A Novel Multi Model Ensemble Learning Approach for Pneumonia Detection

Davoud Moradi
Department of Computer Science
University at Buffalo
davoudmo@buffalo.edu

Mohiuddeen Khan
Department of Computer Science
University at Buffalo
mohiudde@buffalo.edu

Rakshitha Shivaraj
Department of Computer Science
University at Buffalo
rhivara@buffalo.edu

Code Link:

https://github.com/mohikhan/CSE-676-A-Novel-Multi-Model-Ensemble-Learning-Approach-for-Pneumonia-Detection/tree/main

Abstract

Chest X-ray imaging technology is a useful tool for diagnosing various lung disorders. The use of this technique in hospitals is well recognized, and it is the most accurate way to diagnose the majority of thoracic disorders. Finding lung conditions from these pictures might take radiologists some time to do and it depends on human performance. However, an automated artificial intelligence system might help radiologists identify lung ailments more rapidly and precisely. Since chest X-ray pictures may be utilized to identify diseases of the chest, we suggest a unique novel ensemble learning strategy in which we combine three models: convolutional neural networks, recurrent neural networks, and VGG 16 model. After giving each model a weight, the output from the three models will be combined using our ensemble learning technique. The models' weights will be assigned mathematically in such a manner that the model that performs the best on a particular batch of experimental sets receives the greatest weight and, consequently, top priority in prediction.

1 Introduction

Pneumonia is a lung inflammation disease that mostly affects the tiny air sacs known as alveoli. A productive or dry cough, chest discomfort, a fever, and breathing difficulties are typical symptoms. To identify pneumonia from X-rays, we'll create a novel weighted ensemble learning model employing three models. The method is novel because it combines the results from three widely used image classification models—CNNs, RNNs, and the VGG-16 model—to provide a result that can be trusted. The production from the three models won't all be tallied equally, though. To specify the three models' contributions to the ensemble classifier's output, a distinct weight allocation strategy will be put into place. The proposed approach can be seen in figure 1 and will be described in a lot of detail later.

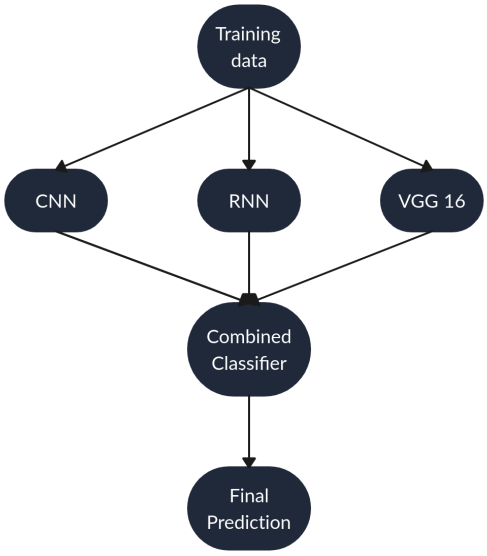


Fig 1. Proposed Ensemble Approach Intuition

2 Dataset Used

For our experiment, we employed a modified version of Paul Mooney's most well-known dataset of Chest X-ray pictures, which are classified as Pneumonia and Normal. The dataset may be accessed by going to https://www.kaggle.com/datasets/paultimothymooney/chest-xraypneumonia. Each picture category (such as pneumonia) and normal have their own subfolders in the data set, which is divided into three sets (train, test, and validation). There are 2 categories (Normal/Pneumonia) and 5,863 X-Ray pictures (JPEG). A sample of the dataset is shown in Fig 2.

Fig 2. Images used in the experiment

3 Models for Ensemble

We deployed a total of 3 models for the ensemble namely Convolutional Neural Networks, Recurrent Neural Networks and VGG 16 model. The next subsections contains the details of the models we used.

3.1 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a type of neural network architecture that is particularly well-suited for image and video recognition tasks. CNNs can also be used for other types of data such as audio, time series, and natural language processing. CNN models are characterized by their ability to extract local features from input data through the use of

convolutional filters. This is achieved through a series of convolutional and pooling layers, which progressively reduce the dimensionality of the input data while preserving important features.

The most common type of CNN architecture is the deep residual network (ResNet), which uses residual connections [1]. Another popular architecture is the convolutional autoencoder (CAE), which can be used for unsupervised feature learning and dimensionality reduction [2]. Training a CNN model typically involves optimizing a loss function using backpropagation and stochastic gradient descent, with various regularization techniques such as dropout and weight decay to prevent overfitting. Recent advances in hardware and optimization algorithms have made it feasible to train large-scale CNN models on massive datasets.

CNNs have achieved state-of-the-art performance on a wide range of image recognition tasks, such as object detection [3], image classification [4], and semantic segmentation [5]. They have also been applied to other types of data such as speech recognition [6] and natural language processing [7]. Generally speaking, CNN models are an important tool for deep learning projects that involve image and video recognition tasks, and have been used to achieve exceptional results in a wide range of applications.

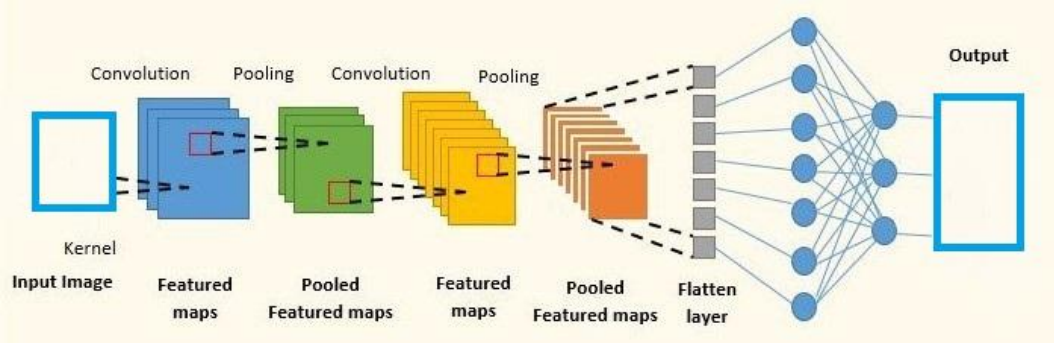
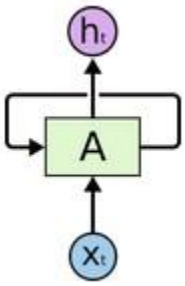


Fig 3. An example of CNN Architecture

Our CNN model consists of multiple convolutional layers followed by batch normalization, activation functions (ReLU), max pooling, and dropout layers. The final layers include flattening, dense layers, and an output layer with sigmoid activation for binary classification. For the training, the model is compiled with the binary cross-entropy loss function and the Adam optimizer. The model consists of 3 blocks and a final layer. The first block consists of a layer with 16 filters and a max pooling layer. The second block consists of a layer with 32 filters and a max pooling layer in these two layers the model has 20 percent dropout. The third block consists of two convolutional layers with 64 filters and kernel size of three and a max pooling layer. For the last block we flatten the data and we used 50 percent dropout. For the first four blocks we used Relu as the activation function and for the output the activation function is sigmoid. We trained the model for three epochs to prevent overfitting. The accuracy and the loss for the CNN model are shown in Table IV.

3.2 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are a type of neural network architecture that is particularly well-suited for sequential data processing. RNNs can be used for different types of deep learning tasks such as natural language processing, speech recognition, time-series prediction, and more. RNN models are known for their ability to process input sequences of variable length, and to maintain a "memory" of previous inputs. This is achieved through the use of recurrent connections between neurons in the network, which allow information to flow both forward and backward in time.



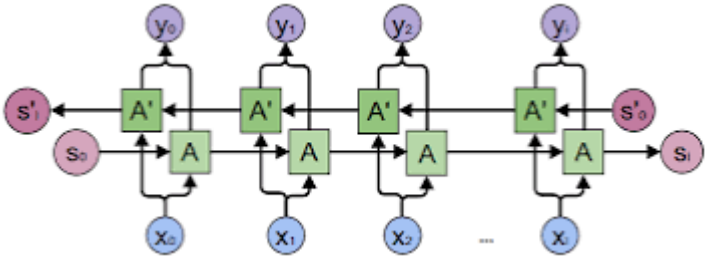


Fig 4. An example of RNN

Fig 5. An example of Bidirectional RNN

The most common type of RNN architecture is the Long Short-Term Memory (LSTM) network, which was specifically designed to address the issue of vanishing gradients that can occur in standard RNNs. LSTMs use a series of gates to control the flow of information through the network, allowing it to selectively remember or forget previous inputs. In this project we used the same network.

Training an RNN model typically involves optimizing a loss function using backpropagation through time (BPTT), which involves computing gradients with respect to the entire input sequence, not just a single time step. This can be computationally intensive, but recent advances in hardware and optimization algorithms have made it feasible for large-scale deep learning projects.

RNNs have been successfully applied to a wide range of tasks in natural language processing, such as machine translation [8], sentiment analysis [9], and speech recognition [10]. They have also been used in time-series prediction tasks, such as predicting stock prices [11] and forecasting electricity demand [12]. In general, RNN models are a powerful tool for deep learning projects that involve sequential data processing, and have been used to achieve state-of-the-art results in a wide range of applications.

Although it is not common to use RNN for classification of images, for educational purposes and for testing this type of model Furthermore, we used RNN model due to the way RNN models handle the data which is different from the other architectures and this is a useful feature based on our multi model ensemble learning approach. Moreover, it gives an additional point of view to verify our method. We implemented a Bi-directional Recurrent Neural Network (BiRNN) with LSTM layers to classify images in the chest x-ray dataset. We used this model as one of the models. We use LSTM as one of the variants of Recurrent Neural Network. LSTM stands for long short-term memory networks, and it is used in the field of Deep Learning, and it is capable of learning long-term dependencies. To classify images using a recurrent neural network, we consider every image row as a sequence of pixels by flattening each image to a 1D array and then we give the array to our model and train our model based on this 1D array, in each step we use same function and same set of parameters and we trained our model for 300 steps. We have defined the BiRNN model using Keras' Model class and defined the forward pass using the call method. We implemented the cross-entropy loss function and applied softmax and used Adam as the optimizer. The accuracy and the loss for the RNN model are shown in Table IV.

3.3 VGG-16 Model

Many deep neural networks trained on images have a curious phenomenon in common: in early layers of the network, a deep learning model tries to learn a low level of features, like detecting edges, colors, variations of intensities, etc. Such kinds of features appear not to be specific to a particular dataset or a task because of no matter what type of image we are processing either for detecting a lion or cars. In both cases, we have to detect these low-level features. All these features occur regardless of the exact cost function or image dataset. Thus learning these features in one task of detecting lions can be used in other tasks like detecting humans. This is what transfer learning is. Nowadays, it is very hard to see people training whole convolutional neural network from scratch, and it is common to use a pre-trained model trained on a variety of images in a similar task, e.g models trained on ImageNet (1.2 million images with 1000 categories), and use features from them to solve a new task.

VGG-16 is a convolutional neural network [13] model proposed by K. Simonyan and A. Zisserman from the University of Oxford in the paper “Very Deep Convolutional Networks for Large-Scale Image Recognition". The model achieves 92.7% top-5 test accuracy in ImageNet, which is a dataset of over 14 million images belonging to 1000 classes. It was one of the famous models submitted to ILSVRC-2014. It makes an improvement over AlexNet by replacing large kernel-sized filters (11 and 5 in the first and second convolutional layer, respectively) with multiple 3×3 kernel-sized filters one after another. VGG-16 was trained for weeks and was using the NVIDIA Titan Black GPU. The architecture of VGG-16 is shown below in Figure 6:

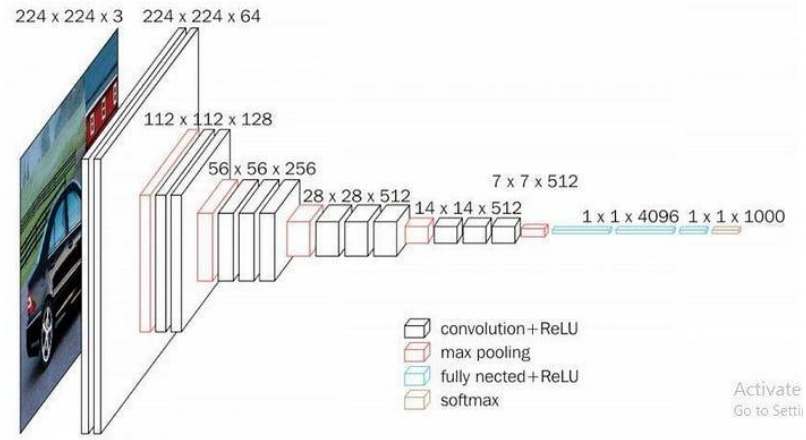


Fig 6. VGG 16 Architecture

In the model, we used the VGG 16 with its layers frozen and RMSprop as the optimizer. This was done to prevent the weights from being updated during training. This allows feature extraction from the pre-trained model.The intuition behind this is that if a model is trained on a large and general enough dataset, this model will effectively serve as a generic model of the visual world. In our first network, our first layer (base layer) is VGG-16 with its layers frozen. After that, we have applied a Global average pooling layer. Then we have a dense layer of 4096 neurons with activation "Relu". Following, we have a dropout layer with rate 0.5 and finally an output "softmax" layer with 2 outcomes. The loss function we used in our network was categorical_crossentropy and the optimizer was RMSprop with learning rate=1e-4.In our first network, our first layer (base layer) is VGG-16 with its layers frozen. After that, we have applied a Global average pooling layer. Then we have a dense layer of 4096 neurons with activation "Relu". For the next step, we have a dropout layer with rate 0.5 and finally an output "softmax" layer with 2 outcomes. The loss function we used in our network was categorical_crossentropy and the optimizer was RMSprop with learning rate=1e-4. We can then take advantage of these learned feature maps without having to start from scratch by training a large model on a large dataset. The accuracy and the loss for the VGG-16 model are shown in Table IV.

4 Weighted Ensemble Model

When faced with binary classification issues, ensemble learning integrates the findings from many models by averaging their outputs, either through hard voting or soft voting.

The class that receives the most votes, or the one that has the greatest chance of being predicted by each classifier, is the projected output class in hard voting. As an illustration, if three classifiers provide [0,1,1] as their outputs for a binary classification issue, the final outcome will be 1 as predicted by the majority of classifiers.

The issue with hard and soft voting is that if a model is really good and resilient in comparison to other models in the ensemble, then its vote is counted equally to all the models in the ensemble. It is a good idea to set up a hierarchical structure for the models so that each one has its own weight and a greater contribution in line with its level of robustness. To construct our agent, we adopted and adapted a method from [14].

The formula for weighing models according to their robustness is given in equations (1) and (2).

$$w_{i,j} = \begin{cases} w_{i-1,j} + r & \text{For correct predictions} \\ w_{i-1,j} & \text{For incorrect predictions} \end{cases} \quad (1)$$

$$r = Y_{wrong}/n \quad (2)$$

where:

$$w = weight,\ r = reward$$

$$Y_{wrong} = number\ of\ classifiers\ that\ did\ wrong\ prediction$$

$$n = number\ of\ classifiers$$

At first, each weight is equal to 1. The weights for the classifiers are adjusted using Equation (1) after traversing each instance in the set 1000 Normal and 1000 Pneumonia images of the training data. For illustration, look at Tables I and II. To update the weights of the three classifiers in Table I, an example dataset is provided in Table II. In the first case, classifier 3 correctly predicts the output, as shown in Table II, but classifiers 1 and 2 predict the incorrect output class. The weights of the first two classifiers are unaffected by the penalty, while the third classifier is rewarded by adding a value from Equation (2). The final weights are determined as given in Table I after repeating the procedure for three occurrences.

Table I

Weight change from each instance

Instances/Classifiers	**C1 prediction**	**C2 prediction**	**C3 prediction**
1	1 + 0 = 1	1 + 0 = 1	$1 + \frac{2}{3} = 1.67$
2	1 + 0 = 1	$1 + \frac{1}{3} = 1.33$	$1.67 + \frac{1}{3} = 2$
3	1 + 0 = 1	1.33 + 0 = 1.33	$2 + \frac{2}{3} = 2.67$
Final weights	**1**	**1.33**	**2.67**

Table II

Dataset Example

Instances/Classifiers	**C1 prediction**	**C2 prediction**	**C3 prediction**	**Actual Class**
1	X	X	Y	Y
2	Y	X	X	X
3	Y	Y	X	X

After giving each model weights by going through each instance of the image set, test data with 234 images of each normal and pneumonia images was sent as input into the ensemble model, with each instance of test data being incorporated into each model. Every model makes an output prediction that is multiplied by the weight that corresponds to it. For example suppose if we have three models in the ensemble, for instance, and the weights for the three models are [1,1.33,2.67] and the outputs from the various classifiers are [1,1,0], then the total output for class 1 is [1*1 + 1.33*1] or 2.33 and the total output for class 0 is [2.67*1] or 2.67. The ultimate outcome will be

class 0 since 2.67 > 2.33. If this had been a simple hard vote, we would have received class 1 as the result because the majority of classifiers gave the result as class 1. Fig 7 shows the above strategy applied to the 3 proposed models.

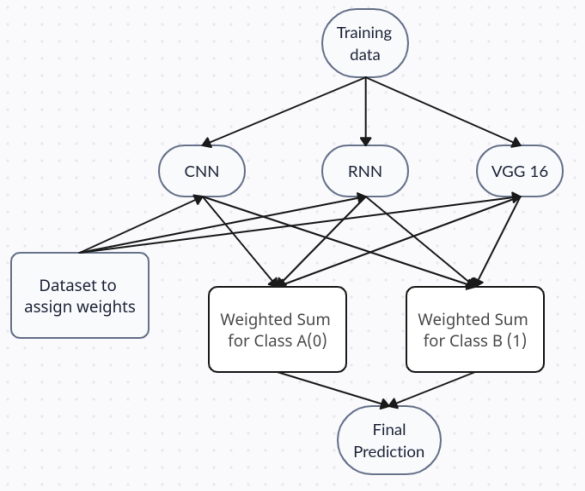


Fig 7. Proposed Model

The described strategy has the benefit of not contributing equally to a good and bad model. Based on its performance, each model in the ensemble will contribute to the final product.We ran the above strategy on the above 3 models namely, CNN, RNN and VGG-16 CNN models. The weights were updated using 1000 images from the training set that included both normal and images with pneumonia and the final updated weights are shown in table III.

Table III

Final weights of each model

Model	**Model Weight**
Convolutional Neural Network	332.00
Recurrent Neural Network	52.33
VGG-16 CNN	325.67

4 Results

After deploying the model onto the test dataset, we got the results as shown in Table IV. We can see the individual test accuracies of different models and the final test accuracy from the weighted ensemble model. The results clearly show the cogency of the proposed approach as the accuracy is higher than the individual models.

Table IV

The accuracy and loss of each model

Model	Test Accuracy(%)	Loss
Convolutional Neural Network	88.14	0.2732
Recurrent Neural Network	78.91	0.5247
VGG-16 CNN	88.30	0.2804
Weighted Ensemble Model	**91.58**	

6 Conclusion

In this research, we employed a weighted ensemble learning method to identify pneumonia from X-ray images. The fundamental premise of this study was that a summarized output computed using several classifiers had more confidence than an output from a specific single model. This study effectively demonstrated that the combined classifier model's accuracy and resilience were improved by using a weighted ensemble technique above any other single model.

This method may be used for any binary classification problem where we have the option to select one optimistic model, which makes the research's potential applications for the future highly exciting. We may just take the aggregate output utilizing the weighted ensemble learning method that we demonstrated in this research, as opposed to selecting any one optimistic model. There are several limitations, however, such as the fact that if one model is significantly superior to others, it is pointless to take into account the votes from those models.

Acknowledgments

We would like to express our gratitude to Professor Changyou Chen and the incredibly helpful Teaching Assistants for their assistance with the project. It would have been very challenging to execute the job successfully without their invaluable advice and assistance.

References

[1] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

[2] Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives. IEEE Transactions on Pattern Analysis and Machine Intelligence, 35(8), 1798-1828.

[3] Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards real-time object detection with region proposal networks. In Advances in neural information processing systems (pp. 91-99).

[4] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems (pp. 1097-1105).

[5] Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 3431-3440).

[6] Abdel-Hamid, O., Mohamed, A. R., Jiang, H., & Penn, G. (2012). Applying convolutional neural networks concepts to hybrid NN-HMM model for speech recognition. In Proceedings of the IEEE international conference on acoustics, speech and signal processing (pp. 4277-4280).

[7] Kim, Y. (2014). Convolutional neural networks for sentence classification. arXiv preprint arXiv:1408.5882.

[8] Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. In Advances in neural information processing systems (pp. 3104-3112).

[9] Tang, D., Qin, B., & Liu, T. (2015). Sentiment analysis with global context attention lstm. arXiv preprint arXiv:1511.08415.

[10] Graves, A., Mohamed, A. R., & Hinton, G. (2013). Speech recognition with deep recurrent neural networks. In 2013 IEEE international conference on acoustics, speech and signal processing (pp. 6645-6649). IEEE.

[11] Fischer, T., & Krauss, C. (2018). Deep learning with long short-term memory networks for financial market predictions. European Journal of Operational Research, 270(2), 654-669.

[12] Kamalaldin, N., Hussain, A. J., Mohammed, S. A., & Ali, A. M. (2017). A long short-term memory approach for electricity demand forecasting. Neural Computing and Applications, 28(4), 711-721.

[13] Simonyan, Karen & Zisserman, Andrew. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv 1409.1556.

[14] Alican Dogan and Derya Birant. A weighted majority voting ensemble approach for classification. In 2019 4th International Conference on Computer Science and Engineering (UBMK), pages 1–6, 2019.