**3.2.1 Scaling and normalization  
  
14°) In this preprocessing, the CT scan data in jpg format are loaded in BGR color space. Then, the BGR color format is interpreted and converted to RGB format.  
  
→ What does this BGR guy look like? Why didn't you convert directly to RGB?**

The decision to initially load the CT scan data in BGR format and subsequently convert it to RGB stems from historical considerations. OpenCV adopted the BGR format due to its popularity among camera manufacturers and image software providers. This choice became a convention that the developers decided to maintain.

The initial representation in BGR format aligns with the native output of some hardware, making the loading process more efficient. Additionally, image processing libraries like OpenCV are optimized for handling BGR images.

However, RGB is the most widely used color space in image display, deep learning models, and various software tools. Converting to RGB ensures consistency and compatibility across different applications and platforms.

In conclusion, the decision to initially load the CT scan data in BGR format and later convert it to RGB is grounded in historical conventions, hardware compatibility, and optimization for image processing libraries.

**References:**

**Gonzalez, R. C., & Woods, R. E. (2018). Digital image processing (4th ed.). Pearson.:** <https://dl.icdst.org/pdfs/files4/01c56e081202b62bd7d3b4f8545775fb.pdf>

**OpenCV documentation: Color Spaces.:** <https://docs.opencv.org/3.4/df/d9d/tutorial_py_colorspaces.html>

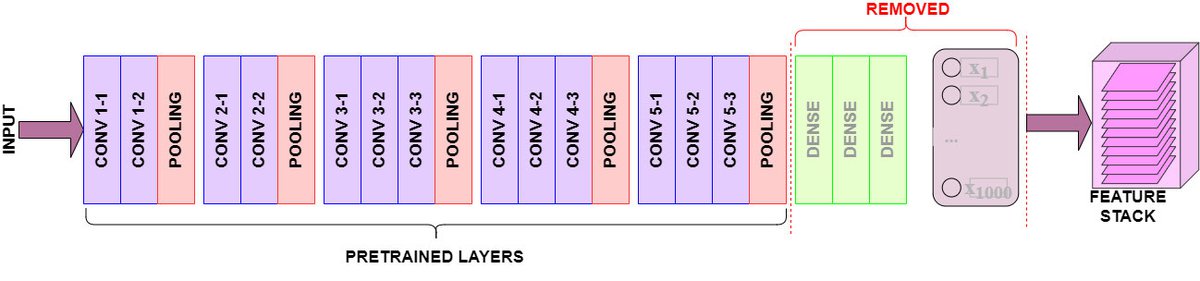
**Bradski, G., & Kaehler, A. (2008). Learning OpenCV: Computer vision with the OpenCV library. O'Reilly** **Media, Inc.:** <https://www.oreilly.com/library/view/learning-opencv/9780596516130/>

<https://learnopencv.com/why-does-opencv-use-bgr-color-format/>

<https://stackoverflow.com/questions/367449/what-exactly-is-bgr-color-space>

**4 Feature Extraction  
  
16°) Figure 4  
→ The VGG16 model has only two convolution layers + max pooling, two convolution layers + max pooling? The VGG16 architecture has more layers. It would be nice to put the entire architecture in the figure. The right thing is max-pooling, not just pooling. The resolution of the text is poor, it should be improved to make it clearer.**

Following is the updated figure:



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

**4.1 Transfer learning for Feature extraction  
  
17°) VGG16 uses just 3×3 convolution layers and 2×2 pooling layers repeatedly, which makes it significantly less complex than other transfer learning models like InceptionNet and enables it to generalize and adapt more effectively to a larger variety of data sets.  
  
→ Do you have any references to support this statement?**

Thank you for your insightful query.

The VGG16 architecture is indeed characterized by its uniformity in using 3x3 convolutional layers, stacked in increasing depth. This uniformity and the relatively small convolutional window size contribute to its lesser complexity compared to models with larger or varying convolutional window sizes.

**Reference:**

**"Very Deep Convolutional Networks for Large-Scale Image Recognition" (2014)”,**

<https://arxiv.org/pdf/1409.1556.pdf>

InceptionNet, particularly the Inception V3 model, employs modules with parallel convolutional layers of varying sizes (1x1, 3x3, 5x5) and a 3x3 max pooling, leading to a more complex network structure.

**Reference:**

<https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/Szegedy_Rethinking_the_Inception_CVPR_2016_paper.pdf>

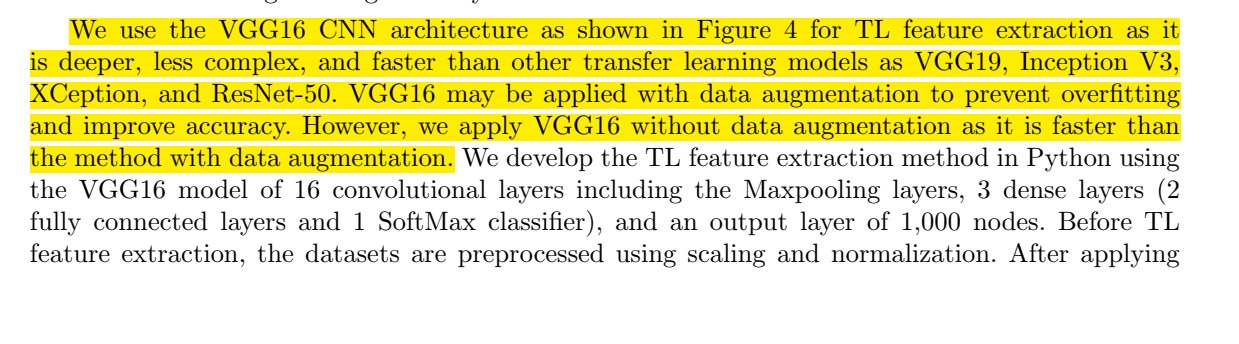
However, it's crucial to note that this claim of simplicity and adaptability is context-dependent. The effectiveness of a model like VGG16 as compared to InceptionNet or other architectures varies based on the specific application and dataset. In our research context, focusing on lung tumor images, VGG16’s architecture provided a balance of depth and simplicity conducive for feature extraction without overfitting.

**18°) We use the VGG16 CNN architecture as shown in Figure 4 for TL feature extraction as it is deeper, less complex, and faster than other transfer learning models as VGG19, Inception V3,  
XCeption, and ResNet-50.  
  
→ who stated this? Need references.**

The statement is not entirely true, and I apologize for any confusion it may have caused. VGG16 has a large network that takes more time to train its parameters and has more than 533MB due to its depth and number of fully connected layers.

Therefore, I would like to revise the statement as follows:

"We use the VGG16 CNN architecture as shown in Figure 4 for TL feature extraction as it has an exceptional feature extraction capability, allowing us to utilize it in our research. However, it is important to note that VGG16 is not faster than some other transfer learning models."



**19°) We develop the TL feature extraction method in Python using the VGG16 model of 16 convolutional layers including the Maxpooling layers, 3 dense layers (2 fully connected layers and 1 SoftMax classifier), and an output layer of 1,000 nodes.  
  
→ the dense layers were removed, right? Did you use any frameworks, like Tensorflow? What version of Python was used?  
  
→ What is the** **size of the attribute vector, extracted with VGG16 in each image?**

Thank you for your insightful comment and inquiry regarding the transfer learning (TL) feature extraction method using the VGG16 model in our study.

**Removal of Dense Layers:** Yes, you are correct. In our TL feature extraction method, we removed the fully-connected dense layers at the top of the VGG16 model.

**Frameworks and Versions:** Yes, we used TensorFlow framework (version 2.10.1) and Python version was 3.9.15.

**Size of the attribute vector:** We extract features from the block4\_pool layer of VGG16, under standard input size conditions (224x224 pixels), is a feature map with dimensions 14x14x512. The size of this vector is 100,352.

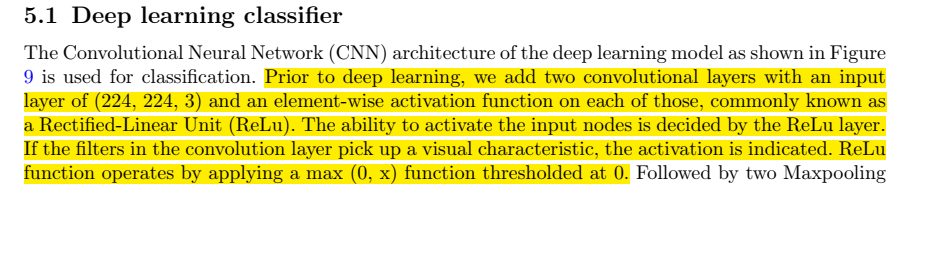
**5.1 Deep learning classifier  
  
24°) The ability to activate the input nodes is decided by the ReLu layer.  
  
→ Do you have a reference for this statement? I believe the role of the ReLU activation function is to normalize the input nodes to the output. And ReLU is not a layer, it is an activation function. Convolution and max-pooling are layers.**

Thank you for your insightful comment and feedback on our work.

You are correct in pointing out that the Rectified Linear Unit (ReLU) is indeed an activation function and not a layer itself.

To address your concern, we would like to modify the relevant section in our journal as follows:

"Prior to deep learning, we add two convolutional layers with an input layer of (224, 224, 3). An element-wise activation function, commonly known as the Rectified Linear Unit (ReLU), is applied to the output of each convolutional layer. The ReLU function operates by applying a max (0, x) function thresholded at 0.”



**29°) The dataset contains 120 CT scan images of the benign class, 561 of the malignant class, and 416 of the normal class.  
  
→ It was used respectively 96, 448 and 332 respectively. Why were cases discarded and not everything used?**

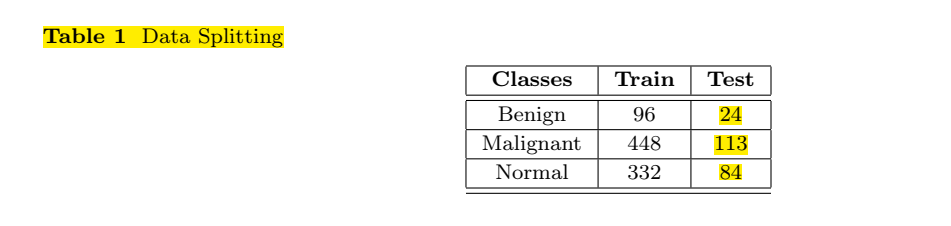
Thank you for your insightful comment on our dataset splitting strategy. We appreciate the opportunity to clarify the rationale behind using specific subsets of CT scan images for training and testing in our study.

In our dataset, we initially had 120 CT scan images for the benign class, 561 for the malignant class, and 416 for the normal class. In our paper, we used 80% of the available images for training and set aside the remaining 20% for testing.

Here is a breakdown of the process:

* **Benign Class:** 80% of 120 images were used for training (96 images), and the remaining 20% were reserved for testing (24 images).
* **Malignant Class:** 80% of 561 images were used for training (448 images), and the remaining 20% were reserved for testing (113 images).
* **Normal Class:** 80% of 416 images were used for training (332 images), and the remaining 20% were reserved for testing (84 images).

No cases were intentionally discarded; rather, the splitting was a deliberate choice to ensure a balanced distribution for training and testing.

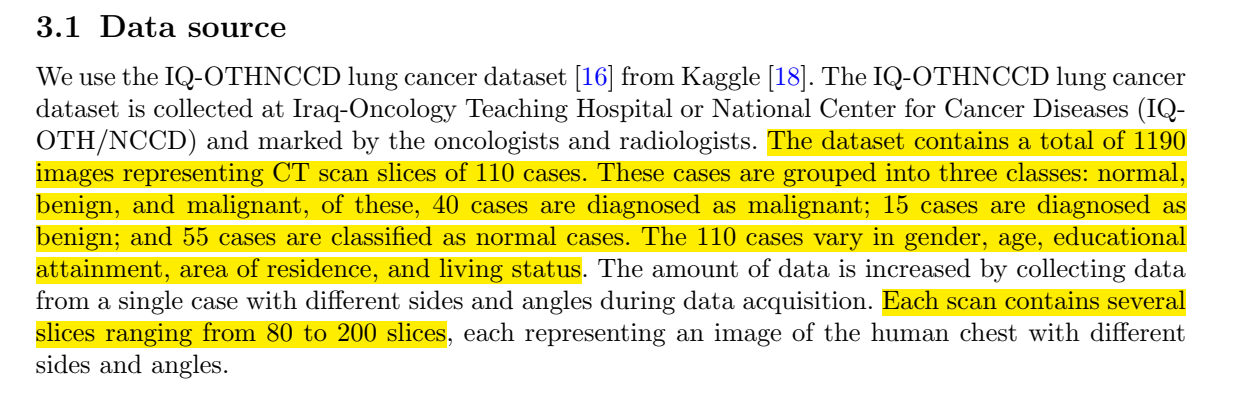


**30°) Is each exam one for a patient, or can there be more than one exam for the same patient?**

Thank you for your question regarding the nature of the exams in our dataset.

Each case in the IQ-OTH/NCCD lung cancer dataset represents a distinct patient. The dataset includes CT scans collected from 110 cases, comprising patients diagnosed with lung cancer in different stages, as well as healthy subjects. The scans are grouped into three classes: normal, benign, and malignant.

To elaborate, each patient's CT scan consists of several slices, with the number of slices ranging from 80 to 200. Each of these slices represents an image of the human chest captured at different sides and angles.



**Reference:**

<https://data.mendeley.com/datasets/bhmdr45bh2/2>

**31°) The libraries include pandas,**[**scipy.io**](http://scipy.io/)**, numpy, matplotlib, seaborn. The classifiers are developed using the Keras, TensorFlow, and Scikit-Learn tools. The morphological feature extraction methods are developed in MATLAB.  
  
→ What are the library versions?**

Thank you for your inquiry regarding the library versions used in our research. Following are the library versions used in our paper:

* **pandas version:** 1.5.2
* **scipy version:** 1.9.3
* **numpy version:** 1.23.5
* **matplotlib version:** 3.6.2
* **seaborn version:** 0.12.1
* **tensorflow version:** 2.10.1
* **keras version:** 2.10.0
* **scikit-learn version:** 1.1.3
* **Python version:** 3.9.15
* **MATLAB version:** 2015a