Comparison of State-of-the-Art Deep Learning Techniques for Pneumonia Detection

Daniel Chen
dschen@ucdavis.edu

Avery Wood
awood@ucdavis.edu

Mahek Jain
mhkjain@ucdavis.edu

Aniket Banginwar
abanginwar@ucdavis.edu

November 2022

Abstract

In recent advancements, Convolutional Neural Networks (CNNs) have proven to be a promising tool to solve several problems in the field of medical science. One such problem is Pneumonia Detection in X-Ray images. In this paper, four CNNs have been compared, namely VGG16, Inception-v3, ResNet152, and DenseNet169, to see how they perform on the pneumonia detection classification task. Vision Transformer, a new architecture framed as an alternative to CNNs, has also been tested to see how well it applies to medical imaging. It was observed that DenseNet169 performed the best among the architectures that were put to the test. Vision Transformers also performed well, indicating a promising future for the architecture in medical imaging. This comparison study helps the readers to understand and choose an architecture based on its strengths and weaknesses for image classification.

1 Introduction

Deep learning (DL) is a powerful, artificial intelligence tool that has seen growing use in the medical field. It is crucial in solving complex computer vision problems such as the analysis of different types of medical images. Specifically, the analysis of X-ray images has been greatly improved by deep learning methods and models, most notably using Convolutional Neural Networks (CNNs). In diagnosis problems, such as diagnosing pneumonia, the feature extraction ability of CNNs allows for greater precision in identifying parts of the X-ray image that correspond to the disease. In this paper, we look specifically at this pneumonia detection problem.

Diagnosing pneumonia is a well-explored problem. Currently, CNNs are the leading deep-learning method for solving this problem. Recent approaches, such as those in [1] and [2], have focused on the early detection of pneumonia, as well as detecting COVID-19. With the onset of the COVID-19 pandemic, diagnosing pneumonia had become a more prominent problem as hospital bed space became quite limited for severe cases, as well as there was the possibility of pneumonia being caused by COVID-19. This re-emphasized the need for early detection as the sooner the pneumonia could get caught, the less likely that case of pneumonia was going to become severe and involve hospitalization. Outside of accounting for COVID-19, other recent work includes testing well-known architectures as well as potentially combining them. Ensemble methods, such as the one in [3], have seen success in combining multiple state-of-the-art models to increase prediction accuracy.

This paper focuses on establishing a baseline for many of the current state-of-the-art CNN models using publicly available data for pneumonia detection. We compare the results from models including VGG16 [4], Inception-v3 [5], ResNet [6], DenseNet [7], and Vision Transformers (ViT) [8, 9]. For each model, we also test how sensitive they are to smaller datasets by reducing the size of the training set. In our results, we find that newer architectures generally perform better. These newer architectures also generally see less of an accuracy decrease when decreasing the size of the training data. These results indicate that the architecture of models themselves has become more robust in solving image classification problems, such as pneumonia detection.

2 Methods

The dataset we are using is the Chest X-ray Images (Pneumonia) dataset on Kaggle [10] from 2017. This dataset contains 5856 chest X-ray images that are primarily labeled as either pneumonia or normal. The pneumonia dataset can be further separated into bacterial and viral pneumonia for a 3-class classification problem which we do for our experiments. The initial training and test split of our data comes from Kaggle itself. This is done to ensure the training and test sets are uniform since it is a public dataset. The initial training/test split is 5232 training images to 624 test images. Of the training images: 1349 are labeled as normal, 2538 are labeled as bacterial pneumonia, and 1345 are labeled as viral pneumonia. For the test images: 234 are labeled as normal, 242 are labeled as bacterial pneumonia, and 148 are labeled as viral pneumonia. In our experiments, we also tried reducing the size of the training set to see how it would effect each model's precision and recall. In these experiments, we reduced the training set by 10%, 20%, and 50% in separate trials.

The images in our data set are also not of a standard size. The images in the train set range from 384x127 pixels to 2916x2663 pixels, and the images in the test set range from 728x344 pixels to 2752x2713 pixels. We resized all of the images in both training and test sets to be 224x224 for all of our experimental models except DenseNet and Inception-v3. DenseNet's default image size as input is 128x128 and Inception-v3's is 299x299.

For all the pre-trained models except for ViT, we add a Batch Normalization layer after retrieving the outputs from the pre-trained models, a Dense layer with 128 units, a Dropout layer with a 0.2 rate, and a final Dense layer for classification.

2.1 Baseline CNN Model

The CNN model we built as a baseline starts with a convolutional layer with 256 units and a kernel size of (3, 3) and a max pooling layer of pool size (2, 2) right after. For all the convolutional layers in this model, the kernel sizes stay the same. The pool sizes are all the same for each max pooling layer as well. Following the first two layers of our baseline model, we add another convolutional layer with 128 units, a max pooling layer, and a convolutional with 64 units. The output gets flattened and fed into a Dense layer with 128 units. Then, a final Dense layer is added for classification.

2.2 VGG16

VGG16 [4] is a convolutional neural network that was published in 2015 and was used to win the previous year's ILSVR (ImageNet) competition. VGG16 is a very deep CNN having 16 weighted layers and around 138 million parameters. It is also unique in that instead of having a large number of hyper-parameters, the filter sizes in the convolution layers and max-pooling layers remain the same throughout the architecture, with the convolutional layers having a filter of size 3x3 stride 1 and the max-pooling layers having a filter of size 2x2 stride 2. The architecture is split into five convolutional blocks separated by the max-pooling layers and followed by 3 fully-connected layers and a softmax layer for output.

As stated previously [11], VGG16 has five convolutional blocks. The first block is applied directly on the 224x224x3 input image. This block contains two convolutional layers followed by a max-pooling layer. Both of the convolutional layers are of size 224x224x64 and have a ReLU activation function. The max-pooling layer then downsamples the image to 112x112x128 and prepares it for input into the next block. The following blocks undergo this same pattern of feature extraction into downsampling by a factor of two. Block 2 has two convolutional layers of size 112x112x128, block 3 has three convolutional layers of size 56x56x256, block 4 has three convolutional layers of size 28x28x512, and block 5 has three convolutional layers of size 14x14x512. This is followed by the final max-pooling layer which has an output of size 7x7x512.

The 5 blocks are the core of VGG16. In the original implementation, the three fully-connected layers that follow the 5 core blocks are considered optional, depending on your specific use case. For the purpose of our experiments, we do use the three fully connected layers and the softmax function for output, as they fit our 3-class classification problem well.

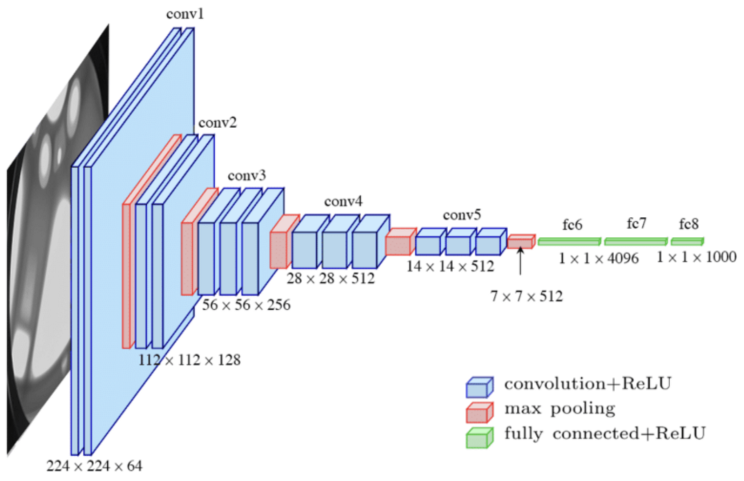

Figure 1: VGG16 architecture

2.3 Inception-v3

Inception-v3 [5] is a 42-layer convolutional neural network for assisting in image analysis and object detection. The inception architecture was introduced with GoogLeNet [12] as Inception-v1 in 2014. The main aim of Inception-v1 was to achieve high-level accuracy with lower computation costs. The major improvements in Inception-v3 over v1 and v2 include factorization into smaller convolutions, spatial factorization into asymmetric convolutions, and efficient grid size reductions. The aim of Inception-v3 is to go deeper as compared to v1 and v2 without compromising the speed of training time and also reducing the computation costs. The input given to the first block is 299x299x3 image and the output is 8x8x2048 further flattened and passed through softmax to get an output for the 1000 classes.

V1 has a 5×5 convolutional layer which was computationally expensive. So to reduce the computational cost the 5×5 convolutional layer was replaced by two 3×3 convolutional layers, as using two 3×3 convolutions reduces the number of parameters. Furthermore, these 3x3 convolutions are replaced by one 3x1 followed by one 1x3 convolution. This process is called spatial factorization into asymmetric convolutions. These improvements help the network to reduce over-fitting and hence go deeper.

The original inception architecture had 2 auxiliary branches to tackle the problem of vanishing gradient. But in v3, only one auxiliary classifier is used. The network performed better when dropout or batch normalization was employed in this branch. V3 avoids bottlenecks early in the network in order to preserve information by using an efficient grid size reduction technique of using two parallel blocks of pooling and convolutional layers. This is also quite computationally efficient.

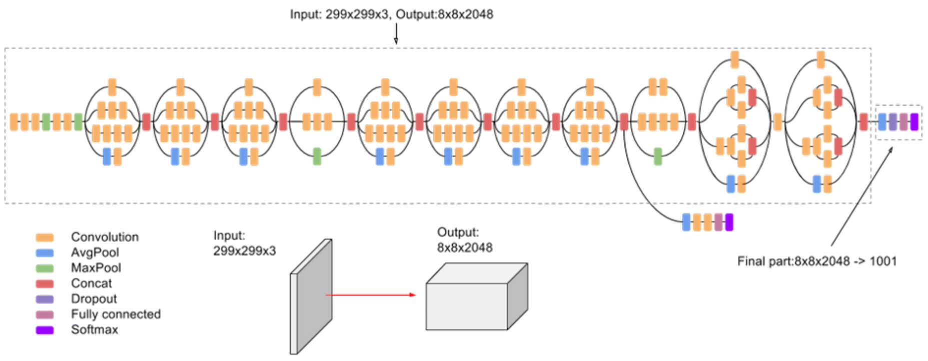

Figure 2: Inception-v3 architecture

2.4 ResNet152

ResNet [6] was published in the year 2015 and was the winner of ILSVR challenge 2015 classification task. The goal of this model was to achieve ultra deep network and solve the issue of vanishing gradient. There can be several types of ResNet based on the number of layers which can vary from 18 to 1202 layers. The most popular kind was ResNet50 which comprises of 48 convolutional layers with one max pooling layer and one average pooling layer. The input given to the first block is 224x224x3 image. Originally, ResNet makes use of a 34 layer, plain architecture, which was initially inspired by the VGG architecture and has the skip connections added to it. They aimed to solve the problem of degradation of the training accuracy by introducing the concept of residual blocks and the skip connections technique.

Skip connections allowed an alternate shortcut path for the gradients to flow through thus solving the vanishing gradient issue. If any layer is diminishing the performance of the architecture then it can be skipped by regularization. These skip connections also help the model to learn the identity functions which makes sure that the higher layer will perform at least as good as the previous layer.

The ResNet architecture consists of a convolutional and pooling layer and is followed by 4 layers behaving in a similar manner. Each of these layers perform a 3x3 convolution with dimension of the feature map as 64, 128, 256 and 512 respectively. For ResNet34, the input bypasses every 2 convolutions (2-layer blocks) whereas for ResNet50, it bypasses every 3 convolutions (3-layer blocks). In case there is a dimension mismatch while adding the output of identity mapping to the output of the stacked layers, it can be solved by padding.

ResNet152 is constructed using adding more 3-layer blocks to the ResNet50 architecture. It is observed that ResNet152 achieves the best accuracy among the ResNet family of networks, hence chosen for this survey.

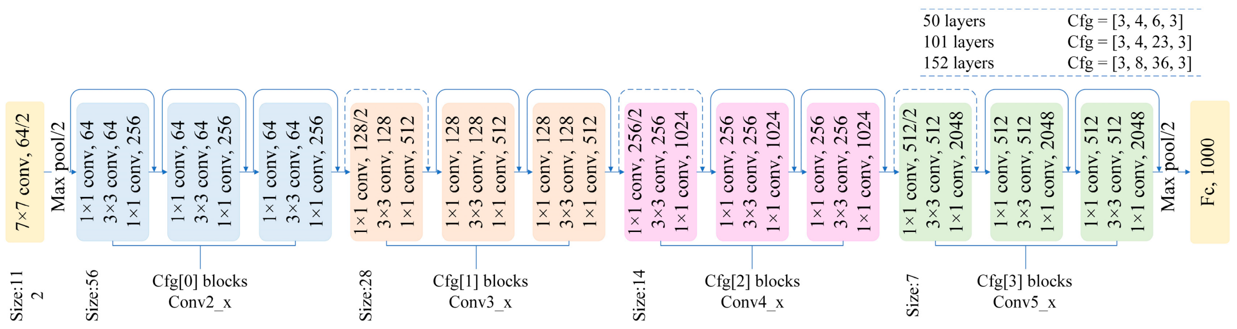

Figure 3: ResNet152 architecture

2.5 DenseNet169

DenseNet [7] is a convolutional neural network that was introduced in the year 2017. This architecture was developed to solve the issue caused by the vanishing gradient in deep neural networks because of the long distance between the input and the output layers. This leads to the information vanishing before it even reached the output. DenseNet has a narrow architecture, i.e., a lesser number of channels. Each layer receives input from all the preceding layers and combines it with its own output to give it to the next layer. This means that each layer is connected with all the other layers in a feed-forward manner. Typically if there are L layers then there would be $\frac{L(L+1)}{2}$ connections.

The DenseNet architecture consists of 4 dense blocks and 3 transition layers. The input given to the first block is 224x224x3 image. Firstly, there is a convolutional layer of size 7x7 with 64 filters and a 3x3 max pooling layer both with stride 2. It is followed by the dense block having 2 convolutions - a 1x1 kernel for the bottleneck and a 3x3 kernel used for performing the convolutions. Then there is a transition layer consisting of 1 convolutional layer and 1 average pool layer. The number of times the dense block and the transition layer repeats depends on the type of DenseNet architecture (DenseNet-121, 169, 201, 264). At the end of the network, there is a global average pooling layer to perform the classification.

The information flow in the network is enhanced by the direct connection of all layers to allow the gradients to flow. This also reduces the problem of overfitting by behaving as a regularizer.

DenseNet169 is chosen because despite having a depth of 169 layers, the architecture handles the vanishing gradient problem well.

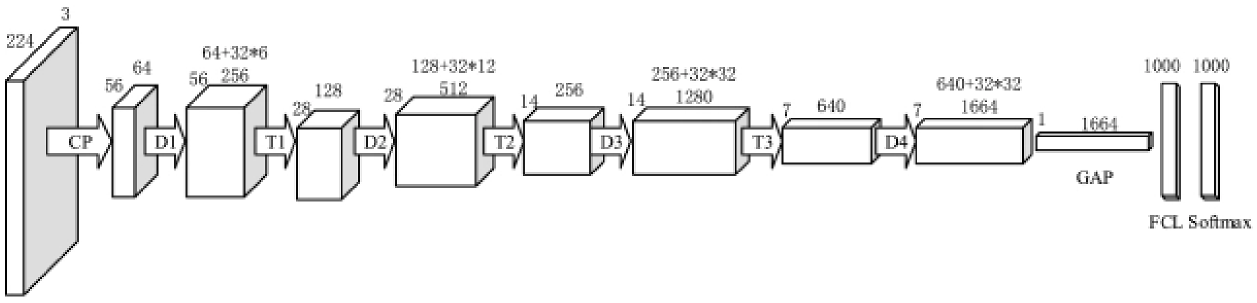

Figure 4: DenseNet169 architecture

2.6 Vision Transformers

Vision Transformers (ViT) [8] is a new state-of-the-art model that was introduced as a conference paper in 2021 and has been shown to outperform models that use convolutional networks for many tasks while requiring significantly fewer computational resources to train. Transformers are popularly used for NLP tasks and have seen lots of success, so the authors experimented with applying a standard Transformer on images with the fewest possible changes.

The model first requires images to be broken down into patches of fixed size. So, an image of size 224x224 would be broken down into 14 patches of 16x16 for example. These patches get flattened out and are used to create linear embeddings. Then, the embeddings are added with positional embeddings and fed into the Transformer encoder layer. Similar to BERT, a special [CLS] token is added in the beginning of the input to use for classification tasks later on. The number of encoder layers, attention heads, and the hidden size depends on the model size - ViT-Base, ViT-Large, or ViT-Huge.

Model	Layers	Hidden Size	Heads
ViT-Base	12	768	12
ViT-Large	24	1024	16
ViT-Huge	32	1280	16

In the case of determining if and what kind of pneumonia a person has, the representation for [CLS] gets fed through a fully-connected layer and softmax is applied to obtain a classification. The ViT models were all pre-trained on large datasets, like ImageNet, ImageNet-21k, and JFT-300M, and then we applied fine-tuned at the end using our X-ray images. For our experiment, we used the base version of ViT that was pre-trained on the ImageNet-21k dataset with the image sizes being 224x224 and the image patches being 16x16. We also tried using the large version as well.

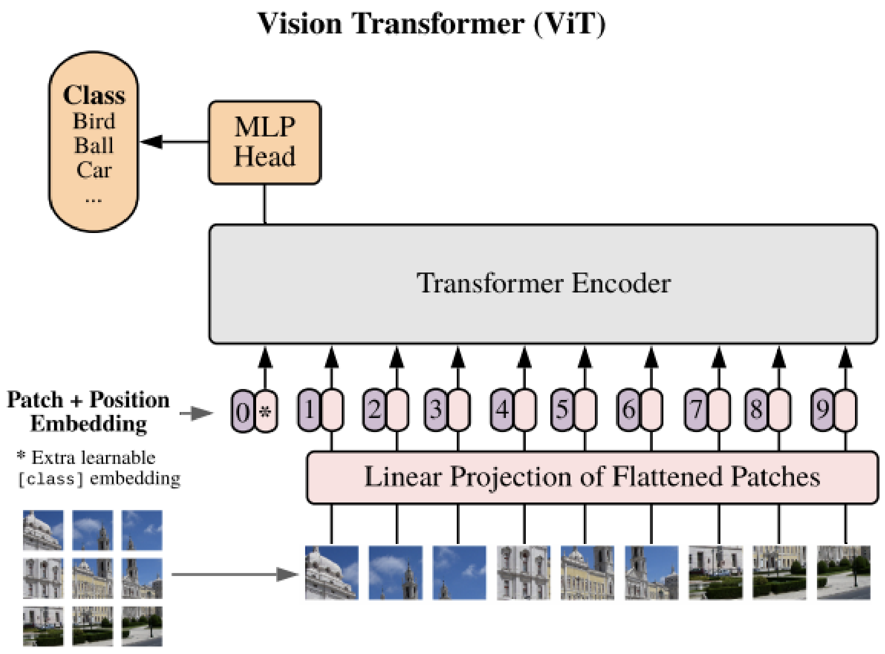

Figure 5: ViT architecture

3 Results

The first pre-trained model tested was VGG16. As seen in table 1, VGG16 had its highest precision, 0.7831, with 20% of the training data removed and used as the validation set. VGG16's recall was highest, 0.7131, for the 10% removed model. However, the 20% removed model performed very closely with a recall of 0.7115. There was a noticeable dip in performance removing 50% of the training data with a precision of 0.7535 and a recall of 0.6859.

Our second model was Inception-v3. The highest precision obtained was 0.7304 and the highest recall was 0.6859. Both of these scores were achieved when only 10% of the training data was removed. There wasn't a significant drop when 50% of the data was removed compared to 10%. ResNet152 performed similarly to VGG16. ResNet152's best test precision and recall were with 20% of the training data removed. The precision and recall for this model were 0.7882 and 0.7276 respectively. The other training splits for ResNet152 also performed very similarly with precisions of 0.7776 and 0.7753 and recalls of 0.7228 and 0.7131 for the 10% and 50% splits respectively. DenseNet169 was the strongest model among the rest. Its best precision was 0.8328 and recall was 0.7740 both when only 20% of the training data was taken out. When another 30% of the training data was removed, precision decreased by only 0.0044 whereas recall dropped by 0.016.

Finally, we tested the base version of the Vision Transformer. This model had the best precision score of 0.8004 when 20% was removed and the best recall score of 0.7771 when 10% of the data was removed. There was a huge drop in both precision and recall when 50% was used as validation data. Precision dropped by 0.113 and recall dropped by 0.2098 compared to when only 20% was removed. We tried experimenting with the large version of ViT with 24 layers, 16 attention heads, and a hidden size of 1024, but didn't achieve any higher scores. The training time of the large model was twice the base model's and the number of parameters for large was more than three times that of the base.

	Precision	Recall
Logistic Regression	0.6929	0.6365
Baseline CNN (20% removed)	0.7895	0.6731
Baseline CNN (50% removed)	0.7206	0.6679
VGG16 (10% removed)	0.7766	0.7131
VGG16 (20% removed)	0.7831	0.7115
VGG16 (50% removed)	0.7535	0.6859
Inception-v3 (10% removed)	0.7566	0.7324
Inception-v3 (20% removed)	0.7704	0.7420
Inception-v3 (50% removed)	0.7609	0.7244
ResNet152 (10% removed)	0.7776	0.7228
ResNet152 (20% removed)	0.7882	0.7276
ResNet152 (50% removed)	0.7753	0.7131
DenseNet169 (10% removed)	0.8310	0.7724
DenseNet169 (20% removed)	0.8328	0.7740
DenseNet169 (50% removed)	0.8284	0.7580
ViT-Base (10% removed)	0.7998	0.7771
ViT-Base (20% removed)	0.8004	0.7522
ViT-Base (50% removed)	0.6874	0.5424
ViT-Large (20% removed)	0.771	0.7458

Table 1: Precision and Recall on the test for each model. Each model was tested with 10%, 20%, and 50% of the training data left out and just added to the validation set.

4 Discussion

The use of pre-trained models is widely popular because of their ease of use, where you can take one off the shelf and apply it to many downstream tasks without the need for much data. The pre-trained models were already trained on huge datasets beforehand to extract important features. Comparing the drop in precision from the baseline CNN when 50% of the data was removed versus the pre-trained models in Table 1, it is clear that not much additional data is needed for pre-trained models. Moreover, the improvement in recall scores from our baseline CNN model compared to the other pre-trained models is apparent too.

Starting with VGG16, the smallest of the tested pre-trained CNN models, the performance is actually quite good. It performs about equally with ResNet152, which had the benefit of being published after VGG16. It is also interesting to see VGG16 outperform Inception-v3 in terms of precision. Inception-v3 is a later iteration of Inception-v1, also known as GoogLeNet. VGG16 and GoogLeNet were competitors for the 2014 ILSVRC (ImageNet) competition with VGG16 performing slightly better than GoogLeNet in terms of the localization error [13]. Even with the later iterations of Inception-v3, this could be part of the reason VGG16 is performing well in comparison. On the other hand, Inception-v3 has higher recall than VGG16, which is an important metric to pay attention to for pneumonia detection. In the medical field, recall is regarded as a safer metric as getting false positives is far less damaging than a true negative. In other words, incorrectly diagnosing someone as having pneumonia will potentially do less damage than incorrectly diagnosing someone as not having pneumonia.

ResNet152 was created in between VGG16 and DenseNet169 and performs as expected, better than VGG16 but not as well as DenseNet169 in terms of both precision and recall. ResNet152 performs better than Inception-v3 in terms of precision but not with recall. This could be due to Inception-v3's innate focus on classification error. As mentioned earlier, VGG16 performed better in terms of localization error, however the original GoogLeNet performed better in classification error [13]. Inception-v3, being a later iteration of GoogLeNet, lends itself to the assumption that the model should perform well in terms of recall. ResNet152 was based-off of VGG16 and VGG19 (a 19-layer iteration of VGG) which may explain why the precision of ResNet152 outperforms that of VGG16 and Inception-v3 while the recall falls short of Inception-v3.

DenseNet169 was the best-performing model that we tested. This isn't too surprising given that it was created after the other CNNs, excluding Vision Transformers. DenseNet169 also had the most layers of any of our CNNs which our results seem to suggest that the deeper a network is, the better it will perform for the given task. Aside from the number of layers in the model, the overall structure of DenseNet169 has direct connections between each of the layers in a feed-forward manner. This reduces the effect that the vanishing gradient problem can have on the model, causing the network to form more meaningful connections between layers. Hence, lending itself to higher accuracy, precision, and recall. This feed-forward structure also reduces the over-fitting capabilities of the model as the connections act like a regularizer throughout the entire network.

The Vision Transformer is the latest core architecture for image classification tasks. Before this model was published, most of the state-of-the-art models used convolution-based architectures. In our pneumonia detection task, we showed that it achieves great results as well, decreasing the reliance on CNN models. Rather than just extracting important features in parts using a filter, what if the entire image could be processed at a time? This is exactly what Vision Transformers are able to do and hence could be the reason for getting promising results.

5 Conclusion

Applying deep learning models to medical tasks is an area that is well-regarded and has a huge impact that could lead to saving lives. We reviewed many of the state-of-the-art pre-trained models for pneumonia detection to see which ones perform better than others. We found that the DenseNet169 model performed the best using precision and recall metrics, with Vision Transformer coming in second. Vision Transformers are relatively new and becoming more popular these days. In the future, variations of the Vision Transformer can be explored and could potentially outperform CNNs for our task. This work provides a baseline for researchers to make an informed decision on what pre-trained model architecture works best for pneumonia detection. In addition, it highlights Vision Transformers as a promising architecture to explore in the field of medical imaging.

The code implemented in this paper can be found here: `https://github.com/DanielSChen61/ECS289G_Project`.

Author Contributions

All team members contributed to the abstract, introduction, results, discussion, and conclusion portions of the paper. Coding was a collective effort via screen sharing on Zoom.

Daniel Chen: Vision Transformers
Avery Wood: VGG16
Mahek Jain: ResNet152, DenseNet169
Aniket Banginwar: Inception-v3

References

[1] Karim Hammoudi et al. "Deep Learning on Chest X-ray Images to Detect and Evaluate Pneumonia Cases at the Era of COVID-19". In: *Journal of Medical Systems* 45 (2021). DOI: `https://doi.org/10.1007/s10916-021-01745-4`.

[2] Abdullahi Umar Ibrahim et al. "Pneumonia Classification Using Deep Learning from Chest X-ray Images During COVID-19". In: *Cognitive Computation* (2021). DOI: `https://doi.org/10.1007/s12559-020-09787-5`.

[3] R Kundu et al. "Pneumonia detection in chest X-ray images using an ensemble of deep learning models". In: *PloS One* 16(9) (2021). DOI: `https://doi.org/10.1371/journal.pone.0256630`.

[4] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *arXiv* (2014). DOI: `https://doi.org/10.48550/arXiv.1409.1556`.

[5] Christian Szegedy et al. "Rethinking the Inception Architecture for Computer Vision". In: *arXiv* (2015). DOI: `https://doi.org/10.48550/arxiv.1512.00567`.

[6] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *arXiv* (2015). DOI: `https://doi.org/10.48550/arxiv.1512.03385`.

[7] Gao Huang et al. "Densely Connected Convolutional Networks". In: *arXiv* (2016). DOI: `https://doi.org/10.48550/arxiv.1608.06993`.

[8] Alexey Dosovitskiy et al. "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". In: *arXiv* (2020). DOI: `https://doi.org/10.48550/arxiv.2010.11929`.

[9] Bichen Wu et al. "Visual Transformers: Token-based Image Representation and Processing for Computer Vision". In: *arXiv* (2020). URL: `https://arxiv.org/pdf/1409.4842.pdf`.

[10] *Dataset of the project (from Kaggle).* URL: `https://www.kaggle.com/datasets/paultimothymooney/chest-xray-pneumonia?datasetId=17810&searchQuery=graph`.

[11] *Step by step VGG16 implementation in Keras for beginners.* URL: `https://towardsdatascience.com/step-by-step-vgg16-implementation-in-keras-for-beginners-a833c686ae6c`.

[12] Christian Szegedy et al. "Going Deeper with Convolutions". In: *arXiv* (2014). DOI: `https://doi.org/10.48550/arXiv.1409.4842`.

[13] *ImageNet competition results.* URL: `https://image-net.org/challenges/LSVRC/2014/results`.