UNIVERSIDAD DEL NORTE

EGFR and KRAS Mutation Prediction on Lung Cancer through Medical Image Processing and Artificial Intelligence

DOCTORADO EN INGENIERÍA DE SISTEMAS Y COMPUTACIÓN

SILVIA CAROLINA MORENO TRILLOS

2016-2022

Thesis for the degree of Ph.D. in Systems Engineering and Computation.
scmoreno@uninorte.edu.co

EGFR and KRAS Mutation Prediction on Lung Cancer through Medical Image Processing and Artificial Intelligence

DEPARTMENT OF SYSTEMS ENGINEERING

UNIVERSIDAD DEL NORTE

THESIS FOR THE DEGREE OF PH.D. IN COMPUTER SCIENCE

BY

SILVIA CAROLINA MORENO TRILLOS

ADVISOR:

Eduardo Enrique Zurek Varela, Ph.D.

BARRANQUILLA – UNIVERSIDAD DEL NORTE

2022

DEDICATION

To my husband and mother, for teaching me that the greatest fight, is the fight for your dreams.

ACKNOWLEDGMENTS

First of all, I want to thank God, to whom I owe everything, and to my family, especially my husband and my mother, who have always encouraged me to follow my dreams and to keep trying even when life gets hard. I could never have accomplished this without their support.

I would also like to thank Dr. Eduardo Zurek for his valuable contributions and all his help in this work. I also thank my international advisors Dr. Dmitry Goldgof and Dr. Lawrence Hall who introduced me to the world of Radiomics and supported me in my research.

Finally, I want to thank Universidad del Norte for granting me the scholarship to study for my PhD and for all the knowledge and skills that I have gained in the process. Also thanks to Fulbright Colombia for sponsoring my internship at the University of South Florida which allowed me to meet great researchers and inspired this thesis.

LIST OF FIGURES

LIST OF TABLES

Table of Content

CHAPTER 1. INTRODUCTION

1.1 Problem Statement

Globally, lung cancer is the leading cause of cancer-related death in men and the second-leading cause in women. In 2018, an estimated 1.8 million lung cancer deaths occurred, with 1.2 million in men and over 576,000 in women, accounting for 1 in 5 cancer-related deaths worldwide [1]. Advances in precision medicine and genomic analyses have resulted in a paradigm shift whereby lung tumors are characterized and classified by biomarkers and genetic alterations (e.g., gene expression, mutations, amplifications, and rearrangements) that are critical to tumor growth and can be exploited with specific targeted agents or immune checkpoint inhibitors. However, there are many limitations of tissue-based biomarkers such as they can be subject to sampling bias due to the heterogeneous nature of tumors, the requirement of tumor specimens for biomarker testing, and the assays can take significant time and be expensive [2]. As such, high-throughput and minimally invasive methods that can improve current precision medicine is a critical need.

In lung cancer, there has been particular interest in predicting EGFR and KRAS mutations [3]. Epidermal Growth Factor Receptor (EGFR) is a protein on the surface of cells that regulates signaling pathways to control cellular proliferation. Some lung cancer cells have too much of this protein, which increases tumor growth. Lung adenocarcinomas with mutated EGFR have a significant response to tyrosine kinase inhibitors [4], which makes the detection of this mutation significant in determining patient treatment. On the other hand, Kristen Rat Sarcoma viral oncogene (KRAS) is also a well-known tumor driver. Mutations of this gene have proven to be a useful biomarker to predict resistance to EGFR-based therapeutics [5]. Furthermore, some studies have shown that KRAS can be targetable with promising results in phase III of NSCLC [6] [7]

Liquid biopsy is a good alternative for a non-invasive way to detect EGFR and KRAS mutations. The use of surrogate sources of DNA, such as blood, serum, and plasma samples, which often contain circulating free tumor (cft) DNA or circulating tumor cells (CTCs), is emerging as a new strategy for tumor genotyping [8]. However, this technique is pretty recent and still has some disadvantages. Different studies have also shown that the amount of cftDNA is correlated with disease stage, which may make it difficult to detect in early stages of cancer. Moreover, non-tumor cfDNA might derive from different processes including necrosis of normal tissues surrounding the tumor cells or lysis of leukocytes after blood collection, which may make mutation difficult to detect. Even when recent versions of liquid biopsy techniques have been approved for clinical use, the sensitivity (or True Positive Rate) of this test is still a weak point [3]. All these concerns provide space for the application of other non-invasive techniques that may be more effective in early stages of cancer and may provide higher sensitivity rates.

Quantitative image features, or radiomics, have the potential to complement and improve current precision medicine. Radiomic features are non-invasive, are extracted from standard-of-care images, and do not require timely and often expensive laboratory testing. Additionally, radiomic features are not subject to sampling bias since the entire tumor is analyzed and represents the phenotype of the entire tumor in 3D and not just the portion that was subjected to biomarker testing.

Radiogenomics is an emerging and important field because it utilizes radiomics to predict genetic mutations, gene expression, and protein expression [9]. In this document, three methods for predicting EGFR and KRAS mutations from CT images are presented and assessed. First, prediction with radiomic features and machine learned models, second, prediction through custom Convolutional Neural Networks (CNNs), and finally, Transfer Learning by applying pre-trained CNNs. In all three cases, first base models are tested and then an ensemble of the best models with a new voting scheme is applied to observe if there is an improvement in the prediction performance.

This work is organized as follows. Chapter 2 provides a general background about topics related to Radiomics, Convolutional Neural Networks, Transfer Learning, and Ensembles. Chapter 3 presents previous work related to EGFR and KRAS mutation prediction through medical image processing. Chapter 4 describes the methodology to predict EGFR and KRAS mutations from CT images applied in this study and the proposed voting scheme to use with Ensembles. Chapter 5 explains the details of the experiment with Radiomic Features and Machine Learned Classifiers. Chapter 6 describes the experiment with custom Convolutional Neural Networks. Chapter 7 describes the experiment with pre-trained Convolutional Neural Networks. Finally, the conclusions about the proposed approach are provided and future work is described.

1.2 Motivation

Lung cancer is the second most common form of cancer in both men and women (not including skin cancer). Lung cancer is by far the leading cause of cancer death among both men and women. More people die of lung cancer every year than of prostate, colon, and breast cancers together. In the United States, the American Cancer Society estimated that for 2018 there were 234.030 new cases of lung cancer (121.680 in men and 112.350 in women) and 154.050 died from lung cancer (83.550 men and 70.500 women) [10]

In the Colombian case, according to the Ministry of Health, every year 33 thousand Colombians die of Cancer, and around 3.875 of them die of Lung Cancer [11]. Analyzing the number of deaths for men and women, in the year 2013 in Colombia 13,4 of 100.000 men died of lung cancer, and 8,1 of every 100.000 women died of the same cause [12].

Statistics on survival in people with lung cancer vary depending on the stage (extent) of cancer when it is diagnosed. Although a small percentage (about 15%) is curable when detected early, the 5-year survival rate remains at about 16.6%. [13]. The early detection and characterization of Lung Cancer for personalized treatment can be crucial for patient survival. In this work, we develop a new voting algorithm for ensembles that improves the performance of Machine Learning and Deep Learning models and apply it to three automated methods for predicting EGFR and KRAS mutations from CT images, which are useful to determine the best treatment for a lung cancer patient in a non-invasive way.

1.3 Research Objectives

The main objective of this research is to develop a framework to improve the performance of Machine Learning and Deep Learning Classifiers with unbalanced datasets and apply it to the prediction of EGFR and KRAS mutation from CT images

Specifically, the research focuses on the following goals:

1) To design an algorithm for implementing a new voting scheme that improves the performance in classification when using ensembles with unbalanced datasets.

2) To develop a methodology for applying machine learning models with unbalanced datasets for predicting EGFR and KRAS mutations from CT images that implements the designed voting algorithm and assess its performance.

3) To develop a methodology for applying custom and pre-trained Convolutional Neural Networks with unbalanced datasets in the prediction of EGFR and KRAS mutations from CT images that applies the designed voting algorithm and assess its performance.

CHAPTER 2. GENERAL BACKGROUND

2.1 Radiomics

The concept of Radiomics was first introduced by Lambin et. al. [14]. According to these authors, Radiomics is the automated high-throughput extraction of large amounts of quantitative features of medical images, with the hypothesis that quantitative analysis of medical image data can provide more and better information than that of a medical expert, such as the differences in tumor shape and texture.

The standard Radiomics workflow is the following: first segmentation is performed on medical images to define tumor region, next features are extracted, such as intensity, texture, and shape. Finally, an analysis is performed using these features, and they are assessed for their prognostic power or relation to cancer stage or gene expression. These are also the stages used for many image processing applications [15].

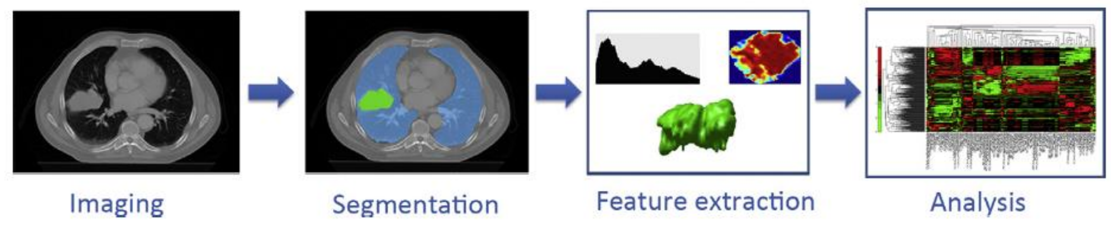

Figure 1. The Radiomics Workflow [14].

The features extracted from medical images can be classified into the following categories [16] [17]:

- **Morphological features**: based on the physical properties of the tumor, such as shape, volume, surface area, sphericity, and mass. Many of these features are useful for cancer prognosis. For example, a strong indicator of cancer aggressiveness is how much time it takes the tumor to double its volume [18]. Mass is another feature that allows detecting nodule growth in an earlier stage of tumor development. [19].

- **Statistical Features**: In this category, we can find the first-order features, related to the image histogram, and higher-order features, related to texture. The histogram shows the range and frequency of the pixel intensity values within the Region of Interest (ROI). Many first-order statistical measurements can be estimated from the histogram, such as the mean, median, standard deviation, kurtosis, energy, entropy, and variance. There have been studies that suggest that these features can be used to determine if a tumor is benign or malignant [20]. Another application of these features is to measure the response to treatment with chemotherapy or radiotherapy. Studies have shown that the first-order statistical features can be a good indicator of treatment response in cases where the size of the tumor is insufficient [21].

  On the other hand, there are also higher-order statistical features related to texture. Among the most used we can find the co-occurrence matrix, the run-length matrix, and the neighborhood gray-tone difference matrix. The co-occurrence matrix (GLCM) is built using the number, distance, and angle of the combination of gray levels that can be found in an image. The run-length matrix (GLRL) measures the continuous pixels with the same gray level in any direction. Finally, the Neighborhood Gray-Tone Difference Matrix (NGTDM) represents how similar or dissimilar are the pixel intensity values within a neighborhood. Other types of texture features are obtained by spatial filtering techniques, which are based on neighborhood operations on the original textured input images. Some examples of commonly used filters for texture analysis include statistical filters like average filter, range filter, and entropy filter or edge filters like Prewitt filter, Sobel filter, Laplacian filter, and Laplacian of Gaussian (LoG) filter. The input image is convolved with the desired kernel to produce filtered images highlighting specific texture information in the original texture image. The resultant filtered images are analyzed using first-order statistics (mean, median, standard deviation, etc.). [22]

.

Apart from statistical and edge kernels, special kernels have also been designed for identifying different types of textures. For example, Laws designed three sets of one-dimensional convolution masks of different sizes corresponding to different types of textures such as level, edge, spot, wave, ripple, undulation, and oscillation.[23].

The frequency of variations in the gray level values in a region of interest is dependent on the scale of the region of interest. The frequency content within an image can be analyzed at different scales using wavelets. Image texture can be analyzed at different scales by representing the image in a pyramid structure. Using a discrete wavelet transform, four low-resolution images can be obtained from the original image. By repeatedly applying discrete wavelet transform at each level, a hierarchical pyramid structure for different resolutions can be created. Texture analysis can be done by computing statistical texture features at each level or averaging the results across multiple resolutions [22]. Many studies have shown that texture features can be used to determine the stage of the tumor, metastasis, response to treatment, survival, and others [24].

- Regional features: these features measure the heterogeneity within the tumor, which means that tumor cells can show different morphological and phenotypic profiles. The regional features find the variation of intensity between regions, which shows how many and how frequent are sub-regions within the tumor. Among the methods for finding sub-regions, we can find data-driven segmentation and threshold-based segmentation, which allow identifying clusters within the ROI. Heterogeneity within the tumor is one of the most promising factors of prognosis that predict patient survival [25].

After extracting quantitative image features the next step is to apply machine learning and statistical techniques to create classification models that can predict different aspects of cancer.

2.2 Machine Learning

Machine Learning is an area of Artificial Intelligence that develops algorithms that build a mathematical model based on sample data, to make predictions or decisions without being explicitly programmed to do so [26]. The type of tasks that can be performed by Machine Learning can be classified into two categories: Supervised Learning and Unsupervised Learning.

In Supervised Learning, the algorithms build a mathematical model from a dataset that contains data of both the input variables and the desired outputs. Classification and Regression algorithms are examples of Supervised Learning. In Classification algorithms, the outputs are restricted to a finite set of values (categories), while in Regression algorithms the outputs take continuous values, which means that they can take any numerical value within a range.

In Unsupervised Learning, a mathematical model is built from a dataset that contains information of the input variables, but it does not have the labels of the desired outputs. Unsupervised algorithms find structures and patterns in data, and they can perform tasks such as grouping the input records, as in Clustering, they can group the inputs in categories, such as in feature learning, or they can reduce the number of features or input variables, such as in dimension reduction [27].

In this work, we apply Supervised Learning, and more specifically Classification algorithms to label an image as Mutant or Wildtype for both the EGFR and KRAS mutations. There are many Classification Algorithms, the following four techniques are applied in our study:

- **Support Vector Machines**: they are supervised learning models that can be used for classification or regression analysis. Given some training data, with examples labeled as belonging to one of two classes, an SVM training algorithm develops a model that assigns new observations to one class or the other. The objective of the support vector machine algorithm is to find a hyperplane in an N-dimensional space (N — the number of features) that distinctly classifies the data points [28]. An SVM constructs a hyperplane in a high-dimensional space and uses it for classification. The hyperplane that causes the best separation is the one that has the largest distance to the nearest training data point of any class (functional margin) [29] (See Figure 2).

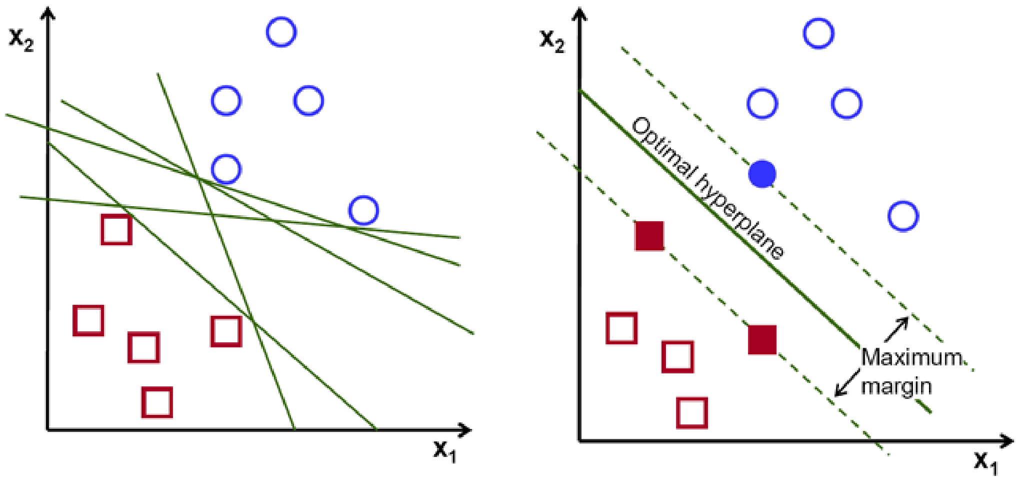

Figure 2. **Possible Hyperplanes, Support Vector Machines [28].**

Maximizing the margin distance provides some reinforcement so that future data points can be classified with more confidence. Hyperplanes are decision boundaries that help classify the data points. Data points falling on either side of the hyperplane can be attributed to different classes. Also, the dimension of the hyperplane depends upon the number of features. If the number of input features is 2, then the hyperplane is just a line. If the number of input features is 3, then the hyperplane becomes a two-dimensional plane (See Figure 3), and so on [28].

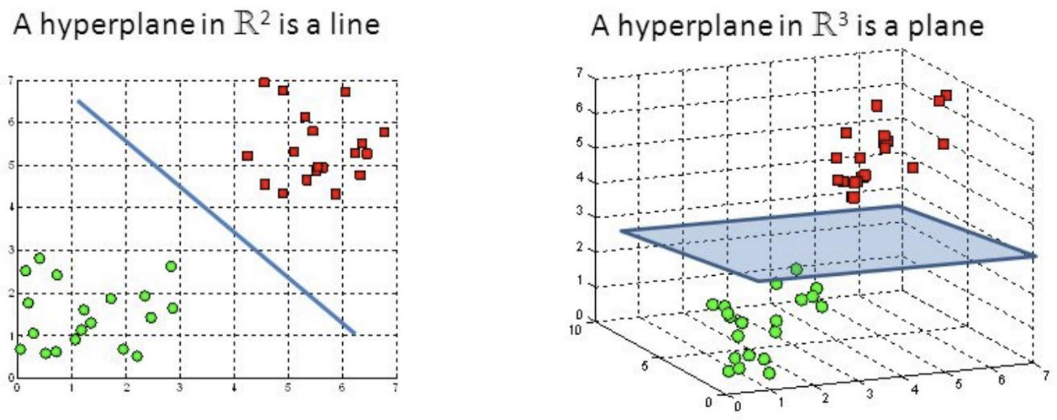

Figure 3. **Hyperplanes in 2D and 3D feature space [28].**

Support vectors are data points that are closer to the hyperplane and influence the position and orientation of the hyperplane. Using these support vectors, we maximize the margin of the classifier. To keep the computational load reasonable, the mappings used by SVM schemes are designed to ensure that dot products of pairs of input data vectors may be computed easily in terms of the variables in the original space, by defining them in terms of a kernel function selected to suit the problem [30], hyperplanes in the higher-dimensional space are defined as the set of points whose dot product with a vector in that space is constant, where such a set of vectors is an orthogonal (and thus minimal) set of vectors that defines a hyperplane.

The original maximum-margin hyperplane algorithm proposed by Vapnik in 1963 constructed a linear classifier [31]. However, in 1992, Bernhard Boser, Isabelle Guyon, and Vladimir Vapnik suggested a way to create nonlinear classifiers by applying the kernel trick to maximum-margin hyperplanes [32]. The resulting algorithm is formally similar, except that every dot product is replaced by a nonlinear kernel function. Some common kernels include Polynomial (homogeneous), Polynomial (inhomogeneous), Gaussian radial basis function, and the Hyperbolic tangent. Although SVMs are defined as non-probabilistic binary linear classifiers, several strategies have been proposed to apply this technique to multiclass problems [33].

- **Random Forest:** A Decision Tree is a method of Machine Learning and Data Mining that creates a model that predicts the value of a target variable based on the values of several other input variables. In a Decision Tree, each interior node represents one of the input variables and each derived branch from that node corresponds to possible values or value ranges of that variable. The leaves of the tree represent values of the target variable given the values of the input variables. Figure 4 presents an example of a Decision Tree.

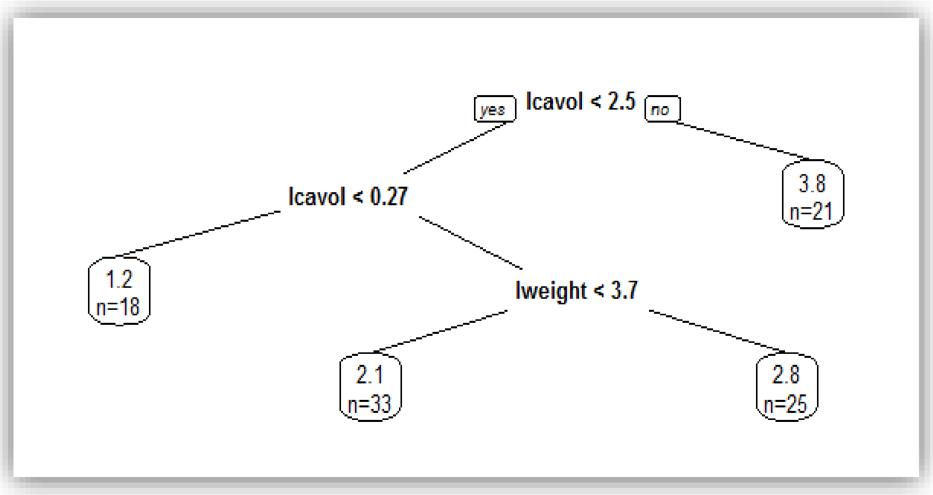

Figure 4. **Decision Tree Example.**

If the target attribute takes a discrete value or class, the model is called a Classification Tree, and if it takes a continuous value, it is called a Regression Tree [34].

Many methods have been proposed for constructing a decision tree using a collection of training samples. The majority of tree construction methods use linear splits at each internal node. A typical method selects a hyperplane or multiple hyperplanes at each internal node, and samples are assigned to branches representing different regions of the feature space bounded by such hyperplanes. n. Such methods can be categorized by the types of splits they produce that are determined by the number and orientation of the hyperplanes [35].:

- **Axis-parallel linear splits:** A threshold is chosen on the values at a particular feature dimension, and samples are assigned to the branches according to whether the corresponding feature values exceed the threshold. These trees can be very deep but their execution is extremely fast.
- **Oblique linear splits**: Samples are assigned to the branches according to which side of a hyperplane or which region bounded by multiple hyperplanes they fall in, but the hyperplanes are not necessarily parallel to any axis of the feature space. A generalization is to use

hyperplanes in a transformed space, where each feature dimension can be an arbitrary function of selected input features. The decision regions of these trees can be finely tailored to the class distributions, and the trees can be small. The execution speed depends on the complexity of the hyperplanes or the transformation function.

- **Piecewise linear splits:** Branches represent a Voronoi tessellation of the feature space. Samples are assigned based on nearest-neighbor matching to chosen anchor points. The anchor points can be selected among training samples, class centroids, or derived cluster centers. These trees can have a large number of branches and can be very shallow.

Figure 5 presents the types of linear splits. Within each category, the splitting functions can be obtained in many ways. For instance, single-feature splits can be chosen by Sethi and Sarvarayudu's average mutual information [36], the Gini index proposed by Breiman et al. [37], Quinlan's information gain ratio [34], or Mingers's G statistic [38]. Oblique hyperplanes can be obtained by Tomek link [39], simulated annealing [40], or perceptron training [41]. Hyperplanes in transformed spaces can be chosen using the support vector machine method [31]. Piecewise linear or nearest-neighbor splits can be obtained by numerous ways of supervised or unsupervised clustering.

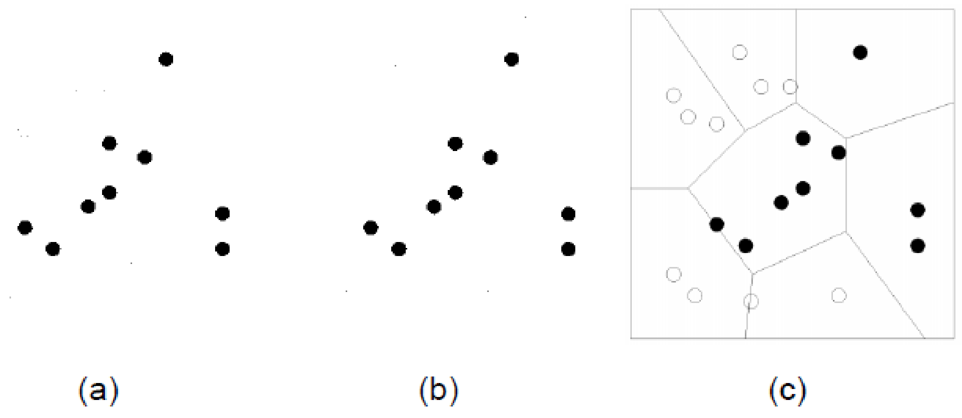

Figure 5. Types of linear splits. (a) Axis-parallel linear splits. (b) Oblique linear splits. (c) Piecewise linear splits [35].

On the other hand, Random Forest is an ensemble learning approach, where several decision tree models are generated on training and the output class is the mode of the models for classification problems, as can be seen in Figure 6, or the average prediction is returned in the case of regression models [42].

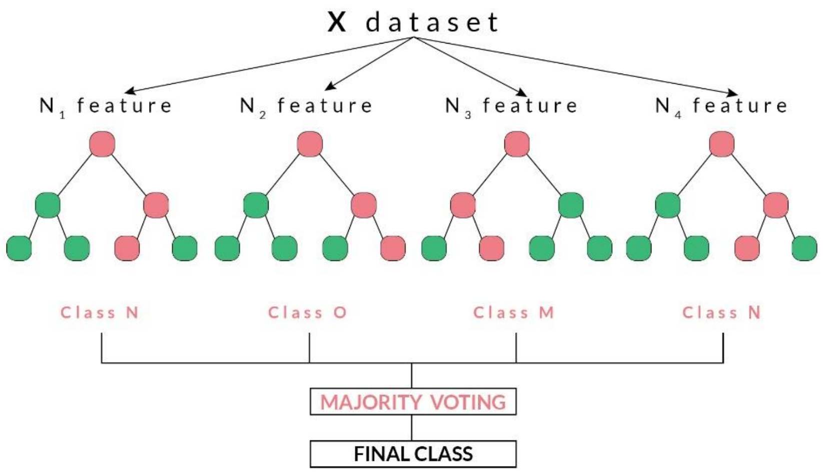

Figure 6. Random Forest Algorithm Scheme [43]

The algorithm was originally proposed by Tin Kam Ho in 1995 [41]. The author claimed that his formulation is a way to implement the "stochastic discrimination" classification approach proposed by Eugen Kleinberg [44]. The algorithm was later extended in the works of Leo Breiman and Adele Cutler [45] who registered "Random Forests" as a trademark in 2006. This work develops Breiman's idea of bagging and random selection of features, initially introduced by Ho, to construct a collection of Decision Trees with controlled variance. The training algorithm for random forests applies the technique of bootstrap aggregating or bagging to tree learners. Given a training set $X = X_{1,}X_2 \dots X_n$ with responses $Y = Y_{1,}Y_2 \dots Y_n$ bagging repeatedly selects a random sample with replacement of the training set and fits trees to these samples. Predictions can then be made by averaging the predictions from all the individual trees in the case of regression, or taking the majority vote in the case of classification. Random forest also uses a modified tree learning algorithm that selects, at each candidate split in the learning process, a random subset of the features. This process is known as "feature bagging". The reason for this process is the correlation of the trees in an ordinary bootstrap sample: if one or a few features are strong predictors for the target variable, these features will be selected in many of the generated trees, thus causing them to be correlated. Random decision forests correct for decision trees' habit of overfitting to their training set [29]. Random forests generally outperform decision trees.

- **Neural Networks:** Neural networks were first proposed in 1944 by Warren McCullough and Walter Pitts [46], two University of Chicago researchers who moved to MIT in 1952. An Artificial Neural Network is a computer model that is inspired by the animal brain, and it is designed to learn how to perform a specific task, such as classification, from a set of observations or examples. An ANN is formed by a network of elements called artificial neurons that receive input data and change their internal state (activation) according to that input and produce an output depending on the input and activation function. The neurons are connected to each other forming a directed weighted graph. Artificial Neurons are aggregated into layers, and these may perform different kinds of transformations on their inputs. The weights and the activation functions can be modified by a learning process, which is directed by a learning rule [46].

  Artificial neural networks (ANNs) are comprised of node layers, containing an input layer, one or more hidden layers, and an output layer (See Figure 7). Most of today's neural nets are organized into layers of nodes, and they're "feed-forward," meaning that data moves through them in only one direction. An individual node might be connected to several nodes in the layer beneath it, from which it receives data, and several nodes in the layer above it, to which it sends data.

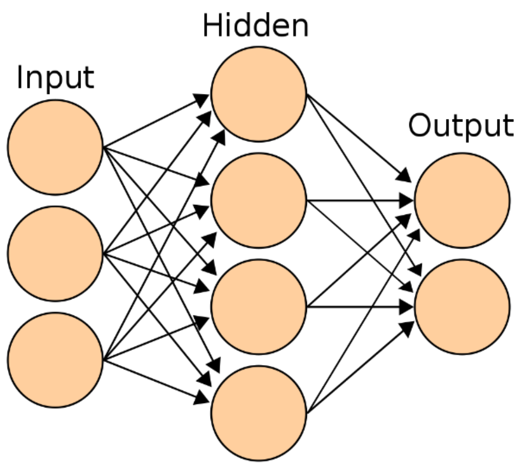

Figure 7. **Artificial Neural Network Scheme [47]**

To each of its incoming connections, a node will assign a number known as a "weight." When the network is active, the node receives a different data item over each of its connections and multiplies it by the associated weight. It then adds the resulting products together, yielding a single number. If that number is below a threshold value, the node passes no data to the next layer. If the number exceeds the threshold value, the node activates, which means sending the number (the sum of the weighted inputs) along with all its outgoing connections.

When a neural net is being trained, all of its weights and thresholds are initially set to random values. Training data is fed to the input layer and it passes through the succeeding layers, getting multiplied and added together in complex ways, until it finally arrives, radically transformed, at the output layer. During training, the weights and thresholds are continually adjusted until training data with the same labels consistently yield similar outputs [48].

Ultimately, the goal is to minimize a cost function to ensure correctness of fit for any given observation. As the model adjusts its weights and bias, it uses the cost function and reinforcement learning to reach the point of convergence or the local minimum. The process in which the algorithm adjusts its weights is through gradient descent, allowing the model to determine the direction to take to reduce errors (or minimize the cost function). With each training example, the parameters of the model adjust to gradually converge at the minimum [49].

Neural networks can be classified into different types, which are used for different purposes. The perceptron is the oldest neural network, created by Frank Rosenblatt in 1958 [50]. It has a single neuron and is the simplest form of a neural network (See Figure 8).

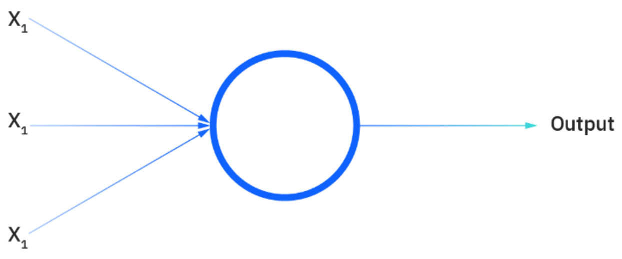

Figure 8. Perceptron Scheme [49]

Feedforward neural networks, or multi-layer perceptrons (MLPs) are comprised of an input layer, a hidden layer or layers, and an output layer. These are mainly the neural networks that have been described in this section. While these neural networks are also commonly referred to as MLPs, it’s important to note that they are actually comprised of sigmoid neurons, not perceptrons, as most real-world problems are nonlinear. Data usually is fed into these models to train them, and they are the foundation for computer vision, natural language processing, and other neural networks [49].

Another type of neural networks are Convolutional Neural Networks which are usually utilized for image recognition, pattern recognition, and/or computer vision. These networks harness principles from linear algebra, particularly matrix multiplication, to identify patterns within an image. This type of network will be discussed further in section 2.3.

Recurrent neural networks (RNNs) are identified by their feedback loops. These learning algorithms are primarily leveraged when using time-series data to make predictions about future outcomes, such as stock market predictions or sales forecasting [49].

Deep Learning and neural networks are terms that may be casually used interchangeably. It’s worth noting that the “deep” in deep learning is just referring to the depth of layers in a neural network. A neural network that consists of more than three layers, can be considered a deep learning algorithm [49].

- **Stochastic Gradient Boosting:** Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. It builds the model in a stage-wise fashion as other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function [51]. Gradient boosting constructs additive regression models by sequentially fitting a simple parameterized function (base learner) to current “pseudo”-residuals by least-squares at each iteration. The pseudo-residuals are the gradient of the loss functional being minimized, with respect to the model values at each training data point evaluated at the current step. It is shown that both the approximation accuracy and execution speed of gradient boosting can be substantially improved by incorporating randomization into the procedure. Specifically, at each iteration, a subsample of the training data is drawn at random (without replacement) from the full training data set. This randomly selected subsample is then used in place of the full sample to fit the base learner and compute the model update for the current iteration [52].

With his “bagging” procedure, Breiman [53] introduced the notion that injecting randomness into function estimation procedures could improve their performance. Early implementations of AdaBoost [54] also employed random sampling, but this was considered an approximation to deterministic weighting when the implementation of the base learner did not support observation weights, rather than an essential part of the process. A few years later Breiman [55] proposed a hybrid bagging-boosting procedure (adaptive bagging) intended for least-squares fitting of additive expansions. This modification replaces the base learner in regular boosting procedures with the corresponding bagged base learner, and substitutes “out-of-bag” residuals for the ordinary residuals at each boosting step. Motivated by this, a minor modification was made to gradient boosting, to incorporate randomness as an integral part of the procedure. At each iteration, a subsample of the traininig data is drawn at random (without replacement) from the full training dataset. This randomly selected subsample is then used, instead of the full sample, to fit the base learner and compute the model update for the current iteration. This is the basis of the stochastic gradient boosting algorithm.

2.3 Convolutional Neural Networks

A Convolutional Neural Network (CNN) is a type of deep neural network that can perform classification tasks directly from images, video, text, or sound. CNNs have demonstrated performance comparable to a human in Computer Vision tasks. They have the advantage over other approaches that they learn directly from image data, eliminating the need for manual extraction of features [56].

CNNs were inspired by biological processes, particularly from the study of the visual cortex performed by Hubel and Wiesel [57]. These authors demonstrated that there are groups of neurons on the visual cortex that have a local receptive field. A receptive field is defined as a limited region of space in which a group of neurons reacts to stimuli. Every group of neurons of the visual cortex focuses on a different region of the visual field. This causes several receptive fields that can superpose and form a mosaic of the whole visual field. These receptive fields are the ones that determine the features to look for in an object to determine what it is. These authors also noticed that some groups of neurons have bigger receptive fields that react to more complex patterns that are combinations of patterns in a lower level. These observations lead to the idea that low-level neurons are based on the results previously obtained by high-level groups of neurons [58].

Just like any other Artificial Neural Network, CNNs are composed of an input layer, an output layer, and many intermediate hidden layers. Among the hidden layers we can find the following types [59]:

- Convolutional Layers: The convolutional layers apply n filters or kernels to the input layer which generates a feature map. These are the type of layers that automatically generates features.
- ReLU: or Rectified Linear Unit: this layer has the function of turning all negative values to zero, to keep the positive values. This layer is known as activation, and only the features that are activated continue to the next layer.
- Pooling: this layer has the purpose of reducing the number of parameters that the network needs to learn. To reduce parameters there is an extraction of statistics such as the mean or the maximum from a region of the feature map. This process causes loss of precision but improves computation.
- Classification: These are the final layers. Fully Connected Layers are used where each pixel of the image is taken as a separate neuron, just like in a Multilayer Perceptron.

Convolutions are a key element in CNNs, however, we have not described what a convolution is. Convolutions are one of the most critical, fundamental building blocks in Computer Vision and Image Processing. When we apply operations such as blurring or smoothing an image, we are already applying this operation. In terms of deep learning, an image convolution is an element-wise multiplication of two matrices followed by a sum. The steps for performing a convolution are the following [60]:

1. Take two matrices (which both have the same dimensions)
2. Multiply them, element-by-element (not the dot product, just a multiplication)
3. Sum the elements together.

An image is a multidimensional matrix. Our image has a width (number of columns) and height (number of rows), just like a matrix. Images also have a depth to them – the number of channels in the image. For a standard RGB image, we have a depth of 3 – one channel for each of the Red, Green, and Blue channels, respectively. Given this knowledge, we can think of an image as a big matrix and a kernel or convolutional matrix as a small matrix that is used for blurring, sharpening, edge detection, and other processing functions. Essentially, this small kernel slides across the big image from left-to-right and top-to-bottom, applying a convolution at each coordinate of the original image, as can be seen in Figure 9. At each pixel in the input image, the neighborhood of the image is convolved with the kernel and the output stored.

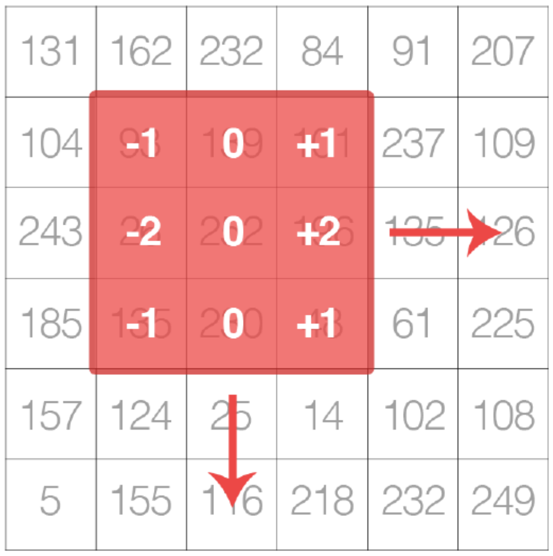

Figure 9. Example of convolution between Image and sliding kernel [60].

Kernels can be of arbitrary rectangular size MxN, provided that both M and N are odd integers. Most kernels applied to deep learning and CNNs are N xN square matrices. We use an odd kernel size to ensure there is a valid integer (x;y)-coordinate at the center of the image. In an image convolution we follow these steps [60]:

1. Select an (x;y)-coordinate from the original image.
2. Place the center of the kernel at this (x;y)-coordinate.
3. Take the element-wise multiplication of the input image region and the kernel, then sum up the values of these multiplication operations into a single value. The sum of these multiplications is called the kernel output.

4. Use the same (x;y)-coordinates from Step #1, but this time, store the kernel output at the same (x;y)-location as the output image.

Below you can find an example of convolving (denoted mathematically as the * operator) a 3x3 region of an image with a 3x3 kernel used for blurring:

$$O_{i,j} = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \star \begin{bmatrix} 93 & 139 & 101 \\ 26 & 252 & 196 \\ 135 & 230 & 18 \end{bmatrix} = \begin{bmatrix} 1/9 \times 93 & 1/9 \times 139 & 1/9 \times 101 \\ 1/9 \times 26 & 1/9 \times 252 & 1/9 \times 196 \\ 1/9 \times 135 & 1/9 \times 230 & 1/9 \times 18 \end{bmatrix}$$

$$O_{i,j} = \sum \begin{bmatrix} 10.3 & 15.4 & 11.2 \\ 2.8 & 28.0 & 21.7 \\ 15.0 & 25.5 & 2.0 \end{bmatrix} \approx 132.$$

After applying this convolution, we would set the pixel located at the coordinate (i; j) of the output image O to $O_{i,j}$= 132. Different kernels are used for different standard image processing operations, such as smoothing, sharpening, and edge detection. By applying convolutional filters, nonlinear activation functions, pooling, and backpropagation, CNNs are able to learn filters that can detect edges and blob-like structures in lower-level layers of the network, and then use the edges and structures as “building blocks”, eventually detecting high-level objects (e.x., faces, cats, dogs, cups, etc.) in the deeper layers of the network [60].

There are many types of layers used to build Convolutional Neural Networks, but the ones we are most likely to encounter include: Convolutional, Activation, Pooling, Fully-connected, Batch normalization, and Dropout.

Convolutional Layers

The Convolutional layer is the core building block of a CNN. This layer parameters consist of a set of K learnable filters (i.e., “kernels”), where each filter has a width and a height, and are nearly always square. These filters are small (in terms of their spatial dimensions) but extend throughout the full depth of the volume. For inputs to the CNN, the depth is the number of channels in the image (depth of three when working with RGB images). For volumes deeper in the network, the depth will be the number of filters applied in the previous layer. For clarification let’s consider the forward-pass of a

CNN, where we convolve each of the K filters across the width and height of the input volume. We can think of each of our K kernels sliding across the input region, computing an element-wise multiplication, summing, and then storing the output value in a 2- dimensional activation map, such as in Figure 10. After applying all K filters to the input volume, we now have K, 2-dimensional activation maps. We then stack our K activation maps along the depth dimension of our array to form the final output volume [60] (Figure 11).

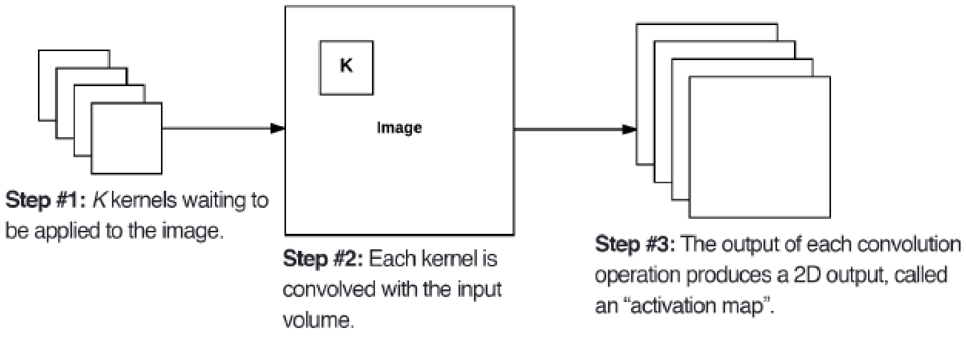
K
Image
Step #1: *K* kernels waiting to be applied to the image.
Step #2: Each kernel is convolved with the input volume.
Step #3: The output of each convolution operation produces a 2D output, called an "activation map".

Figure 10. Convolutional layer of a CNN scheme [60].

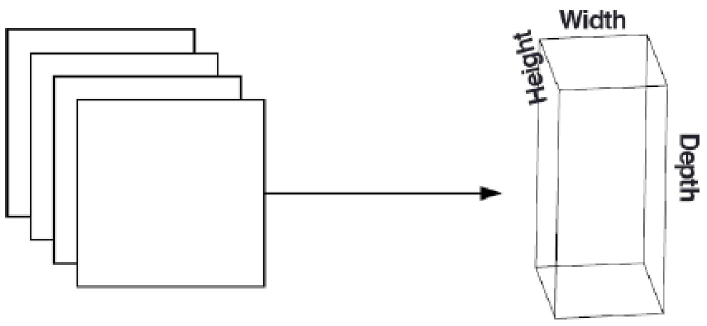

Figure 11. Convolutional layer of a CNN, final volume [60].

Every entry in the output volume is thus an output of a neuron that "looks" at only a small region of the input. In this manner, the network "learns" filters that activate when they see a specific type of feature at a given spatial location in the input volume. In lower layers of the network, filters may activate when they see edge-like or corner-like regions. Then, in the deeper layers of the network, filters may activate in the presence of high-level features, such as parts of the face, the paw of a dog, the hood of a car, etc.

Three parameters control the size of an output volume: the depth, stride, and zero-padding size [60].

- **Depth:** This parameter of the output volume controls the number of neurons (i.e., filters) in the Convolutional layer that connects to a local region of the input volume. Each filter produces an activation map that “activates” in the presence of oriented edges, blobs, or color. For a given Convolutional layer, the depth of the activation map will determine the number of filters we are learning in the current layer. The set of filters that are “looking at” the same (x;y) location of the input is called the depth column.
- **Stride**: we described a convolution operation as “sliding” a small matrix across a large matrix, stopping at each coordinate, computing an element-wise multiplication and sum, then storing the output. This description is similar to a sliding window that slides from left to right and top-to-bottom across an image. The stride is the size of the step we take when sliding the window or kernel. In the context of CNNs, for each step, we create a new depth column around the local region of the image where we convolve each of the K filters with the region and store the output in a 3D volume. When creating our Convolutional layers we normally use a stride step size S of either S = 1 or S = 2. Smaller strides will lead to overlapping receptive fields and larger output volumes. Conversely, larger strides will result in less overlapping receptive fields and smaller output volumes.
- **Zero padding**: We need to “pad” the borders of an image to retain the original image size when applying a convolution. The same is true for filters inside of a CNN. Using zero padding, we can “pad” our input along the borders such that our output volume size matches our input volume size. The amount of padding we apply is controlled by the parameter this parameter. Without zero padding, the spatial dimensions of the input volume would decrease too quickly, and we wouldn’t be able to train deep networks (as the input volumes would be too small to learn any useful patterns from).

Activation Layers

After each Convolutional layer in a CNN, we apply a nonlinear activation function, such as ReLU, ELU, or any of the other Leaky ReLU variants. Activation layers are not technically "layers" (since no parameters/weights are learned inside an activation layer) and are sometimes omitted from network architecture diagrams, as it's assumed that an activation immediately follows a convolution. An activation layer accepts an input volume and then applies the given activation function. Since the activation function is applied in an element-wise manner, the output of an activation layer is always the same as the input dimension [60]. ReLU or Rectified Linear Unit is a layer that has the function of turning all negative values to zero, to keep the positive values, as can be seen in Figure 12.

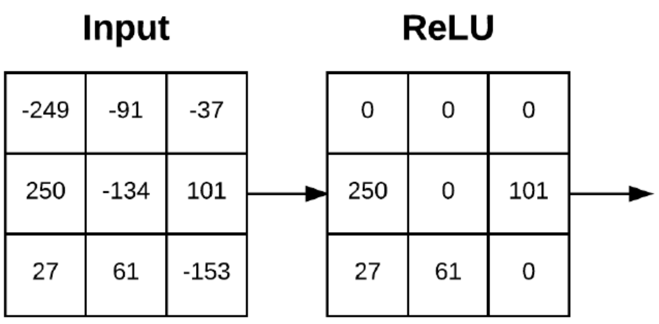

Figure 12. **Example of an input volume going through a ReLU activation [60].**

Pooling Layers

The primary function of the Pool layer is to progressively reduce the spatial size (i.e., width and height) of the input volume. Doing this allows to reduce the amount of parameters and computation in the network. Pooling also helps to control overfitting. It is common to insert Pool layers in-between consecutive convolutional layers in CNN architectures. Pool layers operate on each of the depth slices of an input independently using either the max or average function. Max pooling is typically done in the middle of the CNN architecture to reduce spatial size, whereas average pooling is normally used as the final layer of the network (e.x., GoogLeNet, SqueezeNet, ResNet) where we wish to avoid using Fully Connected layers entirely. The most common type of Pool layer is max pooling, although this trend is changing with the introduction of more exotic micro-architectures [60]. When working with Pool layers, we require two parameters: the receptive field size F (also called the “pool size”) and the stride S. In Figure 13 we can see an example of Max Pooling with two different strides.

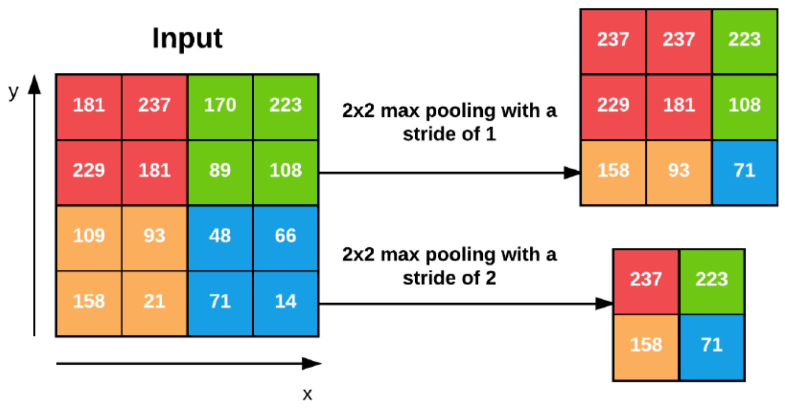

Figure 13. Example of Max Pooling [60].

Notice for every 2x2 block, we keep only the largest value. We can further decrease the size of our output volume by increasing the stride.

Batch Normalization

First introduced by Ioffe and Szegedy [61], batch normalization layers are used to normalize the activations of a given input volume before passing it into the next layer in the network. Activations leaving a batch normalization layer will have approximately zero mean and unit variance (i.e., zero-centered). Batch normalization is extremely effective at reducing the number of epochs it takes to train a neural network. Batch normalization also has the added benefit of helping “stabilize” training, allowing for a larger variety of learning rates and regularization strengths. Using batch normalization doesn’t alleviate the need to tune these parameters, but it will make the learning rate and regularization less volatile and more straightforward to tune. Also, a final loss and a more stable loss curve will be obtained when using batch normalization [60].

The biggest drawback of batch normalization is that it can slow down the time it takes to train the network (even though fewer epochs will be needed to obtain reasonable accuracy) due to the computation of per-batch statistics and normalization.

Dropout

Dropout is a form of regularization that aims to help prevent overfitting by increasing testing accuracy, perhaps at the expense of training accuracy. For each mini-batch in the training set, dropout layers, with probability p, randomly disconnect inputs from the preceding layer to the next layer in the network architecture. The reason to apply dropout is to reduce overfitting by explicitly altering the network architecture at training time. Randomly dropping connections ensures that no single node in the network is responsible for “activating” when presented with a given pattern. Instead, dropout ensures there are multiple, redundant nodes that will activate when presented with similar inputs [60]. This in turn helps the model to generalize. Figure 14 examples of neural network layers without dropout, and with 0.5 dropout.

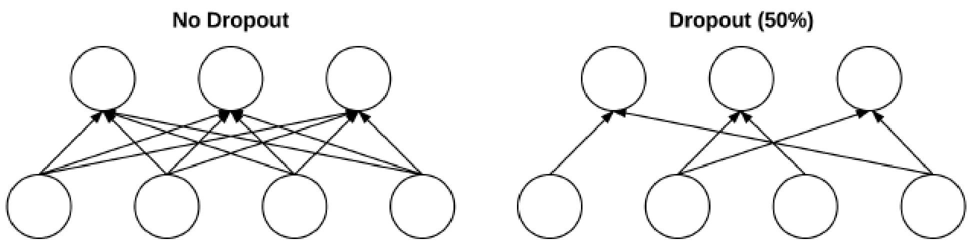

Figure 14. Example of Dropout layer [60].

Fully-connected Layers

Neurons in Fully-connected (FC) layers are fully-connected to all activations in the previous layer, as is the standard for feedforward neural networks. FC layers are always placed at the end of the network, usually followed by a softmax classifier which will compute the final output probabilities for each class [60].

CNN architectures

One aspect that significantly affects the performance of a CNN is its architecture. Several famous architectures have achieved outstanding performance for image classification after being trained on ImageNet. This is an image database organized according to the WordNet hierarchy, in which each node of the hierarchy is depicted by hundreds and thousands of images. Currently, they have an average of over five hundred images per node [62]. As part of the ImageNet Large Scale Visual Recognition Challenge [63], some of the architectures that have achieved outstanding performance have thus become available to the public as part of the Keras library. Three of these pre-trained networks are used in the Transfer Learning Experiment:

- VGG16: This is a convolutional neural network model proposed by K. Simonyan and A. Zisserman [64] from the University of Oxford. The authors propose networks of increasing depth using an architecture with very small (3 × 3) convolution filters, which shows that a significant improvement on the prior-art configurations can be achieved by pushing the depth to 16–19 weight layers. The model achieves 92.7% top-5 test accuracy in ImageNet. Figures 15 and 16 display the architecture of the VGG16 network.

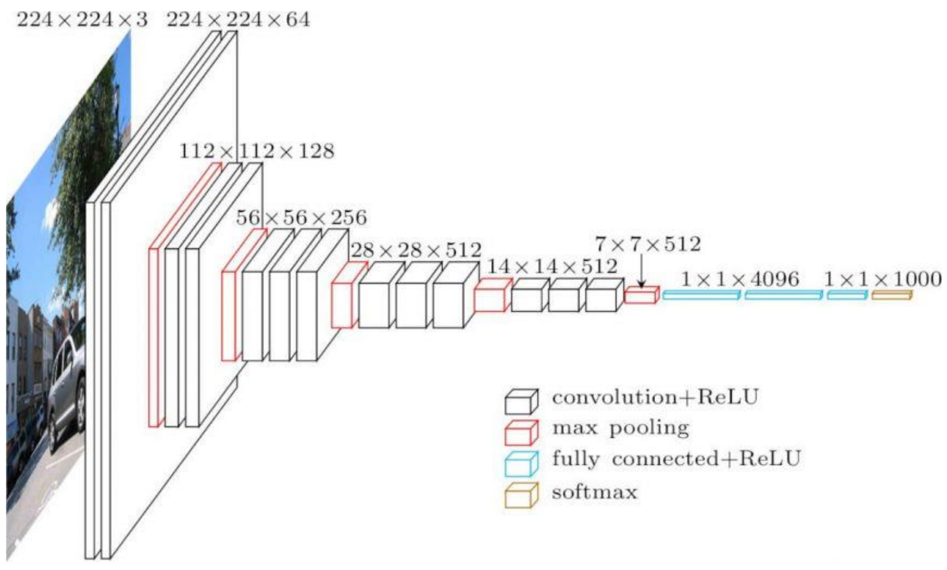

Figure 15. **VGG16 Architecture [65].**

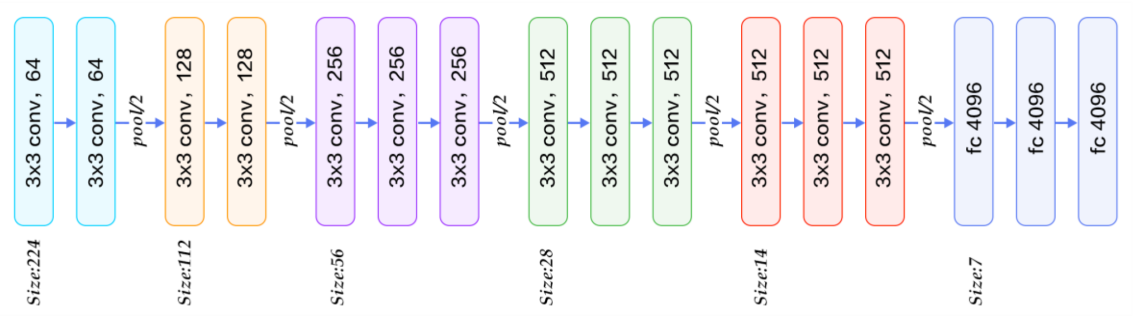

Figure 16. **VGG16 Layers [66].**

- Inception: This architecture is proposed by Google. The main hallmark of this architecture is the improved utilization of the computing resources inside the network. The authors increased the depth and width of the network while keeping the computational budget constant. To optimize quality, the architectural decisions were based on the Hebbian principle and the intuition of multi-scale processing. The model achieves 93.3% top-5 test accuracy in ImageNet [67]. Figure 17 shows the architecture of Inception V3.

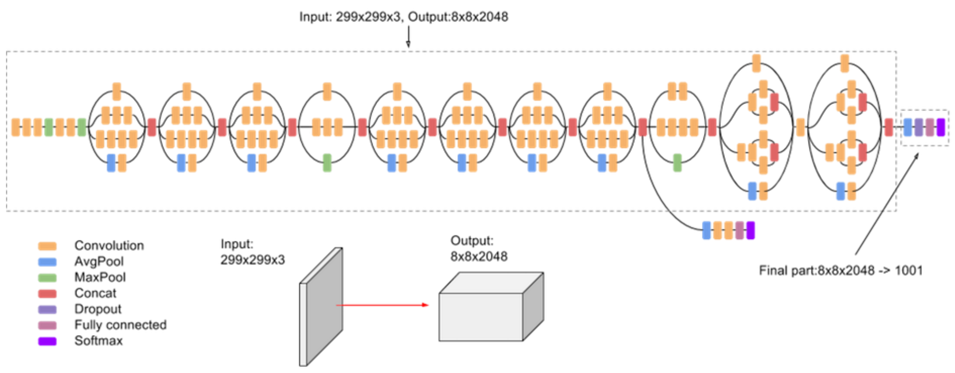

Figure 17. **Inception V3 network architecture [68].**

- ResNet-50: This architecture is proposed by Microsoft. A residual neural network (ResNet) is an artificial neural network (ANN) o that builds on constructs known from pyramidal cells in the cerebral cortex. Residual neural networks do this by utilizing skip connections, or shortcuts to jump over some layers. Typical ResNet models are implemented with double or triple-layer skips that contain nonlinearities (ReLU) and batch normalization in between. ResNet achieves 96.4% top-5 test accuracy in ImageNet [69]. Figure 18 shows the scheme for the ResNet Architecture:

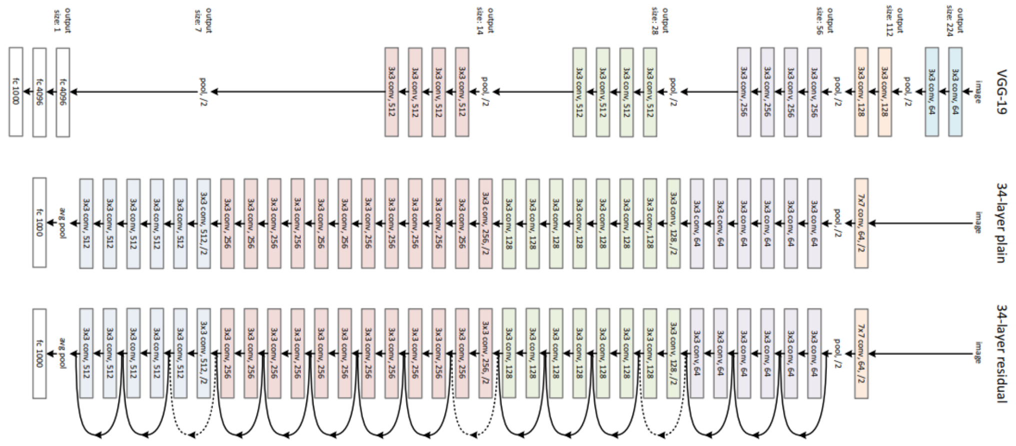

Figure 18. **ResNet network architecture scheme [70].**

2.4 Metrics in Classification Problems

The objective of this study is to predict the mutation status of a lung nodule from a CT image. Two mutations are considered: EGFR and KRAS. For both mutations only two values are possible, either the nodule is Wildtype or it is Mutant. As can be seen, this is a case of a binary classification problem. In classification problems, the outputs are restricted to a finite set of values (categories), as was explained before. Several metrics have been proposed to evaluate the performance of classifiers, both for traditional machine learning models and for deep learning ones. We will describe the ones applied in this study. Most classification performance metrics are derived from the confusion matrix. This is a matrix where each row represents the instances in an actual class while each column represents the instances in a predicted class, or vice versa. The name stems from the fact that it makes it easy to see whether the system is confusing two classes (i.e. commonly mislabeling one as another). It is a special kind of contingency table, with two dimensions ("actual" and "predicted"), and identical sets of "classes" in both dimensions (each combination of dimension and class is a variable in the contingency table) [71].

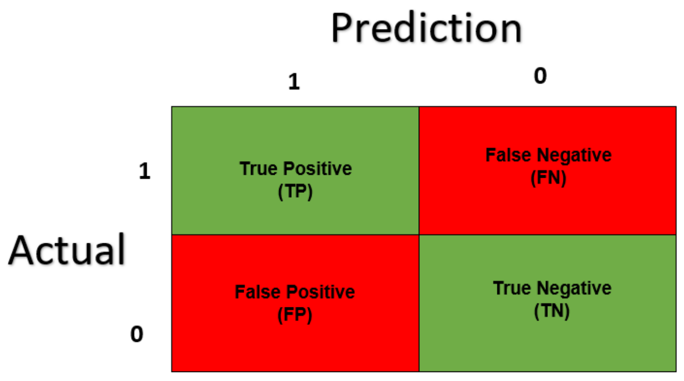

Figure 19. Confusion Matrix [72].

As can be seen in Figure 19, our observations can be classified into four categories:

- True Positives (TP): The model predicted positive and the label was actually positive.
- True Negatives (TN): The model predicted negative and the label was actually negative.
- False Positives (FP): The model predicted positive and the label was actually negative. That is the observation was falsely classified as positive.
- False Negatives (FN): The model predicted negative and the label was actually positive. That is the observation was falsely classified as negative.

In the confusion matrix, we state the number of observations that correspond to each of the four categories. From this table we can estimate certain well known and commonly used metrics [73]:

- **Accuracy:** This is one of the most common metrics for classifiers. Accuracy gives an idea of how many of the observations were correctly classified, both for the positive and negative cases. The mathematical expression for accuracy is the following:

$$ACC = \frac{TP + TN}{P + N} = \frac{TP + TN}{TP + FP + TN + FN}$$

- **Sensitivity, Recall or True Positive Rate (TPR):** This metrics gives us an idea of how good is the classifier to detect the positive cases. The mathematical expression for this metric is the following:

$$Sensitivity = \frac{TP}{P} = \frac{TP}{TP + FN}$$

- **Specificity or True Negative Rate (TNR):** This metric gives us an idea of how good is the classifier at detecting the negative cases. The mathematical expression for this metric is the following:

$$Specificity = \frac{TN}{N} = \frac{TN}{TN+FP}$$

- **Fall-out or False Positive Rate (FPR):** This metric gives an idea of how many of the observations are incorrectly classified as positive. The mathematical expression for this metric is the following:

$$Fall\ out = \frac{FP}{N} = \frac{FP}{FP + TN} = 1 - Specificity$$

- **Precision or Positive Predicted Value (PPV):** It is another common metric that tells us that from the labels our classifier has labeled positive, the amount that is actually positive. The mathematical expression for this metric is the following:

$$Precision = \frac{TP}{TP + FP}$$

- **F1 Score**: This metric is the harmonic mean of precision and sensitivity. The mathematical expression for this score is the following:

$$F1\,Score = 2 \times \frac{PPV \times TPR}{PPV + TPR} = \frac{2TP}{2TP + FP + FN}$$

There are other performance metrics commonly used in binary classification problems that are derived from the **Receiver Operating Characteristic (ROC) curve.** The ROC curve is a graphical plot that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied. The ROC curve is created by plotting the true positive rate (TPR) or Sensitivity against the false positive rate (FPR) or Fall-out at various threshold settings, as can be seen in Figure 20. In general, if the probability distributions for both detection and false alarm are known, the ROC curve can be generated by plotting the cumulative distribution function (area under the probability distribution from $-\infty$ to the discrimination threshold) of the detection probability in the y-axis versus the cumulative distribution function of the false-alarm probability on the x-axis [73].

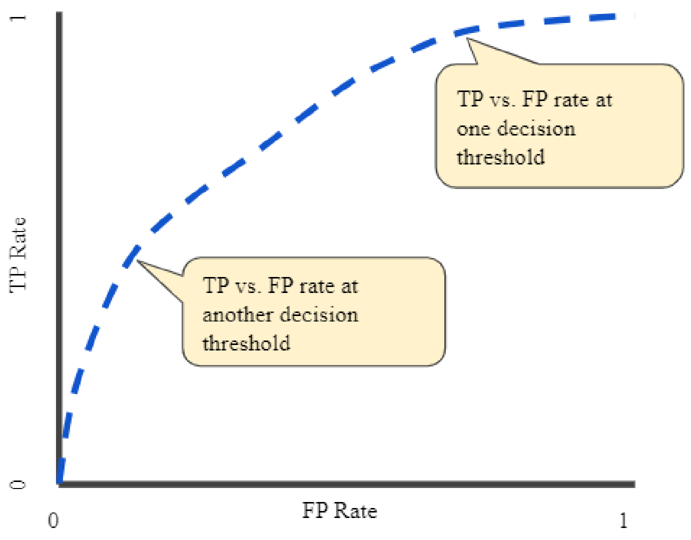

Figure 20. Example of ROC Curve [74].

ROC analysis provides tools to select possibly optimal models and to discard suboptimal ones independently from the cost context or the class distribution. From this curve, we can derive another commonly used metric in classification problems: the AUC. This stands for Area Under the ROC curve, and it represents the degree or measure of separability. It tells how much the model is capable of distinguishing between classes. The higher the AUC, the better the model is at predicting 0 classes as 0 and 1 classes as 1. The AUC takes values from 0 to 1. An excellent model has AUC near to 1 which means it has a good measure of separability. A poor model has an AUC near 0 which means it has the worst measure of separability. In fact, it means it is reciprocating the result. It is predicting 0s as 1s and 1s as 0s. And when AUC is 0.5, it means the model has no class separation capacity whatsoever [75]. The AUC values are directly related to how separate the probabilistic distributions for class 0 and class 1 are from each other, as can be seen in Figures 21 to 24.

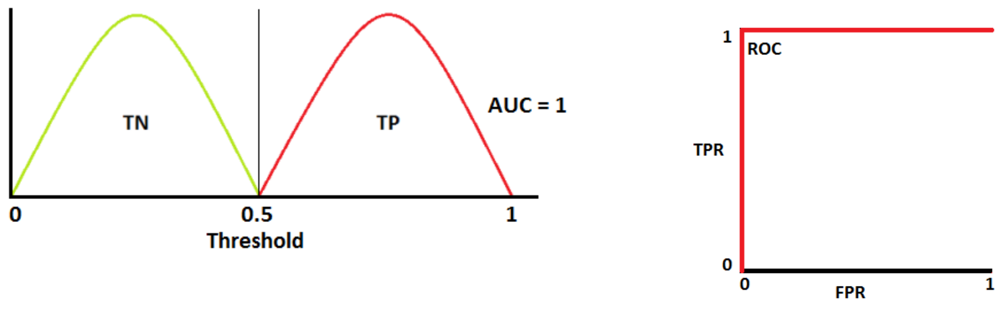

Figure 21. Distribution and ROC curve, case AUC=1 [75].

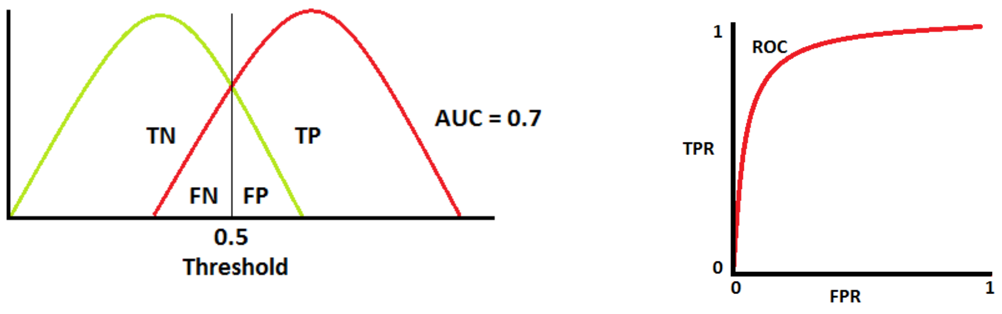

Figure 22. Distribution and ROC curve, case AUC=0.7 [75].

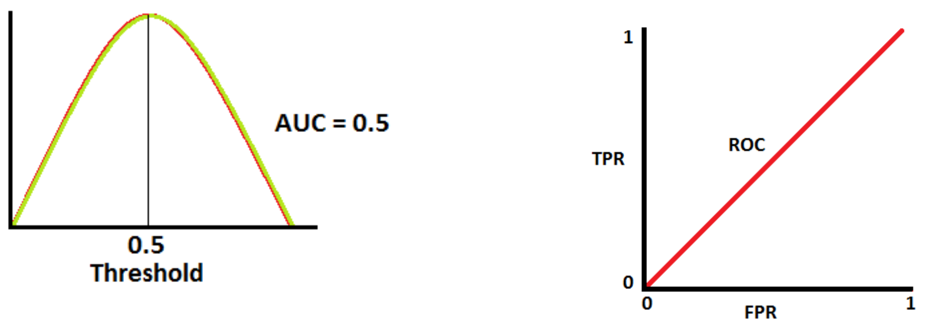

Figure 23. **Distribution and ROC curve, case AUC=0.5 [75].**

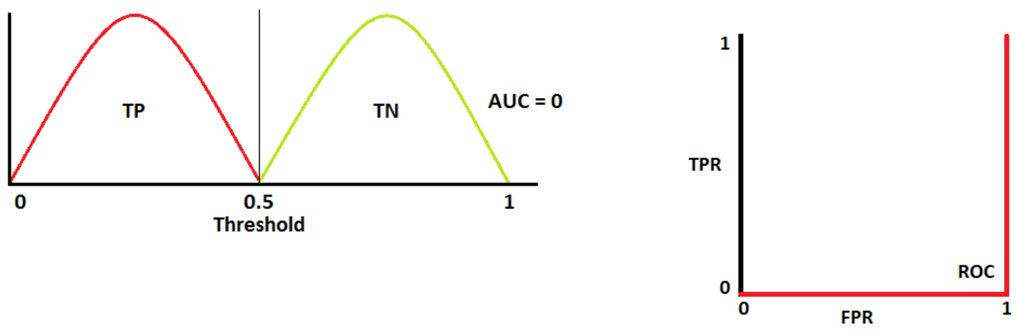

Figure 24. **Distribution and ROC curve, case AUC=0 [75].**

The AUC is one of the metrics more commonly used for evaluating classifiers in medical image classification problems, and is the metric that is more frequently used in other works of the state of the art.

CHAPTER 3. STATE OF THE ART

3.1 Radiomics in Lung Cancer

Lung cancer tumors show significant phenotypic differences that can be visualized in a non-invasive way through medical images. Radiomics, a concept introduced by Lambin et al. [14] takes care of quantifying tumor phenotypes by estimating several quantitative features of the images. These features have proven to be useful for the classification of pulmonary nodules (diagnostic) and prognosis of an already identified cancer [76].

Many radiomic quantitative features can be extracted from medical images and several previous studies have shown that they have classifying and prognostic values [17] . For this reason, recent work has focused on determining which features, or sets of features, have the greatest predicting value and have higher reproducibility, so they can be used to develop predictive models.

One example of these works is the one presented by Saad and Choi [77]. Non-Small Cell Lung Cancer (NSLC) can be classified as adenocarcinoma, squamous cell carcinoma and large cell carcinoma, however 20% of pathology reports of these tumors are unclassified. These authors analyze through Radiomics spatial variations to decode unclassified tumors. Twelve spatial descriptors extracted from the 3D co-occurrence matrix were profiled into each one of the sub-groups. Afterward, a classifier based on these profiles was built applying Support Vector Machines (SVM). Sixteen multi-class classifier models with an 81% average accuracy and descriptor subset size ranging from 12 to 144 were reported. The average area under the curve was 86.3% at a 95% confidence interval and a 0.03-0.08 standard error.

In the study performed by Ma et al. [78], 583 radiomic features are used to distinguish between malignant and benign nodules. These features measure intensity, shape, heterogeneity, and information in multiple frequencies, and are later used as input for a classification system that applies a Random Forest Technique. This method was tested with the 79 CT images from the TCIA (The Cancer Imaging Archive), and an 82.7% of classification accuracy was obtained.

Another recent work is the one presented by Huang et al. [79]. These authors seek to obtain a biological marker or signature from the features obtained through Radiomics that can be used to predict survival in patients diagnosed with stages I and II of NSCLC. 132 texture features obtained from the image histogram and the Gray Levels Co-occurrence Matrix (GLCM) were estimated. Then the LASSO (Least Absolute Shrinkage and Selection Operator) technique was applied to determine which of these features were discriminant of patient survival. The generated signature was statistically significantly associated with the survival index of the patients and independent of other clinical or pathological risk factors.

In the work presented by Wu et al. [80], a relationship between radiomic features and the histological subtypes of tumors, adenocarcinoma and squamous cell carcinoma, is searched. A total of 440 radiomic features are extracted from segmented tumor volumes, which measured phenotypical features such as shape, size, intensity values statistics, and texture. Afterward, univariate and multivariate statistical analysis is performed to determine which of these features are discriminant. In the univariate analysis, it was observed that 53 features were associated with the tumor histology, most of them obtained by applying wavelet transforms. In the multivariate analysis, the feature selection method ReliefF obtained the higher prediction accuracy, and the best performance was obtained with the Bayesian classifier using five features obtained through wavelets and related to the gray-level run-length, median, and obliquity, among others.

Another recent study focused on determining which radiomic features are more significant for cancer evolution is Emaminejad et al. [81]. In this study, the association between several radiomic features and two genomic biomarkers is evaluated, with the purpose of predicting survival without recurrence in cancer patients after surgery. 35 features were estimated from CT images and analyzed with the data mining software Weka. Afterward, ten non-redundant features were selected. Among these features were entropy, standard deviation, mean, irregularity of tumor boundary, density values of minimum and maximum pixel, obliquity, and kurtosis. The prediction capacity between the quantitative image features, the subjective features given by a radiologist, and the previously mentioned biomarkers were compared, and they all showed to be relevant classifiers. Between these options, the radiomic quantitative features presented the higher predictive power.

3.2 Radiomics in prediction of EGFR and KRAS Mutations

Other authors have previously tried to predict EGFR and KRAS mutations in Non-Small Cell Lung Cancer (NSCLC) from image features. In the work presented by Gevaert et al. [3], the authors attempted to predict these mutations from semantic image features provided by radiologists. From a dataset of 186 images of thin-slice CT scans, 89 semantic image features were specified by a radiologist. Then Decision Trees were built to predict EGFR and KRAS mutations. The final model uses only four semantic features. The presence of emphysema is at the root of the tree, determining EGFR wildtype tumors, followed by tumors with airway abnormalities also determining EGFR wildtype tumors. Next, tumors that have smooth or irregular margins are again predicted to have no EGFR mutation. Next, for the remaining tumors that have lobulated, spiculated, or poorly defined margins, if they contain any ground glass component, they are predicted to be EGFR mutated. Finally, for purely solid lesions, when the margins are lobulated, spiculated, or poorly defined, the model predicts the presence of an EGFR mutation. The proposed predictive model for the EGFR mutation achieved an AUC of 0.89; however, conclusive results for the KRAS mutation were not obtained.

Pinheiro et al. [82] also found a correlation between imaging features and mutation status for EGFR mutation (AUC of 0.745) but could not find the same for the KRAS mutation. Two main types of input features were considered: radiomic and semantic (given by a radiologist). The semantic ones were further divided into features that only describe the nodule, features that only describe structures external to nodule and a hybrid between the previous two. Radiomics were not further divided as they only describe the nodule. When using Principal Component Analysis (PCA) followed by t-distributed Stochastic Neighbour Embedding (t-SNE) for dimensionality reduction, the authors concluded that the separation of classes between mutated and wildtype EGFR gene status is better when using hybrid semantic features. However, the separation is not perfect, as there are samples outside their cluster, which illustrates the level of complexity faced in a classification process. Contrarily, for KRAS, there was no visible separation between classes with any type of input features. The images were taken from The NSCLC-Radiogenomics dataset [83], where 116 cases were considered for the EGFR mutation model and 114 were considered for the KRAS mutation model. A set of 1218 radiomic features were extracted using the open-source package PyRadiomics [84]. The classifier used in this work was Extreme Gradient Boosting (XGBoost), which is a scalable and accurate implementation of gradient boosted trees algorithms.

On the other hand, Wang et al. [85] utilized semantic features to predict EGFR and KRAS mutation, and found a significant correlation between EGFR and KRAS mutations and lesions with a low ground glass opacity (GGO). In particular, the authors found that L858R point mutations, exon 19 deletions and KRAS mutations were more common in lesions with a lower GGO proportion (P=0.029, 0.027 and 0.018, respectively).

Mei et al. [86] utilized texture features to predict mutations in EGFR at exon 19 and exon 21. In a retrospective study with 306 patients, 3D radiomic features were extracted using the open-source library PyRadiomics [84]. Ninety-four texture features, including first-order features (19 features), gray-level-co-occurrence matrix (GLCM) features (27 features), gray-level-run-length matrix (GLRLM) features (16 features), gray-level size zone matrix (GLSZM) features (16 features), and shape features (16 features), were extracted from the segmented lesions. Then Logistic Regression Analysis was applied to build a predictive model. The authors reported an AUC of 0.66 for predicting EGFR exon 21 mutation using a model that included sex, non-smoking status, and the Size Zone Non-Uniformity Normalized radiomic feature.

Shiri et. al. [87] created machine learning models from PET and CT image features to predict both EGFR and KRAS mutations. The authors worked with a dataset of 211 NSCLC patients, each including diagnostic CT (CTD), low dose CT (CTA), and PET, as well as mutation status for KRAS and EGFR. From this dataset 186 patients were selected that had manual tumor segmentation on PET images and 175 patients with segmentation were selected from the CTD and CTA images. The images were preprocessed with Laplacian of Gaussian (LOG) wavelet decomposition (WAV) and discretization to 64 bin (BIN64). Then several radiomic features were extracted: first-order statistics and SUV based (19 features), Shape-based (16 features), gray level co-occurrence matrix (GLCM; 23 features), gray level run length matrix (GLRLM; 16 features), gray level size zone matrix (GLSZM; 16 features), neighboring gray-tone difference matrix (NGTDM; 5 features), and gray level dependence matrix (GLDM; 14 features). In this study, different algorithms were used for data splitting, feature selection, and classification. For feature selection, Select K Best (KB), Select K Best & Mutual Info Regression (KB-MIR), Select from Model (SM), and Variance Threshold & Select from Model (VT-SM) were applied. For data splitting, 10-fold cross-validation was used. Several classification algorithms were used: random forest (RF), Bagging (BG), Logistic Regression (LREG), Naive Bayesian (NB), and Support Vector Machine (SVM). In their best results, these authors obtained an AUC of 0.75 for both mutations in CT images by applying a combination of K-Best and variance threshold feature selector with logistic regression. Incorporating PET kept AUC values around 0.74.

Liu et al. [88] utilized radiomics features and clinical data to predict EGFR status. 298 patients with surgically resected peripheral lung adenocarcinomas were investigated and Definiens Developer XD© [89] (Munich, Germany) was used the image analysis platform to perform tumor segmentation and feature extraction. The authors extracted a total of 219 features from each of the 3D objects. These features were divided into eight categories, including tumor size, shape, location, air space, pixel intensity histogram, run-length & co-occurrence, laws texture, and wavelets. The correlation between features was investigated to address the collinearity issue. The highly correlated features (correlation > 0.9) were regarded as dependent features which were not considered in the analysis. Multiple logistic regression analysis was performed. The final model was selected using the backward elimination method. Further, various predictive models were developed by the support vector machine (SVM) and the principal component analysis (PCA) and were compared with the logistic regression model. Finally, the authors found an AUC of 0.647 with the best model based on 5 radiomic features which improved to 0.709 AUC by combining radiomic features and clinical data.

Deep learning has recently been applied in the diagnosis of different types of cancer [90], and other authors like Wang et. al. [91] have applied these techniques to mutation prediction. These authors utilized deep learning to the prediction of EGFR mutational status. The authors worked with a dataset of 844 lung adenocarcinoma patients with pre-operative CT images, EGFR mutation, and clinical information from two hospitals. For applying the deep learning model, a cubic region of interest (ROI) containing the entire tumor was manually selected. The authors applied Convolutional Neural Networks to create the predictive model. The authors used transfer learning to train the first 20 convolutional layers from the ImageNet dataset, applying the DenseNet [92] pre-trained network, and then the last four convolutional layers were trained using 14.926 CT images from lung adenocarcinoma tumors in the primary cohort. The authors found an AUC of 0.81 on an independent validation cohort.

Other recent studies have applied clinical nomograms to predict EGFR mutation status. In the work presented by Zhang et al. [93], the authors combined CT features and clinical risk factors and used them to build a prediction nomogram. They obtained a 0.74 AUC on the validation cohort. Table 1 summarizes the methods and results of previous work.

Table 1 Previous Work Summary

Authors	Methods	Results
Gevaert et al.	Semantic Features, Machine Learning	EGFR: 0,89 AUC KRAS: inconclusive
Pinheiro et al.	Quantitative Image Features	EGFR: 0.745 AUC KRAS: inconclusive
Mei et al.	Quantitative texture features	EGFR: 0.66 AUC
Shiri et. al.	Quantitative features from CT and PET, machine learning	EGFR y KRAS: 0.75 AUC
Liu et al.	Quantitative features, clinical data, Machine Learning	EGFR: 0.709 AUC
Wang et.al.	Deep Learning	EGFR: 0,81 AUC
Zhang et al.	Semantic features, Risk factors	EGFR: 0.74 AUC

CHAPTER 4. MATERIALS AND METHODS

For this study, a cohort of 99 patients from the TCIA was obtained [94], whose data included CT images with tumor segmentation on the CT image, genomic data (KRAS mutational status, and EGFR mutational status), and clinical data (age, sex, smoking status, pathological T stage, pathological N stage, pathological M stage and histology type). Details of the cohort and corresponding data are published in a previous study [3] Patients with unknown mutational status were eliminated from the analysis, which resulted in 83 patients for the analysis. This type of data, with curation, is difficult to obtain. This set, while small, allows for comparisons. The summary of the study cohort is presented in Table 2. Table 3 summarizes the clinical features of the study cohort. Part of this study has been published in [95].

Table 2 Mutation Status Data Summary

Variable	Values	Number of Cases (%)
EGFR Mutation Status	Mutant	12 (14%)
	Wildtype	71 (86%)
	Total	83 (100%)
KRAS Mutation Status	Mutant	20 (24%)
	Wildtype	63 (76%)
	Total	83 (100%)

For the EGFR mutation case, there is not a significant difference observed between the mutant and wildtype statuses in terms of age. In terms of gender, for the mutant status there seems to be a more balanced distribution between the genders, while the wildtype status seems to be significantly more common among men. In terms of smoking history, the EGFR mutant status seems to be found more often among former smokers, and non-smokers in second place, while the wildtype status seems to be more common among former and current smokers. There is no significant difference between the groups in terms of T cancer stage, although wildtype status seems to be more common for patients with stage T1a. Cases with stages N1 and N2 seem to more frequently present wildtype status, as well as patients with M1b stage. In terms of histology type, none of the Squamous Cell Carcinoma patients present the EGFR mutation; this is only present in Adenocarcinoma cases.

Table 3 Clinical Features Data Summary

Variable	Overall Dataset	EGFR Mutant	EGFR Wildtype	KRAS Mutant	KRAS Wildtype
Median Age (Range)	69 (46 – 85)	72 (55 –85)	69 (46 – 84)	68 (50- 81)	69 (46-85)
Gender					
Male	65 (78%)	7 (8%)	58 (70%)	16 (19%)	49 (59%)
Female	18 (22%)	5 (6%)	13 (16%)	4 (5%)	14 (17%)
Smoking Status					
Current	18 (22%)	1 (1%)	17 (21%)	6 (6%)	12 (16%)
Former	56 (67%)	8 (9%)	48 (58%)	14 (17%)	42 (50%)
Non-smoker	9 (11%)	3 (4%)	6 (7%)	0 (0%)	9 (11%)
Pathological T Stage					
Tis	3 (4%)	1 (1%)	2 (3%)	0 (0%)	3 (4%)
T1a	17 (21%)	1 (1%)	16 (20%)	4 (5%)	13 (16%)
T1b	19 (23%)	5 (6%)	14 (17%)	3 (3%)	16 (20%)
T2a	26 (31%)	3 (3%)	23 (28%)	7 (8%)	19 (23%)
T2b	6 (7%)	1 (1%)	5 (6%)	1 (1%)	5 (6%)
T3	8 (9%)	1 (1%)	7 (8%)	5 (6%)	3 (3%)
T4	4 (5%)	0	4 (5%)	0 (0%)	4 (5%)
Pathological N Stage					
N0	65 (78%)	10(12%)	55 (66%)	16 (20%)	49 (58%)
N1	8 (10%)	1 (1%)	7 (9%)	1 (1%)	7 (9%)
N2	10 (12%)	1 (1%)	9 (11%)	3 (3%)	7 (9%)
Pathological M Stage					
M0	80 (96%)	12(14%)	68 (82%)	19 (23%)	61 (73%)
M1b	3 (4%)	0 0%	3 (4%)	1 (1%)	2 (3%)
Histology					
Adenocarcinoma	66 (80%)	12(14%)	54 (66%)	19 (23%)	47 (57%)
Squamous cell carcinoma	14 (17%)	0 (0%)	14 (17%)	0 (0%)	14 (17%)
NSCLC NOS	3 (3%)	0 (0%)	3 (3%)	1 (1%)	2 (2%)

For the KRAS mutation, there are no significant differences in terms of age and gender between the mutant and wildtype cases. In terms of smoking history, it can be observed that none of the non-smokers presented the KRAS mutation. For the pathological stage, it seems that most patients with stage N1 and N2 are wildtype cases.

Moreover, as seen with the EGFR mutation, mutant status is only found in Adenocarcinoma.

Three experiments were conducted, first with traditional radiomic features and machine learned models, second with custom Convolutional Neural Networks (CNNs), and third, with pre-trained CNNs and fine-tuning. The experiments were executed in a computer with Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz, 16 GB of RAM memory and an NVIDIA GeForce GTX 1060 GPU. This hardware configuration limited some of the experiments as is explained in future sections. All the experiments consisted of two stages: first base classifier performance was assessed and second ensembles of several models were tested. An ensemble model is created by generating multiple models and combining them to produce an output classification. To combine the different models, a voting process is performed among them to determine the final result. There are different types of voting, for example, average voting, in which the average of the probabilities for each class of all the models is computed, and then a classification is performed based on the average probability. Another type of voting is maximum probability, in which for each case the base model with the higher pseudo-probability is selected, and the classification of the case is performed based on the pseudo-probabilities of this classifier alone.

In this study, three types of voting were tested: average, maximum, and the method proposed here, Selective Class Average Voting (SCAV). SCAV is a voting technique that is particularly useful when dealing with an unbalanced dataset, where one class (majority class) is much more frequent than the other (minority class). In SCAV, first we count how many models predicted the minority class (in our case, the mutant status), and if this quantity is above a threshold value, the final outcome is the minority class. The pseudo-probability of this particular case is computed by averaging the scores of all the models where the final result was the minority class. If the value is below the chosen threshold, the final outcome is the majority class (in this case, the wildtype status), and the class pseudo probability is computed by finding the average of probabilities of all the models where the final result was the majority class. Once the probabilities are averaged according to the previous process, a threshold of 0.5 is applied to the final score to determine if the sample belongs to the minority (mutant) or to the majority (wildtype) class. To select the best thresholds for SCAV, that is the threshold for how many models must vote for the minority class, the performance of the ensemble on the Training set was assessed, and the thresholds that enabled a higher AUC on this data were selected and applied to the Test data. Figure 25 describes the algorithm used by SCAV.

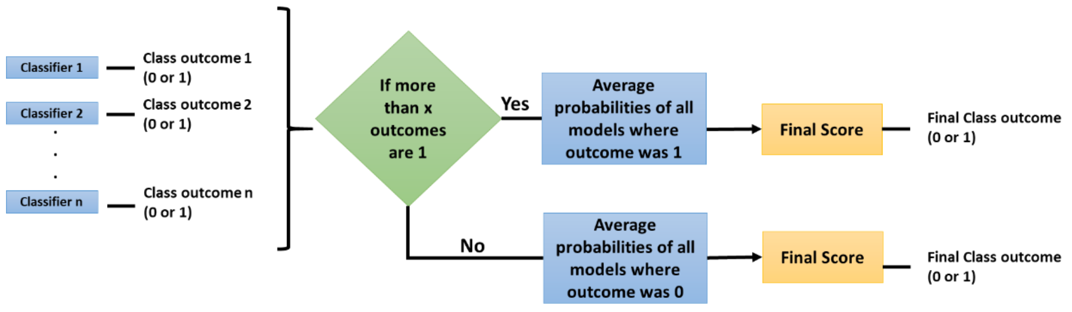
Classifier 1
Class outcome 1 (0 or 1)
Classifier 2
Class outcome 2 (0 or 1)
Classifier n
Class outcome n (0 or 1)
If more than x outcomes are 1
Yes
No
Average probabilities of all models where outcome was 1
Average probabilities of all models where outcome was 0
Final Score
Final Score
Final Class outcome (0 or 1)
Final Class outcome (0 or 1)

Figure 25. Selective Class Average Voting (SCAV) algorithm.

The use of ensembles increases the probability of obtaining better results, since we have several diverse models as inputs, and their errors tend to be outvoted by the full set of classifiers. This enables better generalization error. However, there are some disadvantages to this approach; first that it consumes more time. Several models have to be trained before an ensemble can be attempted, and it requires more computing power and resources, since we have several classifiers running at the same time. This last item creates a limitation in how many total models can be used in the ensemble.

4.1 Experiment 1: Radiomic features and Machine Learned Classifiers

Quantitative image features (N=266) presented in [96] were extracted from the segmented regions which included texture and non-texture features. Non-texture features include tumor size, tumor shape, and tumor location categories, and texture features include pixel histogram, run length, co-occurrence, Laws, and Wavelet features. To extract these features, Definiens Developer XD© (Munich, Germany) was used [89]. Definiens is based on the Cognition Network Technology that allows the development and execution of image analysis applications. Here, the Lung Tumor Analysis application was used. Most of the features were implemented within the Definiens platform, whereas some were computed with an implementation of the algorithms in C/C++ developed in a previous work by some of the advisors of this work [57].

For stage 1, the following experimental workflow was applied to predict mutation status from image features. First, the data was divided into Train and Test sets as part of a 10-fold cross-validation. Second, on each fold feature selection was applied to select the image features with the most predictive power, third, the SMOTE algorithm [97] was applied to balance the dataset, fourth, a classifier was trained with the previously selected features as inputs on the balanced training data, and finally, the resulting model was applied to the test set. Figure 26 summarizes the presented workflow.

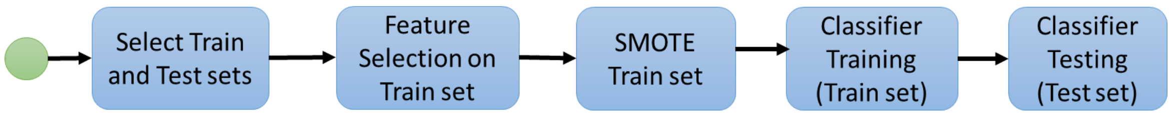
Select Train and Test sets
Feature Selection on Train set
SMOTE Train set
Classifier Training (Train set)
Classifier Testing (Test set)

Figure 26. Experiment 1 Workflow.

Feature selection was used to determine the image features with more predictive power. Sets of 5, 10, 15, and 20 features were tested. For feature selection, two methods were separately applied. The selection approaches were the Mann Whitney test [98], and ReliefF [99]. In particular, the Mann Whitney filter is not a standard feature selection algorithm, and its use for feature selection is a proposal of this work.

Since this is a case of an unbalanced dataset (one class is much more abundant than the other) an optional application of the SMOTE algorithm [97] was performed to create synthetic samples of the minority class. The SMOTE algorithm was applied with the default settings. These settings make the dataset approximately balanced by class.

Finally, a classifier was trained with the selected features. Four machine learned classifiers were used: Random Forests [41], Support Vector Machines [31], Stochastic Gradient Boosting [52] and Neural Networks [100].

For every experiment, standard metrics were computed: including accuracy, sensitivity, and specificity (assuming the mutant status as the positive case), and Area Under the ROC Curve (AUC) [73]. The workflow was applied in a ten-fold cross-validation scheme, where iteratively nine folds were used to select features and train the classifier, and the left-out fold was used for testing the model.

The whole process was coded and executed in R 3.5.1 using the package FSelector [101] for the ReliefF feature selection, package DMwR [102] for the SMOTE algorithm, and package caret [103] to test the four different classifiers. For the four classification algorithms, the classifiers were executed with the default settings of the caret package. For the Random Forest algorithm, 500 trees were generated and variables randomly sampled as candidates at each split. For the Support Vector Machines algorithm, a linear kernel was used in all the experiments, and for the cost (c parameter) c=1. For the Stochastic Gradient Boosting algorithm the following parameters were used: trees = 1000, interaction.depth =6, shrinkage = 0.1, n.minobsinnode = 10. Finally, for the neural networks, the nnet package of caret with 2 units in the hidden layer, a 100 iterations and logistic (sigmoid) as activation function were used. In the package nnet the neural network always has a single layer where you specify the number of nodes.

The implementation of the Mann-Whitney Feature Selector was coded in R.3.5.1 using the wilcox.test function to compute the p-value of every feature (every column of the dataset) and then features were sorted by this value in increasing order.

In the second stage of the experiment an ensemble of the base models was applied. Sets of 5, 10, and 20 models were tested. To select which base models would be part of the ensemble, the average performance on the training set was considered. The base models were sorted according to their average AUC on the train set, and the top 5, 10 and 20 were selected.

4.2 Experiment 2: Custom Convolutional Neural Networks

In the second experiment, CNNs were applied to the problem of predicting EGFR and KRAS mutations. CNNs are a type of deep neural networks that have proven to be useful in detecting patterns on images [104]. From the same TCIA dataset, the CT images from the 83 patients that had both tumor segmentation and mutation information were selected and processed so a volume with only the tumor would be obtained. Then images of the Region of Interest (ROI) with a uniform size of 128 × 128 pixels per slice were extracted. From the whole volume, up to three slices per patient were selected to be part of the final dataset. The slice that had the largest tumor area was selected by manual visual inspection by the lead author of the segmented images. Then, we left one out in both directions of the z-axis and selected the two slices that where closest to the chosen slice up and down. The immediately consecutive slices were not used, assuming they were too similar to the central image. A slice without a clear piece of tumor in it was discarded. The dataset was then split into three: training (65%), validation (15%), and test (20%) datasets. For this experiment, and the transfer learning experiments that also involve CNNs, a fixed train and test dataset was used instead of the 10-fold cross validation. This was performed this way since training a CNN is computationally more complex than training a machine learning model, and repeating the training 10 times takes too much time and machine power for the available hardware. Since there was more than one image from each patient, we verified that images from the same patient were assigned to the same dataset.

In the first stage of the second experiment. Several CNN models were applied to predict EGFR and KRAS mutations, varying conditions such as the CNN architecture, data augmentation, the optimizer, the learning rate, and the number of epochs of training. Augmentation was applied in these experiments with two modalities: first augmenting only the minority class, and second augmenting both classes. For both cases, the original images were augmented 4 times through rotations with 90 degrees of difference (0, 90, 180, and 270 degrees). Augmentation was only applied to the training dataset. Since this is a very small dataset, small CNN architectures were tested. Ten different architectures were applied with a different number of convolutional layers. Figures 27 to 36 show the applied ten architectures.

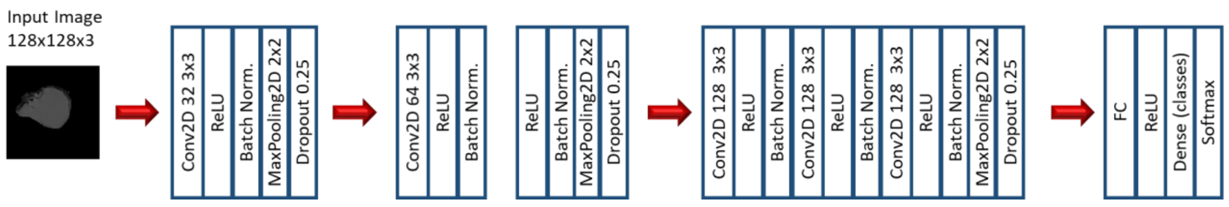

Figure 27. Architecture 1 of CNN base models.

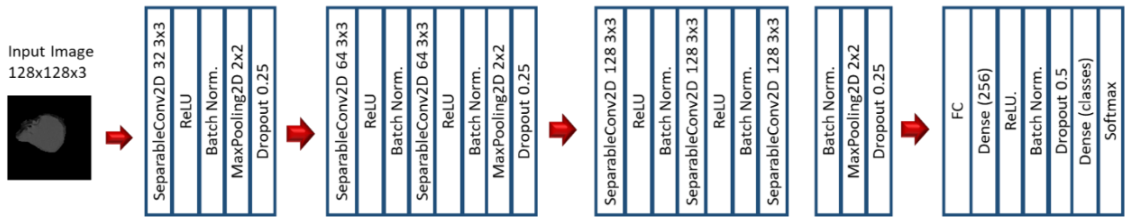

Figure 28. Architecture 2 of CNN base models.

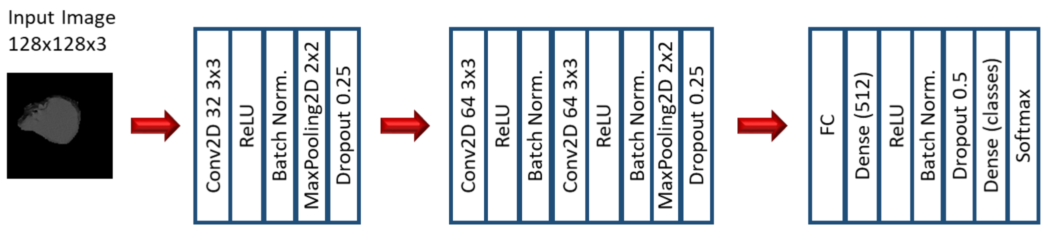

Figure 29. Architecture 3 of CNN base models.

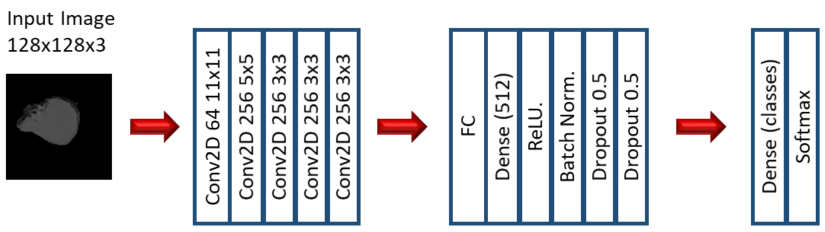

Figure 30. Architecture 4 of CNN base models.

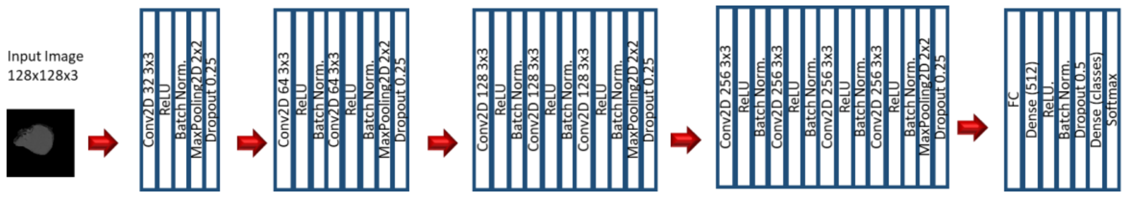

Figure 31. Architecture 5 of CNN base models.

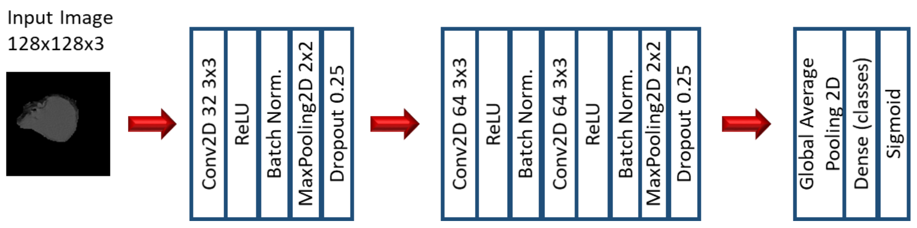

Figure 32. Architecture 6 of CNN base models.

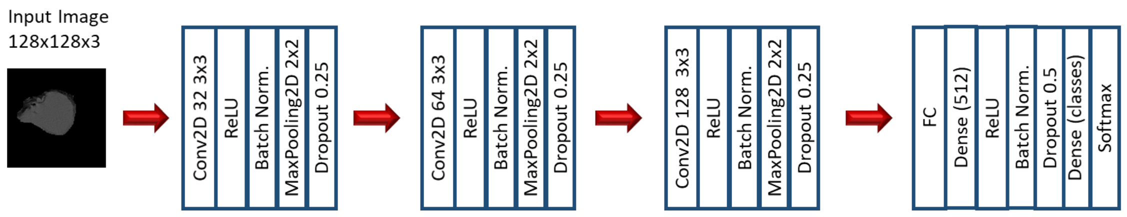

Figure 33. **Architecture 7 of CNN base models.**

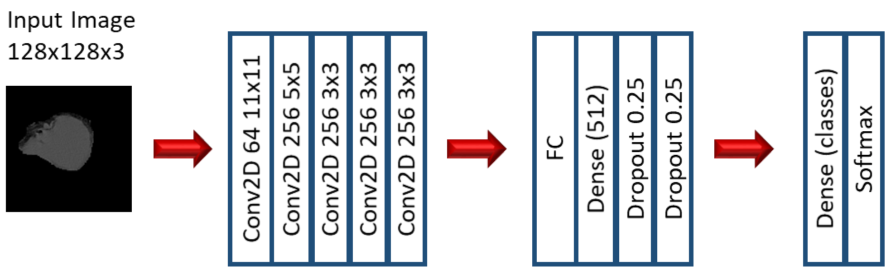

Figure 34. **Architecture 8 of CNN base models.**

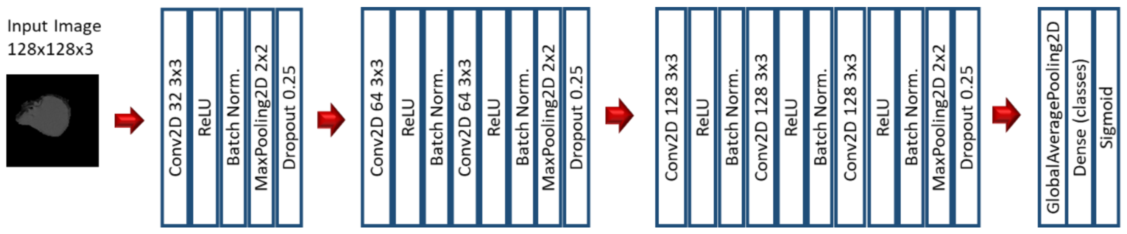

Figure 35. **Architecture 9 of CNN base models.**

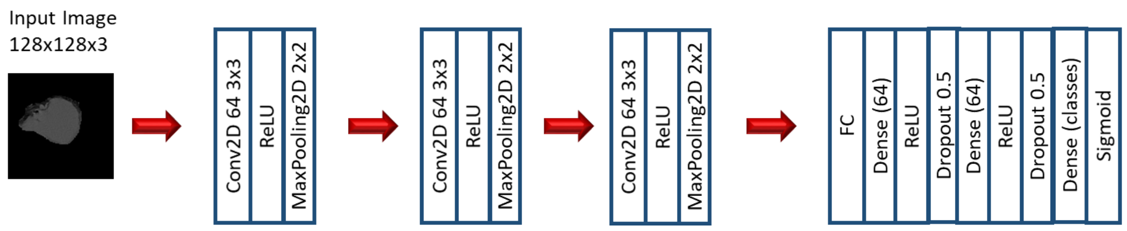

Figure 36. **Architecture 10 of CNN base models.**

Most of this architectures were inspired by the VGG-16 architecture, but just with the first few layers. Different variations of number of layers and filters following the VGG-

16 pattern were tested in the different architectures with a few exceptions. Architectures 4 and 8 were inspired by the CNN-F architecture [105] initially proposed by Chatfield et al. [106] that affirmed to work well with small datasets.

When enough good results were obtained using the base CNN models, a second stage of ensembles of CNN models was performed. Combinations of three and five models were tested from the models that performed best on training, and different types of voting were applied: average, maximum, and SCAV voting. The second experiment was coded in Python 3, and the library OpenCV was used for the image processing tasks. For the CNN generation, the library Keras with TensorFlow backend was utilized.

4.3 Experiment 3: Pre-trained CNNs and Fine Tuning

For the third experiment Transfer Learning was applied. Three well-known CNNs: VGG16, ResNet50, and Inception, pre-trained in the image database ImageNet, were applied to the problem of predicting EGFR and KRAS mutations. These networks were implemented from the library Keras. From these models, the Convolutional layers and their weights from the ImageNet training were taken and frozen, then the models were complemented with two alternative architectures which included Fully Connected Layers, and these layers were trained on the Mutations Dataset and labels (Fine Tuning). Figures 37 and 38 show the two architectures tested in the Transfer Learning experiment. The images used for this experiment were the same 128x128 images that were extracted for the custom CNN's experiment.

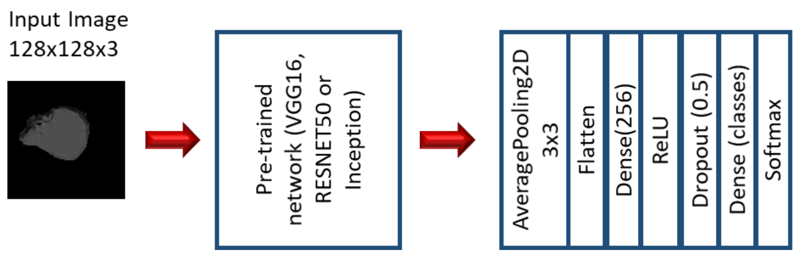

Figure 37. Architecture 1 for Transfer Learning Experiments.

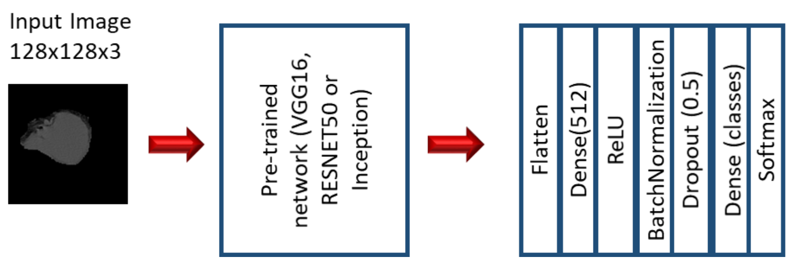

Figure 38. Architecture 2 for Transfer Learning Experiments.

First combinations of the three previously mentioned CNNs with the two complementary alternative custom architectures were tested as single classifiers. Augmentation was applied in these experiments with two modalities: first augmenting only the minority class, and second augmenting both classes. Again in both cases, the images in the training dataset were augmented 4 times through rotations, just like in the custom CNNs experiment. Different options of initial learning rate, optimizer, and number of epochs were tested. Also for this experiment fixed Train and Test datasets were used: training (65%), validation (15%), and test (20%) datasets. Again this option was chosen since 10-fold cross-validation for CNNs was too computationally expensive for the available hardware.

For stage two, Ensembles of several classifiers were tested. Average, Maximum, and SCAV voting were applied. Finally, the best models of the custom CNNs and the pre-trained networks were combined in an Ensemble.

CHAPTER 5. Radiomic features and Machine Learned Classifiers

5.1 Machine Learned Models: EGFR Mutation

The ten best results of the performance of the base classifiers for the EGFR mutation are presented in Table 4, sorted by their AUC. The classifiers are gradient based method (gbm), random forest (RF), support vector machine (SVM), and neural network (nnet).

Table 4 EGFR mutation prediction results on Test dataset, base classifiers

Feature Selection	**Classifier**	**SMOTE**	**Accuracy**	**Sensitivity**	**Specificity**	**AUC**
MW (5 features)	nnet	No	0.83	0.00	0.98	0.43
ReliefF (15 features)	SVM	Yes	0.76	0.66	0.78	0.68
ReliefF (10 features)	RF	Yes	0.76	0.41	0.82	0.67
ReliefF (15 features)	nnet	Yes	0.76	0.58	0.79	0.67
ReliefF (5 features)	RF	Yes	0.77	0.50	0.82	0.64
ReliefF (20 features)	RF	Yes	0.73	0.16	0.83	0.63
ReliefF (20 features)	SVM	Yes	0.68	0.66	0.69	0.63
ReliefF (5 features)	nnet	Yes	0.71	0.50	0.75	0.60
ReliefF (5 features)	SVM	Yes	0.79	0.25	0.89	0.59
ReliefF (15 features)	RF	Yes	0.72	0.25	0.80	0.57
MW (5 features)	gbm	Yes	0.68	0.16	0.78	0.53

The highest AUC for EGFR mutation prediction was 0.68 (standard deviation of 0.25 between folds) with an SVM classifier. This model had an Accuracy of 0.76 with a standard deviation of 0.154. For this mutation, much better results were obtained with ReliefF as feature selector. This result suggests that SVM works best because there is a clear margin of separation in the high dimensional space, and this separation follows a linear pattern, since a linear kernel is being used for these experiments.

Then, ensembles of different numbers of models with three different types of voting were tested. Table 5 presents the best results with ensembles.

Table 5 EGFR mutation prediction best results on Test dataset, ensembles.

Ensemble Combination	**Classifiers**	**Accuracy**	**Sensitivity**	**Specificity**	**AUC**
Ensemble SCAV thresh 3 (10 models)	gbm, SVM, nnet	0.59	0.75	0.57	0.70
Ensemble SCAV thresh 6 (10 models)	gbm, SVM, nnet	0.80	0.33	0.89	0.68
Ensemble Average (10 models)	gbm, SVM, nnet	0.78	0.16	0.89	0.68
Ensemble Average (5 models)	All	0.78	0.16	0.89	0.67
Ensemble Average (5 models)	RF, SVM, nnet	0.79	0.33	0.87	0.66
Ensemble Maximum (10 models)	gbm, SVM, nnet	0.75	0.41	0.82	0.59

The best AUC was 0.70 (0.149 standard deviation between folds) with SCAV. This model also had the higher sensitivity (0.75). Moreover, in another model, an accuracy of 0.80 with a standard deviation of 0.119 was obtained with a 0.68 AUC (0.20 standard deviation). It can be observed for the machine learning experiment that a higher accuracy, sensitivity, specificity, and AUC can be obtained by applying ensembles and SCAV. Different ensemble combinations can be used to favor certain metrics.

5.2 Machine Learned Models: KRAS Mutation

The results of the performance of the ten best base classifiers for the KRAS mutation are presented in Table 6. The best AUC is 0.65 with a standard deviation of 0.173 between folds. This model also had an accuracy of 0.70 with a standard deviation of 0.193 For this mutation, similar results could be obtained with both feature selection methods, though ReliefF was still best. Again SVM was the algorithm that performed better. This suggests that there is a linear separation margin between the two classes in the high dimensional space. Then, ensembles of different numbers of models with three different types of voting were tested. Table 7 presents the best results with ensembles. The ensemble approach resulted in an improved AUC of 0.71 (0.207 standard deviation) with a 0.72 accuracy (0.163 standard deviation) using SCAV. Again, the models with best accuracy and best AUC were obtained with the proposed voting scheme. This was the best AUC that could be obtained for the KRAS mutation with the machine learning models.

Table 6 KRAS mutation prediction results on Test dataset, base classifiers

Feature Selection	Classifier	SMOTE	Accuracy	Sensitivity	Specificity	AUC
MW (10 features)	nnet	No	0.72	0.10	0.93	0.44
Relief (10 features)	nnet	No	0.75	0.00	1.00	0.44
ReliefF (5 features)	**SVM**	**Yes**	**0.70**	**0.35**	**0.81**	**0.65**
MW (15 features)	SVM	Yes	0.64	0.40	0.72	0.64
ReliefF (5 features)	gbm	Yes	0.64	0.60	0.65	0.63
ReliefF (20 features)	gbm	Yes	0.63	0.50	0.67	0.63
MW (10 features)	SVM	Yes	0.64	0.45	0.70	0.63
MW (20 features)	SVM	Yes	0.71	0.40	0.81	0.63
ReliefF (15 features)	SVM	Yes	0.67	0.45	0.75	0.62
MW (5 features)	SVM	Yes	0.71	0.35	0.83	0.62
MW (15 features)	gbm	Yes	0.67	0.40	0.77	0.62
ReliefF (15 features)	RF	Yes	0.62	0.40	0.70	0.60

Table 7 KRAS mutation prediction best results on Test dataset, ensembles.

Ensemble Combination	Classifiers	Accuracy	Sensitivity	Specificity	AUC
Ensemble SCAV thresh 8 (10 models)	SVM, nnet	0.72	0.20	0.89	0.71
Ensemble SCAV thresh 6 (10 models)	SVM, nnet	0.73	0.30	0.87	0.69
Ensemble Average (5 models)	SVM	0.70	0.35	0.81	0.67
Ensemble Maximum (5 models)	SVM	0.70	0.40	0.80	0.66
Ensemble Average (10 models)	SVM, nnet	0.66	0.35	0.76	0.65

CHAPTER 6. CUSTOM CONVOLUTIONAL NEURAL NETWORKS

6.1 Convolutional Neural Networks: EGFR Mutation

The best results of EGFR mutation prediction applying CNNs on the Test set are presented in Table 8. Although Stochastic Gradient Descent (SGD) and Adam were tested as optimizers, it can be observed that all the best results were obtained with SGD as optimizer. This suggests that SGD can be a good choice when dealing with small datasets with small CNN architectures. Also, according to some authors [107], SGD is more locally unstable and is more likely to converge to the minima at the flat or asymmetric basins/valleys which often have better generalization performance over other types of minima. The best result was obtained with Architecture 4, which is presented in Figure 30. This model had an AUC of 0.846 and an accuracy of 0.800. This result is significantly better than the one obtained with the machine learning approach. Among the reasons why this architecture works best, we can highlight that Architecture 4, unlike most of the other architectures, applies a 11x11 kernel, while the rest apply smaller kernels (3x3). This suggests that maybe the texture patterns that characterize EGFR mutation are bigger and can be better detected with kernels of a greater size.

Table 8 EGFR mutation best results on Test dataset, CNNs.

Model	Optimizer	Learning Rate	Epochs	Accuracy	Sensitivity	Specificity	AUC
Arch. 4	SGD	0.0005	30	0.800	0.667	0.846	0.846
Arch. 6	SGD	0.0005	30	0.771	0.222	0.961	0.752
Arch. 1	SGD	0.01	8	0.400	1.000	0.192	0.688
Arch. 6	SGD	0.01	10	0.657	0.666	0.654	0.675
Arch. 3	SGD	0.01	7	0.543	0.778	0.461	0.671
Arch. 4	SGD	0.01	10	0.543	0.778	0.461	0.628
Arch. 1	SGD	0.01	30	0.514	0.778	0.423	0.623
Arch. 2	SGD	0.01	30	0.542	0.667	0.538	0.571
Arch. 4	SGD	0.01	20	0.600	0.444	0.654	0.559

After the base CNN models were obtained, an ensemble of the best CNN models was created. Table 9 presents the results of the best ensembles of CNN models. The best result in terms of AUC was 0.820 and an accuracy of 0.828, this result was obtained with a combination of the three best models and SCAV. An even better accuracy (0.857) was obtained with the combination of the five best models. This was the best accuracy for the

EGFR mutation. Even if in this case there was not an increase in performance in terms of AUC, a better accuracy was obtained by applying SCAV.

Table 9 EGFR Mutation Best Results, Ensembles of CNNs

Model	Accuracy	Sensitivity	Specificity	AUC
Ensemble (3 models) SCAV thresh 3	0.828	0.667	0.885	0.820
Ensemble (5 models) SCAV thresh 5	0.857	0.667	0.923	0.778
Ensemble (3 models) Average	0.486	0.778	0.385	0.743
Ensemble (5 models) Average	0.628	0.778	0.577	0.641
Ensemble (3 models) Maximum	0.371	0.778	0.231	0.624

6.2 Convolutional Neural Networks: KRAS Mutation

The best results of KRAS mutation prediction using CNNs on the Test set are presented in Table 10. Analyzing the results for KRAS, we can see there is not a model that performs well according to all three metrics. The best result according to AUC is 0.739, however, the sensitivity of this model is zero, so none of the mutant cases were detected. The model with the best accuracy has 72.2% and a sensitivity of 0.25, so it is a more balanced result, however, the AUC is only 0.566.

Table 10 KRAS mutation best results on Test dataset, CNNs.

Model	Optimizer	Learning Rate	Epochs	Accuracy	Sensitivity	Specificity	AUC
Arch. 1	SGD	0.01	60	0.667	0.000	1.000	0.739
Arch. 6	Adam	0.005	10	0.333	1.000	0.000	0.607
Arch. 6	Adam	0.001	10	0.667	0.000	1.000	0.593
Arch. 1	Adam	0.005	15	0.722	0.250	0.958	0.566
Arch. 1	SGD	0.01	90	0.667	0.000	1.000	0.555
Arch. 1	SGD	0.01	10	0.555	0.667	0.500	0.531

In order to improve the results, an ensemble of the best CNN models was created. Table 11 shows the best results with ensembles of CNNs. The best AUC that could be obtained in this stage was 0.778, which was obtained with an ensemble of the three best models and average voting. This was the best AUC that could be obtained for the KRAS mutation with custom CNNs, and is significantly better than the result obtained with the machine learning models. This was the only stage where the best results in terms of AUC were not obtained with SCAV; however, an equal accuracy could be obtained by applying SCAV with the best three models.

Table 11 **KRAS mutation best results on Test dataset, ensembles of CNNs.**

Model	**Accuracy**	**Sensitivity**	**Specificity**	**AUC**
Ensemble (3 models) Average	0.722	0.250	0.958	0.778
Ensemble (3 models) SCAV thresh 2	0.722	0.250	0.958	0.722
Ensemble (4 models) SCAV thresh 3	0.722	0.250	0.958	0.642
Ensemble (7 models) SCAV thresh 4	0.694	0.416	0.833	0.618
Ensemble (7 models) SCAV thresh 5	0.694	0.083	1.000	0.604

CHAPTER 7. PRE-TRAINED CNNS AND FINE TUNING

7.1 Pre-trained CNNs: EGFR Mutation

For the first stage of the Transfer Learning Experiments combinations of a pre-trained network (VGG16, ResNet50, or Inception) and a few additional layers (Models 1 and 2, shown in Figures 37 and 38) were tested. Table 12 presents the best results of single classifiers with pre-trained networks and fine-tuning

Table 12 EGFR Mutation Best Results, Pre-trained CNNs

Model	Optimizer	Learning Rate	Epochs	Accuracy	Sensitivity	Specificity	AUC
VGG16 – Model 1	**SGD**	**0,0005**	**10**	**0,686**	**0,889**	**0,615**	**0,820**
ResNet50 - Model 2	SGD	0,0005	5	0,600	0,778	0,538	0,778
VGG16 – Model 2	SGD	0,0005	10	0,600	0,889	0,500	0,747
ResNet50 – Model 2	SGD	0,0005	8	0,600	0,555	0,615	0,658
ResNet50 - Model 1	Adam	0,005	10	0,743	0,000	1,000	0,504
Inception - Model 2	SGD	0,0005	8	0,514	0,555	0,500	0,602
Inception - Model 1	SGD	0,005	10	0,686	0,111	0,885	0,551
Inception - Model 1	SGD	0,0005	10	0,800	0,222	1,000	0,547

The best result in terms of AUC was 0.820 and was obtained with VGG16 and architecture 1. The Sensitivity of this model is also high (88.9%). After the single models were evaluated, ensembles of pre-trained CNNs were tested. In this step, the three best models in terms of AUC in training and the three best models in terms of accuracy on training were tested in the Ensembles. Table 13 presents the best results. Models with higher AUC (0.837) and higher accuracy (85,7%) could be obtained with SCAV.

Table 13 **EGFR Mutation Best Results, Ensembles of Pre-trained CNNs**

Ensemble Combination	Accuracy	Sensitivity	Specificity	AUC
Ensemble Average (3 best models AUC)	0,657	1,000	0,538	0,837
Ensemble SCAV thresh 2 ((3 best models AUC)	0,657	0,889	0,577	0,761
Ensemble SCAV thresh 3 (3 best models AUC)	**0,857**	**0,667**	**0,923**	**0,829**
Ensemble SCAV thresh 2 (3 best models accuracy)	0,743	0,778	0,731	0,765
Ensemble SCAV thresh 3 (3 best models accuracy)	0,800	0,222	1,000	0,564

Finally, ensembles of the best custom CNNs and the best pre-trained models were tested. The results are presented in Table 14.

Table 14 **EGFR Mutation Best Results, Ensembles of Pre-trained and Custom CNNs**

Ensemble Combination	Accuracy	Sensitivity	Specificity	AUC
Ensemble SCAV (3 best pre-trained models, 2 best custom CNNs) thresh 3	0,828	0,778	0,846	0,855
Ensemble SCAV (3 best pre-trained models, 2 best custom CNNs) thresh 4	**0,886**	**0,556**	**1,000**	**0,914**

Combining the best custom CNNs and the best pre-trained models we obtained the best AUC of all the experiments, 0.914. This ensemble combination also has very good accuracy (88.6%). It can be concluded that this is the best model for EGFR mutation prediction. This result is slightly better than the one obtained by Gevaert. et. al. [3] (0.89) with the same dataset, but has the advantage that this result was obtained with features extracted automatically from the image and not given manually by a radiologist. This result is also better than the one obtained by Wang. et. al. [91] with deep learning techniques (but with a different dataset). Figure 39 shows sample images of correct and incorrect classification of EGFR mutant cases with our best model. Figure 40 presents a sample image of a correctly classified EGFR wildtype case with the best model. Since the specificity of this model is 1, there were no incorrectly classified wildtype cases. These images suggest that the classifier works best with big tumors.

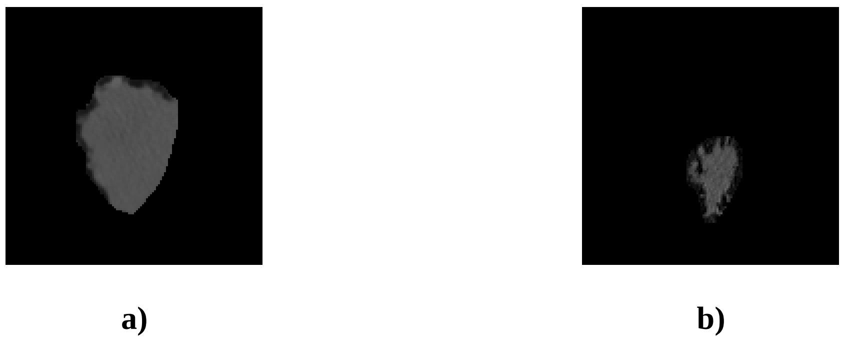

Figure 39. **Examples of EGFR Mutant cases a) Correctly Classified, b) Incorrectly Classified.**

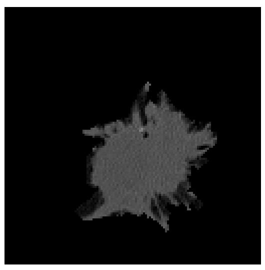

Figure 40. **Example of a correctly classified EGFR Wildtype case.**

7.2 Pre-trained CNNs: KRAS Mutation

For the KRAS mutation, the same combinations of one of the three pre-trained networks (VGG16, ResNet50, and Inception) with fine-tuning with the architectures presented in Figures 37 and 38 were tested. First, single classifiers were assessed. Table 15 presents the best results of the base classifiers. The best model was VGG16 with model 2, with an AUC of 0.778 and an accuracy of 72.2%. This is the best and more balanced result obtained with the base classifiers for the KRAS mutation, which shows the effectiveness of the Transfer Learning approach for this particular problem. Compared to the results obtained with the EGFR mutation, where the best result for the base classifier was achieved with a custom architecture, we can speculate that the texture patterns of the KRAS mutation are smaller, since they work better with the 3x3 kernels traditional to the VGG-16 architecture, and can be perceived with more standard feature extraction filters, since the model benefits from its training in Imagenet, a generic image database.

Table 15 **KRAS Mutation Best Results, Pre-trained CNNs.**

Model	Optimizer	Learning Rate	Epochs	Accuracy	Sensitivity	Specificity	AUC
VGG16-Model2	**Adam**	**0,0005**	**50**	**0,722**	**0,667**	**0,750**	**0,778**
VGG16-Model2	Adam	0,0005	10	0,667	0,333	0,833	0,778
Inception-Model1	SGD	0,0005	10	0,583	0,916	0,417	0,767
VGG16-Model1	SGD	0,0005	50	0,722	0,333	0,917	0,712
VGG16-Model2	Adam	0,0005	70	0,694	0,500	0,792	0,712

After single classifiers were tested, ensembles of pre-trained CNNs were assessed. The best results of this approach are presented in Table 16.

Table 16 **KRAS Mutation Best Results, Ensembles of Pre-trained CNNs.**

Ensemble Combination	Accuracy	Sensitivity	Specificity	AUC
Ensemble Average (5 best models)	0,722	0,583	0,792	0,809
Ensemble Average (3 best AUC models)	**0,750**	**0,667**	**0,792**	**0,809**
Ensemble SCAV thresh 2 (3 best AUC models)	0,778	0,667	0,833	0,771
Ensemble SCAV thresh 3 (3 best AUC models)	0,694	0,250	0,917	0,767
Ensemble SCAV thresh 2 (5 best models)	0,778	0,750	0,792	0,757

With the ensembles of pre-trained CNNs, the best results in terms of AUC was 0.809 obtained both with an ensemble with average voting of the 5 and the 3 best models, however, the accuracy of the ensemble with the 3 best models was higher. Even if in this case the best result was not obtained using SCAV, good results were also obtained with this voting approach, in particular, the model with the higher accuracy of the ensembles (77,8%) was obtained with SCAV.

For this mutation ensembles of Pre-Trained and custom CNNs were also attempted, however the results were not better than the ones obtained with only Pre-Trained CNNs

The results obtained with Ensembles of Pre-Trained CNNs were the best for the KRAS mutation prediction, and particularly the 0,809 AUC was the highest obtained for this mutation. This result is very promising, especially considering that in the work presented by Gevaert et. al. [3] with the same dataset, a conclusive predictive model for the KRAS mutation could not be obtained. Also, in the state of the art, the higher AUC obtained for the KRAS mutation was 0.75, which is lower than the AUC obtained in this work, however, this result was obtained with a different dataset. Figure 41 shows sample images of a correctly and incorrectly classified KRAS mutant case for our best model. Figure 42 presents a correctly and incorrectly classified case of KRAS wildtype tumor with our best model. Again these images suggest that the classifier works better with bigger tumors.

Figure 41. Examples of KRAS Mutant cases a) Correctly Classified, b) Incorrectly Classified.

Figure 42. Examples of KRAS Wildtype cases a) Correctly Classified, b) Incorrectly Classified.

CHAPTER 8. CONCLUSIONS AND FUTURE WORK

In this study, we analyzed the effectiveness of using ensembles in the prediction of EGFR and KRAS mutations in a small dataset, in particular, we assessed the performance of a proposed voting scheme SCAV. We tested this scheme with both ensembles of machine learning models, ensembles of custom CNNs, and ensembles of pre-trained CNNs, and a significant improvement from the base classifiers was observed.

For the EGFR mutation and machine learning models, the best AUC was obtained by creating a model with ReliefF's 15 best features and a Support Vector Machine (SVM) as classifier. The AUC increased when applying an Ensemble with SCAV. The image features that were more frequently associated with this mutation are texture features, such as 3D Laws and Wavelet features, others related to tumor shape, such as Flatness and Asymmetry, and finally GLSZM features related to gray level zones in the image. For the deep learning models with custom CNNs, the highest AUC (0.846) was obtained by a CNN with Architecture 4. In this case we did not see an increase in terms of AUC when we applied Ensembles, but the accuracy increased when using an Ensemble with SCAV voting. The performance further increased when we applied Ensembles of Pre-Trained CNNs and custom CNNs with SCAV voting (0,914 AUC). This was the best result that could be obtained for the EGFR mutation, and it indicates that ensembles, and in particular our voting scheme can significantly increase performance of classifiers when dealing with a small dataset and a not so outstanding performance of the base models. Our results were better than the ones obtained by Gevaert et. al. [3] with the same dataset, and our model did not require semantic features manually specified by a radiologist. Further, our best model obtained a higher

AUC than the ones presented by the most recent works that used deep learning [91] and nomograms [93], as is summarized in Table 1.

For the KRAS mutation, an increase in performance was also observed, both with the machine learning models and the deep learning ones. The AUC of the machine learning models went from 0.65 to 0.71 with an ensemble of 10 models and SCAV for the voting method. Among the image features that were associated with the KRAS mutation, we could find texture features, particularly 3D Laws features, and features related to grey level non-uniformity. For the CNNs, the performance went from an AUC of 0.739 to an AUC of 0.778 with an ensemble of CNNs. The performance also increased when we built ensembles of Pre-Trained CNNs (0,809 AUC). These results are much better than the ones obtained in [3] where a conclusive model for KRAS mutation could not be found for this same dataset. This result also outperforms other recent works of the state of the art.

For both mutations, in most cases better results could be obtained by applying SCAV as the voting method, rather than average and maximum voting, however, a more rigorous method to determine the best threshold is still necessary. Also, we proposed a new Feature Selection algorithm based on the Mann Whitney test. Although in some cases this filter had good performance, better results were obtained with the standard feature selection algorithm, Relief. Furthermore, higher Sensitivity was obtained when applying the SMOTE algorithm for the machine learning models, which is an effective strategy to handle unbalanced classification datasets.

This work showed novel ways to use ensembles of CNNs and non-neural classifiers on small data to achieve state-of-the-art results. Our proposed approach, which is to use ensembles with SCAV, shows in this study that the performance of classifiers can be improved, even when the base models do not perform that well, and this is an important contribution from this work. Since larger datasets will enable better models, we firmly believe that if our approach is applied with more data it will yield outstanding performance and may generate models that can be used in clinical practice. This indicates a promising future for detecting these mutations in a non-invasive way.

REFERENCES

[1] American Cancer Society *et al.*, “Global cancer facts & figures 4th Edition,” *Am. Cancer Soc.*, pp. 1–76, 2018.

[2] R. J. Gillies, P. E. Kinahan, and H. Hricak, “Radiomics: Images are more than pictures, they are data,” *Radiology*, vol. 278, no. 2, pp. 563–577, 2016.

[3] O. Gevaert *et al.*, “Predictive radiogenomics modeling of EGFR mutation status in lung cancer,” *Sci. Rep.*, vol. 7, no. 1, pp. 1–8, 2017.

[4] G. Bethune, D. Bethune, N. Ridgway, and Z. Xu, “Epidermal growth factor receptor (EGFR) in lung cancer : an overview and update,” 2010.

[5] G. J. Riely and M. Ladanyi, “KRAS mutations: an old oncogene becomes a new predictive biomarker,” *J. Mol. Diagn.*, vol. 10, no. 6, pp. 493–495, Nov. 2008.

[6] E. M. Gaughan and D. B. Costa, “Genotype-driven therapies for non-small cell lung cancer: Focus on EGFR, KRAS and ALK gene abnormalities,” *Ther. Adv. Med. Oncol.*, vol. 3, no. 3, pp. 113–125, 2011.

[7] J. W. Goldman *et al.*, “A Randomized Phase III Study of Abemaciclib Versus Erlotinib in Patients with Stage IV Non-small Cell Lung Cancer With a Detectable KRAS Mutation Who Failed Prior Platinum-Based Therapy: JUNIPER,” *Front. Oncol.*, vol. 10, no. April 2017, pp. 1–12, 2020.

[8] F. Fenizia *et al.*, “EGFR mutations in lung cancer: From tissue testing to liquid biopsy,” *Futur. Oncol.*, vol. 11, no. 11, pp. 1611–1623, 2015.

[9] A. M. Rutman and M. D. Kuo, “Radiogenomics : Creating a link between molecular diagnostics and diagnostic imaging,” vol. 70, pp. 232–241, 2009.

[10] American Cancer Society, “Key Statistics for Lung Cancer.” [Online]. Available: https://www.cancer.org/cancer/non-small-cell-lung-cancer/about/key-statistics.html. [Accessed: 04-Dec-2018].

[11] C. Pardo and R. Cendales, *Cáncer en Colombia 2007-2011*. 2015.

[12] Grupo IMD, “El cáncer de pulmón en Colombia.” [Online]. Available: https://www.grupoimd.com.co/blog/cancer-de-pulmon-colombia/. [Accessed: 05-Nov-2020].

[13] Y. Balagurunathan *et al.*, “Reproducibility and Prognosis of Quantitative Features Extracted from CT Images,” *Transl. Oncol.*, vol. 7, no. 1, pp. 72–87, 2014.

[14] P. Lambin *et al.*, “Radiomics: Extracting more information from medical images using advanced feature analysis,” *Eur. J. Cancer*, vol. 48, no. 4, pp. 441–446, 2012.

[15] N. Leal, S. Moreno, and E. Zurek, “Simple method for detecting visual saliencies based on dictionary learning and sparse coding,” 2019, vol. 2019-June.

[16] G. Lee *et al.*, “Radiomics and its emerging role in lung cancer research, imaging biomarkers and clinical management: State of the art,” *Eur. J. Radiol.*, vol. 86, pp. 297–307, 2017.

[17] S. Moreno, M. Bonfante, E. Zurek, and H. S. Juan, “Study of medical image processing techniques applied to lung cancer,” 2019, vol. 2019-June.

[18] F. C. Detterbeck and C. J. Gibson, “Turning Gray: The Natural History of Lung Cancer Over Time,” *J. Thorac. Oncol.*, vol. 3, no. 7, pp. 781–792, 2008.

[19] B. de Hoop, H. Gietema, S. van de Vorst, K. Murphy, R. J. van Klaveren, and M. Prokop, “Pulmonary Ground-Glass Nodules: Increase in Mass as an Early Indicator of Growth,” *Radiology*, vol. 255, no. 1, pp. 199–206, Mar. 2010.

[20] A. Kamiya, S. Murayama, H. Kamiya, T. Yamashiro, Y. Oshiro, and N. Tanaka, “Kurtosis and skewness assessments of solid lung nodule density histograms: Differentiating malignant from benign nodules on CT,” *Jpn. J. Radiol.*, vol. 32, no. 1, pp. 14–21, 2014.

[21] Y. Chong *et al.*, “Quantitative CT variables enabling response prediction in neoadjuvant therapy with EGFR-TKIs: Are they different from those in neoadjuvant concurrent chemoradiotherapy?,” *PLoS One*, vol. 9, no. 2, pp. 1–8, 2014.

[22] J. S. Peper and R. E. Dahl, “Radiomics: a new application from established techniques,” *Expert Rev. Precis. Med. Drug Dev.*, vol. 1, no. 2, pp. 207–226, 2016.

[23] K. I. Laws, “Rapid Texture Identification,” in *Proc.SPIE*, 1980, vol. 0238.

[24] T. Pyka *et al.*, “Textural features in pre-treatment [F18]-FDG-PET/CT are correlated with risk of local recurrence and disease-specific survival in early stage NSCLC patients receiving primary stereotactic radiation therapy,” *Radiat. Oncol.*, vol. 10, no. 1, p. 100, 2015.

[25] T. Win *et al.*, “Tumor heterogeneity and permeability as measured on the CT component of PET/CT predict survival in patients with non-small cell lung cancer,” *Clin. Cancer Res.*, vol. 19, no. 13, pp. 3591–3599, 2013.

[26] C. Bishop, *Pattern recognition and machine learning*. New York: Springer, 2006.

[27] S. Russell, *Artificial intelligence : a modern approach*. Upper Saddle River, New Jersey: Prentice Hall, 2010.

[28] R. Gandhi, “Support Vector Machine — Introduction to Machine Learning Algorithms,” 2018. [Online]. Available: https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47. [Accessed: 27-Sep-2021].

[29] T. Hastie, R. Tibshirani, and J. Friedman, “The Elements of Statistical Learning,” *Springer Series in Statistics*, 2009. [Online]. Available: https://web.stanford.edu/~hastie/ElemStatLearn/data.html. [Accessed: 01-Aug-2017].

[30] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, “Section 16.5. Support Vector Machines,” in *Numerical Recipes: The Art of Scientific Computing*, New York: Cambridge University Press, 2007.

[31] C. Cortes and V. Vapnik, “Support-vector networks,” *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, 1995.

[32] B. E. Boser, V. N. Vapnik, and I. M. Guyon, “Training Algorithm Margin for Optimal Classifiers,” *Perception*, pp. 144–152, 1992.

[33] K.-B. Duan and S. S. Keerthi, “Which Is the Best Multiclass SVM Method? An Empirical Study BT - Multiple Classifier Systems,” 2005, pp. 278–285.

[34] J. R. Quinlan, “Induction of decision trees,” *Mach. Learn.*, vol. 1, no. 1, pp. 81–106, 1986.

[35] T. K. Ho, “The Random Subspace Method for Constructing Decision Forests,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 8, pp. 832–844, 1998.

[36] I. K. Sethi and G. P. R. Sarvarayudu, “Hierarchical Classifier Design Using Mutual Information,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-4, no. 4, pp. 441–445, 1982.

[37] L. Breiman, *Classification and regression trees*. New York: Chapman & Hall, 1993.

[38] J. Mingers, “An empirical comparison of selection measures for decision-tree induction,” *Mach. Learn.*, vol. 3, no. 4, pp. 319–342, 1989.

[39] Y. Park and J. Sklansky, “Automated design of piecewise-linear classifiers of multiple-class data,” in *[1988 Proceedings] 9th International Conference on Pattern Recognition*, 1988, pp. 1068–1071 vol.2.

[40] S. K. Murthy, S. Kasif, and S. Salzberg, “A System for Induction of Oblique Decision Trees,” *J. Artif. Int. Res.*, vol. 2, no. 1, pp. 1–32, 1994.

[41] T. K. Ho, “Random decision forests,” in *Proceedings of 3rd International Conference on Document Analysis and Recognition*, 1995, vol. 1, pp. 278–282 vol.1.

[42] T. K. Ho, “The random subspace method for constructing decision forests,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 8, pp. 832–844, 1998.

[43] S. Tahsildar, “Random Forest Algorithm In Trading Using Python,” 2019. [Online]. Available: https://blog.quantinsti.com/random-forest-algorithm-in-python/. [Accessed: 29-Sep-2021].

[44] E. M. Kleinberg, “An overtraining-resistant stochastic modeling method for pattern recognition,” *Ann. Stat.*, vol. 24, no. 6, pp. 2319–2349, Dec. 1996.

[45] L. Breiman, “Random Forests,” *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.

[46] W. S. McCulloch and W. Pitts, “A logical calculus of the ideas immanent in nervous activity,” *Bull. Math. Biophys.*, vol. 5, no. 4, pp. 115–133, 1943.

[47] Wikipedia, “Artificial neural network.” [Online]. Available: https://en.wikipedia.org/wiki/Artificial_neural_network. [Accessed: 04-Oct-2021].

[48] L. Hardesty, “Explained: Neural networks,” 2017. [Online]. Available: https://news.mit.edu/2017/explained-neural-networks-deep-learning-0414. [Accessed: 04-Oct-2021].

[49] IBM, “Neural Networks.” [Online]. Available: https://www.ibm.com/cloud/learn/neural-networks. [Accessed: 22-Oct-2021].

[50] F. ROSENBLATT, “The perceptron: a probabilistic model for information storage and organization in the brain.,” *Psychol. Rev.*, vol. 65, no. 6, pp. 386–408, Nov. 1958.

[51] L. Breiman, “ARCING THE EDGE Leo Breiman Technical Report 486 , Statistics Department University of California, Berkeley CA. 94720,” vol. 4, pp. 1–14, 1997.

[52] J. H. Friedman, “Stochastic gradient boosting,” *Comput. Stat. Data Anal.*, vol. 38, no. 4, pp. 367–378, 2002.

[53] L. Breiman, “Bagging Predictors,” *Mach. Learn.*, vol. 24, no. 2, pp. 123–140, 1996.

[54] Y. Freund and R. E. Schapire, “Experiments with a New Boosting Algorithm,” *Proc. 13th Int. Conf. Mach. Learn.*, pp. 148–156, 1996.

[55] L. Breiman, “USING ADAPTIVE BAGGING TO DEBIAS REGRESSIONS,” 1999.

[56] Mathworks, “Convolutional Neural Network,” 2020. [Online]. Available:

https://www.mathworks.com/solutions/deep-learning/convolutional-neural-network.html. [Accessed: 11-Jun-2020].

[57] D. H. Hubel and T. N. Wiesel, “Receptive fields, binocular interaction and functional architecture in the cat’s visual cortex,” *J. Physiol.*, vol. 160, no. 1, pp. 106–154, Jan. 1962.

[58] I. E. Cicero, “Utilización de redes neuronales convolucionales para la detección de tipos de imágenes(Tesis de especializacion),” 2018.

[59] E. Buelvas, K. Martelo, O. Movilla, and S. Moreno, “Image-based search engine for touristic recommendations ,” *RISTI - Rev. Iber. Sist. e Tecnol. Inf.*, vol. 2020, no. E36, pp. 102–113, 2020.

[60] Adrian Rosebrock, *Deep Learning for Computer Vision with Python*. PyImageSearch.com, 2019.

[61] S. Ioffe and C. Szegedy, “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift,” in *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, 2015, pp. 448–456.

[62] Stanford University, “Imagenet,” 2016. [Online]. Available: http://www.image-net.org/. [Accessed: 09-Nov-2020].

[63] O. Russakovsky *et al.*, “ImageNet Large Scale Visual Recognition Challenge,” *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, 2015.

[64] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 2015.

[65] Muneeb ul Hassan, “VGG16 – Convolutional Network for Classification and Detection,” 2018. [Online]. Available: https://neurohive.io/en/popular-networks/vgg16/. [Accessed: 10-Nov-2021].

[66] P. Monasse, “TP : Implémentez votre premier réseau de neurones avec Keras.” [Online]. Available: https://openclassrooms.com/fr/courses/4470531-classez-et-segmentez-des-donnees-visuelles/5097666-tp-implementez-votre-premier-reseau-de-neurones-avec-keras. [Accessed: 10-Nov-2021].

[67] C. Szegedy *et al.*, “Going deeper with convolutions,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2015, vol. 07-12-June, pp. 1–9.

[68] Szegedy et al., “Inception-v3.” [Online]. Available: https://paperswithcode.com/method/inception-v3#. [Accessed: 10-Nov-2021].

[69] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016, vol. 2016-Decem, pp. 770–778.

[70] Acervo Lima, “Redes residuales (ResNet): deep learning.” [Online]. Available: https://es.acervolima.com/2021/02/09/redes-residualesresnet-deep-learning/. [Accessed: 11-Nov-2021].

[71] D. M. W. Powers, “Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation,” no. May, 2020.

[72] K. Pykes, “Confusion Matrix ‘Un-confused,’” 2020. [Online]. Available:

https://towardsdatascience.com/confusion-matrix-un-confused-1ba98dee0d7f. [Accessed: 17-Nov-2021].

[73] T. Fawcett, “An introduction to ROC analysis,” *Pattern Recognit. Lett.*, vol. 27, no. 8, pp. 861–874, 2006.

[74] Google, “Classification: ROC Curve and AUC.” [Online]. Available: https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc. [Accessed: 19-Nov-2021].

[75] S. Narkhede, “Understanding AUC - ROC Curve,” 2018. [Online]. Available: https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5. [Accessed: 19-Nov-2021].

[76] M. Scrivener, E. E. C. de Jong, J. E. van Timmeren, T. Pieters, B. Ghaye, and X. Geets, “Radiomics applied to lung cancer: a review,” *Transl. Cancer Res.*, vol. 5, no. 4, pp. 398–409, 2016.

[77] M. Saad and T.-S. Choi, “Deciphering unclassified tumors of non-small-cell lung cancer through radiomics,” *Comput. Biol. Med.*, vol. 91, pp. 222–230, 2017.

[78] J. Ma, Q. Wang, Y. Ren, H. Hu, and J. Zhao, “Automatic lung nodule classification with radiomics approach,” vol. 9789, p. 978906, 2016.

[79] Y. Huang *et al.*, “Radiomics Signature: A Potential Biomarker for the Prediction of Disease-Free Survival in Early-Stage (I or II) Non—Small Cell Lung Cancer,” *Radiology*, vol. 281, no. 3, pp. 947–957, Jun. 2016.

[80] W. Wu *et al.*, “Exploratory Study to Identify Radiomics Classifiers for Lung Cancer Histology,” *Front. Oncol.*, vol. 6, no. March, pp. 1–11, 2016.

[81] N. Emaminejad, S. Yan, Y. Wang, W. Qian, Y. Guan, and B. Zheng, “Applying a radiomics approach to predict prognosis of lung cancer patients,” *Prog. Biomed. Opt. Imaging - Proc. SPIE*, vol. 9785, pp. 1–7, 2016.

[82] G. Pinheiro *et al.*, “Identifying relationships between imaging phenotypes and lung cancer-related mutation status: EGFR and KRAS,” *Sci. Rep.*, vol. 10, no. 1, pp. 1–9, 2020.

[83] S. Bakr *et al.*, “A radiogenomic dataset of non-small cell lung cancer,” *Sci. Data*, vol. 5, no. 1, p. 180202, 2018.

[84] Computational Imaging & Bioinformatics Lab - Harvard Medical School, “PyRadiomcis,” 2017. [Online]. Available: http://www.radiomics.io/pyradiomics.html. [Accessed: 29-Dec-2021].

[85] T. Wang, T. Zhang, X. Han, X. Liu, N. Zhou, and Y. Liu, “Impact of the international association for the study of lung cancer/american thoracic society/European respiratory society classification of stage IA adenocarcinoma of the lung: Correlation between computed tomography images and EGFR and KRAS gene mutati,” *Exp. Ther. Med.*, vol. 9, no. 6, pp. 2095–2103, 2015.

[86] D. Mei, Y. Luo, Y. Wang, and J. Gong, “CT texture analysis of lung adenocarcinoma: can Radiomic features be surrogate biomarkers for EGFR mutation statuses,” *Cancer Imaging*, vol. 18, no. 1, p. 52, 2018.

[87] I. Shiri *et al.*, “PET/CT Radiomic Sequencer for Prediction of EGFR and KRAS Mutation Status in NSCLC Patients,” in *2018 IEEE Nuclear Science Symposium and Medical Imaging Conference Proceedings (NSS/MIC)*, 2018, pp. 1–4.

[88] Y. Liu *et al.*, “Radiomic Features Are Associated With EGFR Mutation Status in Lung Adenocarcinomas,” *Clin. Lung Cancer*, vol. 17, no. 5, pp. 441-448.e6, 2016.

[89] M. Baatz, J. Zimmermann, and C. G. Blackmore, “Automated analysis and detailed quantification of biomedical images using Definiens Cognition Network Technology.,” *Comb. Chem. High Throughput Screen.*, vol. 12, no. 9, pp. 908–916, Nov. 2009.

[90] Y. Feng *et al.*, “A Deep Learning Approach for Targeted Contrast-Enhanced Ultrasound Based Prostate Cancer Detection,” *IEEE/ACM Trans. Comput. Biol. Bioinforma.*, vol. 16, no. 6, pp. 1794–1801, Nov. 2019.

[91] S. Wang *et al.*, “Predicting EGFR mutation status in lung adenocarcinoma on computed tomography image using deep learning,” *Eur. Respir. J.*, vol. 53, no. 3, 2019.

[92] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, “Densely Connected Convolutional Networks,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2261–2269.

[93] G. Zhang *et al.*, “Nomogram based on preoperative CT imaging predicts the EGFR mutation status in lung adenocarcinoma,” *Transl. Oncol.*, vol. 14, no. 1, 2021.

[94] S. Bakr, Shaimaa; Gevaert, Olivier; Echegaray, Sebastian; Ayers, Kelsey; Zhou, Mu; Shafiq, Majid; Zheng, Hong; Zhang, Weiruo; Leung, Ann; Kadoch, Michael; Shrager, Joseph; Quon, Andrew; Rubin, Daniel; Plevritis, Sylvia; Napel, “Data for NSCLC Radiogenomics Collection,” *The Cancer Imaging Archive*, 2017. [Online]. Available: https://wiki.cancerimagingarchive.net/display/Public/NSCLC+Radiogenomics.

[95] S. Moreno *et al.*, “A Radiogenomics Ensemble to Predict EGFR and KRAS Mutations in NSCLC,” *Tomography* , vol. 7, no. 2. 2021.

[96] Y. Balagurunathan *et al.*, “Reproducibility and Prognosis of Quantitative Features Extracted from CT Images,” *Transl. Oncol.*, vol. 7, no. 1, pp. 72–87, Feb. 2014.

[97] N. V Chawla, K. W. Bowyer, and L. O. Hall, “SMOTE : Synthetic Minority Over-sampling Technique,” *J. Artif. Intell. Res.*, vol. 16, pp. 321–357, 2002.

[98] H. B. Mann and D. R. Whitney, “On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other,” *Ann. Math. Stat.*, vol. 18, no. 1, pp. 50–60, 1947.

[99] K. Kira and L. A. Rendell, “The Feature Selection Problem: Traditional Methods and a New Algorithm,” in *Proceedings of the Tenth National Conference on Artificial Intelligence*, 1992, pp. 129–134.

[100] H. K. D. H. Bhadeshia, “Neural Networks in Materials Science,” *ISIJ Int.*, vol. 39, no. 10, pp. 966–979, 1999.

[101] P. Romanski and L. Kotthoff, “Package ‘ FSelector ,’” 2018. [Online]. Available: https://cran.r-project.org/web/packages/FSelector/FSelector.pdf.

[102] L. Torgo, “Package ‘ DMwR ,’” 2015. [Online]. Available: https://cran.r-project.org/web/packages/DMwR/DMwR.pdf.

[103] K. Max *et al.*, “Package ‘ caret ,’” 2018. [Online]. Available: https://cran.r-project.org/web/packages/caret/caret.pdf.

[104] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” *Adv. Neural Inf. Process. Syst.*, vol. 2, pp. 1097–1105, 2012.

[105] R. Paul, S. H. Hawkins, L. O. Hall, D. B. Goldgof, and R. J. Gillies, “Combining deep neural

network and traditional image features to improve survival prediction accuracy for lung cancer patients from diagnostic CT," *2016 IEEE Int. Conf. Syst. Man, Cybern. SMC 2016 - Conf. Proc.*, pp. 2570–2575, 2017.

[106] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman, "Return of the Devil in the Details: Delving Deep into Convolutional Nets." 2014.

[107] P. Zhou, J. Feng, C. Ma, C. Xiong, S. Hoi, and W. E, "Towards Theoretically Understanding Why SGD Generalizes Better Than ADAM in Deep Learning." 2021.