CPSC 66 Final Report: Classifying Chest X-Ray Images Using Convolutional Neural Networks

Guy Berreby GBERREB1@SWARTHMORE.EDU
Xavier Serrano XSERRAN1@SWARTHMORE.EDU
Bilal Soukouna BSOUKOU1@SWARTHMORE.EDU

Abstract

The aim of this project was to design a machine learning algorithm that could accurately and efficiently classify chest x-ray images as positive or negative for Pneumonia. The goal was to apply the concepts learned in this class to a biomedical application: medical imaging and diagnosis. 5856 images were gathered online for use in training and testing the algorithm. Convolutional Neural Networks (CNNs) were primarily used as an accessible introduction to one of the most common image classification algorithms in machine learning. Images were preprocessed to work better and more uniformly with CNNs. Different models including Efficient Net, Inception ResNet, MobileNet, and ResNet, were used for transfer learning. The results showed, as could probably be expected, that transfer learning generally performed better than the algorithm made from scratch, with the EfficientNetB7 transfer learning algorithm performing the best. Though it did not perform as well as the transfer learning models, the from-scratch algorithm was able to manage an accuracy of over 70% after 20 epochs. The results demonstrate that such an algorithm could be a viable tool in diagnostics, especially if stronger computers and algorithms are accessible for use, which would thereby produce better results.

1. Introduction

This project aimed to create a relatively elementary algorithm with the purpose of classifying images. The goal of the project was to apply machine learning image classification to a biomedical application to demonstrate this kind of application's usability and potential to become commonplace in healthcare in the future. In this case, the images to be classified are full chest x-ray images gathered by Daniel Kermany, Kang Zhang, and Michael Goldbaum of the University of California San Diego. Though preprocessing was required for the algorithms used, the data did come labeled (pneumonia, normal) and pre-split into training, testing, and validation sets. Researchers have demonstrated the ability of machine learning to perform quite well in biomedical imaging applications. For example, Esteva et al. were able to create a classification algorithm that was able to "achieve performance on par with all tested experts across both tasks" in a comparison with 21 board-certified dermatologists (Esteva et al., 2017). This project aimed to, on a lower scale, demonstrate the ability of a similar image classification algorithm to classify pneumonia images. With approximately 1.5 million cases and 40 thousand deaths per year in the United States, pneumonia is a very serious infectious lung disease that is one of the top causes of death in the United States (CDC, 2021). It stands to reason that if such an accurate classifier can be created for skin cancer, which is identified visually by doctors, a similarly accurate classifier can be created for pneumonia using chest x-rays, the means by which doctors are able to diagnose pneumonia. Due to the fact that we lack access to extremely high-performance computers, and that we will be designing the algorithm ourselves, it can't necessarily be expected that the algorithm we create will perform on par with medical professionals as the classifier by Esteva et al. did. However, even with a basic classifier and relatively limited dataset, this project can certainly serve as a "proof of concept" of a pneumonia classifier. Even a moderate level of success, though at least better than 50% (no better than random guessing in this case), could serve as a successful proof of concept, as performance could be expected to increase with a larger training set and better computer performance capability. For reference, approximately six thousand images were used for this classification algorithm, a relatively modest number when compared with the Esteva et al. paper, which used over 120 thousand images for training. The classification algorithm we create will demonstrate the plausibility of a similar, but more advanced algorithm to serve in a clinical setting, aiding med-

CPSC 66 Machine Learning Proceedings, Swarthmore College, Spring 2021.

ical professionals in the diagnosis of pneumonia. Eventually, the application of such algorithms to various visually-diagnosed diseases, such as skin cancer and other tumors, can become commonplace in healthcare settings, inexpensively assisting clinicians in making accurate diagnoses of a broad variety of maladies.

2. Methods

Before looking at our results, we will first give an overview of convolutional neural networks (CNNs) and transfer learning, as well as our specific implementation of these techniques.

2.1. Neural Networks

Neural Networks are a powerful supervised machine learning model often used for classification. They consist of multiple components called layers. Each of these layers can be thought of as a function, with each layer taking in the output of the previous one and creating a new output to feed into the next one.

The most common type of layer is called a fully connected or dense layer. This layer consists of multiple nodes sometimes called neurons. Each neuron takes in a copy of the output of the last layers, performs some sort of classification regression (called the activation function) with that output, and then outputs the result of that regression to the next layer. The last layer of a network is always going to be a fully connected layer, with a number of neurons corresponding to the number of classes you are trying to predict. A technique commonly used with fully connected layers is dropout. This is a mechanism which randomly turns off a certain percentage of neurons for each example in training. This helps reduce overfitting in testing, as it forces the model to rely on fewer features.

Neural networks are trained using stochastic gradient descent. This is done by taking the final output of the model on some training sample and comparing it against the actual true result via some loss function, and then taking the gradient of that loss function. Using the gradient, it is then possible to adjust the weights of all the regressions so that it will be more correct in that example in the future.

2.2. Convolutional Neural Networks

Convolutional neural networks (CNNs) are a powerful tool for supervised image classification tasks. CNNs are differentiated from a regular neural network through the inclusion of the convolutional and pooling layers. In convolutional layers, instead of sending every feature (pixels, in the case of an image) into a series of classification regressions, they instead take each input image as a whole, and run a number of $m \times m$ matrices filters over each image.

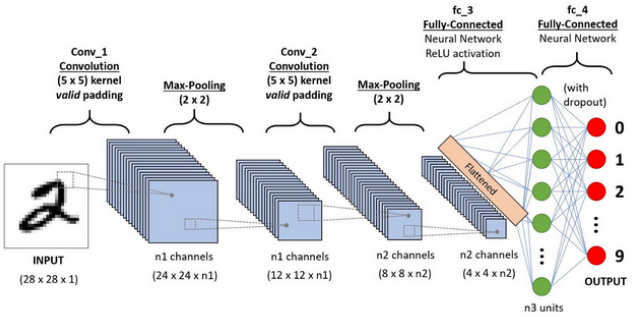


Figure 1. Schematic image of a convolutional neural network working on an MNIST image.

The filters are typically of a smaller size, with $m \leq 10$. Each filter is dotted with an $m \times m$ segment of the input image, producing a scalar value. The filter is then moved across the image, each time taking the dot product with whatever segment of the image the filter is on. This creates a new "image" generated by the filter. This process is then repeated multiple times with different filters to generate a multitude of new images. These new images can then be run through another convolutional layer, a pooling layer, or a fully connected layer.

Pooling layers are used to downsize images before running them through another set of layers. This provides a handful of benefits. First, by reducing the image size, it decreases the number of operations it takes for the model to train. This is important as training times are often a restriction on the complexity of neural networks. Second, it can help reduce overfitting, as the downsizing can get rid of any noise that was generated by the filtering process. The general structure of a CNN model is to first have multiple convolutional layers, with pooling layers interspersed between them, and then to have the last few layers be fully connected. Intuitively, this architecture can be thought of as first finding relevant features in the images you are looking at, with earlier convolutional layers looking at low-level features such as edges and later convolutional layers looking at higher-level features. Then, the fully connected layers take in these higher-level features and use them to determine what class the image falls into. This is the architecture that the CNN we trained from scratch uses.

2.3. Transfer Learning

Transfer learning is the process of taking already existing pre-trained models, and slightly adjusting them to work for the problem you are working on. Generally, you want to use a model that has been trained on a dataset far larger than what you have available for the task you are attempting. While it may seem strange to take a model trained on one set of data, and use it to classify a completely different set

of data, it has actually been shown to produce results that can be better than models trained from scratch (Giorgi & Bader, 2018). For an intuitive explanation, one can think of the description of a convolutional model described earlier. While the exact combination of features used to do the final classification may vary for different classification tasks, the image features created by the convolutional layers may be very similar across the two. To do transfer learning, we use the pre-trained models that TensorFlow has available. We strip off the very last fully connected layer of the model, and then add a new one corresponding to our classification task. We then retrain the model, only adjusting the weights of the final layer we added.

3. Experiments and Results

First we will describe the dataset we used, and then we will elaborate on the model we created and the transfer learning models we used.

3.1. Dataset

For our data, we used the Chest X-Ray Images (Pneumonia) dataset from the researchers at the University of California San Diego (Kermany et al. 2018). This dataset consists of 5856 images of chest x-rays, where roughly half the images represent people with Pneumonia, and the other half are people without pneumonia. The data was collected from children, as pneumonia is a leading cause of child mortality. Overall, there were 624 testing images, with 234 images of healthy people and 390 of people with pneumonia, and 5216 training images, with 1341 images of healthy people and 3875 of people with pneumonia. The images were of varying size, each being approximately $1000 \pm 200 \times 1000 \pm 200$ pixels. This was a problem for us for two reasons. First, neural networks generally require inputs of the same size. Second, these images are quite large, so it would not be possible to train a model using them, as we would not have enough RAM to do so. To solve this problem, we downsized all of the images to 128x128 pixels. We tried a variety of different image sizes from 32x32 to 512x512, but found that 128x128 was the largest we could feasibly make the pictures without running into RAM issues on the computers we had.

3.2. CNN Construction and Tuning

For our CNN, we went with the following architecture:

- 1 Convolutional layer with a 3x3 filter size, 64 channels, and ReLu activation function
- 1 max-pooling layer, with 2x2 pooling filters
- 1 Convolutional layer with a 3x3 filter size, 128 channels, and ReLu activation function
- 1 Fully connected layer with 128 neurons, ReLu activation function and 20% dropout
- 1 Fully connected output layer with 1 neuron and sigmoid activation function

We tested models with both more convolutional layers and more fully connected layers but found that the additional layers caused worse performance, most likely due to overfitting. In addition, we experimented with the number of channels and neurons in the convolutional and fully connected layers and found that these numbers offered the best results. We may have been able to achieve better results if we added substantially more layers to the model, but due to hardware constraints, doing so would not be feasible.

3.3. Transfer Learning Models

For transfer learning, we used the EfficientNetB7, ResNet101V2, MobileNetV2, and InceptionResNetV2 models in the Tensorflow applications module. Using the Tensorflow API, we imported these with the very last layer stripped off, and we added a fully connected layer with a single neuron using a sigmoid activation function. The output of this layer is a single number between 0 and 1, representing the probability that an image is of a person with pneumonia or not. We then retrained the new model, freezing the weights of all the layers save for the very last one we added.

3.4. Results

There was a difference in results across the models we used for transfer learning. The results are as follows:

Average Validation over All Epochs	Average Validation over last half of epochs	Validation on last Epoch
EfficientNetB7 (82.1%)	EfficientNetB7 (83.2%)	EfficientNetB7 (84.3%)
InceptionResNetV2 (81.2%)	InceptionResNetV2 (82.9%)	ResNet101V2 (82.9%)
ResNet101V2 (80.8%)	ResNet101V2 (82.6%)	InceptionResNetV2 (82.5%)
MobileNetV2 (79.1%)	MobileNetV2 (80.9%)	MobileNetV2 (81.7%)
ourModel (72.8%)	ourModel (72.6%)	ourModel (72.0%)

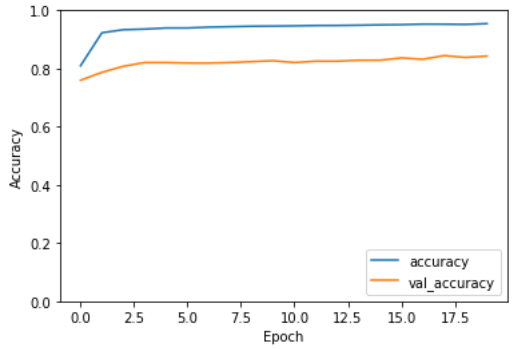
Figure 2. Table showing average validation accuracy for each model over all epochs, over the last half of epochsm and on the last epoch.

Overall, all of the transfer learning models did significantly better than the model we created. EfficientNetB7 seemed to consistently do slightly better than all the rest, while InceptionResNetV2 and ResNet101V2 both tended to perform similarly to each other. MobileNetV2 always seemed to be last in terms of accuracy, which makes sense given the fact that it is specifically designed to be lighter-weight than the

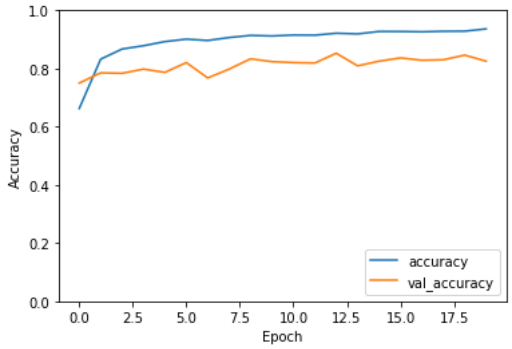
other three so it can be used in settings that have less computational power, like onboard a mobile device. In the last row of the table we can see that our model performed noticeably worse than even the worst transfer learning model, indicating that transfer learning is a viable avenue for improved performance.

4. Discussion

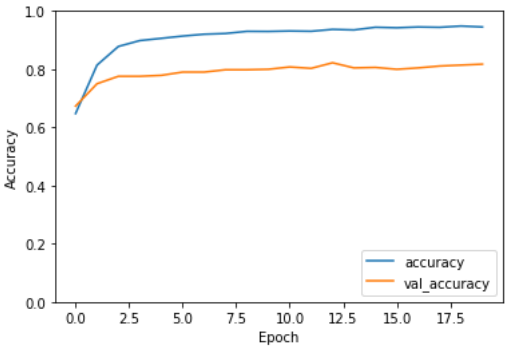
```
Stats For EfficientNetB7:
Average Validation Accuracy Over All Epochs: 0.821
Average Validation Accuracy Over Second Half of Epochs: 0.832
Validation Accuracy Over Last Epoch: 0.843
Average Training Accuracy Over All Epochs: 0.938
Average Training Accuracy Over Second Half of Epochs: 0.950
Training Accuracy Over Last Epoch: 0.954
```



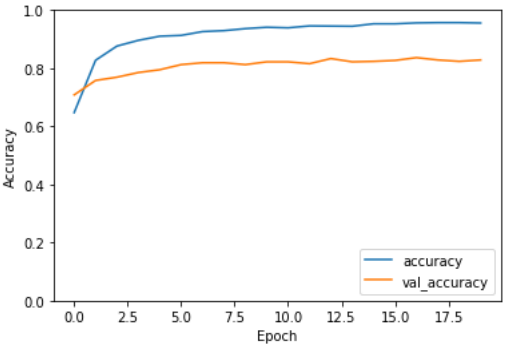

```
Stats For InceptionResNetV2:
Average Validation Accuracy Over All Epochs: 0.812
Average Validation Accuracy Over Second Half of Epochs: 0.829
Validation Accuracy Over Last Epoch: 0.825
Average Training Accuracy Over All Epochs: 0.895
Average Training Accuracy Over Second Half of Epochs: 0.924
Training Accuracy Over Last Epoch: 0.936
```




```
Stats For MobileNet V2:
Average Validation Accuracy Over All Epochs: 0.791
Average Validation Accuracy Over Second Half of Epochs: 0.809
Validation Accuracy Over Last Epoch: 0.817
Average Training Accuracy Over All Epochs: 0.908
Average Training Accuracy Over Second Half of Epochs: 0.940
Training Accuracy Over Last Epoch: 0.945
```




```
Stats For ResNet101V2:
Average Validation Accuracy Over All Epochs: 0.808
Average Validation Accuracy Over Second Half of Epochs: 0.826
Validation Accuracy Over Last Epoch: 0.829
Average Training Accuracy Over All Epochs: 0.916
Average Training Accuracy Over Second Half of Epochs: 0.951
Training Accuracy Over Last Epoch: 0.956
```




```
Average Validation Accuracy Over All Epochs: 0.728
Average Validation Accuracy Over Second Half of Epochs: 0.726
Validation Accuracy Over Last Epoch: 0.720
Average Training Accuracy Over All Epochs: 0.973
Average Training Accuracy Over Second Half of Epochs: 0.996
Training Accuracy Over Last Epoch: 1.000
```

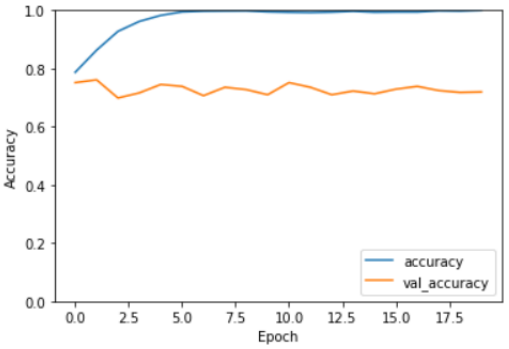


Figure 3. Graphs of validation and training accuracy for all four transfer learning models and our from-scratch model.

The results from the experiment show that the convolu-

tional neural networks are effective for processing chest x-ray images and identifying x-rays that display signs of pneumonia. We had originally hoped that the model we trained ourselves would perform at an accuracy level of about 90% or higher. Although we did not meet this goal, our CNN model produced an accuracy rate of about 70% after a fair amount of hyperparameter tuning. We believe that this level of performance is still sufficient to be used in real-world applications. Because a misdiagnosis could have severe negative consequences for both patients and doctors, we did not intend on creating a model that would replace a doctor . With this in mind, we sought out to create a model that could identify pneumonia in x-ray images with a high enough level of accuracy that it would help streamline the process of diagnosing by making it more efficient and helpful for doctors. Since the model that we trained performs at an accuracy level of 70%, we believe that it can serve as a complement to a doctor's diagnosis, especially when transfer learning is used.

As we previously mentioned in our results, when using transfer learning, we noticed a significant improvement in the performance. Since the pretrained models were trained on significantly larger datasets, they were better able to capture the more important features of the images and, as a result, better classify the images we gave it. Essentially, transfer learning allowed us to create a more powerful model that is able to better generalize when looking at new data. Transfer learning, as a result, would prove to be even better suited for real-world applications than the model made from scratch.

Throughout the development of our project, we became more familiar with some of the limitations of CNNs and how to circumvent some of these limitations. One of the biggest challenges that we encountered while training our CNN model was that we simply did not have enough computing power to run our experiments as freely as we had originally intended. One consequence of our lack of GPU power was that we had to considerably downsize our images in order to run our model. When we began our experiments, we naïvely resized the images to be 512x512 pixels, believing that we could attempt to run our model on larger images once we created a working model. However, none of the machines in the computer science department could run our model on these images without running out of memory. As a result, we had to resize the images to a size of 128x128 pixels in order to be able to train and test our model. The original sizes of the x-ray images, as mentioned before, were each approximately 1000x1000 pixels plus or minus 200 pixels in size. This means that we had to reduce the images to about a tenth of their original size which resulted in a lot of information being lost. Ideally, we would have liked to train our models on images about the size of 800x800 pixels, with the hopes of preserving as much information as possible.

The next challenge we faced was trying to improve the accuracy of our models. In the case of our CNN model there were a considerable number of hyperparameters that we had to adjust in order to improve the performance of our model, we had difficulty trying to adjust our model in a manner that would help it generalize better. One of the hyperparameters that we had to consider was the architecture of our CNN. In other words, we had to decide the number of layers we wanted to use, the types of layers to use, how to order the layers, and how many filters to use in each layer. The number of hyperparameters made it difficult to ascertain which combinations of choices we made for our hyperparameters would allow us to maximize the performance of our model. We did not completely deal with this challenge in regards to our transfer learning models. However, choosing the right pre-trained models out of many proved to be difficult. Given more time, we may have found models that had been pre trained to solve image classification problems that were similar to ours in order to increase the performance of our models even further.

Overall, we were able to confirm that CNN models perform well with image classification. Their performance was further enhanced by using transfer learning, which allowed us to use very complex high-depth models without needing the resources to train them from scratch. We sought out to use machine learning to help doctors diagnose patients with pneumonia. In this regard, we believe that we have succeeded in creating models that can identify pneumonia in x-ray images. Our project has demonstrated that transfer learning can significantly improve the accuracy of classification algorithms on images. With many biomedical applications requiring visualization of some sort, such as the x-rays in this example or pictures of lesions in the example by Esteva et al., further research should be done into higher accuracy classifiers using more advanced transfer learning datasets and computer hardware.

5. Conclusions

In summary, we were able to use a convolutional neural network to classify x-ray images and accurately identify pneumonia within the images. When we used transfer learning to classify the x-ray images, we were able to achieve a slight improvement in performance. Although none of the models achieved a mean accuracy of 90% or higher, they all performed well enough for real-world applications. Given more time to complete this project, we would have focused more on improving the performance of the models that used transfer learning. The process of using the transfer learning models was a lot simpler than the CNN models and the transfer learning models also performed better. This suggests that transfer learning may be better suited

for classifying x-ray images and so more time and effort should be put into finding more suitable pre-trained models and improving their performance to solve our classification problem.

Acknowledgments

We would like to thank Professor Ben Mitchell for his assistance on this project. We'd like to thank the youtube channel 3Blue1Brown for introducing us to neural networks. We would also like to thank the Computer Science Department at Swarthmore College for providing the resources we needed to complete this project.

References

CDC. Pneumonia. 2021. URL https://www.cdc.gov/dotw/pneumonia/index.html.

Esteva, Andre, Kuprel, Brett, Novoa, Roberto, Ko, Justin, Swetter, Susan, Blau, Helen, and Thrun, Sebastian. Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 542:115–118, 2017. URL https://www.nature.com/articles/nature21056.

Giorgi, John and Bader, Gary. Transfer learning for biomedical named entity recognition with neural networks. *Bioinformatics*, 34:4087–4094, 2018. URL https://doi.org/10.1093/bioinformatics/bty449.