Capstone report

I. Definition

Project Overview

The project I have chosen for my capstone project is images classification of X-ray images of people lunges some consist of 2 categories some are: (Normal and Pneumonia) - "Pneumonia is an infection in airways or lungs usually caused by infection with viruses or bacteria or less commonly by other microorganisms" [1] . The reason for I think it will be a relevant topic is because I think AI will be the future in for Healthcare and nursing and will be the next step in the future to get into more advanced tech to help other people, some have the potential to save and help a lot beyond what we use today. Therefore, have I chosen x-ray image classification with neural networks and because neural networks are a subject, I intend to work with in the future so it will be the most logical I also use it in my capstone project.

One Related topic some are like my project in many ways is Deepcare some stands for a deep, dynamic neural network(https://truyentran.github.io/papers/pakdd_16.pdf). Deepcare is an "end-to-end deep dynamic neural network that reads medical records, stores previous illness history, infers current illness states and predicts future medical outcomes" cited from page 1/12 in the research paper [2]. It does this out from (ERM) electronic medical records some contains "details including family history, reason for initial complaint, diagnosis and treatment, prescription medications, lab tests, and other vital details needed to provide assistance to each patient" [3] out from these record will Deepcare have the ability to predict illness in the future of patients some are similar to my project where I just diagnose the patients out from X-rays images instead ERM's to determine if a patient the illness or not.

Problem Statement

The problem some can be help with machine learning, in this case, is to diagnose the partitions some have Pneumonia in their lungs out for X-ray images of the patients. The use for such an algorithm can be to help out doctors with determining if a patient has Pneumonia or not based on finding patterns in the X-ray images some might be a challenging thing to find for a human eye in some cases and base on that can help out patients to get a correct diagnosis.

Metrics

For preforms metrics have I used the loss of the validation set some will be checked for every epoch my model has run with the model checkpoint callback [4] some save the best weight of my model to a file every time the loss is going down for the validation set. I have then used the model load function to load all the save weights to my model and used Keras model evaluate function to see the accuracy and loss of the testing set for measure the performers of my model.

II. Analysis

Data Exploration

The data some I Have used is form Kaggle.com (https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia) it is X-ray images there are "selected from retrospective cohorts of pediatric patients of one to five years old from Women and Children's Medical Center in Guangzhou. All chest X-ray imaging was performed as part of patients' routine clinical care." [5]

"The dataset is organized into 3 folders (train, test, val) and contains subfolders for each image category (Pneumonia/Normal). There are 5,863 X-Ray images (JPEG) and 2 categories (Pneumonia/Normal)" [5]

and each category contains:

- Train 5,219 Files, Normal 1,342 Files, Pneumonia 3,876 Files, .DS_Store 3 File
- Val 19 Files, Normal 8 Files, Pneumonia 8 Files, .DS_Store 3 File
- Test 625 Files Normal 234 Files, Pneumonia 390 Files, .DS_Store 1 File

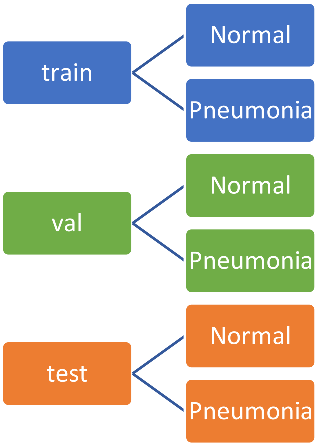
all the images are 1 color dimensions (black and white). The images have all different dimensions so there is not a default resolution for the images in the dataset. [5]

The original data is from data.mendeley.com and the Contributor(s) is: Daniel Kermany, Kang Zhang, Michael Goldbaum(https://data.mendeley.com/datasets/rscbjbr9sj/2)

Exploratory Visualization

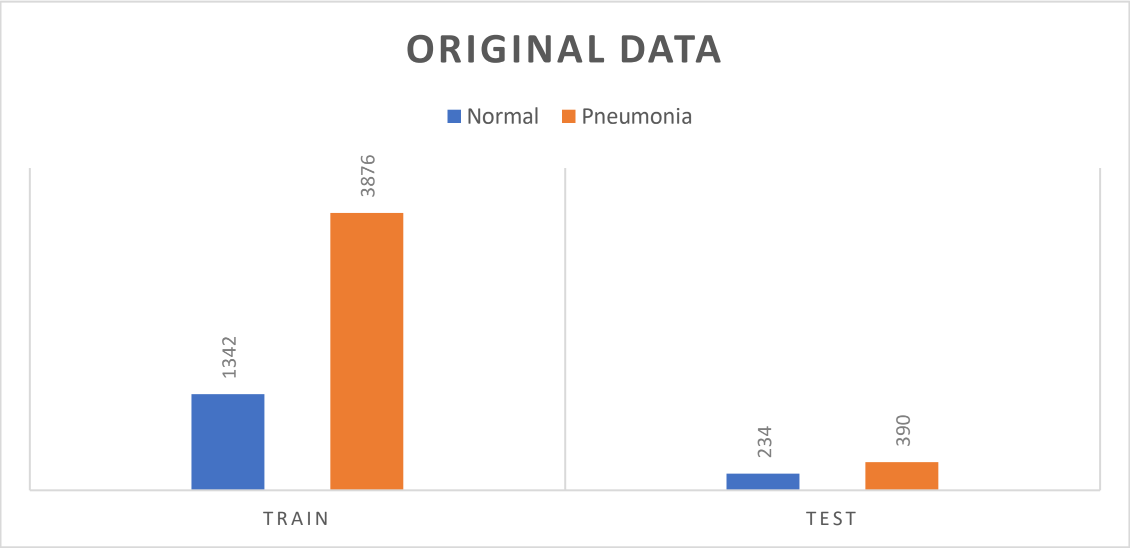
The data I am using is structured in to 3 different folders(train, Val and test) with 2 categories in each(Normal and Pneumonia) the reason for this is because the data will be used with Keras data generators [7] some will take each folder and create batches of all the data in each category based on the folder structure and the subfolders(Normal and Pneumonia) to each top directories(train, Val and test). This is a need way to structure the data and simultaneous label the data instead of having all the data in one folder and label and path in a CSV file some can be much more confusing to work with and less forgiving to changes of the data structure for preprocessing.

The structure I am using looks like this:



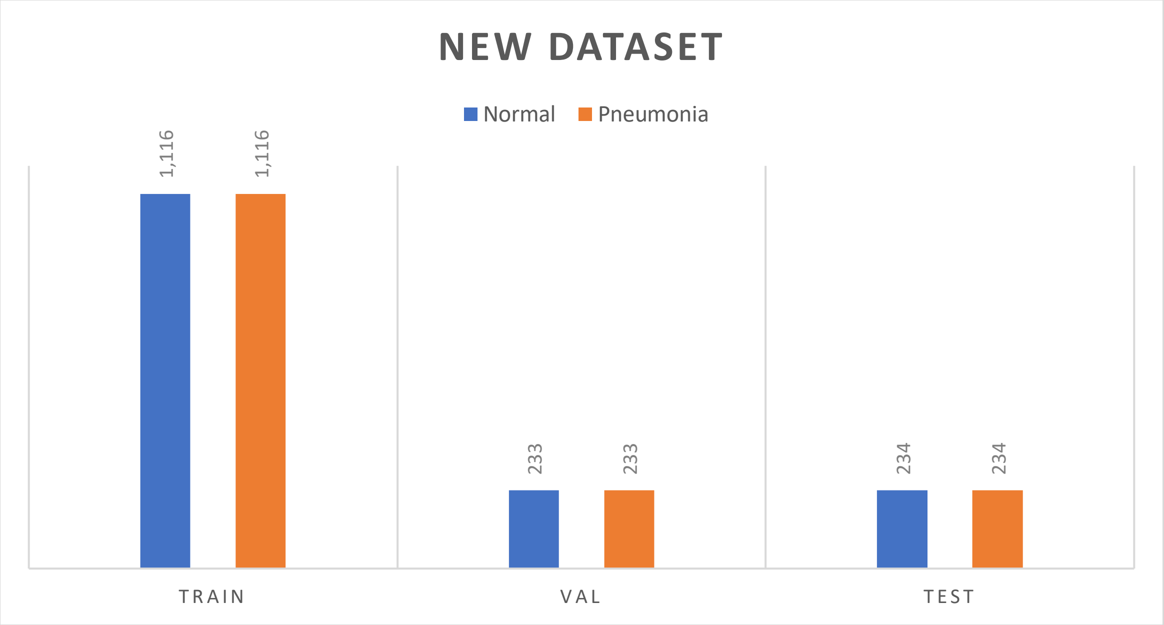
The data I am using is quite unbalanced where most of the images are in pneumonia category where only the validation set balanced with 8/8 samples. This is a problem when training the model because the model will have a chance to be unbiased towards the Normal category, therefore, will I downscale the number of images to the Normal category so the images will have 50/50 in each folder.

The original data for train and test looks like this:



Another problem some there is with the data are there are only 16 images in total for validation where there are 8 in each folder so the data are not balanced but the problem is when training the model will the model compare its result to the validation set and compute a result but the model will not have enough data to compare with from the validation set so the model will fast learn the 16 images there are and thereafter just get stuck on 100% accuracy for the validation set or get worse. So, to fix this problem will I take 225 images of each category in the training set so the new validation set will be similar to the test set in size.

So, after balancing train and test sets and adding 225 images to the validation set from train folder will the new dataset look like this:



Algorithms and Techniques

The model I have used consist of 2x4 CNN layers with batch normalization and with max poling layers after every 2 CNN layers. I have then use one flatten layer so the tensor will be converted to 1D tensor before the output layer.

The reason for I have used batch normalization and max-pooling layers in my neural network are:

Batch normalization helps the input data to genialize better with normalizes the input to the activation function in the neural network with standardizing the learning process with normalizing mean to the activation function close to 0 [8].

Max-pooling is for bringing down the data size with down scale the input data and their bye help to bring down the training time [9].

Flatten is for covering the data to a 1D tensor so the output dense layer will accept it when feeding it from the CNN layers.

For making the model structure have I used Keras library some looks like: [6]

Conv2D input	Filters 32, input shape (64 ,64 ,1)
Batch Normalization	
Conv2D	Filters 32
MaxPooling2D	pool size (2, 2)
Conv2D	Filters 64
Batch Normalization	
Conv2D	Filters 64
MaxPooling2D	pool size (2, 2)
Conv2D	Filters 128
Batch Normalization	
Conv2D	Filters 128
MaxPooling2D	pool size (2, 2)
Conv2D	Filters 256
Batch Normalization	
Conv2D	Filters 256
MaxPooling2D	pool size (2, 2)
flatten	

the reason for I have chosen this model structure is because the original structure from my proposal doesn't live up to the performers I have hoped some takes inspection form VGG16 [11] model so I look into newer model structures and found Resnet 50 [12] model structure so I made a similar version and drop the 2 dense layers and change them out with 2 conv2D layers for better performers.

Benchmark

For benchmarking the model have I chosen F-beta2 score some should work well on balance data some as well on unbalancing data so I can get the model results on both the balance test set I have made and the original for Klagge [5] and compare the results. I have also used Precision, Recall, and accuracy so it will be possible to see where the model will lake in term of detecting false positives, true positives and false negatives and true negatives. The evaluation metrics are from Sklearns library [13] some I have used for benchmarking.

III. Methodology

Data Preprocessing

Balancing the data

The first step in preprocessing the data was to balance all the data. I have done this with OS, glob and Shutil library.

- Glob for importing the file path to the images [14]
- OS for creating new directories [15]
- Shutil for copy and moving files [16]

First, I scan all the files from the dataset to see the number of images there are in each directory (Train, Val, and test) and subdirectories (Normal and Pneumonia) my output looks like this:

```
Files in chest_xray directory

Train image files.

Normal: 1341 Pneumonia: 3875 in total 5216

validation image files.

Normal: 8 Pneumonia: 8 in total 16

Test image files.

Normal: 234 Pneumonia: 390 in total 624
```

thereafter created I a function there will balance any of the directories I put into it. the function is structured as followed:

1. Importing subdirectory paths (Normal and Pneumonia)
2. Making an if statement some will check if “Normal” directory has fewer images in it or the same number as “Pneumonia” directory
3. If “Normal” have less run a while loop some will run for the number of images there is in “Normal” directory
4. In the while loop copy images from “Normal” and “Pneumonia” to a new directory, some are given to the function.

5. else if “Pneumonia” have less in Normal repeat steps above just with the count number for copying the images is the same as “Pneumonia” directory size instead of the “Normal” directory size.

after creating my function for balancing the data. Created I all the new directories for the balance data set (Train, Val, and Test) and all the subdirectories (Normal and Pneumonia) for each top directory.

then I used the balance function for all the subdirectories and copy all the images over from the original data to the new directories. So now all the subdirectory has the same number of images in each subdirectory as the related as subdirectory has (Normal or Pneumonia).

Lastly, I made a while loop some will run for 225 times and move images from the new Train directory to Val. So, the Validation set will now have 225 more images in each subdirectory and then I printed out the new directory file count for the balanced data set. The output looks like this:

```
Files in balanced_chest_xray directory

Train image files.

Normal: 1116 Pneumonia: 1116 in total 2232

validation image files.

Normal: 233 Pneumonia: 233 in total 466

Test image files.

Normal: 234 Pneumonia: 234 in total 468
```

Converting images to Tensors

The next step for preprocessing the data is to convert all the data into batches of tensors “A tensor is a generalization of vectors and matrices to potentially higher dimensions. Internally,” cited from: https://www.tensorflow.org/guide/tensors The reason for this is so the neural network can read the images so it need to be in vector from. I have done this with Keras image Data generators [6] with a method some can be used on the data generators some is flow from directory [6] some does:

Image data generator

Is a function there generates tensors of data in batches with Real-time data augmentation [6].

Flow from directory

Flow from directory method takes the path from a directory and generates batches of data from it [6].

I then used image data generator to generate all the batches from the new balanced data set and made a generator for every directory(Train, Val, Test) I then loaded the images from their file path with flow from directory method and there categories(Pneumonia and Normal) in to batch with these parameters:

Train (Target_size = 64x64, Classes= Pneumonia, Normal, Color_mode = grayscale, batch_size=16, shuffle=True)

Val (Target_size = 64x64, Classes= Pneumonia, Normal, Color_mode = grayscale, batch_size=1, shuffle=False)

Test (Target_size = 64x64, Classes= Pneumonia, Normal, Color_mode = grayscale, batch_size=1, shuffle=False)

The parameters mean:

Target_size

Target size are the size of the images in each batch when there are converted to tensors.

Classes

is the categories or labels of the data (Pneumonia and Normal)

Color_mode

are the mode of color the images will have when being converted e.g. grayscale (black and white) or RGB (full color).

Batch_size

the number of data point in each batch

Shuffle

randomized order in with the images are loaded.

The reason for I have chosen these parameters:

Target_size

my reason for loading the images in 64x64 size is because the image needs to be small when training on them for making the process faster. So, I choose 64x64 because I find the size to get the right balance between time and quality.

Batch_size

I have chosen 16 for batch size some are the half of the standard parameter 32 because I find the neural network to works better under training with 16 batches for training instead of standard 32 out from training runs. I have chosen 1 for Val and Test because when loading the data can I use the len function [16] some count the length of an object so I can be sure to have all the images loaded in ones.

Shuffle

the reason I have used shuffled on the training set is because when training the data will the data be in a different order for every batch so the network is forced to learn the patterns in the images instead of the order in which there come from the directory so it helps the network to generalize better when training.

applying augmentation

I have chosen to use data augmentation [7] on the training set because there is not a lot of data point so for helping the neural network not to overfit on the data have, I used augmentation with these parameters:

(rotation_range=10, height_shift_range=0.2, width_shift_range=0.2, horizontal_flip=True)

Some does:

rotation_range

rotate the image in a degree randomly with a degree as input.

height_shift_range and width_shift_range

shift the image to a side accordingly to the total wide of the image with a wide as input.

horizontal_flip

flips the image horizontally.

I have chosen not to augment the data too much so I have only rotated and shifted the images slightly so it will not be the same image as the original some should help the network to treat it as a new image and there bye add more different examples to the training set.

Implementation

Building the model

For my image detection, neural network have I used a Convolutional Neural Networks some are network some have a sequence of layers, of Convolutional Layers(CNN layers) and can have fully connected layers also call dense layers but not in my case where I have chosen the more modern version with only using CNN layers.

in my neural network have I chosen to use 8 CNN layers. The reason I have chosen 8 CNN layers for my model is because I find it was a good balance of layers to pick up all the patterns I need for max performers of the X-ray images but in the same time hold down the cost of the network in term of process power and time.

I have chosen to use 32 filters as the 2 first convolutional layer and thereafter double up the filters for every 2 new convolutional layers (32, 64, 128 and 256). The reason for that is because the first filter size need to be small for picking up more hard pattern some shapes and edges for mapping out the object so I picked the standard size 32 from Keras models [17] because I find it will give me the best result out from testing the model with training runs. I then add double the amount of filters for the next 2 CNN layers for detecting finer details and then keep doing this for every 2 CNN layers out of the 8 for detecting the most complex shapes from the X-ray images and then flatten all the data to the output layer so it will accept the data.

I have chosen a stride of 3x3 of all the layers because I find it the give me the best result after tuning the model.

I have used max poling layers with a pool size of 2x2 half the size for every 2 CNN to keep down the time and processor power and used batch normalization in-between every 2 CNN layers because I find the model to yield better results with normalized data. I have used Keras Sequential model API for the model structure [14]

Compiling and fitting the model

for compiling the model have I used Keras compile [14] and for the parameters have I used "Adam" for the optimizer with the loss function binary cross entropy [15]. The reason I have used "Adam" is because "Adam" often give some of the best results and is simple to implement and are lightweight so it is also one of the fastest optimizers out there and can handle big and small dataset as well some shown in this article where Adam gets compared to some of the most popular optimizers: https://medium.com/octavian-ai/which-optimizer-and-learning-rate-should-i-use-for-deep-learning-5acb418f9b2

the reason I used binary cross entropy [15] because I only use 2 categories and the categories I use are one hot encoded to 0 and 1 so it will be the most ideal choice to pick the binary version of cross entropy. The reason for I have chosen cross entropy for the loss function is because out from test run I self-have made have I gotten the best result from cross entropy out of the 3 most commonly used loss function some are hinge, mean squared error and cross entropy [15] where cross entropy score a lot higher in the other 2.

For fitting the model to the data have I used Keras model.fit_generator [7] with these parameters:

step_per_epoch = 155

validation_steps = len(test set)

callbacks = ModelCheckpoint

epochs = 100

The reason for I have chosen these parameters is because:

step_per_epoch

The step I have chosen is 155 some is the number of batch times the number of the data point. The reason for this is when running the model will the model load all the data in for every epoch, so all the data get used.

validation_steps

for the validation steps have I chosen to use len function on the test set so I will only get the whole data set and noting more or less to insure I get the right results.

callbacks

I have used model check point as callback [4] to get all the best weight for the model and save them to a file. This helps with only get the best run of the epochs so I can run the model for longer time without overfitting.

Epochs

I used 100 epochs because I find the model to even out about 80-100 epochs for experience from all the training runs, I have made with fine-tuning the model.

Refinement

For the refinement of the model have I changed somethings compared to propose model some are the 2 dense layers at the end of my model. The reason for that is because I don't find my model to get the performers, I have hope for with accuracy on 76% on the test set where my goal for my model is 85%. So I look into newer models some Resnet 50 [12] and took inspection out from that with only using CNN layers some have the advantages to not overfit as much because CNN layers are not fully connected and therefore also do not take up nearly as much process power so the model will also be faster compared to the original. So, I change the layers to CNN layers and turned the parameters to the right balance and my result got much better with accuracy on 88% from the new model on the balance test set.

For turning the model have I tuned a couple of things. The way I turned the model was by running the model for 25 epochs and then look for the result per parameter I have tried out and then change the parameter to the best one out from the results the model gave me where I used Keras evaluate generator function [14] to do that with some shows the loss and accuracy on the test data. I have done this 2 time per parameter because the model can show different result for a run to run but not really big changes, so it was only to ensure I got the best results. It might not be the best way to tune the model with, but it has worked fine for me, so I have not tried out more complex methods.

the parameters I have tried out for the model are:

filters for the last 2 layers (2 x 128, 2 x 256, 2 x 516) the best result was 128 filters

kernel_size ((1, 1), (2, 2), (3, 3), (4, 4), (5, 5)) the best result was (3, 3)

strides ((1, 1), (2, 2), (3, 3)) the best result was (1, 1)

the parameters I have tried out for the loss function are:
hinge, mean squared error and cross entropy where cross entropy gave me the best result and most stable under the training tests.

IV. Results

Model Evaluation and Validation

I will say the model is appropriate fitted and can produce good results out from many tests runs of different parameters for finding the best ones for the task. For the final benchmark test have I used the test set for the balance data set and the original test set to get the most accurate results of each categories and the balance test set is 20% in size compared to the training set so there should be enough images to determine accurate results out of the to test sets I have used benchmark test on where I have used Precision and recall to see the model ratio of False negative and False positive vs True positive and False negative where the balance model preform equally as good on both Precision and recall and the original test set and the original have gone down in Precision to 79% but have the same result for recall. I have then used F-beta score to see the combined result of Precision and Recall for both models where the balance test set has 88% and original test set have 87% so not a big difference in terms of overall performances of the to test sets. The beta I have used for the F-beta score are 2 some should weight precision more in recall, so the model is not perfect in terms of precision but still produces some good results with only 1% in difference in overall performance between the two test tests out from the F beta score.

Justification

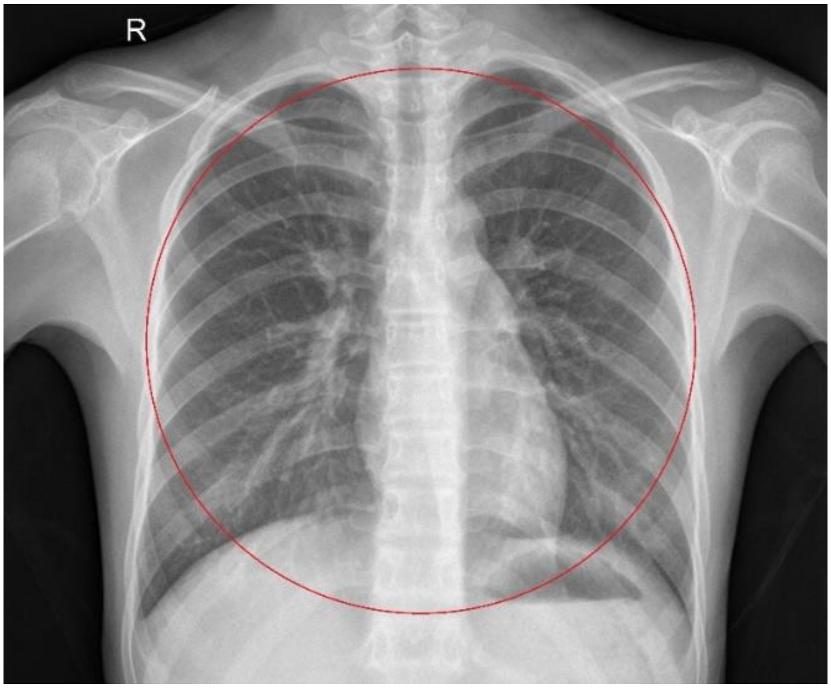
I will say F beta score is a good measurement to justify the preforms of the model because it not only shows the models performs in predicting the data set correct but also shows the ratio of False negative and False positive vs True positive and true negative in precision and recall some are weighted between each other to one result out from a beta value some can be changed to fit the user needs to favor precision over recall or the other way around or equally weighted some will give a more realistic result instead of accuracy alone.

I will say my model can be justified to have successfully solved the problem it has been given out from the results it can produce of the test sets and have been tested on many different images with multiple measurements with good results.

V. Conclusion

Free-Form Visualization

One key aspect about the model is it are self-learning and can find objects out from patterns some relate to Pneumonia disease and filter all other patterns apart some will be a tricky thing to do in hard coding itself but the network has done it bye its own in about 20 min training session some I think is one of the strongest features about the neural networks and model can do this only out from the categories of the images and the image itself. Like in this project if you look on the image are there many things some have nothing to do with Pneumonia disease itself some the shoulders and bones in the body and a little noise but the network have successfully find the patterns in the lungs of the images to predict patients with Pneumonia disease with a good result and thereby have the potential to help other people in the future.



Reflection

The challenges in this project was to take X-ray images and predict if a patient has suffered from Pneumonia disease in there lunges with a neural network some will learn the patterns in the image and find the differences there are between a healthy patient and ill patient with Pneumonia disease. This is done out from X-ray images some are already decided if the person has the illness or not and these images will the neural network find patterns in out from the pixels in the images and link these patterns together to find some unique features some can be categories between a normal person and one with Pneumonia disease.

The most interesting aspect of this project was to create the structure for the neural network and thereafter see it perform on the data some can be like magic when it works and are definitely one of the more fun parts of machine learning.

The most difficult aspects in this project was to find the right parameters some will make the best neural network because there are not some standard way to make a network there will work on all data so most of the time was it trial and error to find the perfect fit out from testing and see the results the model produced.

The final model definitely fit my expectation where I have only one goal to make a model there will make it prediction right 85% of the times some have been a long process to meet the criteria but I end up with something better so it has been worth it in the long run.

Improvement

For improving my model will it help a lot to have more data in I had in the normal category and one of the biggest problems was to balance the data because the data set was so uneven and has far from enough data point in it validation set to train the model with to get accurate results so I need to take some data point from the training set some cloud help the model to improve more and when look back will it probably be smarter to upscale the normal category instead of down scaling Pneumonia because there was so many unique examples of images in data set some properly will have help the model to learn more patterns for predicting the images. One other thing I find hard to deal with was to find the perfect parameters some cloud be done from a function like GridSearchCV [19] but Keras those not support a function like that out from my knowledge so I have not look more in to it in that but can properly be done with a custom function.

Referencer

[1] »wikipedida Pneumonia,« [Online]. Available: https://en.wikipedia.org/wiki/Pneumonia.

[2] »Deepcare research paper,« [Online]. Available: https://truyentran.github.io/papers/pakdd_16.pdf.

[3] A. d. s. c. -. S. O'Connor, »How Do EHR Systems Work,« [Online]. Available: https://www.adsc.com/blog/how-do-ehr-systems-work.

[4] »keras docmuntation - Usage of callbacks,« [Online]. Available: https://keras.io/callbacks/.

[5] P. Mooney, »kaggle,« [Online]. Available: https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia.

[6] »keras documentation - ImageDataGenerator,« [Online]. Available: https://keras.io/preprocessing/image/#imagedatagenerator-methods.

[7] »Keras documentation - BatchNormalization,« [Online]. Available: https://keras.io/layers/normalization/.

[8] »Keras documentation - MaxPooling,« [Online]. Available: https://keras.io/layers/pooling/.

[9] »keras documentation,« [Online]. Available: https://keras.io/layers/convolutional/.

[10] M. u. Hassan, »VGG16 – Convolutional Network for Classification and Detection,« [Online]. Available: https://neurohive.io/en/popular-networks/vgg16/.

[11] P. Guillou, »Understand how works Resnet,« [Online]. Available: https://medium.com/@pierre_guillou/understand-how-works-resnet-without-talking-about-residual-64698f157e0c.

[12] »scikit-learn - Model evaluation,« [Online]. Available: https://scikit-learn.org/stable/modules/model_evaluation.html.

[13] »glob — Unix style pathname pattern expansion,« [Online]. Available: https://docs.python.org/3/library/glob.html.

[14] »os — Miscellaneous operating system interfaces,« [Online]. Available: https://docs.python.org/3/library/os.html.

[15] »shutil — High-level file operations,« [Online]. Available: https://docs.python.org/3/library/shutil.html.

[16] »programiz - Python len(),« [Online]. Available: https://www.programiz.com/python-programming/methods/built-in/len.

[17] »keras documentation - Sequential models,« [Online]. Available: https://keras.io/getting-started/sequential-model-guide/.

[18] »The Sequential model API,« [Online]. Available: https://keras.io/models/sequential/.

[19] »keras documentation - Usage of loss functions,« [Online]. Available: Usage of loss functions.

[20] »sklearn - GridSearchCV,« [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html.

[21] K. Z. M. G. Daniel Kermany, »data.mendeley,« [Online]. Available: https://data.mendeley.com/datasets/rscbjbr9sj/2.

[22] A. Budhiraja, »Dropout in (Deep) Machine learning,« [Online]. Available: https://medium.com/@amarbudhiraja/https-medium-com-amarbudhiraja-learning-less-to-learn-better-dropout-in-deep-machine-learning-74334da4bfc5.

[23] stanford, »bytes and bites,« [Online]. Available: https://web.stanford.edu/class/cs101/bits-bytes.html.