Despite this integration of a pretrained model and a state of the art algorithm, the algorithm had only been able to achieve a validation accuracy of 87.5% for training accuracy of 97.18%. This model, despite overfitting, had been able to achieve a testing accuracy of 88.3%. Hence, we see that CapsNet models have been much more reliable and sensitive to detecting pneumonia from CXR images.

A holistic analysis of all the algorithms that have been aforementioned in the literature had been conducted. This had been done in terms of the accuracy they achieved on the dataset they were meant to utilize which had mostly been CXR image dataset or ChestX-ray14 dataset used in case of CheXNet. A comparative analysis of the methods discussed above has been mentioned below.

**Table 1:** Comparative Analysis of various deep learning models in accord with the literature.

| MODELS USED | ACCURACY ACHIEVED |
|---|---|
| CheXNet [21] | ~76.80 % |
| Can J Saul et al [22] CNN | ~78.73 % |
| Xception pretrained model | ~82 % |
| VGG16 pretrained model [23] | ~87 % |
| VGG19 pretrained model | ~92 % |
| R. H. Abiyev et al [27] CNN | 92.40 % |
| Okeke Stephan et al [28] CNN | 93.73 % |
| ResNet-152 | ~88 % |
| VGG16 pretrained model [30] | 88.89 % |
| A. Sagar CNN model [31] | 91.02 % |
| J. Erthal CNN model [34] | ~80 % |
| ResNet 34 | ~91 % |

| | |
|---|---|
| VGG16 + CapsNet | 88.30% (10 epochs) |

A further extrapolation of Table 1 has been done in figure 2 given below. It shows that despite that the combination of VGG16 model and CapsNet had been able to get accuracy of only 88.30%, it can still perform better as the model had been run on just 10 epochs. Hence, this depicts that CapsNet have a lot of potential to increase the accuracy of detection of pneumonia.

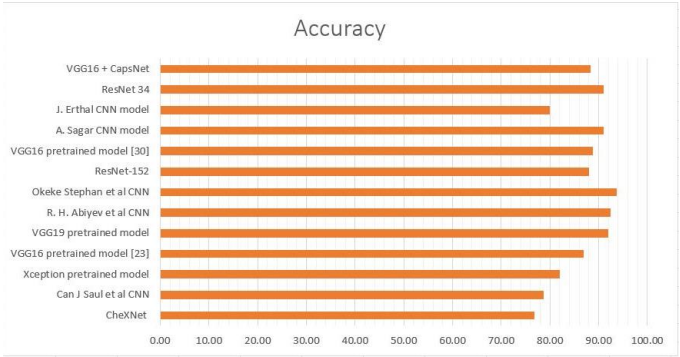


**Figure 2:** Pneumonia Detection using CXR image through deep learning models in accordance with Literature.

# 3. PROPOSED METHODOLOGY

## 3.1 Dataset Description

The dataset consists of CXR images that were taken from L-OCT and CXR image dataset [32]. This dataset consists of 5857 CXR images. These have been divided into 2 classes – "Normal" and "Pneumonia". The distribution for these images has been shown below in table 2 while the total class distribution of the dataset has been shown in the figure 3.

**Table 2:** Distribution of Pneumonia CXR images

| | TRAIN SET | VALIDATION SET | TEST SET |
|---|---|---|---|
| Normal | 1108 | 238 | 237 |
| Pneumonia | 2992 | 641 | 641 |

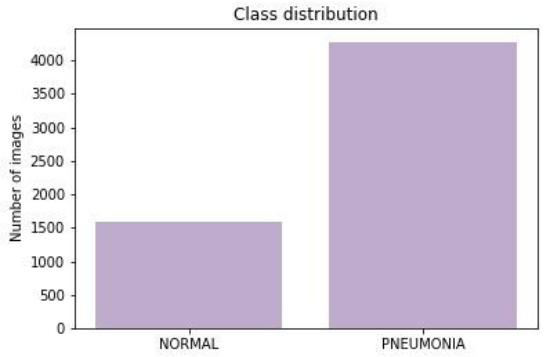


**Figure 3.** The class distribution of the complete CXR image dataset.

This depicts that there were 1583 images for "Normal" class whereas there were 4274 images for "Pneumonia". This depicts that about 27% of the data had been represented through "Normal" class while about 63% of the CXR images were having "Pneumonia". The data had not been subjected to under-sampling or oversampling. As the CXR images can be sensitive to slight changes in shearing, rotation or zooming, hence, data augmentation had been avoided in the CXR image dataset. The sample of images for each of the set have been displayed below as in figure 4.

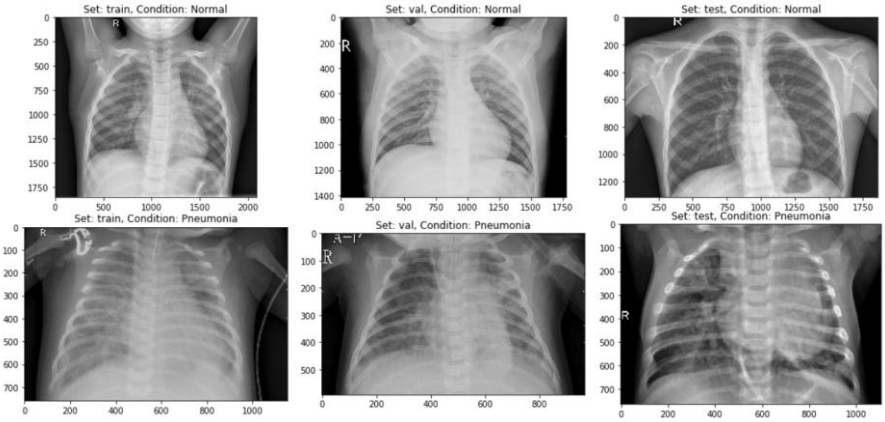


**Figure 4.** A sample of CXR images having both the classes separated into Train, Validation and Test dataset used in this research.

### 3.2 Dropouts

To reduce overfitting, the third model has been accommodated to regularization known as dropout which had been employed after pooling layers in one of the dense convolution subnetworks of the model. Dropout had been set to 0.5 (as illustrated later in figure 9), which depicts that the weights of the half of the neurons have randomly been set to zero while remainder give original inputs multiplied by weights. This ensures that the neurons in one of the convolutional model had been independent of the other CNN in ensemble of CNNs.

### 3.3 Model Architectures

Initially, a CapsNet [11] architecture had been used (as shown in figure 5) to check if the model could predict pneumonia on par with the work done in [28] where 93.73 % accuracy had been achieved. This simple architecture has only two convolutional layers in addition to one fully connected layer. The ReLU Conv1 had 100, 3×3 kernels of convolutions with single stride for feature vector of 98×98. As can be seen from name of the layer, Conv1 introduces non-linearity using Rectified Linear Unit (ReLU) activation function [40]. Here, values of local features have been obtained using the pixel intensities of 100×100 (resized as number of parameters exceeded 40M with 200×200) CXR images, which have then been used in primary capsules (in layer named Primary Caps). The function for ReLU has been discussed below and the graph for the function has been plotted in figure 5(a) using Numpy [41] and Matplotlib [42].

$$ReLU\ (z) = \begin{cases} z, & z \geq 0 \\ 0, & z < 0 \end{cases} \tag{1}$$

As has been explicable from the equation, the ReLU function uses half rectified output to introduce non-linearity to local features. It can be seen from equation (1) to be nullifying feature vectors with negative values while retaining the positive local features. It, along with its derivatives can be proved to be monotonic in nature. It has been used while routing the feature vectors from convolutions to the initial capsule layer. The equation for its derivative has been given in equation (2).

$$ReLU'(z) = \begin{cases} 1, & z \geq 0 \\ 0, & z < 0 \end{cases} \tag{2}$$

In both equation (1) and (2), $z$ demonstrates the local feature vector that had been extracted from the local pixel intensity values from CXR images. Figure 5(a-b) demonstrate the monotonicity of both the ReLU function and its derivative. This means that both the functions only increase or remains the same when local feature vectors increase, and decrease or remains same when the vectors' values decreases.

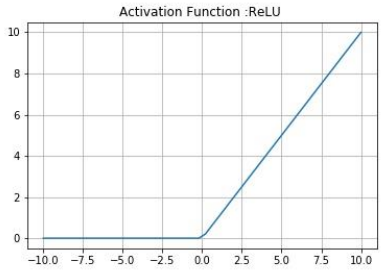


**Figure 5(a).** A representation of ReLU (Rectified Linear Unit) activation unit function

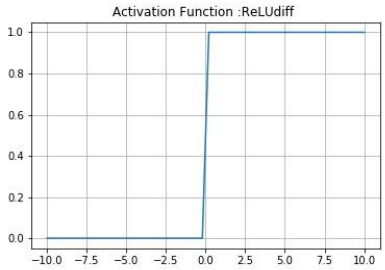


**Figure 5(b).** A representation of Sigmoid activation unit function

The Primary Caps i.e. the second layer have multi-dimensional entities that can be placed at the lowest level, hierarchically. This layer when activated, corresponds to the inverse process of rendering an image. This second layer is a convolutional layer having 16 channels for convolutional 8-D capsules (so every primary capsule contributes 8 units similar to feature maps in CNNs, with 3×3 kernels and 2 as stride). Each of these primary capsules has been provided with a 100×9 units from ReLU Conv1. The Primary Caps have 16×48×48 outputs from the capsules (where each output is 8-D). Here, each capsules in 48×48 feature map share their weight with each other. For this block the non-linearity used has been provided by the function given below in equation (3).

$$v_j = \frac{\|s_j\|^2}{1 + \|s_j\|^2} \frac{s_j}{\|s_j\|} \tag{3}$$

where $s_j$ is the total input vector while $v_j$ is output vector from capsule $j$. Here, the probability of entity being present in the input vector, that the capsule represents has been depicted by the length of capsule. This function ensures that short vector get shrunk to approximately zero while large vector get shrunk to length that has been slightly less than unity and has hence, been coined **squashing** function [11]. The final layer (output capsule layer (or OutputCaps)) has 16-D capsule unit for both "Normal" and "Pneumonia" classes (i.e. 16×2 vector). As there have been only two capsule layers (Primary Caps and OutputCaps) which have been consecutive, hence, routing has just done between these two. As the output from ReLU Conv1 can just be 1-D, there have been no orientation in which routing can be done here. The routing algorithm has been briefly explained below in accordance with [11].

Every capsule layer except the first capsule layer (here, Primary Caps), has $s_j$ (of equation (3)) as a weighted sum over all predictions of entity $\alpha_{j|i}$ from the capsule layer below. Value of $s_j$ is calculated by multiplying output $\alpha_i$ of capsule by the matrix of weight $\mathbf{W}_{ij}$. This has been given in the equation (4) below.

$$s_j = \sum_i k_{ij}.\alpha_{j|i}\,, \qquad \alpha_{j|i} = \mathbf{W}_{ij}.\alpha_i \tag{4}$$

where iterative dynamic routing can be described using coupling coefficients, $k_{ij}$. The coupling coefficients between capsules above the layer having capsule $i$ and itself sum well above unity which has been determined by softmax (which can hence be called routing softmax, and has been given in equation (5)) whose initial class units $b_{ij}$ are log a priori probabilities that capsule $i$ and capsule $j$ have been coupled.

$$k_{ij} = \frac{e^{b_{ij}}}{\sum_1^K e^{b_{ik}}} \tag{5}$$

The log a priori values can be learned simultaneously as all other weights. This results in them depending on locations and type of two capsule layers instead of the current CXR image (i.e. even if current CXR image is disoriented, these routings work well). This signifies that the CapsNet model follows equivariance instead of invariance for the orientation of objects in the images. The whole routing algorithm has been explained more intricately in [11]. First all the class units have been initialized to zero values, after which a capsule output ($\alpha_i$) is sent to both parent capsules ($v_0$ and $v_1$; or which has been determined by number of classes) with equal probabilities ($k_{ij}$). The final architecture had been created

using Keras [43] and Tensorflow API [44]. The compilation of this model had been done using RMSProp optimizer [45] for mean square error (MSE) and margin loss (which has been given in equation (6) below).

$$L_k = T_k \max(0, m^+ - ||v_k||)^2 + \lambda (1 - T_k) \max(0, ||v_k|| - m^-)^2 \quad (6)$$

As explained earlier, the length of the vector represents the possibility of a capsule's entity to exist in the pre-squashed vector. The top level capsule has been used for output class to have long instantiation vector (in this model, it had been 16-D) only if the class represented in the vector has been denoted by the image. As this is a binary classification problem, $T_K$ represents if the image is "Pneumonia" or "Normal" and $m^+$ and $m^-$ will be 0.5, respectively. The down-weighting factor $\lambda$ puts an impasse to shrinking the activity vector for "Normal" or "Pneumonia" in case "Pneumonia" or "Normal" has been depicted by the receptive fields from the capsules, respectively. Total loss can have said to be the loss of both "Normal" and "Pneumonia" capsules. Finally, $Y_{out}$ has been used for determining the probability of an CXR image to be "Normal" or showing "Pneumonia". These values as mentioned above have been obtained by the application of equation (5).

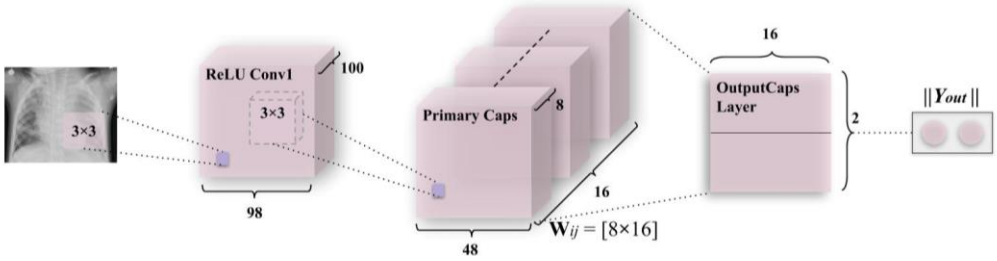


**Figure 6(a).** A CapsNet having 3 layers (1 convolutional layer and 2 capsule layers). This simple model is comparable to deep CNN (such as Okeke Stephan et al [28]). The length of class instantiation activity vector of both capsule in "Normal" and "Pneumonia" layer indicates presence of both classes which is used to calculate classification loss. As discussed above, weight matrix $\mathbf{W}_{ij}$ is a matrix of weights between both $\alpha_i$, $i \in (1, 16\times48\times48)$ in Primary Caps layer and $v_j$, $j \in (1,2)$ where 1 represents "Normal" and 2 represents "Pneumonia".

To encourage the class capsules to encode parameters of the input "Normal" and "Pneumonia" CXR images, a reconstruction layer has been added. This has been done in order to counter the effects of attacks such as the one performed in [46] when one pixel had been perturbed to disrupt the classification performed by DNN. Although this had been performed on CIFAR-10 dataset [47], such types of security threats if done on CXR images can cause fatalities in case of prediction of pneumonia or any other disease. An instance of this can be observed when a single pixel is perturbed, and it causes a CXR scan of "Pneumonia", to be detected as "Normal", which can ultimately lead to a fatality. Hence, while training this layer, all layers except that of third layer (also, second capsule layer), have been masked. From this layer, activity instantiation vector is utilized to reconstruct the CXR image and finally all parameters are backtracked to encode the pixel intensity values which makes model more robust to aforementioned pixel attacks. The output of this layer along with the 2 classes mentioned earlier, has been fed into decoder consisting of fully connected layers that model pixel intensities as has been represented in figure 6(b). The sum of squared differences between logistic unit output and intensity of pixels of actual image, is minimized by a very small amount, so, the margin loss is not obscured during this process, because the main task of this network is to detect pneumonia, rather than to work as generative model. This encoding-cum-reconstruction layer has been depicted below in figure 6(b).

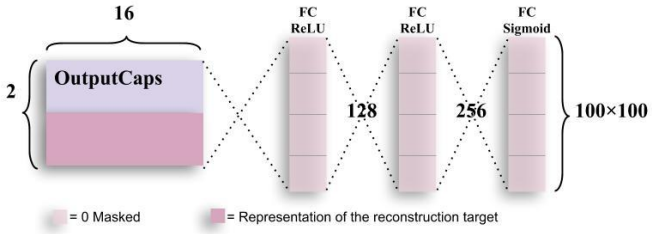


**Figure 6(b).** Decoder structure for reconstruction of CXR images from the class representation. The mean square error is minimized during the training phase of the network. Labels have also been used during reconstructing CXR images during training. The decoder has comparatively subservient role to the CapsNet in this research as it is just used for encoding of the

activity vector. In this research, the decoder will mostly be stuck in a local optimum which leads to a lower accuracy of the image.

The first two fully connected layers use ReLU non-linearity which has been explained earlier using equation (1), (2) and figure 5(a-b). These layers have 128 and 256 neurons respectively. They take an activity instantiation vector of 16×2 (where 2 is no. of classes) which is given from the third layer of the CapsNet architecture explained above. In addition to this, another 2-D vector, that states if the class is "Normal" or "Pneumonia" has been augmented to the activity vector. Finally, the last fully connected layer can give the output image using sigmoid function [48] given in equation (7) which has been explained briefly.

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{7}$$

This function takes all real values and output values between 0 and 1. It asymptotes at 0 and 1 for values of $-\infty$ and $\infty$, respectively. At $z = 0$, this function gives a value of 0.5. The sigmoid function has been depicted in figure 7(a). It is S-shaped curve which introduces no binary activations (such as those in step functions) and hence remains one of the most prominent non-linearity for binary classification (in form of "Normal" or "Pneumonia" CXR images). Here, the sigmoid gives a probability pixel for each masked vector values (augmentation of activity vector and class of that vector). This function is monotonic, but, its derivative is not monotonic as described in equation (8) and figure 7(b).

$$\sigma'^{(z)} = \sigma^{(z)}\left(1 - \sigma^{(z)}\right), \qquad or \qquad \frac{e^{-z}}{(1 + e^{-z})^2} \tag{8}$$

In both equation (7) and (8), $z$ demonstrates the local feature vector that had been extracted from the local pixel intensity values from CXR images. Figure 7(a-b) demonstrate the monotonicity and non-monotonicity of the sigmoid function and its derivative, respectively. This means that the function only increase or remains the same when local feature vectors increase, and decrease or remains same when the vectors' values decreases for sigmoid and vice versa can be proven true for derivative of the sigmoid function.

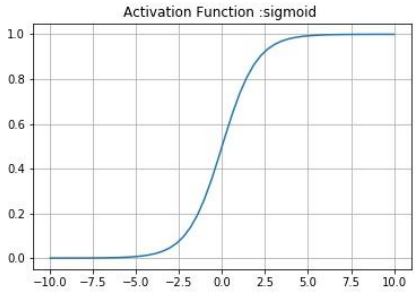


**Figure 7(a).** A representation of ReLU (Rectified Linear Unit) activation unit function

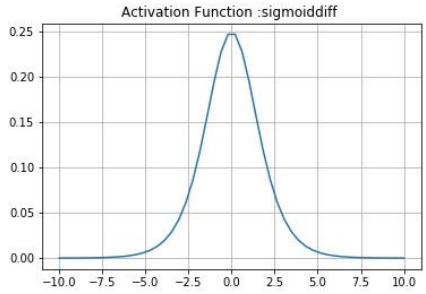


**Figure 7(b).** A representation of Sigmoid activation unit function

The second model – Integration of convolutions with capsules (ICC) depicted in figure 8, used for detecting pneumonia, has been a combination of convolutions with capsule layers among which (simple capsules in the bottom of figure 8), the routing occurs. This network has two subnetworks – the dense convolution neural subnetwork (or also known as dCNS), and the dynamic routing subnetwork (or also known as DRS).

The dCNS in ICC model consists of 4 convolutions which also have 4 maxpool layers in the formers correspondence. These convolutions uniformly introduce non-linearity using ReLU activation function (described in equation (1-2) and figures 5(a-b)) using a kernel of size 3×3, while maxpool is uniformly imposed on pool size of 2×2. The ReLU Conv1 layer has 32 feature maps which utilize 198×198 feature vectors from these, which have later been subjected to maxpooling to get 99×99 feature vectors. These 2 layers have been integrated into one block for the sake of simplicity as has been depicted in the figure. This process has been repeated through ReLU Convolutional layers and Max_pool2d layers (through ReLU Conv2, Max_pool2d_2, ReLU Conv3, Max_pool2d_3, ReLU Conv4, Max_pool2d_4)

in order to obtain feature maps of vectors having 64×97×97 and 64×48×48 (2nd convolution-maxpool block), 128×46×46 and 128×23×23 (3rd convolution-maxpool block), and 128×21×21 and 128×10×10 (4th convolution-maxpool block) in the same order.

The DRS – the other part of the ICC, consists of the simple CapsNet explained earlier, albeit, with a shrunken activity instantiation vector due to earlier convolutions on the CXR images. Here, ReLU Conv5 consists of feature vectors of dimensions 8×8. This is then mapped to Primary Caps layer which has 16 channels of 8-D capsules (so every primary capsule contributes 8 units, with 3×3 kernels and 2 stride). Each of these primary capsules has been provided with a 128×9 units from the ReLU Conv5 layer. The Primary Caps have 16×6×6 outputs from the capsules (where each output is 8-D). Here, each capsule in 6×6 feature map share their weight with each other for equivariance to the perceptive fields of capsules. For this block the non-linearity remains the same as has been provided by the function in equation (3). Apart from this, all the configuration for the capsule layers (i.e. their routing, their margin loss, etc.) remains as had been used in the simple CapsNet discussed earlier. Finally, as earlier, $Y_{out}$ has been used for determining the probability of an CXR image to be "Normal" or showing "Pneumonia".

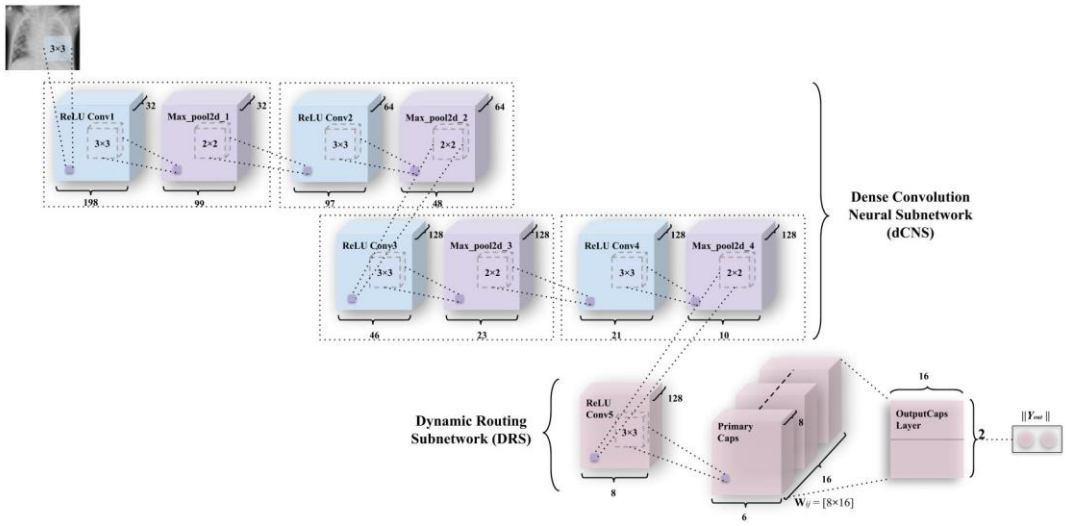


**Figure 8.** An integration of convolutions and capsules which has been named ICC (Integration of convolutions with capsules). This model outperforms earlier simple capsule model (in figure 6(a)). As discussed earlier, weight matrix $\mathbf{W}_{ij}$ has been a matrix of weights between both $\alpha_i$, $i \in (1, 16\times6\times6)$ in Primary Caps layer and $v_j$, $j \in (1,2)$ where 1 represents "Normal" and 2 represents "Pneumonia".

As mentioned in simple CapsNet earlier, to promote the class capsules to encode parameters of the CXR "Normal" and "Pneumonia" images, a reconstruction layer has been added as in the earlier model. The reason for the encoding these activity instantiation vector has also been discussed earlier. Hence, the encoding-cum-reconstruction layer that had been depicted in figure 6(b) remains the same for ICC model as well. This model has been inspired from model such as [50-51] which utilize capsules in conjunction with convolutions with or without addition to a feedback, respectively.

The third and the final model – Ensemble of convolutions with capsules (ECCs) presented in figure 9, has been a combination of ensemble of convolutions similar to the earlier model, except for the ensemble of convolutions that have been introduced in this layer. This network also has two subnetworks – the ensemble dense convolution neural subnetwork (known as EdCNS), and the dynamic routing subnetwork (known as DRS).

The EdCNS in ECC model consists of 8 convolutions to which, the 6 maxpool layers correspond to (divided among two ensembles of 4 convolutions and 3 maxpooling). These convolutions uniformly retain the configuration (non-linearity, kernel size, and pool size) from ICC model. Here, one naming technique to note is that **(e_1_)** refers to the first set of convolutions in ensemble, and **(e_2_)** refers to the second set of convolutions in ensemble. The e_1_ReLU Conv1 layer has 32 feature maps which utilize 198×198 feature vectors from these, which have then been subjected to maxpooling to get 99×99 feature vectors. The figure 9 combines both of these layers into one block as had been done in ICC. This process is repeated through ReLU Convolutional layers and Max_pool2d layers (through e_1_ReLU Conv2, e_1_Max_pool2d_2, e_1_ReLU Conv3, e_1_Max_pool2d_3) in order to obtain feature maps of vectors having 64×97×97 and 64×48×48 (2nd block of convolution-maxpool), 128×46×46 and

128×23×23 (3rd block of convolution-maxpool), and finally 128×21×21 through e_1_ReLU Conv4. In the second set of convolutions, the e_2_ReLU Conv1 layer has 16 feature maps which utilize 198×198 feature vectors from the last layer, which have again been subjected to maxpooling to get 99×99 feature vectors. Then a dropout (e_2_Dropout_1) of 0.5 is applied (as had been mentioned in section 3.2). The figure 9 combines all 3 – convolutional, maxpool, dropout layers (through e_2_ReLU Conv2, e_2_Max_pool2d_2, e_2_Dropout_2, e_2_ReLU Conv3, e_2_Max_pool2d_3, e_2_Dropout_3), in order to obtain feature maps of vectors having 32×97×97 and 32×48×48 (2nd block of convolution-maxpool-dropout), 64×46×46 and 64×23×23 (3rd block of convolution-maxpool-dropout), and conclusively 128×21×21 using e_2_ReLU Conv4, in order.

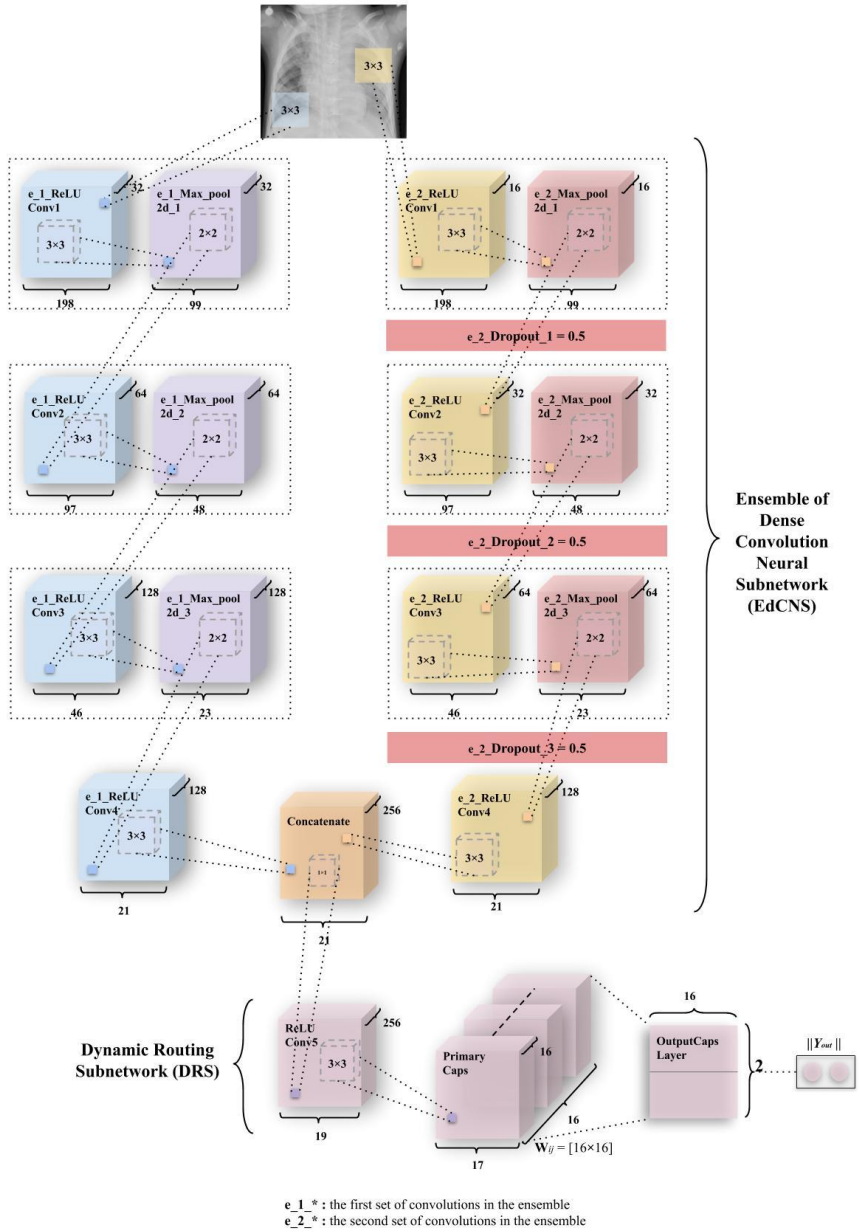


**Figure 9.** An integration of convolutions and capsules which has been named ECC (Ensemble of convolutions with capsules). This model outperforms earlier simple CapsNet model (in figure 6(a)) and comparable to ICC (depicted in figure 8). As discussed earlier, weight matrix $\mathbf{W}_{ij}$ is a matrix of weights between both $\alpha_i$, $i \in (1, 16\times17\times17)$ in Primary Caps layer and $v_j$, $j \in (1,2)$ where 1 represents "Normal" and 2 represents "Pneumonia".

The output from the e_1_ReLU Conv4 and e_2_ReLU Conv4 is merged using "concatenate" function of the keras package similar to [49] where instead of concatenating ReLU units, this model concatenates different feature maps and dropouts. The DRS – the next part of ECC, consists of the simple CapsNet explained earlier. This is then mapped to Primary Caps layer which has 16 channels of 8-D capsules (same configuration as in ICC). Each of these primary capsules has been provided with 256×9 units through ReLU Conv5. The Primary Caps have 16×17×17 outputs from the capsules where each

output is 16-D (which is the only difference in first two models and this model in terms of capsule layers). Here, each capsules in 17×17 feature map behave as explained earlier and the configuration for the capsules (i.e. their routing, their margin loss, etc.) remains as had been used in the simple CapsNet and ICC model. Conclusively, $Y_{out}$ has been used for determining "Normal" or "Pneumonia" from CXR images. Similar to the ICC, ECC boast the decoder network (figure 6(b)) for the same reason of encoding activity instantiation vectors so they can be resistant and robust to one pixel attacks discussed earlier. This is also done as a regularization technique for CapsNet. But apart from their purpose to resist such attacks, they haven't been optimized here to generate "Pneumonia" or "Normal" CXR images. This remains true for all three models explained above. This model has been inspired by other integrated model such as those represented through Multi-Lane Capsule Networks [52], and integrated stacking models such as those represented by [53] where capsules act as the level-1 meta learners from level-0 learners which have been the ensembles of convolutions.

# 4. RESULTS ANALYSIS

These three models discussed above had been trained on a set of 4100 images, validated on 879 images and had been tested with 878 images. Figure 10(a-e) represents the training curve for the simple CapsNet. These figures are representative of the CapsNet accuracy, CapsNet loss, Decoder accuracy, Decoder loss, and the total loss (sum of all losses) from figure 10(a-e) for simple CapsNet, in order. All these figures had been created using the history of training from simple CapsNet. The packages used were Matplotlib [42], Keras [43] and Seaborn [54]. This model converged after 161 epochs and had CapsNet validation accuracy of 93.75% (hence being comparable to model in [28]). The other values have also been listed in Table 3 given below.

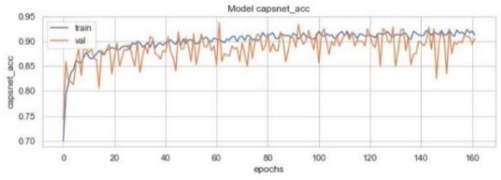


**Figure 10(a).** Validation (in orange) & Training Accuracy (in blue) during the training of CapsNet for the classification of CXR images.

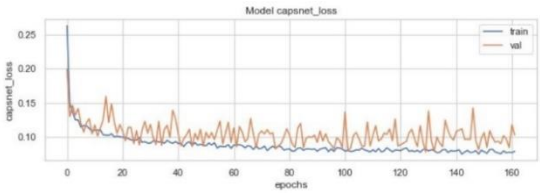


**Figure 10(b).** Validation (in orange) & Training Loss (in blue) during the training of CapsNet for the classification of CXR images.

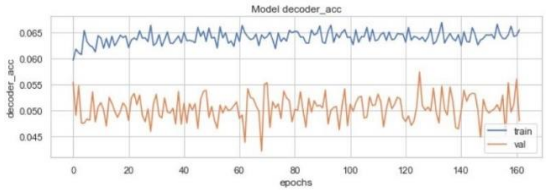


**Figure 10(c).** Validation (in orange) & Training Accuracy (in blue) during the training of Decoder for the classification of CXR images.

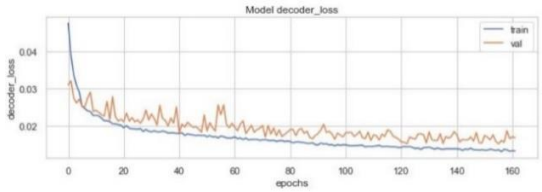


**Figure 10(d).** Validation (in orange) & Training Loss (in blue) during the training of Decoder for the classification of CXR images.

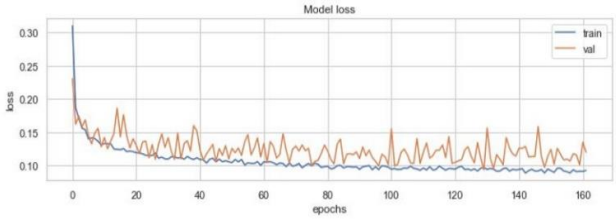


**Figure 10(e).** Validation (in orange) & Training Loss (in blue) during the training of CapsNet + Decoder for the classification of CXR images.

Figure 11(a-e) represents the training curve for the ICC model described earlier. These figures are representative of the CapsNet accuracy, CapsNet loss, Decoder accuracy, Decoder loss, and the total loss (sum of all losses) from figure 11(a-e) for ICC model, in order. All these figures had been created using the history of training from ICC. This model converged after 100 epochs and had CapsNet validation accuracy of 95.22%. The other values have also been listed in Table 3 given below. It's important to mention here that the CapsNet accuracy and all other terms mentioned here, have CapsNet because the whole architecture having capsule layers is usually divided into 2 different networks which can help encoding of data through activity instantiation vector. So, "simple CapsNet" discussed above and the "CapsNet accuracy" discussed here and later onwards don't refer to the first model's accuracy.

Rather it's the accuracy of the discriminative model and term "Decoder accuracy" is generative model's accuracy.

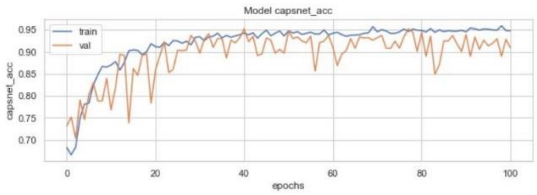


**Figure 11(a).** Validation (in orange) & Training Accuracy (in blue) during the training of ICC model's CapsNet for the classification of CXR images.

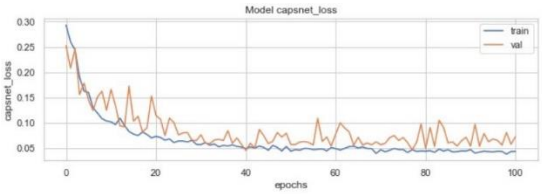


**Figure 11(b).** Validation (in orange) & Training Loss (in blue) during the training of ICC model's CapsNet for the classification of CXR images.

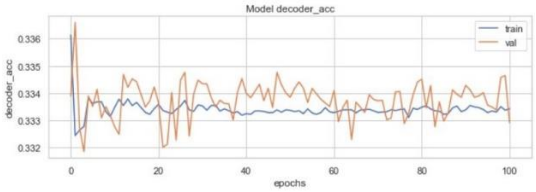


**Figure 11(c).** Validation (in orange) & Training Accuracy (in blue) during the training of ICC model's Decoder for the classification of CXR images.

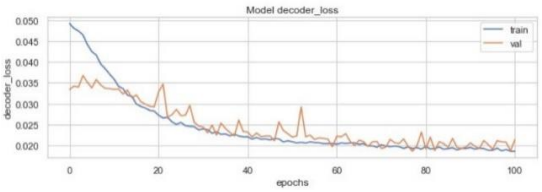


**Figure 11(d).** Validation (in orange) & Training Loss (in blue) during the training of ICC model's Decoder for the classification of CXR images.

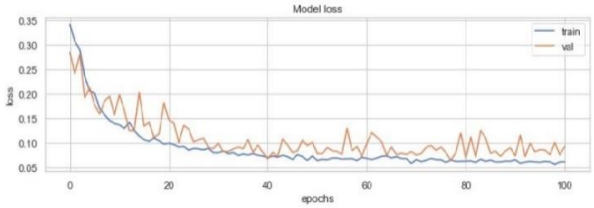


**Figure 11(e).** Validation (in orange) & Training Loss (in blue) during the training of ICC model's CapsNet + Decoder for the classification of CXR images.

Figure 12(a-e) represents the training curve for the ECC model described earlier. These figures are representative of the CapsNet accuracy, CapsNet loss, Decoder accuracy, Decoder loss, and the total loss (sum of all losses) from figure 12(a-e) for ECC, in order. All these figures had been created using the history of training from ECC. This model converged after 149 epochs and had CapsNet validation accuracy of 95.41% (just slightly higher than that of ICC model). The other values have also been listed in Table 3 given below.

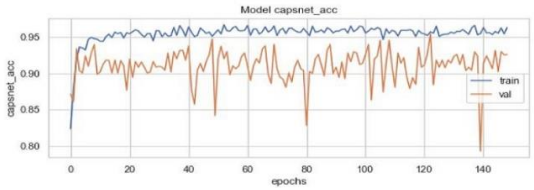


**Figure 12(a).** Validation (in orange) & Training Accuracy (in blue) during the training of ECC model's CapsNet for the classification of CXR images.

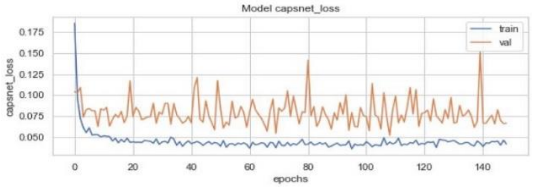


**Figure 12(b).** Validation (in orange) & Training Loss (in blue) during the training of ECC model's CapsNet for the classification of CXR images.

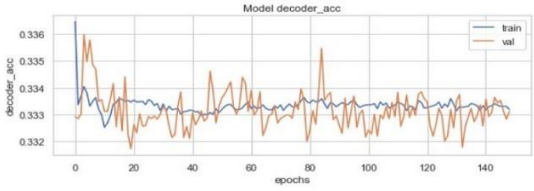


**Figure 12(c).** Validation (in orange) & Training Accuracy (in blue) during the training of ECC model's Decoder for the classification of CXR images.

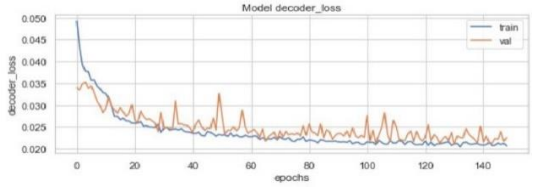


**Figure 12(d).** Validation (in orange) & Training Loss (in blue) during the training of ECC model's Decoder for the classification of CXR images.

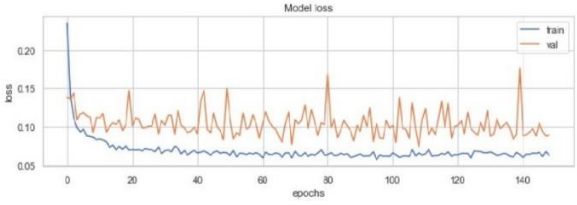

**Figure 12(e).** Validation (in orange) & Training Loss (in blue) during the training of ECC model's CapsNet + Decoder for the classification of CXR images.

The plot of confusion matrix for all 3 of the models whose training history has been mentioned earlier in figure 10(a – e), figure 11(a – e), and figure 12(a – e) for simple CapsNet, ICC, ECC respectively have been given in figure 13(a – c). It can be observed that the simple CapsNet model (figure 13(a)) has 35 false negatives (FNs) (~5.46% of "Pneumonia" CXR images) from the CXR test set of 878 images. In pneumonia detection, FNs can prove to be fatal as someone who has pneumonia may be denied of its possibility and hence, may get the wrong treatment for it. On the other hand, this model has 18 false positives (FPs) (~7.59% of "Normal" CXR images) which have otherwise been normal CXR images predicted to have pneumonia.

The ICC model, discussed earlier has its confusion matrix in figure 13(b). It classifies 625 instances of pneumonia positively (~97.5%) and has 16 false negatives (FNs) (~2.5% of "Pneumonia" CXR images) from the CXR test set of 878 images. On the other hand, this model has 25 false positives (FPs) (~10.55% of "Normal" CXR images) which have otherwise been normal CXR images predicted to have pneumonia. It may be observed that although the amount of FNs decrease, the number of FPs have increased but FPs may not be as harmful as FNs. This is due to the fact that someone who doesn't have pneumonia and has got a positive test, would be subjected to medications which would otherwise only get rid of pneumonia. So, the amount of FPs is acceptable within a certain scope until and unless patients aren't suffering from different diseases. A more detailed study should be conducted if the aforementioned case is true.

Finally, the plot of confusion matrix for ECC has been given in figure 13(c). This model has further improved the test accuracy of the Pneumonia dataset and has also led to reducing the number of FNs to 14 (i.e. ~2.18% of "Pneumonia" CXR images) and 22 false positives (FPs) (~10.55% of "Normal" CXR images) It also has less misclassified points, better CapsNet validation accuracy compared to simple CapsNet model and ICC model as has been discussed with reference to Table 3.

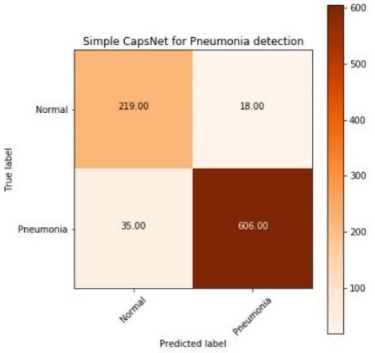


**Figure 13(a).** A plot of confusion matrix for detection of Pneumonia using simple CapsNet.

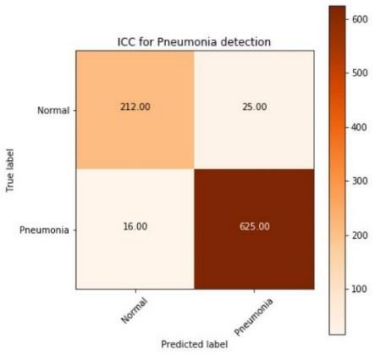


**Figure 13(b).** A plot of confusion matrix for detection of Pneumonia using ICC (i.e. Integration of convolutions with capsules) model.

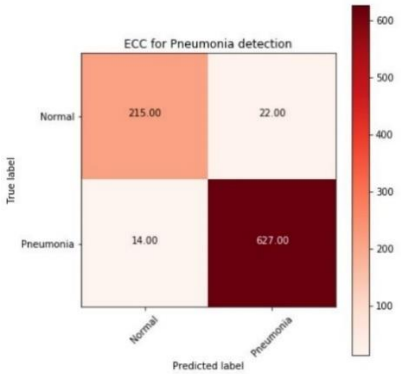


**Figure 13(c).** A plot of confusion matrix for detection of Pneumonia using ECC (i.e. Ensemble of convolutions with capsules) model (best model with least FNs).

A detailed comparison of all three of the models that have been implemented has been done in Table 3, which has been given below. Note that ECC has been the best models amongst the models mentioned here. This has been due to the fact that 2 convolutions compute the weights to be given to the

concatenation layer which is then routed through the capsules, as discussed earlier in section 3. The calculation of features from both the convolutions make ICCs more robust and increases its equivariance to spatial correlation. If the model had had more number of convolutions, it may work even better than it currently does.

**Table 3:** A holistic comparison of different models used for the detection of Pneumonia from CXR images. These algorithms have been compared on basis of CapsNet accuracy (CapsNet $Acc_{TRAIN}$, CapsNet $Acc_{VALID}$, CapsNet $Acc_{TEST}$), CapsNet loss (CapsNet $Loss_{TRAIN}$, CapsNet $Loss_{VALID}$), Decoder accuracy (Decoder $Acc_{TRAIN}$, Decoder $Acc_{VALID}$), Decoder loss (Decoder $Loss_{TRAIN}$, Decoder $Loss_{VALID}$), and the total loss (sum of all losses) ($Loss_{TRAIN}$, $Loss_{VALID}$).

| MODEL NAME | CapsNet $Acc_{TRAIN}$ | CapsNet $Acc_{VALID}$ | CapsNet $Acc_{TEST}$ | CapsNet $Loss_{TRAIN}$ | CapsNet $Loss_{VALID}$ | Decoder $Acc_{TRAIN}$ | Decoder $Acc_{VALID}$ | Decoder $Loss_{TRAIN}$ | Decoder $Loss_{VALID}$ | $Loss_{TRAIN}$ | $Loss_{VALID}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Simple CapsNet | 90.25 % | 93.75 % | 93.96 % | 0.0888 | 0.0858 | 6.63 % | 4.91 % | 0.0167 | 0.0188 | 0.1055 | 0.1046 |
| ICC Model | 94.29 % | 95.22 % | 95.33 % | 0.0482 | **0.0448** | 33.32 % | 33.4 % | 0.0221 | 0.0232 | 0.0702 | **0.068** |
| ECC Model | **95.70 %** | **95.41 %** | **95.90 %** | **0.0432** | 0.0618 | **33.33 %** | **33.42 %** | **0.0213** | **0.0215** | **0.0647** | 0.0873 |

An exegesis of the results suggests that ICC model converges the fastest for the dataset described in section 3. Despite this and its low value of loss (CapsNet $Loss_{TRAIN}$), it has a comparatively lower accuracy (CapsNet $Acc_{TRAIN}$, CapsNet $Acc_{VALID}$ and CapsNet $Acc_{TEST}$). Here, we can see that one of the best models in terms of accuracy (CapsNet $Acc_{TRAIN}$, CapsNet $Acc_{VALID}$ and CapsNet $Acc_{TEST}$) is the ECC model. Apart from this, a mention of Decoder Accuracy (Decoder $Acc_{TRAIN}$ and Decoder $Acc_{VALID}$) and Loss (Decoder $Loss_{TRAIN}$ and Decoder $Loss_{VALID}$) is necessary over here. It's important to note that this work has been a classification task and hence used decoder for encoding and regularizing the activity instantiation vector which is then sent back through receptive fields of the capsule layer above the connected capsule layer. As, all the models mentioned used capsule layers here, the decoder didn't have to be having complex architecture which led to the lower accuracy of the Decoder network.

# DISCUSSIONS AND CONCLUSIONS

The models introduced in this research, ICC and ECC achieved a training accuracy (CapsNet $Acc_{TRAIN}$) of 94.29% and 95.70%, and training loss (CapsNet $Loss_{TRAIN}$) of 0.0482 and 0.0432, respectively on set of 4100 images. These models had been robust to overfitting and pixel attacks using decoder and achieved a validation accuracy ($ACC_{VALID}$) of 95.22% and 95.41%, and validation loss (CapsNet $Loss_{VALID}$) of 0.0448 and 0.0618 on a set of 879 images for ICC and ECC, respectively. These models have been tested on yet another 878 images and achieved an accuracy ($ACC_{TEST}$) of 95.33% and 95.90% as has been observed from confusion matrices in figure 13(b-c). Both of these models combined convolution operations with capsule routing to achieve these results, the only difference being that one model combined one set of convolutions, while the other combined two sets of convolutions. All three models (simple CapsNet, ICC, and ECC) had been trained on the ratio of 14:3:3 for train, validation, a test sets of images (with 5857 being the total number of images).

Although both of these model achieved a higher testing and validation accuracy and lower validation loss compared to the models mentioned earlier in table 2, they still suffer from some drawbacks. This has been ascribed to the fact that all these models aren't able to efficiently utilize their generative capabilities to produce accurate computer generated images (CGIs) for CXR from the given set of images, even though the generative capabilities of the models lie outside the scope of this research. This work can further be extended through the utilization of a more robust generative model (or decoder) to generate high quality images for CXR to get better generative capabilities. Some more areas where this work can be extended have been discussed below. ICCs and ECCs used convolutions and capsules that had been generated through trial-and-error method which is not an efficient means for defining a good classification or generative model. It's important to mention that genetic algorithms can be employed to design an optimal blend of convolutions for classification purposes [54-55]. Along with this, generation of optimal capsule layers can also be secured through genetic algorithm (just as in case of DNNs) [56-57] or other evolutionary strategies [58] such as CMA-ES (Covariance Matrix Adaptation Evolution Strategy) [59] and PEPG (Parameter-Exploring Policy Gradients) [60] for which research is being conducted.

# REFERENCES


[1] Ruuskanen, O., Lahti, E., Jennings, L. C., & Murdoch, D. R. (2011). Viral pneumonia. *The Lancet, 377*(9773), 1264-1275.

[2] McLuckie, A. (Ed.). (2009). *Respiratory disease and its management.* Springer Science & Business Media.

[3] Katsuragawa, S., Doi, K., & MacMahon, H. (1989). Image feature analysis and computer-aided diagnosis in digital radiography: Classification of normal and abnormal lungs with interstitial disease in chest images. *Medical physics, 16*(1), 38-44.

[4] Kido, S., Ikezoe, J., Naito, H., Tamura, S., & Machi, S. (1995). Fractal analysis of interstitial lung abnormalities in chest radiography. *Radiographics, 15*(6), 1457-1464.

[5] Ishida, T., Katsuragawa, S., Kobayashi, T., MacMahon, H., & Doi, K. (1997). Computerized analysis of interstitial disease in chest radiographs: Improvement of geometric-pattern feature analysis. *Medical Physics, 24*(6), 915-924.

[6] Loog, M., van Ginneken, B., & Nielsen, M. (2004, May). Detection of interstitial lung disease in PA chest radiographs. In *Medical Imaging 2004: Physics of Medical Imaging* (Vol. 5368, pp. 848-855). International Society for Optics and Photonics.

[7] Abe, H., MacMahon, H., Shiraishi, J., Li, Q., Engelmann, R., & Doi, K. (2004, October). Computer-aided diagnosis in chest radiology. In *Seminars in Ultrasound, CT and MRI* (Vol. 25, No. 5, pp. 432-437). WB Saunders.

[8] Ronneberger, O., Fischer, P., & Brox, T. (2015, October). U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention* (pp. 234-241). Springer, Cham.

[9] Badrinarayanan, V., Kendall, A., & Cipolla, R. (2017). Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence, 39*(12), 2481-2495.

[10] Mortazi, A., Karim, R., Rhode, K., Burt, J., & Bagci, U. (2017, September). CardiacNET: segmentation of left atrium and proximal pulmonary veins from MRI using multi-view CNN. In *International Conference on Medical Image Computing and Computer-Assisted Intervention* (pp. 377-385). Springer, Cham.

[11] Sabour, S., Frosst, N., & Hinton, G. E. (2017). Dynamic routing between capsules. In *Advances in neural information processing systems* (pp. 3856-3866).

[12] Afshar, P., Plataniotis, K. N., & Mohammadi, A. (2019, May). Capsule Networks for Brain Tumor Classification Based on Mri Images and Coarse Tumor Boundaries. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 1368-1372). IEEE.

[13] Mobiny, A., & Van Nguyen, H. (2018, September). Fast capsnet for lung cancer screening. In *International Conference on Medical Image Computing and Computer-Assisted Intervention* (pp. 741-749). Springer, Cham.

[14] Zhang, X., & Zhao, S. (2019). Blood Cell Image Classification Based on Image Segmentation Preprocessing and CapsNet Network Model. *Journal of Medical Imaging and Health Informatics, 9*(1), 159-166.

[15] Oliveira, L. L. G., e Silva, S. A., Ribeiro, L. H. V., de Oliveira, R. M., Coelho, C. J., & Andrade, A. L. S. (2008). Computer-aided diagnosis in chest radiography for detection of childhood pneumonia. *International journal of medical informatics*, 77(8), 555-564.

[16] Sousa, R. T., Marques, O., Soares, F. A. A., Sene Jr, I. I., de Oliveira, L. L., & Spoto, E. S. (2013). Comparative performance analysis of machine learning classifiers in detection of childhood pneumonia using chest radiographs. *Procedia Computer Science*, 18, 2579-2582.

[17] Qin, C., Yao, D., Shi, Y., & Song, Z. (2018). Computer-aided detection in chest radiography based on artificial intelligence: a survey. *Biomedical engineering online, 17*(1), 113.

[18] Moh'd Rasoul, A., Dorgham, O., & Razouq, R. S. (2006). Pneumonia identification using organizing map algorithm.

[19] Parveen, N., & Sathik, M. M. (2011). Detection of pneumonia in chest X-ray images. *Journal of X-ray science and technology, 19*(4), 423-428.

[20] Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., & Fei-Fei, L. (2009, June). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition* (pp. 248-255). Ieee.

[21] Rajpurkar, P., Irvin, J., Zhu, K., Yang, B., Mehta, H., Duan, T., ... & Lungren, M. P. (2017). Chexnet: Radiologist-level pneumonia detection on chest x-rays with deep learning. *arXiv preprint arXiv:1711.05225*.

[22] Saul, C. J., Urey, D. Y., & Taktakoglu, C. D. (2019). Early Diagnosis of Pneumonia with Deep Learning. *arXiv preprint arXiv:1904.00937*.

[23] Ayan, E., & Ünver, H. M. (2019, April). Diagnosis of Pneumonia from Chest X-Ray Images Using Deep Learning. In *2019 Scientific Meeting on Electrical-Electronics & Biomedical Engineering and Computer Science (EBBT)* (pp. 1-5). IEEE.

[24] Islam, M. T., Aowal, M. A., Minhaz, A. T., & Ashraf, K. (2017). Abnormality detection and localization in chest x-rays using deep convolutional neural networks. *arXiv preprint arXiv:1705.09850*.

[25] Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

[26] Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1251-1258).

[27] Abiyev, R. H., & Ma'aitah, M. K. S. (2018). Deep convolutional neural networks for chest diseases detection. *Journal of healthcare engineering, 2018*.

[28] Stephen, O., Sain, M., Maduh, U. J., & Jeong, D. U. (2019). An Efficient Deep Learning Approach to Pneumonia Classification in Healthcare. *Journal of healthcare engineering, 2019*.

[29] Martinez, V. (2019). Detecting Pneumonia in Chest X-Rays with Custom Vision and PyTorch. Retrieved 14 November 2019, from https://medium.com/datadriveninvestor/detecting-pneumonia-in-chest-x-rays-with-custom-vision-and-pytorch-e270e071e982

[30] Imran, A. (2019). Training a CNN to detect Pneumonia. Retrieved 14 November 2019, from https://medium.com/datadriveninvestor/training-a-cnn-to-detect-pneumonia-c42a44101deb

[31] Sagar, A. (2019). Deep Learning for Detecting Pneumonia from X-ray Images. Retrieved 14 November 2019, from https://towardsdatascience.com/deep-learning-for-detecting-pneumonia-from-x-ray-images-fc9a3d9fdba8

[32] Kermany, D. K., & Goldbaum, M. (2018). Labeled optical coherence tomography (OCT) and Chest X-Ray images for classification. *Mendeley Data, 2*.

[33] Tian, Y. (2018). Detecting Pneumonia with Deep Learning. Retrieved 14 November 2019, from https://becominghuman.ai/detecting-pneumonia-with-deep-learning-3cf49b640c14

[34] Erthal, J. (2018). Detecting Pneumonia with Deep Learning: A Soft Introduction to Convolutional Neural Networks. Retrieved 14 November 2019, from https://medium.com/datadriveninvestor/detecting-pneumonia-with-deep-learning-a-soft-introduction-to-convolutional-neural-networks-b3c6b6c23a88

[35] Frosst, N., Sabour, S., & Hinton, G. (2018). DARCCC: Detecting adversaries by reconstruction from class conditional capsules. *arXiv preprint arXiv:1811.06969*.

[36] Hinton, G. E., Sabour, S., & Frosst, N. (2018). Matrix capsules with EM routing.

[37] Qin, Y., Frosst, N., Sabour, S., Raffel, C., Cottrell, G., & Hinton, G. (2019). Detecting and Diagnosing Adversarial Images with Class-Conditional Capsule Reconstructions. *arXiv preprint arXiv:1907.02957*.

[38] Kosiorek, A. R., Sabour, S., Teh, Y. W., & Hinton, G. (2019). Unsupervised Object Discovery via Capsule Decoders.

[39] Cardoso, R. (2019). Using VGG + CapsNet in to Diagnose Pneumonia | Kaggle. Retrieved 14 November 2019, from https://www.kaggle.com/rodcardoso92/using-vgg-capsnet-in-to-diagnose-pneumonia

[40] Pedamonti, D. (2018). Comparison of non-linear activation functions for deep neural networks on MNIST classification task. *arXiv preprint arXiv:1804.02763*.

[41] Oliphant, T. E. (2006). *A guide to NumPy* (Vol. 1). Trelgol Publishing USA.

[42] Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in science & engineering*, *9*(3), 90.

[43] François, C. (2015, March 27). keras. Retrieved from https://github.com/fchollet/keras

[44] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., ... & Ghemawat, S. (2016). Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*.

[45] Kurbiel, T., & Khaleghian, S. (2017). Training of Deep Neural Networks based on Distance Measures using RMSProp. *arXiv preprint arXiv:1708.01911*.

[46] Su, J., Vargas, D. V., & Sakurai, K. (2019). One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*.

[47] Krizhevsky, A., Nair, V., & Hinton, G. (2014). The CIFAR-10 dataset. *online: http://www. cs. toronto. edu/kriz/cifar. html*, 55.

[48] Marreiros, A. C., Daunizeau, J., Kiebel, S. J., & Friston, K. J. (2008). Population dynamics: variance and the sigmoid activation function. *Neuroimage*, *42*(1), 147-157.

[49] Shang, W., Sohn, K., Almeida, D., & Lee, H. (2016, June). Understanding and improving convolutional neural networks via concatenated rectified linear units. In *international conference on machine learning* (pp. 2217-2225).

[50] Zhang, W., Tang, P., & Zhao, L. (2019). Remote Sensing Image Scene Classification Using CNN-CapsNet. *Remote Sensing*, *11*(5), 494.

[51] Hoogi, A., Wilcox, B., Gupta, Y., & Rubin, D. L. (2019). Self-Attention Capsule Networks for Image Classification. *arXiv preprint arXiv:1904.12483*.

[52] do Rosario, V. M., Borin, E., & Breternitz Jr, M. (2019). The Multi-Lane Capsule Network (MLCN). CoRR.

[53] Brownlee, J. (2018, December). How to Develop a Stacking Ensemble for Deep Learning Neural Networks in Python with Keras. Retrieved 23 September 2019, from https://machinelearningmastery.com/stacking-ensemble-for-deep-learning-neural-networks/.

[54] Sun, Y., Xue, B., Zhang, M., & Yen, G. G. (2018). Automatically designing CNN architectures using genetic algorithm for image classification. *arXiv preprint arXiv:1808.03818*.

[55] Sun, Y., Xue, B., Zhang, M., & Yen, G. G. (2019). Evolving deep convolutional neural networks for image classification. *IEEE Transactions on Evolutionary Computation*.

[56] Castillo, P. A., Arenas, M. G., Castillo-Valdivieso, J. J., Merelo, J. J., Prieto, A., & Romero, G. (2003). Artificial neural networks design using evolutionary algorithms. In *Advances in Soft Computing* (pp. 43-52). Springer, London.

[57] Ding, S., Li, H., Su, C., Yu, J., & Jin, F. (2013). Evolutionary artificial neural networks: a review. *Artificial Intelligence Review*, *39*(3), 251-260.

[58] Hansen, N., & Ostermeier, A. (2001). Completely derandomized self-adaptation in evolution strategies. *Evolutionary computation*, 9(2), 159-195.

[59] Hansen, N. (2006). The CMA evolution strategy: a comparing review. In *Towards a new evolutionary computation* (pp. 75-102). Springer, Berlin, Heidelberg.

[60] Sehnke, F., Osendorfer, C., Rückstieß, T., Graves, A., Peters, J., & Schmidhuber, J. (2010). Parameter-exploring policy gradients. *Neural Networks*, *23*(4), 551-559.