RCKettel / **PhaseFour_ImageClassification**

0 stars 0 forks

Star Unwatch

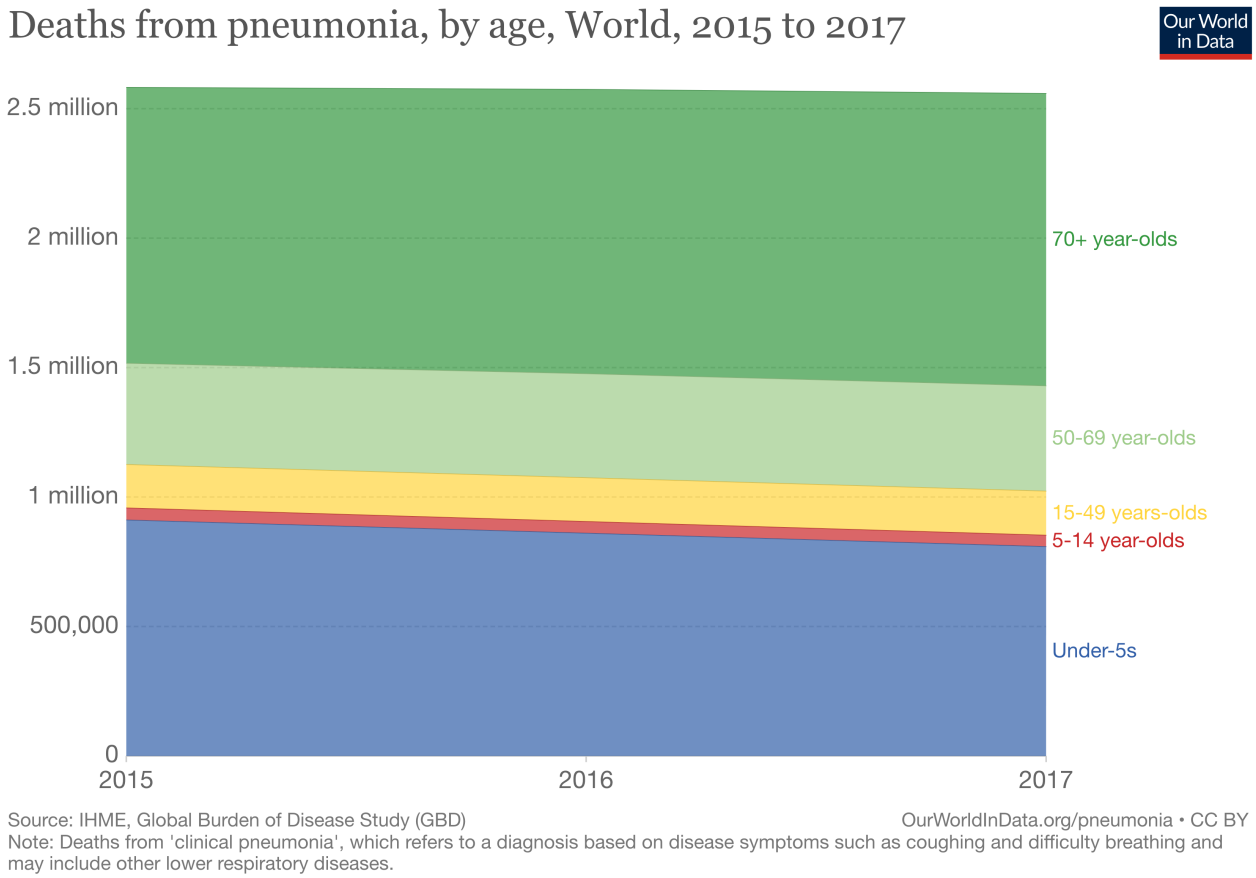
Code Issues Pull requests Actions Projects Wiki Security

main

RCKettel Update README.md ... 6 minutes ago 14

View code

README.md



Pneumonia Diagnosis Project

Directory

Data

Notebooks

References

Reports

src

Business Understanding

For this project a Convolutional Neural Network deep learning model was created for a proof of concept that would predict if a patient had lungs infected with pneumonia. The best model showed a high level of accuracy which would show the least number of results that are that would predict the patient does not have pneumonia when they actually do. Or in other words, are false negatives.

Accessing the Data

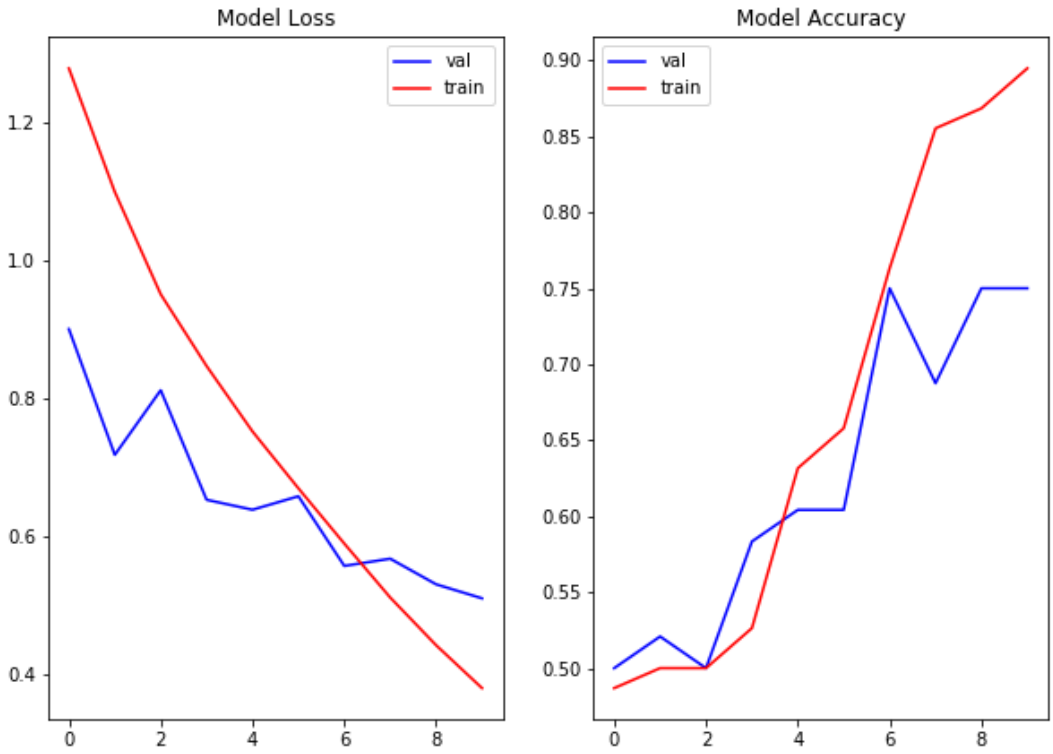
Due to the number of images and the size of the dataset its recommended that it is downloaded to a folder and keras Image Data Generator and flow_from_directory methods are used to access it. The data can be downloaded from Keras here. The data will be downloaded as a .zip file and will require a program to access it. Once opened exploration will reveal a number of redundant or empty files that contain much of the same data or no data at all such as MACOSX. These files were deleted since they were considered unnessecessary. The approach used for this project was to take the train, val, and test files from their parent folder and access them directly to expedite access and reduce the amount of code.

The Data

The data contains pediatric X-ray images of lungs from 5863 patients. These images were split into three different sets: Train, Val, Test each containing subsets: Normal, and Pneumonia. This data took very little preperation since the data was already split into training and validation sets. Due to a nearly three to one imbalance of images with healthy lungs to images with pneumonia the normal lungs set was resampled with SMOTE to better balance the data.

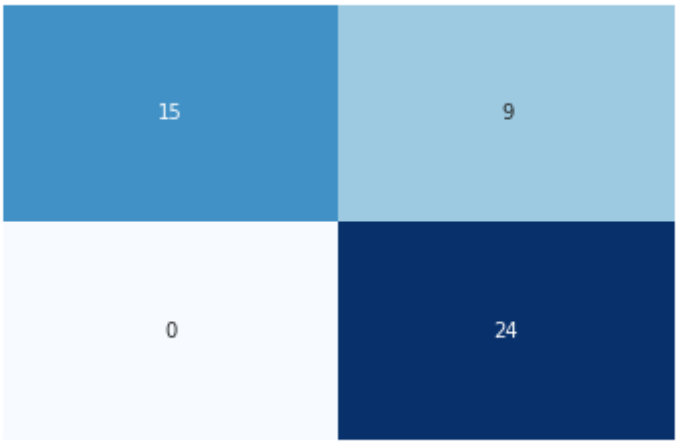
First Simple Model

The first simple model focused on accuracy and was very basic being made up of an instantiating convolutional layer followed by a layer with maxpooling and a final sigmoid layer. the activator used for the model was relu as it does better in shallow networks. The optimizer and the loss functions were the standard adam and binary crossentropy that are often used in models whose output layer activation is sigmoid. The fit method contained non-standard class weights that were calculated when instantiating the flow_from_directory method and are based on the compared datapoints of the training and target data from the training dataset. These were used to help balance the weights of the model to reduce variance.



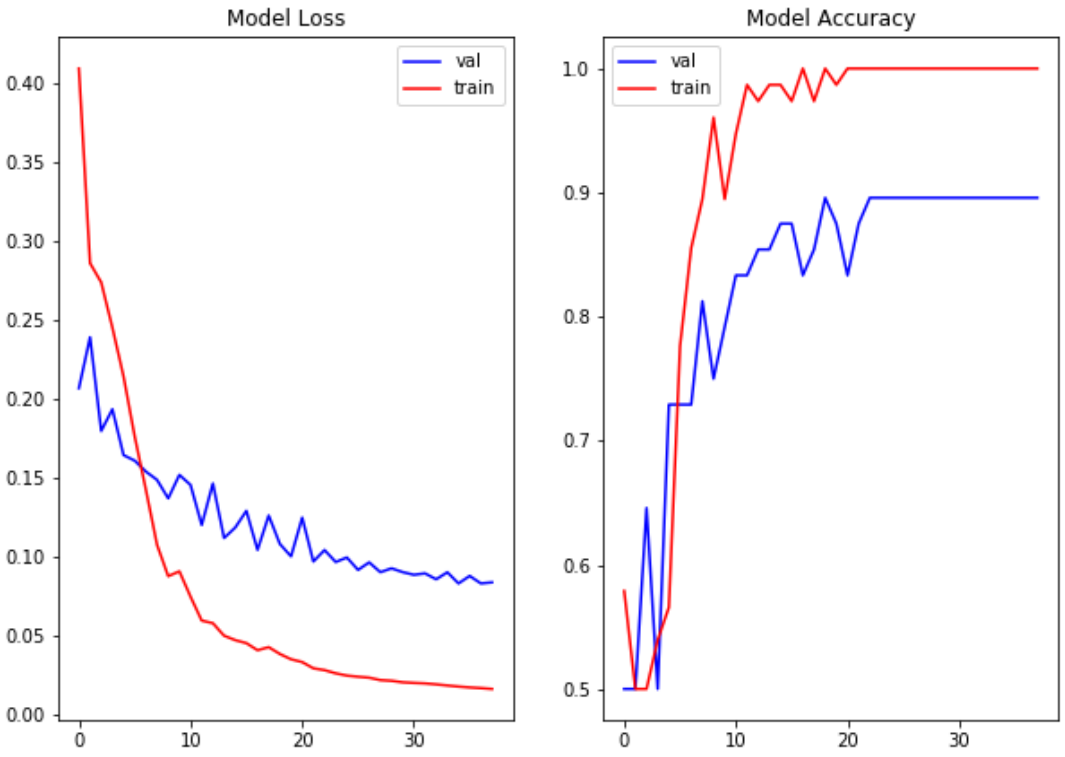
Evaluation of FSM

Though the results of the model as tested by the validation and test scores show high accuracy ratings of .71 and .76 respectively the loss in both models is fairly high, above .5. This is likely due to how the function checks against false negatives. Since the data set was balanced by SMOTE, it is likely giving high penalizations to bad predictions causing a higher loss. In addition to the high loss, the disproportionate performace by the training data as compared to the validation and test data show evidence of an overfit model. This problem is usually caused by a model that is too complicated for the data it is working with. In Convolutional Neural Networks this can be rectified by tuning the hyperparameters of the model.



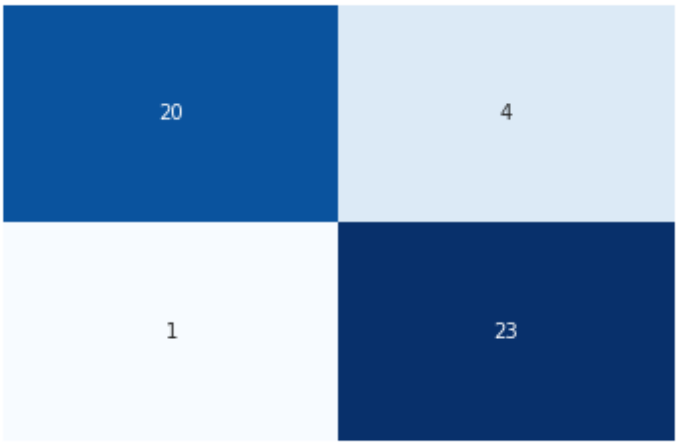
The Final Model

The final model also focused on accuracy, was made up of eight layers. The instantiating convolutional layer two hidden concolutional layers, all followed by maxpooling layers, an added dropout layer, a flattening layer and a final dense sigmoid output layer. The activation of each convolutional layer, used a tanh activation which through expreimentation turned out to yeild the best results. The dropout rate for the model ended as the standard twenty percent and the optimizer was adam as in the previous model. Since the Convolutional Neural Network didn't have an instantiated kernel, bias regularization was tested on the model but consistently yeilded bad results often lowering the accuracy of the data. For these reasons an l2 activity weight regularizer was used to penalize the outgoing activity of each layer and an l2 regularizer may lower a sigmoids activity but it wont drop it entirely. Early stopping was used as well to halt the progression of the models development before it began to degrade. Finally, a non-standard focal loss function was added to penalize the effect of difficult to learn data and allow the model to more easily learn the data all which reduced loss, one of the major issues in the FSM.



Evaluation of Final Model

This model shows much less evidence of being overfit. Though the accuracy of the training data at 1.0 is still unrealistic in comparison to the val data being at .89, the two correlate much better in the plot of thier results. The loss is also much lower as the focal loss function lowered the number of bad predictions by making the model learn the data at a more even rate. The final test scores showed the model was able to generalize to unseen data as it showed an accuracy of .80 and a loss of .126.



Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Languages

- **Jupyter Notebook** 100.0%