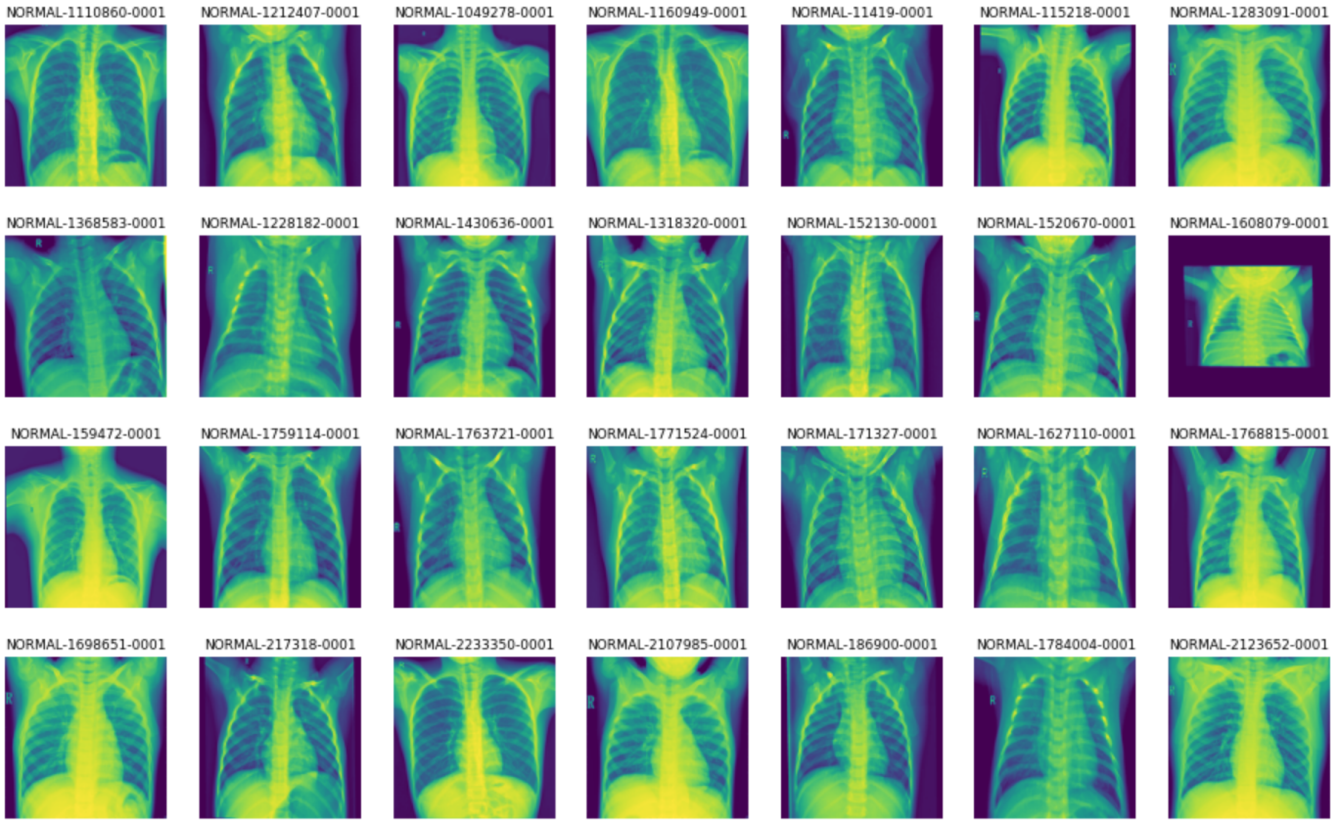
MASTER DEGREE PROGRAM IN DATA SCIENCE AND ADVANCED ANALYTICS | NOVA IMS

Deep Learning Project Report



Pneumonia Detection from Chest X-Ray Images

Github Link: https://github.com/sergiocorrals/DL-Group19-2022

April, 2022

Guzel Bayazitova m20210699
Mohamed Ali Felfel m20211322
Sergio Alfonso Corral Sobrevilla, m2021101
Jessica Routzahn, m20210987

NOVA IMS Information Management School

1. Introduction

Deep learning algorithms are proving to be effective tools for medical research by streamlining and simplifying detailed data analysis in order to make improvements to diagnoses possibly leading to earlier detection of disease and illness. This project attempts to use deep learning algorithms for medical diagnosis of pediatric pneumonia using chest X-ray images. Pneumonia is a form of acute respiratory infection that affects the lungs by causing inflammation in the air sacs in one or both lungs. It kills more children aged 5 and younger each year than any other infectious disease, such as HIV infection, malaria, or tuberculosis. According to the World Health Organization it is the single largest infectious cause of death in children worldwide[1]. Diagnosis is often based on symptoms and physical examination. Chest X-rays may help confirm diagnosis, however there is a finite amount of time in which medical practitioners can view and analyze chest x-ray images. This is where deep learning algorithms come in. Research project development in this paper attempts to aid in expediting the diagnosis and referral of this treatable condition, thereby facilitating earlier treatment, resulting in improved clinical outcomes.

This project is largely based on research from a study entitled “Identifying Medical Diagnoses and Treatable Diseases by Image-Based Deep Learning” which aimed to develop an effective transfer learning algorithm to process medical images to provide an accurate and timely diagnosis of key pathology in each image.[2] Like the original research, the primary illustrations of this project are validated Chest X-Ray images.[3]

1.1. Task/Goal

The goal or task of this project is to use Convolutional Neural Networks on chest x-ray images to determine and distinguish normal healthy patients from those with pneumonia. However, successfully training CNNs requires a large amount of data. The *Chest X-Ray Images* dataset that was used for this project only contains 5,856 images. This was not enough to generalize the network effectively. For this reason a data augmentation technique was needed in order to improve the generalizability of the neural network. There are four approaches for this image classification. First a simple Convolutional Neural Network (CNN) was applied to the existing data in the dataset. Second, we attempt to optimize the CNN through parameters.Third, deep convolutional generative adversarial networks (DCGANs) were used in an attempt to improve the training of CNNs by artiffitialy generating new data. And lastly, the newly generated images were applied again for all six CNN models and used for original test data prediction.

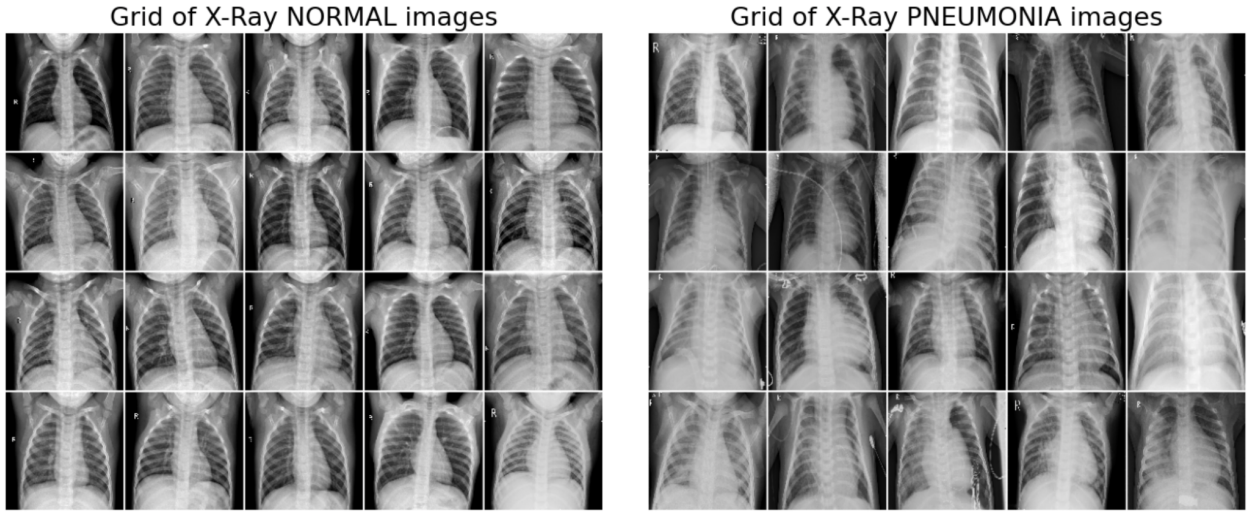
2. Dataset Description

The "*Chest X-Ray Images*" dataset contains 5,856 validated Chest X-Ray images. The images are split into a training set and a testing set of independent patients. There are 1583 “NORMAL,” patients with no present pneumonia, and 4273 “PNEUMONIA,” patients with

[1] Pneumonia, Accessed 4/01/2022 at: https://www.who.int/news-room/fact-sheets/detail/pneumonia
[2] Identifying Medical Diagnoses and Treatable Diseases by Image-Based Deep Learning, Accessed 4/01/2022 at: https://www.cell.com/cell/fulltext/S0092-8674(18)30154-5
[3] Chest X-ray Images, Accessed 4/01/2022 at: https://www.kaggle.com/datasets/tolgadincer/labeled-chest-xray-images

pneumonia, images. The training sample contains 1349 normal images and 3883 pneumonia images with a total of 5232 images in the training sample. The testing sample contains 234 normal images and 390 pneumonia with a total of 624 images in the testing sample. In **figure 1** examples are of the original “NORMAL” images and the “PNEUMONIA” images.

Figure 1: Grid for NORMAL and PNEUMONIA Images



3. Basic Convolutional Neural Networks CNNs

Convolutional Neural Networks (CNNs) are a popular deep learning algorithm choice when training a model with images and videos. CNNs are effective at reducing the number of parameters without affecting the quality of the model. Images are considered to have high dimensionality, as each pixel is considered a feature. CNNs are able to reduce the dimensionality of images. It is for this reason the group’s first approach to achieve the task at hand was to apply CNNs to the dataset.

3.1. Pre-Processing

The “NORMAL” and “PNEUMONIA” images came pre-split in the dataset into test and training sets, but some preprocessing was needed to prepare the images to be fed into the models. Using the Keras Image Data Generator, the group applied some initial and early augmentation to the images. The group defined two generators for this project, *train_datagen* and *test_datagen*, to rescale the training and test sets. Four transformation specifications were applied to the *train_datagen*: rescale was used to target specific RGB coefficients in the original images targeting values between 0 and 1 by scaling the a 1/255 factor, shear_range was set to 0.2 to randomly apply shearing transformations, zoom_range was set to 0.2 to randomly zoom inside pictures, and horizontal_flip was set to *True* in order to randomly flip half of the image horizontally.[4] The *test_datagen* was just reshaped to 1/255, without further data augmentation,

[4] Building powerful image classification models using very little data, Accessed 4/15/22 at: https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html

since the data is not used for tuning the model's parameters. The group applied these transformations on the training and test sets using the flow_from_dataframe method.

3.2. Basic CNN Model and Initial Results

Before running the CNN model1 the group added several callbacks to automate some tasks after every epoch to create more control over the training process. First the group added early stopping to avoid overfitting the model with the metric monitor. Next the group saved the model as a checkpoint after each successful epoch. Sigmoid activation was used as this is a binary classification CNN model.

The first basic CNN model, Model1, contained three convolutional layers, pooling layers, and one dense layer of 128 resulting in a model performance of 0.93. This performance of 0.93 was a great benchmark to start with. However, the group wanted to try to optimize this through various parameter choices. This is in the next section.

4. Parameter Optimization

4.1. Dropout

The group moved to its second approach to the task of this project of CNN parameters tuning. One optimization technique used by the group was adding a dropout layer. This was an attempt to reduce possible overfitting and make the nodes in the network more robust.[5] The group used dropouts after convolutional and pooling layers. An initial dropout layer of 0.25 and final dropout layer of 0.5 was applied to four models (Model2, Model3, Model4, and Model5), each performing worse than the initial basic CNN model. Their respective f1-scores can be seen in **table 1**.

Table 1: Comparison and Performance of All Models

No	Convolutional Layers	Layer Dropout	Dense Layers	Layer Dropout	Batch Normalization	Regularizers	f1-score	f1-score after DCGAN
1	32,64,128		128				0.95	0.80
2	32,64,128	0.25	128	0.5			0.88	0.74
3	32,64,128	0.25	128	0.5	X		0.55	0.33
4	16-32,32-64,128	0.25	128	0.5	X		0.65	0.62
5	16-32,32-64,128	0.25	128	0.5	X	X	0.73	0.62
6	64,128,256		256				0.96	0.86

CVL filter size (3,3), Max-Pooling (2,2), activation function: ReLu, Padding: Same and last layer activation function: Sigmoid

4.2. Batch Normalization

The Batch Normalization layer was used for the Model 3, Model 4 and Model 5. The idea of the method is to transform input data by standardizing, converting mean to zero and standard deviation to one. This technique is widely used in order to increase the speed of training process, and sometimes improve performance of the model via a modest regularization effect.[6]

[5] How to Reduce Overfitting With Dropout Regularization in Keras, Accessed 4/10/22 at: https://machinelearningmastery.com/how-to-reduce-overfitting-with-dropout-regularization-in-keras/

[6] How to Accelerate Learning of Deep Neural Networks With Batch Normalization, Accessed on 4/8/22 at:

In this project the group didn't observe dramatic improvement, and the model performance significantly worsened in all three cases (dropping F1 scores to 0.55, 0.65 and 0.73 respectively).

4.3. Two additional Convolutional Layers

In the Model 4 the group attempted adding two more convolutional layers to perform more in-depth analysis of local patterns in the image. The obtained F1 score is 0.65, so it can be concluded that this method wasn't successful in this case.

4.4. L1 and L2 Regularization

Regularizers are a component that helps to generalize the algorithm by lowering the matrix weights, assuming that the neural network must be smaller, and thus greatly reducing overfitting. The group chose to incorporate this parameter in the Model 5 with L1 set equal to 0.001 and L2 equal to 0.001, which improved the f1-score over the previous one but fell short of the initial models that did not.

4.5. Dense Layers

A Dense of 128 was included in Models1 through Model5. The application of a dense dense layer of 256 was only applied to the sixth and final model the group produced and resulted in a model performance increase from 0.95 (Model1) to 0.96 (Model6).

5. Data Augmentation with DCGANs

Now that the CNN models had been optimized, the group proceeded with their third approach of applying GANs to further improve the accuracy score. Traditional data augmentation techniques include manipulation of brightness, contrast, sharpness, blurriness, etc. These methods are effective for simple classification tasks but not for medical imaging as the augmentations might distort the original scans. We need something more intelligent like Generative Adversarial Networks (GANs). Recently GANs has gained popularity as a possible solution for deep learning models in the medical field.[7] This is because of the model's ability to generate additional "fake" data when original data is limited. The GAN model involves two sub-models: the *generator* model which generates the examples or additional data and the *discriminator* which classifies whether these newly generated examples are from the original dataset or are generated by the model.[8] These models are trained together. The generator creates a batch of samples. These samples along with the original or real examples from the dataset are then classified by the discriminator as real or fake. With each additional round, or epoch, the discriminator is updated to get better at discriminating real and fake samples in the next round, and the generator is updated based on how well, or not, the generated samples fooled the discriminator.[9]

---

https://machinelearningmastery.com/how-to-accelerate-learning-of-deep-neural-networks-with-batch-normalization/

[7] Data augmentation using Generative Adversarial Networks (GANs) for GAN-based detection of Pneumonia and COVID-19 in chest X-ray images, Accessed on 4/10/22 at: https://www.sciencedirect.com/science/article/pii/S2352914821002501

[8] A Gentle Introduction to Generative Adversarial Networks (GANs, Accessed on 4/12/22 at: https://machinelearningmastery.com/what-are-generative-adversarial-networks-gans/

[9] A Gentle Introduction to Generative Adversarial Networks (GANs, Accessed on 4/12/22 at: https://machinelearningmastery.com/what-are-generative-adversarial-networks-gans/

This project and the data used would not allow the use of normal GANs. Instead deep convolutional generative adversarial networks were used in place of fully-connected networks (DCGANs). DCGAN is more appropriate for image and video datasets. For the experimentation with DCGANs the group used the same chest X-ray dataset containing images in two categories "NORMAL" and "PNEUMONIA." The images needed to be resized to 128 × 128 pixels and given appropriate titles.

5.1. Pre-Processing and Initial Augmentation

The loaded images were normalized to have values from -1 to 1 as these have pixel values from 0 to 255. Then they are shuffled randomly in order to produce good training via stochastic gradient descent, since it requires the training dataset be shuffled prior to each epoch. Further, the images were rearranged, so that generated tensors can be transferred by PyTorch to the GPU. As a result, the initial tensors were reshaped from (128, 128, 3) to torch size ([3, 128, 128]) and passed further for training.

5.2. *Discrimator, Generator, Parameters, and DCGAN Model*

The group set parameters before defining the discriminator and generator. Starting with the number of epochs, 200 epochs or rounds were set. The batch size during training was set to 128, as specified above. The size of the generator input, or latent vector, was set to 100. The learning rate for optimizers was set to 0.0002. Prior to developing the architecture for the discriminator, generator, and DCGANs model, the group did extensive research on similar problem sets and models used for medical image generation. Much of the parameters and discussions made about the architecture of this group's model takes influence and guidance from a model originally posted on Kaggle, "*Deep Convolutional Generative Adversarial Network(DCGAN)*[10]," by Djiby Balde.

The discriminator architecture was constructed with a sequential model. The group used convolutional layers followed by the Leaky ReLU activation functions. Leaky ReLUlayers were used instead of ReLU activation because it substitutes zero value with some small value, so gradient descent will continue learning without reaching a dead end. The final layer flattens the architecture and uses a dense layer containing one node to make the appropriate predictions and uses the sigmoid activation.

The generator architecture was constructed with a sequential model. The initial model was set up with an input _dim as latent_dim for receiving incoming random noise and passed through further layers of convolutional upsampling with striding layers. After the first convolutional layer, each consecutive one has a padding type "same", since the output size was intended to be the same as input. There are no max-pooling layers in the network. As with the discriminator, Leaky ReLU layers were used over ReLU and the final layer uses tanh activation.

[10] DCGAN-Keras (Chest X-Ray Images), Accessed on 4/10/22 at: https://www.kaggle.com/code/djibybalde/dcgan-keras-chest-x-ray-images/notebook

After completion of the initial pre-processing and setup of the discriminator and generator, the group built and trained the DCGANs model. Some critical steps needed to take place in setting up the DCGANs model before training. First the loss and optimization needed to be defined for the compilation of the model. The group did not create a checkpoint for this project, but it is something that could be done in the future this way the model could be re-trained after a specific set of epochs. Additionally, all parameters and tf_optimizer (lr =0.0002 and beta_1 = 0.5) needed to be set. Finally the group trained the model and visualized the generated images.

5.3. Results

The DCGANs model was applied to both Normal and Pneumonial images, each with 200 epochs, producing 10000 images per class. This was done in order to partially tackle the imbalance data issue, but mostly to create a new bigger size dataset that is so difficult to get in the real world scenario, then further use for training the model and testing on unseen real data (in our case test data set again).

The Normal class images, produced with DCGANs, can be described as having good or acceptable quality (see **figure 2**), which bring to conclusion that the model is performing well, and 200 epochs is a sufficient amount for training. Unfortunately, the Pneumonial group, being much larger, showed much worse results, the images came significantly blurred and in some parts smeared (**figure 2**). The group attempted rerunning the model with more epochs (300 in particular) in order to improve the images quality, however without much success. It was decided to proceed anyway with further training using these newly generated images, to evaluate their performance and compare results of the original dataset.

Figure 2: Fake Normal (Left) and Pneumonia (Right) Chest X-Ray from DCGANs - After 200 Epochs

5.4. Limitations

The biggest limitation the group had experienced from the data augmentation using DCGANs model was the computing power required to execute the model. The images were improving with each epoch, but the computing power needed to continue making improvements was greater than any processor available to any member of the group. The group initialized the DCGANs model with 200 epochs, but wanted to increase this number to 500 or even 1000

epochs. However, this was just not possible for this project. There is the possibility that given additional computing power the artificial data generator by the DCGANs model could have been improved through additional epochs, but the group was not able to test this.

6. CNN Models with DCGAN Images

The previous CNN models produced good performance, when accounting for f1 scores and accuracy, and DCGAN has its limitations, but it seemed a waste of computational efforts to not do more with the data that was produced from that model. This brought the group to its final approach to achieve the task of this project. The group now tried to implement the approach of DCGAN to improve the training of CNNs. The original images in the dataset were limited as there were only a little over 5,000 images. While DCGAN appears to be the most useful technique that should increase generalization of the data, computational limitations did not make this successful. However, now that the 5,232 training data were used to generate 10,000 images for the normal X-rays and 10,000 images for the Pneumonia X-rays, why not use this new data with the CNN models.

The group used exactly the same CNN models for the new generated training data (DCGAN data) and the results were worse than the CNN models with previous training data as shown in the **tabel 1**. This can be explained by the bad quality of generated Pneumonial class images. Given the GPUs that the group members have, while generating the images, the parameters of DCGAN were limited and low, as a consequence the generated images were not of good quality.

7. Conclusion and Final Model

The stated task of this project was to use CNNs on chest X-ray images to build a predictive model to help identify and distinguish between normal chest X-ray images and those of pneumonia patients. Four approaches were used to achieve this outcome. The most successful, rather than the approach that produced the highest performance measurement, was model6 which used a basic CNN with three convolutional layers and a dense 256 layer.

The final model trained for 20 epochs and consisted of:

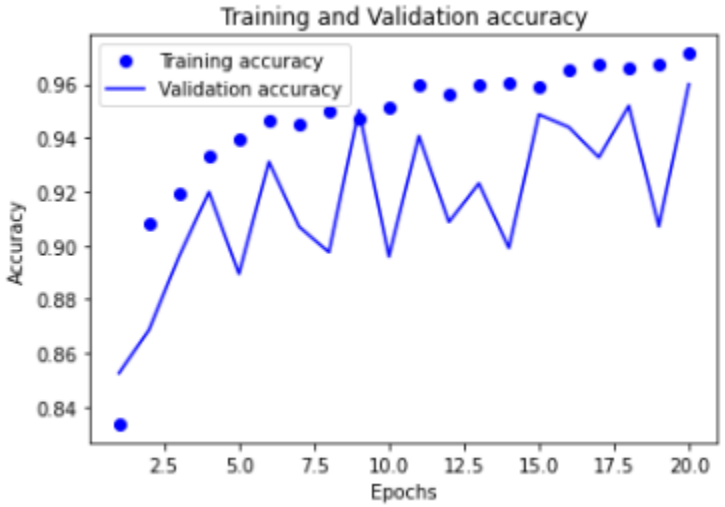
- Conv2D with 64 filters (3,3)
- Max Pooling (2,2)
- Conv2D with 128 filters (3,3)
- Max Pooling (2,2)
- Conv2D with 256 filters (3,3)
- Max Pooling (2,2)
- Flatten Layer
- Dense 256
- Dense 1 (Sigmoid)
-

Besides the output layer, all layers have ReLU Activation function.

7.1. Evaluation Measures

In order to select the final model, the group considered the f1 score to be the determining factor for the model evaluation when comparing CNN models. Despite this, throughout the project process the group did consider more than just accuracy as an evaluation measure. For example, when deciding whether or not to proceed with an approach (DCGANs) an important consideration was the running time for each epoch. The intensive computation needs led the group to abandon DCGANs as a possibility to move forward. Considering the accuracy from training and validation plots in **figure 3** there is a lot of oscillation of the validation accuracy, this could be due to the size of data.

Figure 3: Training and Validation Accuracy in Model6



When looking at the shared models in Kaggle for the same dataset competition, it's clear that most individuals chose pre-trained models like InceptionV3, alexNet or ResNet152V2, which had good and consistent performance at first appearance, with f1 values around 95-96%. On the other hand, there were individuals who chose to build their own models, which yielded a broader range of results ranging from 40 to 95%.

7.2. Tradeoffs of Accuracy and Efficiency

The group implemented four-fold training of the dataset, by applying 6 types of CNN model, creating DCGAN images and then reproducing the same prediction, but with the newly generated data. It is an unusual precedent of creating more data to be used on unseen data, the question is: is it worth it? The quality of pneumonial images is much lower than required for sensible training, and accuracy of the predictions is hence worse. To synthesize data of acceptable level would require bringing high costs, and only from a certain stage the balance between accuracy of predictions and efficiency of the time spent, memory and power can be achieved and considered desirable. Having said that, if creating high quality images would advance the medical industry in diagnosing pneumonia at early stages, such a method would certainly have high potential for the future.