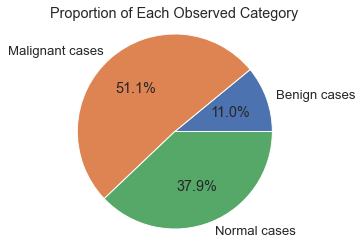
**Title**

**About the Dataset**

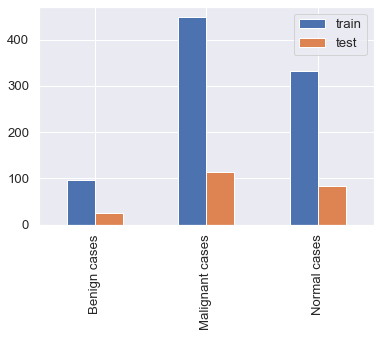
The dataset consists of three classes: benign, malignant, and normal. There are 120 CT scan images for the Benign Class, 561 for the Malignant Class, and 416 for the Normal Class. In Kaggle, the Dataset is accessible to everyone.



**Splitting Dataset**

80% of the data set is for the training set, while 20% is for the test set.

|  |  |  |
| --- | --- | --- |
| Classes | Train | Test |
| Benign | 96 | 24 |
| Malignant | 448 | 113 |
| Normal | 332 | 84 |



We have randomly shuffled the train pictures into a state of 25. Using a factor of 255.0, scale each pixel. The majority of picture data has integer pixel values between 0 and 255.

Small weight values are processed by neural networks, while high integer values might interfere with or slow down learning. Since each pixel value should range from 0 to 1, normalizing the pixel values is a good option.

To do this, divide all pixel values by 255, which is the biggest pixel value. Regardless of the actual range of pixel values that are present in the image, this is done across all channels.

**Raw Data Machine Learning**

We employed machine learning methods for image classification, including Decision Tree Classifier, Random Forrest Classifier, Extra Tree Classifier, KNN, XG Boosting, Logistic Regression, and Support Vector Machine, all with default parameters.

Precision Recall and F1 Score were used for the classifications report.

We applied Mean Absolute Error, Mean Squared Error, and Root Mean Square Error for the evaluation matrix.

The end outcome of using machine learning algorithms is as follows.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Algorithms | Precision | Recall | F1Score | MAE | MSE | RMSE | Accuracy |
| DT | 0.876 | 0.869 | 0.871 | 0.181 | 0.281 | 0.529 | 86.878% |
| RF | 0.958 | 0.955 | 0.951 | 0.086 | 0.167 | 0.409 | 95.475% |
| KNN | 0.969 | 0.968 | 0.967 | 0.054 | 0.099 | 0.316 | 96.833% |
| ET | 0.991 | 0.991 | 0.991 | 0.014 | 0.023 | 0.150 | 99.095% |
| XGB | 0.996 | 0.995 | 0.995 | 0.005 | 0.005 | 0.067 | 99.548% |
| LR | 1.000 | 1.000 | 1.000 | 0.000 | 0.000 | 0.000 | 100.00% |
| SVM | 0.965 | 0.964 | 0.963 | 0.059 | 0.104 | 0.323 | 96.380% |

**Table**: Classification Result Table of Raw Data Machine Learning Result

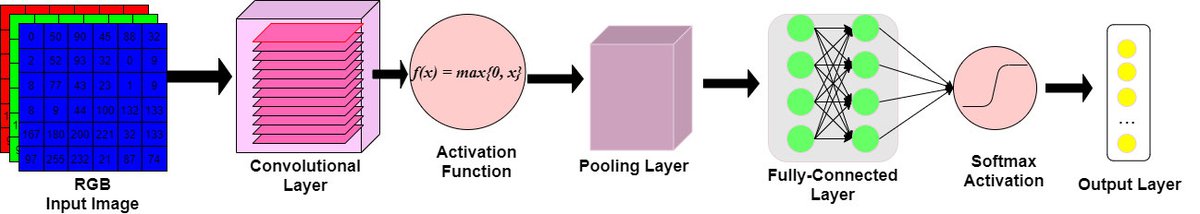
**Raw Deep Learning Architecture**

Prior to deep learning, we added two CONV layers with an input layer of (224,224,3) and elementwise activation function of Relu on each, commonly known as a Rectified-Linear Unit (ReLu). A specific input node's ability to "fire" will be decided by the ReLu layer. If the filters in the convolution layer have picked up a visual characteristic, this "firing" indicates it. ReLu functions operate by applying a function and thresholding at 0.

Followed by two Maxpooling layers of (2,2) a down-sampling strategy is applied to reduce the width and height of the output volume.

Then, after adding a flatten layer, two dense layers with 128 neurons and three neurons output respectively were added.

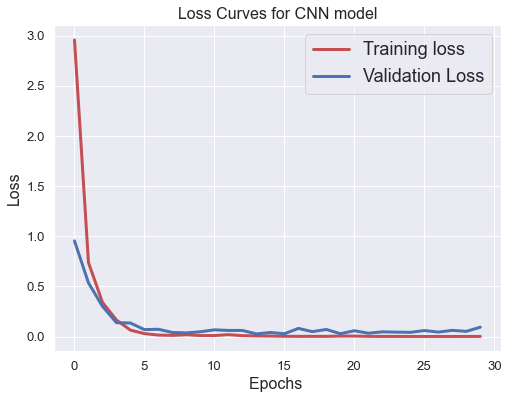
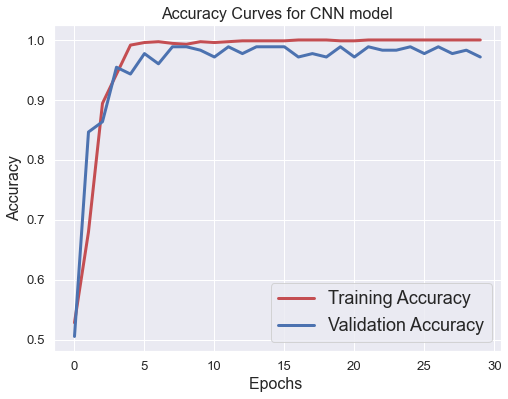
We have chosen the optimizer Adam and the loss function sparse categorical cross entropy with a batch size of 64 for the compilation stage.



**Figure**: CNN Architecture

The Classification report is as follows:

|  |  |
| --- | --- |
| Parameters | Score |
| Training Accuracy | 100.00% |
| Test Accuracy | 98.190% |
| Training Loss | 0.001 |
| Test Loss | 0.044 |
| Precision | 0.983 |
| Recall | 0.982 |
| F1 Score | 0.981 |
| MAE | 0.036 |
| MSE | 0.072 |
| RMSE | 0.269 |



**Feature Extraction**

We have employed two methods, such as processed data and transfer learning feature extraction, for the feature extraction portion.

**Processed Data:**

1. **About Processed Data**
2. **Work flow of the processed Data**
3. **Methods**
4. **Before and After Processed DATA Images**

The end outcome of using machine learning algorithms of Processed Data is as follows.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Algorithms | Precision | Recall | F1Score | MAE | MSE | RMSE | Accuracy |
| DT | 0.897 | 0.896 | 0.896 | 0.136 | 0.199 | 0.446 | 89.593% |
| RF | 0.962 | 0.959 | 0.955 | 0.072 | 0.136 | 0.368 | 95.928% |
| KNN | 0.935 | 0.937 | 0.935 | 0.104 | 0.186 | 0.431 | 93.665% |
| ET | 0.969 | 0.968 | 0.966 | 0.054 | 0.099 | 0.316 | 96.833% |
| XGB | 0.971 | 0.968 | 0.967 | 0.059 | 0.113 | 0.336 | 96.833% |
| LR | 0.978 | 0.977 | 0.977 | 0.041 | 0.077 | 0.277 | 97.738% |
| SVM | 0.966 | 0.964 | 0.961 | 0.068 | 0.131 | 0.362 | 96.380% |

**Table**: Classification Results of Machine Learning using Processed Data as feature Extraction

The end outcome of using Deep Learning algorithms of Processed Data is as follows.

|  |  |
| --- | --- |
| Parameters | Score |
| Training Accuracy | 100% |
| Test Accuracy | 94.57% |
| Training Loss | 0.00019783 |
| Test Loss | 0.259 |
| Precision | 0.945 |
| Recall | 0.946 |
| F1 Score | 0.945 |
| MAE | 0.090 |
| MSE | 0.163 |
| RMSE | 0.404 |

**Transfer Learning Feature Extraction Method:**

**Transfer Learning**

A model that has been trained for one job is repurposed for a different, related task using the machine learning approach known as transfer learning.

When modeling the second task, transfer learning is an optimization that enables quick advancement or increased performance.

In transfer learning, the learned features are first applied to a base network that is trained on a base dataset and task, and then the learned features are transferred to a second target network that is trained on a target dataset and task. If the traits are general—that is, applicable to both the base task and the target task—rather than task-specific, this procedure is more likely to succeed.

[[1411.1792] How transferable are features in deep neural networks? (arxiv.org)](https://arxiv.org/abs/1411.1792)

[A Gentle Introduction to Transfer Learning for Deep Learning (machinelearningmastery.com)](https://machinelearningmastery.com/transfer-learning-for-deep-learning/)

**VGG16 Model**

[ImageNet (image-net.org)](https://www.image-net.org/)

[[1409.1556] Very Deep Convolutional Networks for Large-Scale Image Recognition (arxiv.org)](https://arxiv.org/abs/1409.1556)

[VGG16 architecture (opengenus.org)](https://iq.opengenus.org/vgg16/)

Convolutional neural network VGG16 was trained using data from a portion of the ImageNet dataset, which consists of approximately 14 million images divided into 22,000 categories. This model was proposed in the 2015 study, Very Deep Convolutional Networks for Large-Scale Image Recognition, by K. Simonyan and A. Zisserman.

VGG16 earned a classification accuracy of 92.7 percent in the 2014 ImageNet Classification Challenge. The fact that it has been trained on millions of photos is more significant.

The architecture of the pre-trained model can no longer be used in its entirety. While our Target Dataset only has 3 classes for prediction, the Fully-Connected layer produces 1,000 possible output labels. In order to "chop off" the Fully-Connected layer, commonly known as the "top" model, we will import a pre-trained model like VGG16.



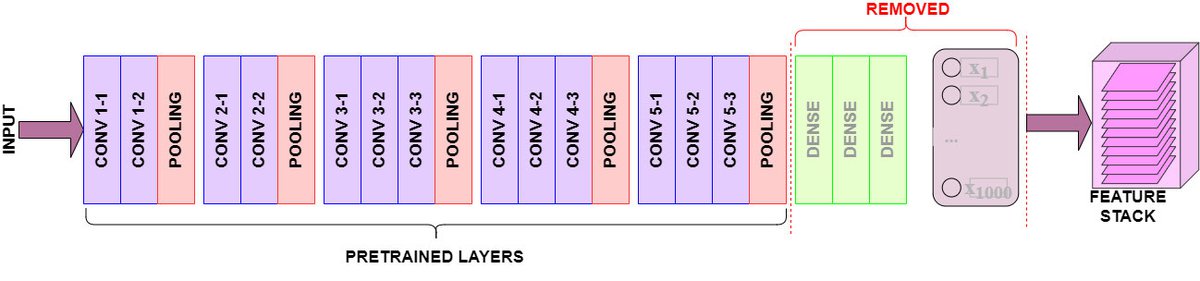
**Figure**: VGG16 Architecture

**Feature Extraction Approach**

In this technique, we build a new dataset from our input photos using the architecture of the pre-trained model. The Convolutional and Pooling layers will be imported; however, the model's "top part" will not be imported (the Fully-Connected layer).

Recall that the VGG16 model we used as an example was trained on millions of photos. It can recognize universal characteristics due to its trained weights and convolutional layers.

Our images will be sent through the convolutional layers of VGG16, which will provide a Feature Stack containing the recognized visual features. The 3-Dimensional feature stack can now be quickly flattened into a NumPy array before we apply our own Convolutional Neural Network Architecture to classify Lung Cancer.



**Figure**: Transfer Learning Feature Extraction Architecture

<https://www.learndatasci.com/tutorials/hands-on-transfer-learning-keras/>

The end outcome of using Machine Learning algorithms of Transfer Learning as feature Extraction is as follows.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Algorithms | Precision | Recall | F1Score | MAE | MSE | RMSE | Accuracy |
| DT | 0.969 | 0.968 | 0.969 | 0.045 | 0.072 | 0.269 | 96.833% |
| RF | 0.987 | 0.986 | 0.986 | 0.027 | 0.054 | 0.233 | 98.643% |
| KNN | 0.969 | 0.968 | 0.967 | 0.063 | 0.127 | 0.356 | 96.833% |
| ET | 0.991 | 0.991 | 0.991 | 0.018 | 0.036 | 0.190 | 99.095% |
| XGB | 0.996 | 0.995 | 0.995 | 0.009 | 0.018 | 0.135 | 99.548% |
| LR | 1.000 | 1.000 | 1.000 | 0.000 | 0.000 | 0.000 | 100.00% |
| SVM | 0.971 | 0.968 | 0.966 | 0.063 | 0.127 | 0.356 | 96.833% |

**Table**: Classification Results of Machine Learning using Transfer Learning as feature Extraction

The end outcome of using Deep Learning algorithms of Transfer Learning as feature Extraction is as follows.

|  |  |
| --- | --- |
| Parameters | Score |
| Training Accuracy | 100.00% |
| Test Accuracy | 95.928% |
| Training Loss | 0.00079309 |
| Test Loss | 0.103 |
| Precision | 0.962 |
| Recall | 0.959 |
| F1 Score | 0.960 |
| MAE | 0.077 |
| MSE | 0.149 |
| RMSE | 0.386 |

**Table**: Classification Results of Deep Learning using Transfer Learning as feature Extraction

**Comparative Analysis**

|  |  |  |  |
| --- | --- | --- | --- |
| Algorithms | Raw Data | Processed Data | Transfer Learning |
| CNN | 0.981 | 0.945 | 0.960 |
| DT | 0.871 | 0.896 | 0.969 |
| RF | 0.951 | 0.955 | 0.986 |
| KNN | 0.967 | 0.935 | 0.967 |
| ET | 0.991 | 0.966 | 0.991 |
| XGB | 0.995 | 0.967 | 0.995 |
| LR | 1.000 | 0.977 | 1.000 |
| SVM | 0.963 | 0.961 | 0.966 |

**Table:** F1 Score Comparison

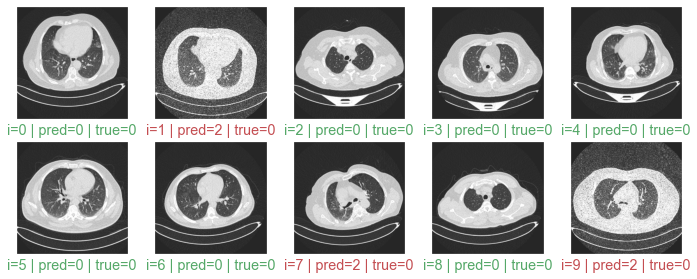
**Figure:** Bar Plot of F1 Score Comparison

|  |  |  |  |
| --- | --- | --- | --- |
| Algorithms | Raw Data | Processed Data | Transfer Learning |
| CNN | 0.269 | 0.404 | 0.386 |
| DT | 0.529 | 0.446 | 0.269 |
| RF | 0.409 | 0.368 | 0.233 |
| KNN | 0.316 | 0.431 | 0.356 |
| ET | 0.150 | 0.316 | 0.190 |
| XGB | 0.067 | 0.336 | 0.135 |
| LR | 0.000 | 0.277 | 0.000 |
| SVM | 0.323 | 0.362 | 0.356 |

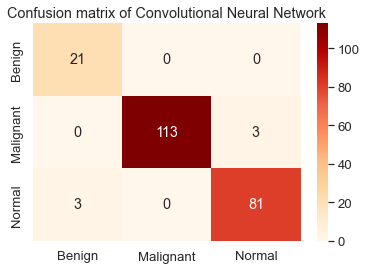
**Table:** RMSE Comparison

**Figure:** Bar Plot of RMSE Comparison

**Prediction Results of Convolutional Neural Network**

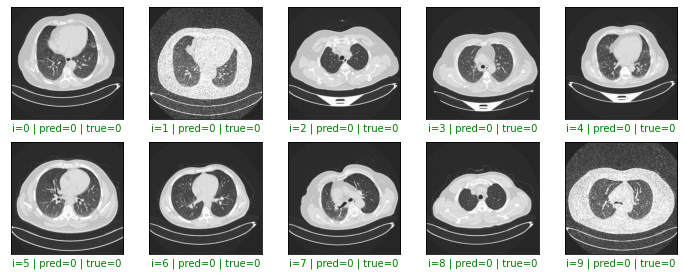


**Figure:** Prediction results of Convolutional Neural Network

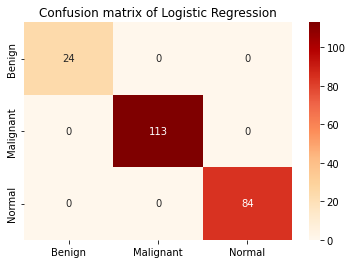
****

**Figure:** Confusion Matrix of Convolutional Neural Network

**Prediction Results of Logistic Regression**

****

**Figure:** Prediction Results of Logistic Regression

****

**Figure:** Confusion Matrix of Logistic Regression