UNIVERSIDAD DE GRANADA

TRABAJO FIN DE MASTER

INGENIERÍA DE TELECOMUNICACIÓN

# Desarrollo de una aplicación para clasificación de imágenes histológicas


**Autor**
Fernando Palomino Cobo

**Directores**
Luis Javier Herrera Maldonado
Francisco Carrillo Pérez




Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación


—

Granada, Septiembre de 2023


# Desarrollo de una aplicación para clasificación de imágenes histológicas

**Autor**
Fernando Palomino Cobo

**Directores**
Luis Javier Herrera Maldonado
Francisco Carrillo Pérez

# Desarrollo de una aplicación para clasificación de imágenes histológicas

Fernando Palomino Cobo

**Palabras clave**: Aprendizaje Automático, Aprendizaje Profundo, Imágenes Digitales Completas, Inteligencia Artificial, Patología Digital, Redes Neuronales Convolucionales, Transfer Learning, Streamlit

## Resumen

El análisis histopatológico implica la minuciosa inspección de muestras de tejido a través del microscopio, una labor que tradicionalmente recae en manos de patólogos expertos. Con la llegada de la patología digital y la creciente disponibilidad de imágenes de alta resolución de muestras completas (WSI), se abre la puerta a una mejora significativa en la precisión y eficiencia del diagnóstico mediante el uso de técnicas de Inteligencia Artificial (IA). Sin embargo, la barrera de entrada para profesionales de la salud, que en ocasiones carecen de conocimientos en programación, puede llegar a frenar la adopción de la IA en el campo de la patología. En este contexto, el presente Trabajo Fin de Máster introduce ***FPathai*** (*Friendly PATHology application for Artificial Intelligence*), una aplicación de código abierto diseñada específicamente para superar esta barrera. *FPathai* ofrece una plataforma intuitiva y accesible que no requiere conocimientos previos en programación, democratizando así el análisis de imágenes histopatológicas potenciado por IA y haciéndolo accesible a un público más amplio, desde científicos de datos hasta patólogos. Este software, al aceptar WSI en formato .svs, automatiza todo el proceso, desde la generación de secciones de la imagen hasta el entrenamiento de modelos mediante Transfer Learning, proporcionando información interpretable a través de Grad-CAM (Gradient-weighted Class Activation Mapping). De esta manera, *FPathai* se presenta como una herramienta fundamental en la evolución de la patología hacia una práctica más precisa y eficiente, al tiempo que democratiza el acceso de usuarios sin conocimientos extensos en programación a las ventajas de la IA en este campo.

# Development of an application for histological images clasification

Fernando Palomino Cobo

**Keywords**: Artificial Intelligence, Convolutional Neural Networks, Deep Learning, Digital pathology, Machine Learning, Streamlit, Transfer Learning, Whole Slide Images

## Abstract

Histopathological analysis involves the examination of tissue samples under a microscope, a task traditionally performed by pathologists. With the advent of digital pathology and the increasing availability of high-resolution whole slide images (WSI), there is an opportunity to enhance diagnostic accuracy and efficiency using AI techniques. However, the barrier to entry for healthcare professionals, who may lack coding expertise, has hindered the adoption of AI in pathology. This master's thesis introduces ***FPathai*** (Friendly PATHology application for Artificial Intelligence), an open-source application designed to bridge this gap. *FPathai* offers an intuitive platform that requires no coding skills, making AI-powered histopathological image analysis accessible to a wider audience, from data scientists to pathologistss. By accepting whole slide images (WSI) in .svs format, FPathai automates the process of patch creation, model training with Transfer Learning, and provides interpretable insights through Grad-CAM (Gradient-weighted Class Activation Mapping). Thus, *FPathai* is presented as a fundamental tool in the evolution of pathology towards a more precise diagnosis, while at the same time democratizing the access to AI techniques to users without a programming background.

Yo, **Fernando Palomino Cobo**, alumno del **Máster en Ciencia de Datos e Ingeniería de Computadores** de la Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada, con DNI 76066820R, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Master en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo: Fernando Palomino Cobo

Granada a 6 de Septiembre de 2023.

D. **Luis Javier Herrera Maldonado**, Profesor Titular del Departamento de Ingeniería de Computadores, Automática y Robótica de la Universidad de Granada.

D. **Francisco Carrillo Pérez**, Investigador Postdoctoral de la Universidad de Stanford.

**Informan:**

Que el presente trabajo, titulado ***Desarrollo de una aplicación para clasificación de imágenes histológicas***, ha sido realizado bajo su supervisión por **Fernando Palomino Cobo**, y autorizamos la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada a 6 de Septiembre de 2023.

**Los directores:**

**Luis Javier Herrera Maldonado** **Francisco Carrillo Pérez**

Yo, **Fernando Palomino Cobo**, alumno del **Máster en Ciencia de Datos e Ingeniería de Computadores** de la Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada, con DNI 76066820R, declaro explícitamente que el trabajo presentado es original, entendiendo en el sentido que no he utilziado ninguna fuente sin citarla debidamente.

Fdo: Fernando Palomino Cobo

Granada a 6 de Septiembre de 2023.

# Agradecimientos

En primer lugar, deseo expresar mi más sincero agradecimiento a los tutores que he tenido la suerte de conocer en este trabajo. Desde el primer día, me han recibido de manera estupenda. Simplemente, espero que mi desempeño haya estado a la altura de su excelencia como orientadores.

Quiero tener la oportunidad también de agradecer a todos esos amigos que me acompañan y me impulsan a crecer, haciendo que me esfuerce en ser un 1% mejor cada día. Ya sabéis quiénes sois. Os llevo en el corazón, aunque estéis dispersos por Graná, Madrid, Valencia, Girona, Barcelona, Bilbo o incluso Japón. Me siento muy afortunado de teneros.

Gracias también a mi familia. Con vosotros estaré eternamente agradecido. Sois los que me hacéis ser más fuerte cuando todo se complica. Sin vosotros no sería quien soy, me lo habéis dado todo y me lo seguís dando día tras día. Os quiero.

Por último, te quiero agradecer a ti, que estás luchando diariamente por ser una mejor versión de ti mismo. Eso es admirable y espero que nunca vuelvas a dudar de tu propio valor. Viaje antes que destino, siempre. Mantén una mente estoica y recuerda que las personas verdaderamente fuertes son amables. Las personas buenas se merecen que le pasen cosas buenas, pero eso ya lo sabes.

BIZIRIK!

# Contents

# List of Figures

# Chapter 1

# Introduction

## 1.1 Digital pathology

**Histology** is the study of the microscopic composition of tissues and organs, aiming to understand the microanatomy of cells, tissues, and organs, and establishing connections between their structure and functions. [1]. **Pathology** focuses on studying the physical and histological changes in the tissue to comprehend the disease process.[2]

During the process of histology imaging obtation, the use of the Hematoxylin and Eosin (H&E) stain stands as the gold standard in histology, providing pathologists with essential tissue insights. This staining method, which has been in use for over a century, remains indispensable for identifying various tissue types and observing the morphological changes that underlie contemporary **cancer diagnosis**. The stain comprises two components: hematoxylin and eosin, each imparting distinct colors and highlighting different cellular components:

1. Hematoxylin binds with basophilic structures (like DNA and RNA) and it stains **nucleic acids within cells** with a deep blue-purple hue.

2. Eosin, binds to acidophilic substances such as positively charged amino acid side chains, like **proteins within cells** with an pink/orange tinting.

In a standard tissue or cell section stained with H&E, the cytoplasm and extracellular matrix exhibit diverse shades of pink coloring attributed to eosin. This sharp demarcation between the blue-stained nuclei and the rosy-tinted cytoplasm effectively delineates distinct cell structures, facilitating pathologists in the recognition of cellular types and anomalies, which makes it very useful for diagnosis purposes.[3]. An example of a tissue stained with H&E can be observed in Figure 1.1.

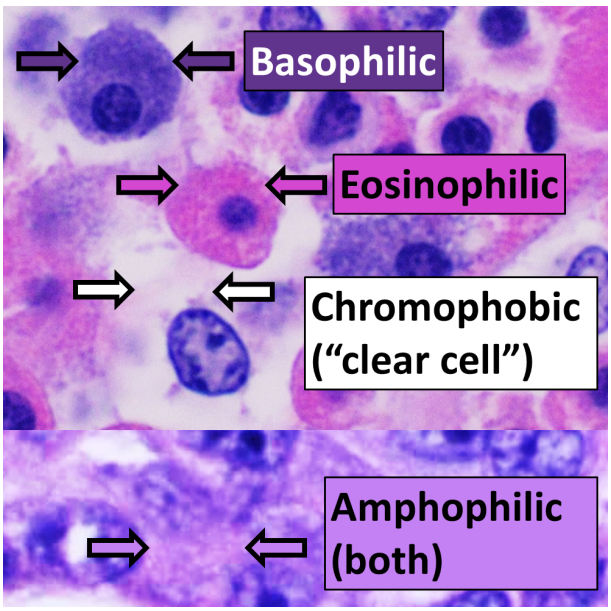


Figure 1.1: Main types of staining seen on H&E stain. The slide corresponds to the concentration of how strongly the stain dyes the tissue [4]

Currently, the examination of histopathological tissue samples relies on pathologists, who meticulously review vast gigapixel-sized images of tissue sections. These specialized medical professionals are responsible for diagnosing and classifying diseases like cancer and inflammatory conditions, utilizing various tissue characteristics as indicators (such as disruptions in tissue structure, the presence or absence of specific cell traits, and the abundance of inflammatory cells).[1]

However, the field of pathology is facing challenges due to a shortage of pathologists and an increasing workload stemming from a higher volume of cases. Additionally, there is a growing need for more comprehensive diagnoses to determine the most effective treatment strategies for patients. This is why the examination of histopathological tissue samples is an area that holds significant promise for the application of Artificial Intelligence (AI). [5]

## 1.2 Artificial Intelligence in digital pathology

The forthcoming decade will witness one of the most significant revolutions in the field of medicine through the incorporation of AI. This transformation is poised to have a profound impact on histopathology, positioning it at the epicenter of this sweeping change. [6] The confluence of enhanced

computing capabilities, swifter network speeds, and more affordable storage solutions has substantially simplified the management of digital slide images for pathologists. [7] This marks a significant departure from the challenges faced in the preceding decade, facilitating seamless image sharing for telepathology and clinical applications. Over the past twenty years, digital imaging in pathology has borne witness to the genesis and progression of whole slide imaging (WSI).

WSI is a digital pathology technique used to capture and digitize entire microscope slides at high resolutions. With WSI, the glass slides are scanned using specialized digital scanners, which create extremely detailed and high-resolution digital images of the entire slide. These digital images can then be viewed, manipulated, and analyzed on computer screens using specialized software. An example of a WSI is 1.2. This technology has the potential to supplant the conventional microscopic approach [6] since it enables pathologists, researchers, and medical professionals to access and assess tissue samples remotely, collaborate more easily, and apply various digital analysis techniques. Notably, WSI stands as a pivotal platform that underpins the integration of AI into the domain of digital pathology. [8]

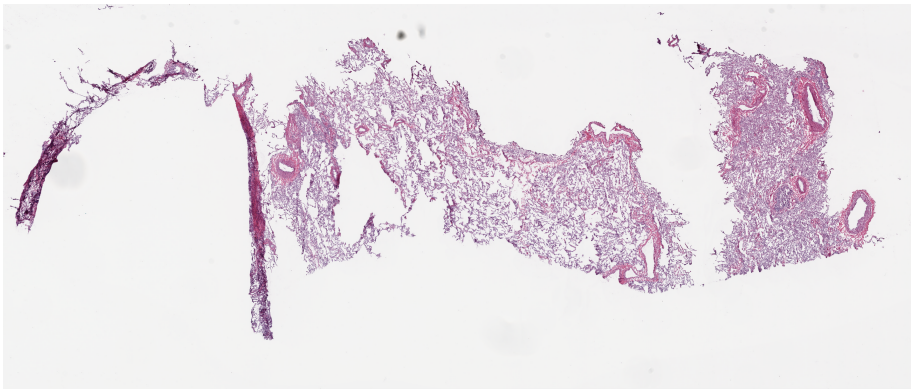

Figure 1.2: Example of a WSI image, obtained from [9]

The images generated through WSI offer a trove of intricate information – their complexity surpasses that of numerous other imaging modalities. This complexity arises from factors such as their substantial dimensions (where sizes of 100,000 x 100,000 are not uncommon), the incorporation of color details (including H&E staining), the absence of inherent anatomical orientation akin to radiology, and the availability of data across multiple scales (e.g., 4x, 20x) and z-stack levels. [10]

The integration of these digital slides into pathology workflows incorporates advanced algorithms and AI-based diagnostic techniques, expanding pathologists' capabilities beyond traditional microscopic analysis. This shift allows harnessing vast knowledge surpassing human capacity, aiding in early

detection, prognosis, and treatment selection. This transformation broadens patient reach without compromising accuracy. [11] Digital pathology, notably using WSI and H&E staining, is now standard. However, manual analysis is time-consuming and lacks scalability. Advanced deep learning in computer vision and accessible datasets have spurred the development of clinical decision support systems, part of precision medicine, offering personalized treatment insights through machine learning and data analysis. [12]

## 1.3 Analysis of the state of the art libraries

As histopathological image analysis gains significant traction in modern scientific circles, a surge of enthusiasm drives the creation of specialized libraries and tools.

Among these libraries, a subset is dedicated to a crucial step: the preprocessing. Noteworthy among them are *Histolab*, *pyHIST*, *deep-histopath*, *compay-syntax*, *py-wsi*, and *wholeslidedata* [13]. These libraries are purposefully designed to proficiently handle WSIs, autonomously identify tissue, extract meaningful tiles. Above these, *QuPath* [14] stands out. QuPath is the leading WSI image analysis software due to its free, open-source nature and compatibility with various image formats. It provides a comprehensive set of tools for tumor identification and high-throughput biomarker assessment. Nevertheless, there is a lack of integration with a mechanism to facilitate deep learning models utilizing the generated images within the libraries.

Other libraries also focuses on deep learning, particularly notable examples include *SlideFlow* [15] and *PathML* [16].

Probably, the most complete one is *SlideFlow*. *Slideflow* stands as a Python package that furnishes a collection of deep learning tools. These tools are implemented using both PyTorch and TensorFlow backends. The toolkit covers various aspects such as patch extraction from WSI, model training and explainability (see 1, 2 and 3 in 1.3 respectively)

Also, it has a user-friendly graphical interface (SlideStudio) to interactive visualize WSI, facilitating real-time generation of predictions and heatmaps. Nonetheless, the installation process poses challenges, as the library is not functional on Windows systems.

In the same fashion, *PathML* is able to accelerate research, reduce barriers to entry for new researchers, and promote open science in computational pathology since it provides pre-built tools, public datasets, and documentation to allow anyone with beginner proficiency in Python to get started. As SlideFlow, it has modules for preprocessing, modelling and explainability (see WSI Preprocessing Pipeline Toolkit, ML Model Design & Evaluation and ML Model Benchmarking in 1.4 respectively).

One significant drawback of these libraries is their reliance on coding

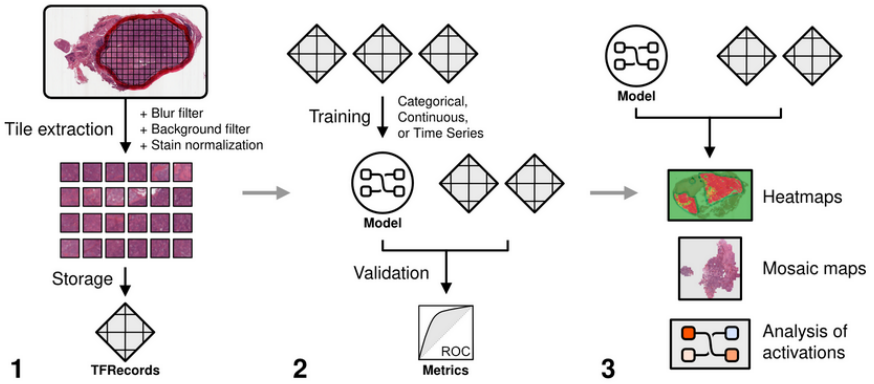


Figure 1.3: SlideFlow Overview [17]

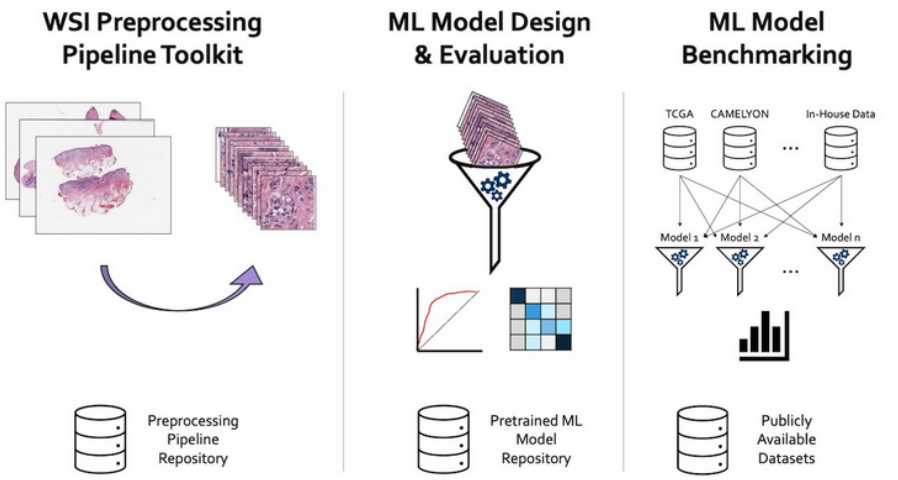


Figure 1.4: PathML overview [18]

expertise. To fully leverage the capabilities of SlideFlow and PathML, users are required to possess a solid understanding of programming languages, data structures, algorithms, and the package itself. This creates a steep learning curve, limiting the accessibility of these tools to individuals with coding backgrounds, such as bioinformaticians, computer scientists, and experienced researchers. The requirement for coding expertise acts as a barrier to entry, potentially excluding domain experts such as pathologists, medical practitioners, and researchers with limited programming experience. This limitation restricts the democratization of image analysis in these fields.

## 1.4 Objectives

Standing on shoulders of these giants libraries, in this work we have developed a functional Streamlit application named ***FPathai***[19], an application tailored for the world of histopathological image analysis that breaks down barriers to entry in the world of machine learning. The logo is shown in the Figure 1.5. While the application proposed in this study might not attain the same degree of complexity and advanced functionalities found in libraries like SlideFlow and PathML, its principal advantage lies in enabling individuals without technical backgrounds to engage in image analysis tasks without requiring coding proficiency. This strategy expands the user demographic, empowers domain experts, and fosters collaborative efforts across disciplines.

Keeping this perspective in mind, the objectives are as follows:

- The core aim of this research is to construct a user-friendly tool that bridges the gap between artificial intelligence and pathology experts, eradicating the necessity for coding on their part.
- Provide a foundational tool for aspiring students seeking to grasp the domain of deep learning as it pertains to histology.
- Attain the capability to generate patches of variable sizes from a collection of WSIs, store them in an optimized format, and establish an efficient method for subsequent access.
- Train a binary classifier with the capability to differentiate between images of cancerous tissue and healthy tissue.
- Train a classifier that is able to distinguish images of two different types of cancers and healthy tissue.


Figure 1.5: *FPathai* logo

# Chapter 2

# Foundations

This chapter uncovers the core foundations of the different algorithms used in this study. The algorithms harnessed for this research are: Supervised Learning, Artificial Neural Networks (ANN), Convolutional Neural Networks (CNN), and Transfer Learning.

## 2.1 Supervised Learning

Machine Learning (ML) is a subset of AI that focuses on developing algorithms and statistical models that enable computer systems to improve their performance on a specific task through experience, without being explicitly programmed.[20] In the context of ML, four distinct types of learning can be delineated: supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning.[21]

In a **supervised learning scenario,** data is provided with labels, and the objective is to gain insights from this data to make predictions for upcoming samples. Supervised learning particularly highlight the notions of classification and regression. **Classification** is the task of assigning a label to an input data point based on its features. The goal is to predict a discrete class from a predefined set of classes. **Regression**, on the other hand, is the task of predicting a continuous numeric value based on input features. The goal is to find a function that maps input variables to a continuous output.

In this work, a classification problem in supervised learning is faced. A supervised learning scenario consists of a dataset denoted as $X$, with dimensions $NxM$, where $N$ represents the dataset's size, and $M$ signifies the number of features present in each individual data sample. Alongside this dataset, there are associated class labels denoted as Y, with each data point possessing a corresponding label.

$$X = \begin{bmatrix} x_{1,1} & x_{1,2} & \dots & x_{1,m} \\ x_{2,1} & x_{2,2} & \dots & x_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n,1} & x_{n,2} & \dots & x_{n,m} \end{bmatrix} \tag{2.1}$$

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \tag{2.2}$$

Hence, the objective is to discover the function $f(X)$ that characterizes the behavior of the data. For classification tasks, this function ideally distinguishes between classes in an optimal manner. Nonetheless, given the inherent complexity of this function,the objective is to identify an approximate function $\hat{f}(X)$ such that $\hat{f}(X)$ closely approximates $f(X)$.

In this pursuit, when working with image data, deep neural networks (DNN) have demonstrated their exceptional potency and efficacy, positioning them as the optimal choice in numerous scenarios.

## 2.2 Deep Neural Networks

In this section, a brief trace of the evolution of deep-learning from its inception to its current prominence will be presented. Following this historical overview, we will delve into the essential concepts crucial for comprehending this work. [22]

In 1958, Rosenblatt introduced the first-generation neural network single-layer perceptron [23]. This perceptron was capable of discerning basic shapes like triangles and squares. Let's define how a neuron works in a neural network. It consists of:

- **inputs** $(a_1, a_2, a_3, ...a_n)$
- **weights** $(w_1, w_2, w_3, ...w_n)$
- **bias** $(b)$ (a constant value for modelling the inputs and weights) and
- an **activation function** $(\sigma)$ that is applied to the sum of $b$ with the product of each corresponding input.

There are graphically shown in in 2.1.

A clear limitation of a single neuron is that it represent a linear function, thereby impeding the incorporation of diverse input interactions. Bearing this in mind, in 1969, Minsky underscored the limitations of a singular-layer perceptron by showcasing its incapability to solve complex problems

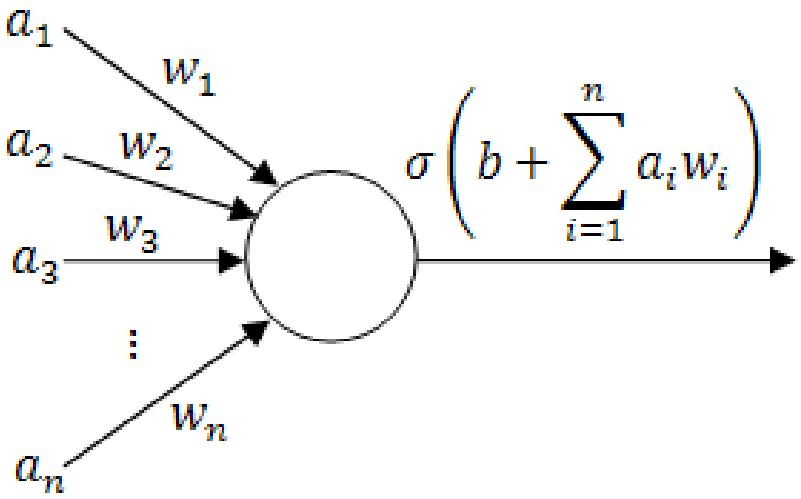


Figure 2.1: Mathematical expression of a single neuron in a neural network [24]

like the XOR operation. This deficiency was compounded by an inadequate learning mechanism [25]. A pivotal turning point occurred in 1986 when the second-generation neural network, conceived by Hinton and colleagues [26], emerged. This network, named *multi-layer perceptron* (MLP) featured the revolutionary backpropagation training method. This breakthrough is a set of single neurons located in layers and marked the onset of the second wave of neural networks[1]. The multi-layer perceptron, that can be seen in 2.2 is formed by:

- **Input layer**: Layer that receive the input data
- **Hidden layer**: Layer that process the information from the previous layers and connect with the next layer.
- **Output layer**: Layer that determines the output of the network

[1]Typically, in discussions about neural networks, we commonly mention fully connected networks, where every neuron in a given layer is linked to all the neurons in the preceding layer.

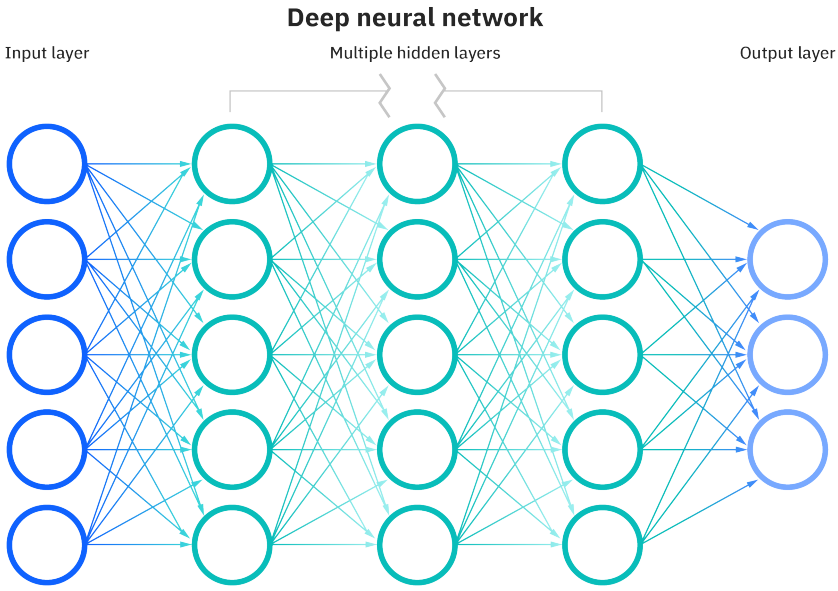


Figure 2.2: Deep Neural Network diagram [27]

When examining the schematic 2.2, one common query that may surface is determining the suitable quantity of neurons and layers to employ.

- Relating to the **number of layers**, adding more layers to a neural network increases its complexity and enables it to learn abstract and intricate data representations, improving model accuracy. However, excessive layers can lead to overfitting, causing the model to struggle with new data. Conversely, reducing layers simplifies the model but may result in decreased accuracy, especially with complex data.

- Relating to the **number of neuron per layer**, happens something similar to the number of layers. Increasing the number of neurons in each layer enhances a neural network's ability to detect patterns and fit training data closely. However, excessive neuron count makes the model more prone to overfitting, causing poor performance on new data. Conversely, reducing neuron count makes the model less complex but may result in underfitting, where it fails to capture essential data patterns due to its simplicity. [20]

**Overfitting** occurs when a model becomes excessively specialized in learning the training data, to the extent that it struggles to generalize effectively to unseen or new data. To prevent it, one of the widely embraced and effective techniques is dropout. **Dropout** involves randomly setting a

percentage of neurons in the network to a 0 output during training. Consequently, in each training batch, a varying number of neurons are temporarily deactivated. This deliberate randomness disrupts the co-adaptation of features, encouraging the network to discover new relationships among neurons. This regularization effect serves to hinder the network from memorizing specific outcomes.[28]

There is no clear answer as to how many neurons and layers to choose since it will depend on the problem. In any case, this network configuration is referred to as a **feed-forward network**, owing to its absence of loops, ensuring a unidirectional flow of information at all times. With the backpropagation method, the weights are updated as follows:

$$w^{k+1} = w^k + \Delta w \tag{2.3}$$

with

$$\Delta w = -\eta \cdot \frac{\partial E}{\partial w} \tag{2.4}$$

$\eta$ represents the learning rate, and $\frac{\partial E}{\partial w}$ denotes the gradient of the errors with respect to the weights. The **earning rate** is a crucial hyperparameter in ML optimization. It dictates the step size taken by the algorithm to minimize the loss function during model training. Choosing a high rate can lead to fast convergence but risk overshooting, while a smaller rate might yield a more accurate result but at a slower pace. The **gradient** quantifies how a function changes concerning the variable being derived. The negative sign is employed because it indicates the direction in which the error seeks to be minimized. This means that the **gradient descent** is the optimization problem that neural network training aims to solve. It iteratively adjusts the network's parameters to minimize a loss function[20], which measures how well the network performs its task. Through the backpropagation algorithm, gradients are computed and used to update the parameters, gradually improving the network's performance until convergence is reached. This process enables neural networks to learn. An illustrative plot can be seen in 2.3.

Various **optimizers** are available for updating a model's parameters during training with the goal of minimizing a loss function. Selecting an optimizer changes the optimization approach of the neural network. Notable optimizers include Stochastic Gradient Descent (SGD), Adaptive Gradient Algorithm (Adagrad), Root Mean Square Propagation (RMSProp), and Adaptive Moment Estimation (Adam) each employing its unique strategy.

- **SGD**: In situations involving a substantial volume of training data, the calculation of gradients for each individual data point within the dataset becomes computationally impractical. To tackle this issue, SGD is employed, where model parameters are iteratively adjusted

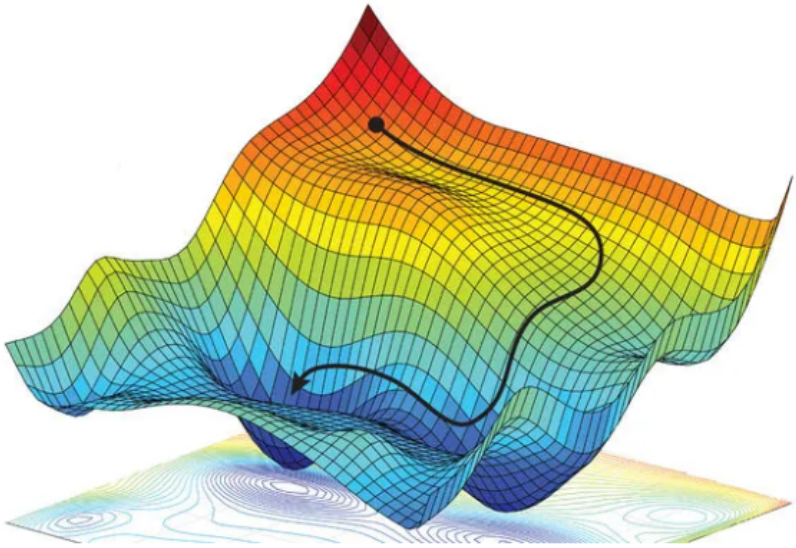

Figure 2.3: Gradient Descent scheme [29]

using a randomly selected mini-batch of training data during each iteration. This infusion of stochasticity into the parameter updates introduces variability into the optimization process, making it noisier but often accelerating convergence. Because of this, SGD can lead to substantial fluctuations in the objective function, potentially prolonging the optimization process and causing convergence difficulties, including getting trapped in saddle points instead of local minima. One effective approach to mitigate these challenges is by incorporating momentum. At a conceptual level, momentum imparts inertia to the search direction, enabling the algorithm to overcome local minima and reduce oscillations in the optimization path.[30]

- **Adagrad** Determining an appropriate learning rate can be a complex task, which is why the **AdaGrad** optimizer is employed as a stochastic optimization technique designed to dynamically adjust the learning rate based on the model's parameters.[31]

- **RMSProp** distinguishes itself from Adagrad in its approach to handling historical gradient information for learning rate adaptation. By employing a moving average mechanism with a decay factor, RMSprop effectively mitigates the rapid decrease of the learning rate that can Adagrad suffer. [20]

- **Adam** can be regarded as a hybrid of RMSprop and Stochastic Gradient Descent (SGD) with momentum. It leverages the concept of scal-

ing the learning rate based on squared gradients, akin to RMSprop. Simultaneously, it incorporates momentum by employing a moving average of gradients, rather than the raw gradients themselves, as seen in SGD with momentum. [32]

As previously explained, rather than conducting training on the entire dataset simultaneously, it is customary to divide the dataset into smaller segments, commonly referred to as **batches**. Batch size refers to the number of data samples processed in a single forward/backward pass during model training. Smaller batch sizes lead to more frequent weight updates, accelerating convergence but increasing model variability and the risk of overfitting. Larger batch sizes reduce update frequency, fostering gradual convergence with lower variance and potential for better generalization. However, they demand more memory and resources, and if too large, may hinder escape from suboptimal solutions. [20] Upon the completion of loading all batch images, the system can be re-initialized by resetting the parameters. It's worth noting that each time a full load of data is accomplished, we refer to it as the completion of an **epoch**.

Having clarified the process of parameter updating, it becomes essential to briefly delve into the topic of activation functions. The activation function would determine if the neuron fires or remain inactive. While various options exist, our focus in this study centers on a comparison between the traditional **sigmoid** function and one of great significance in the history of neural networks: **the Rectified Linear Unit (ReLU)**.

The sigmoid activation function presents a noteworthy limitation: as values approach -1 or 1 (see 2.4), its slope diminishes to zero, resulting in a gradient of 0, which hinders the learning process during backpropagation. In 2011, the introduction of the Rectified Linear Unit (ReLU) activation function by Glorot et al. [33] proved to be a pivotal solution for mitigating the previously mentioned vanishing gradient issue. Additionally, ReLU offers the benefits of improved computational efficiency (since there is no need to calculate an exponential function) and superior convergence performance. These advantages firmly establish ReLU as the preferred activation function in the realm of deep learning.

As mentioned, the distribution of output values within the activation units plays a crucial role in neural network training. **Batch normalization** [35] aims to regularize this distribution by encouraging the input of activation units across all layers to adhere to a standard normal distribution and must be defined since it is normally used in practise. This standardization involves subtracting the batch mean and dividing by the batch standard deviation. Additionally, Batch Normalization promotes individual learning for each layer within a network, fostering a degree of independence among layers during training.

While the elements explained thus far can yield satisfactory results at

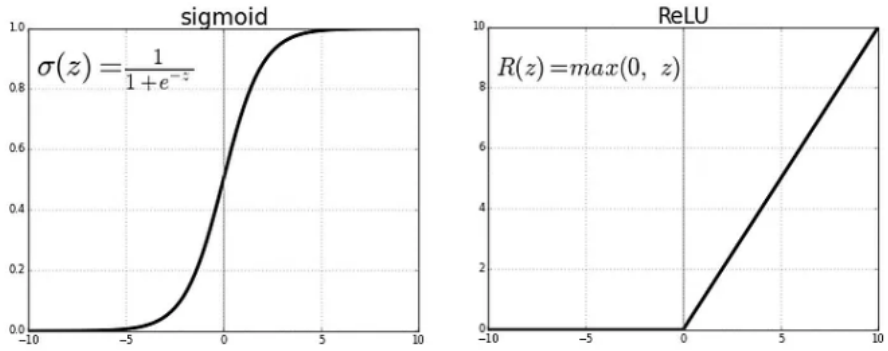


Figure 2.4: Sigmoid and ReLU activation functions [34]

the neuron output level, it's essential to acknowledge that when applied to the field of computer vision, particularly with high-resolution images, this connectivity leads to a significant proliferation of parameters. For example, for a color image of size 224x224 pixels, a fully connected layer would have millions of weights. To address these issues, Convolutional Neural Networks (CNNs) were developed. [36]

## 2.3 Convolutional Neural Networks

The pivotal moment in CNN history occurred in 1989 when LeCun et al. [37] introduced this groundbreaking network, initially applied to digital handwriting recognition. However, the true watershed moment transpired in 2012 during the ImageNet image recognition competition when Hinton and his team unveiled AlexNet[38]. It vastly outperformed the second-place Support Vector Machine (SVM) method, marking a monumental turning point in the field [39]. What distinguishes CNNs is their innate ability to automatically extract features from data. This is done by incorporating convolutional layers.

A **convolutional layer** relies on convolutional operations applied across the entire spatial domain of the data, whether it's an image or a signal. The fundamental aim of these operations is to capture inherent data patterns, transforming the input into an alternative feature representation. These filters possess the same depth as the input image and are applied iteratively along both the first and second dimensions. An illustrative example can be observed in Figure 2.5. A common approach involves employing multiple filters to extract a wider array of features from each position within the image. The concept behind this strategy is that each filter is designed to capture specific image attributes that collectively contribute to creating a rich and expressive representation of the image.

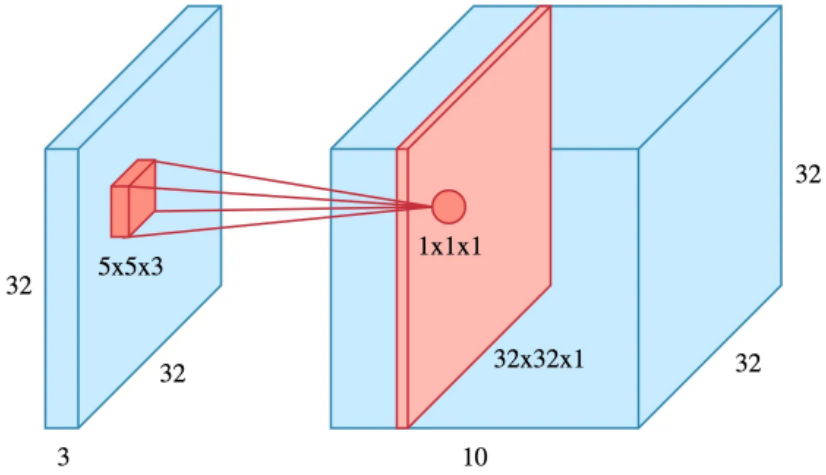


Figure 2.5: Example of application of a convolutional filter[40]

To enhance the manageability of representations by reducing their size and computational complexity, a solution arises in the form of **pooling layers**. These layers efficiently diminish the number of required parameters. Pooling layers achieve this by condensing the data dimensions, consolidating the outputs from clusters of neurons in one layer into a single neuron in the subsequent layer. This condensation process reduces the information carried by the convolution filters. An example of a max pooling operation, where the maximum value from a 2x2 filter is selected, is illustrated in Figure 2.6.

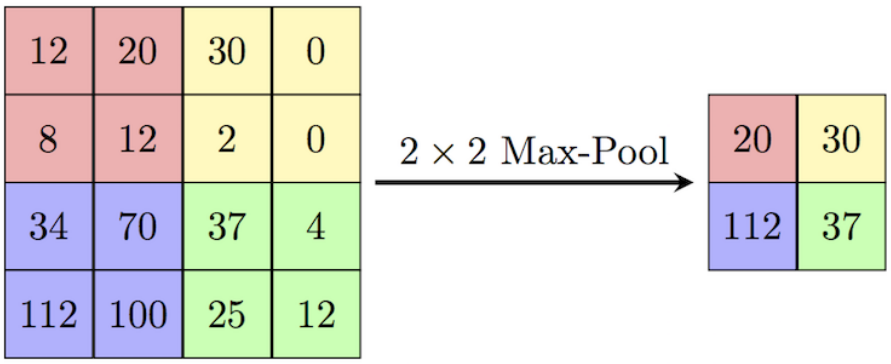


Figure 2.6: Example of application of MaxPooling operation[41]

To derive the ultimate representations of a CNN following the convolutional and pooling stages, we introduce **fully connected layers**. Fully

connected layers establish connections between all neurons in one layer and all neurons in another layer, essentially replicating the structure of a conventional multi-layer perceptron. These layers utilize the features acquired from the convolutional layers to facilitate the classification process. In the context of a classification problem within a CNN, the final layer typically takes the form of a fully connected layer, as it necessitates an output neuron for each class. [20]

Despite their presented remarkable capabilities, training CNNs from scratch demands significant computational resources and extensive datasets. This is where the concept of Transfer Learning must be defined.

## 2.4 Transfer Learning

Transfer learning typically denotes a procedure in which a pre-trained model that has been trained on a specific task is leveraged in some capacity for a different but related task.

In the context of deep learning, transfer learning constitutes a methodology wherein a neural network model is initially trained on a problem analogous to the one under consideration. Later, specific layers from this pre-trained model are incorporated into a novel model, which is being developed to tackle the targeted issue. The advantage lies in leveraging the learned weights of this architecture as a starting point, resulting in a notably accelerated learning process. [42]

This influential paradigm involves the initial pretraining of a model using an extensive dataset, followed by fine-tuning on a smaller, domain-specific dataset tailored to a particular task. In this regard, the invaluable resource of ImageNet comes into play to fulfill this requirement effectively. [43] The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) has become the standard benchmark for large-scale object recognition. In total, the dataset contains 14,197,122 annotated images. Each image is meticulously annotated, contributing to a comprehensive resource that aids in training and evaluating image classification algorithms. An example of the images in ImageNet is represented in the Figure 2.7.

AlexNet, as mentioned, marked a historic milestone as the inaugural deep learning model to secure victory in this competition[38]. Subsequently, the field has witnessed the emergence of progressively sophisticated CNNs. There exist some effective models designed for image recognition that are available for download, serving as foundational tools for tasks associated with image recognition and computer vision. Among these, three particularly renowned models are:

- VGG (such as VGG16 or VGG19) [44]
- GoogLeNet (for instance, InceptionV3) [45]
- Residual Network (like ResNet50) [46]
- MobileNet (like MobileNetV2 or MobileNetV3 [47])

These models find widespread application in transfer learning due to their remarkable performance and their role as pioneering examples that introduced distinctive architectural advancements. Specifically, these advancements encompass consistent and iterative structures in the case of VGG, inception modules in GoogLeNet, residual modules within ResNet and computacional limitations in the case of MobileNet. [20]

Figure 2.7: Example of the ImageNet dataset [48]

### 2.4.1 VGG16

VGGNet, an architectural sibling to AlexNet, is distinguished by its greater depth and the employment of compact 2x2 constant-sized filters. Two prominent variations of this architecture exist: VGG16 and VGG19, named for their respective layer counts of 16 and 19. [44] A comparison can be seen in Figure 2.8 where it is noticeable the augment of neurons and layer of the VGG architectures.

This study primarily centers around the VGG16 model due to its lower parameter count and faster training and deployment capabilities. The whole structure of the VGG16 can be seen in Figure 2.9

### 2.4.2 MobileNetV2

MobileNet is a family of lightweight CNN architectures specifically designed for **efficient deployment** on mobile and embedded devices. These networks aim to achieve a good balance between accuracy and computational efficiency, making them suitable for applications with limited computing resources. MobileNets use depth-wise separable convolutions, which split the convolution operation into a depth-wise convolution followed by a point-wise convolution. This reduces the number of parameters and computations, leading to faster inference on mobile devices. [47]. The architecture of MobileNetV2 can be seen in Figure 2.10.

### 2.4.3 ResNet50

ResNet represents a groundbreaking architectural innovation by introducing the concept of **residual learning**. In conventional deep networks, the challenge of the vanishing gradient problem hampers the effective training of exceedingly deep models. ResNet tackles this issue by incorporating "residual blocks" equipped with skip connections, facilitating the uninterrupted flow of information across network layers. Basically, a skip connections are direct connections that skips over some layers of the model and serve as a robust defense against the vanishing gradient problem, enhancing the overall stability of the training process.[46] An example of the ResNet50 is in the Figure 2.11.

### 2.4.4 InceptionV3

In images, significant features can exhibit varying sizes, making the selection of filters complex. Larger filters have the capacity to encompass globally distributed information, whereas smaller ones excel at capturing local intricacies. However, incorporating an excessive number of layers can introduce the risk of overfitting. The Inception approach mitigates this concern by em-

ploying filters of diverse sizes on the same input, resulting in wider networks as opposed to exceptionally deep ones. [45]

InceptionV3 (see Figure 2.12) stands as a CNN architecture that places a strong emphasis on multi-scale feature extraction through the utilization of "Inception modules." These modules harness filters of varying sizes (1x1, 3x3, 5x5) and pooling operations in parallel to effectively capture features across different scales. InceptionV3 was meticulously crafted to enhance both accuracy and computational efficiency when compared to earlier iterations of the Inception architecture. It leverages techniques such as factorized convolutions and batch normalization to elevate training efficiency and generalization capabilities.

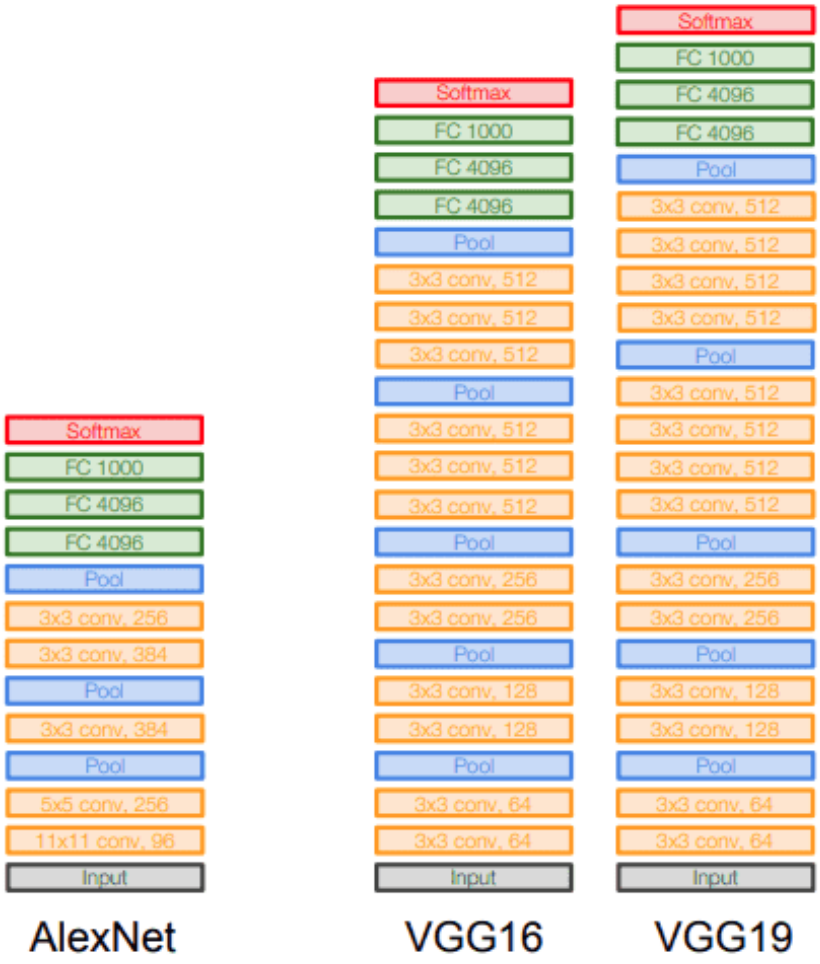


Figure 2.8: Comparison between AlexNet, VGG16 and VGG19 [49]

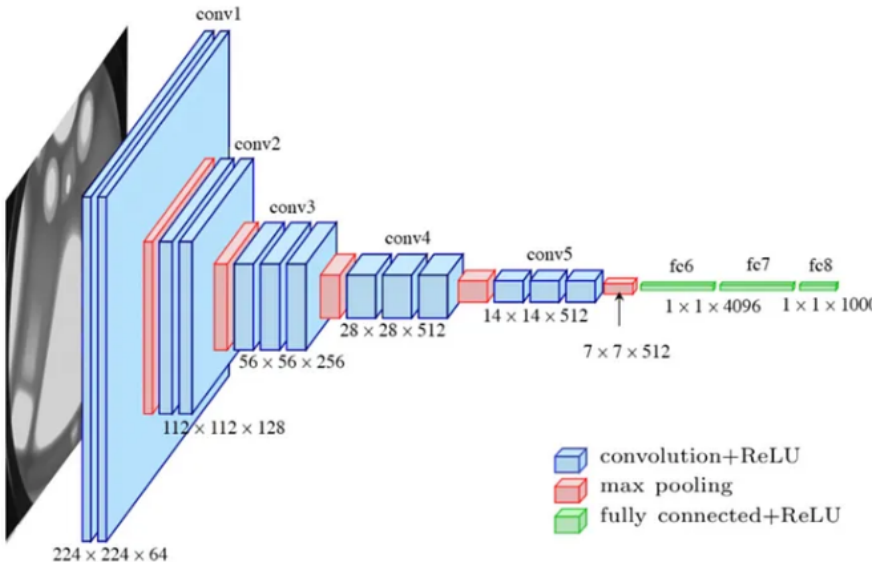


Figure 2.9: Structure of the VGG16 [50]

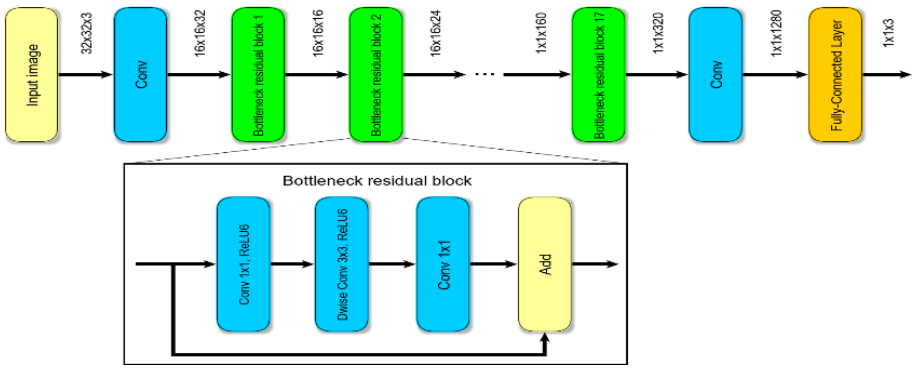


Figure 2.10: Structure of the MobileNetV2 [51]

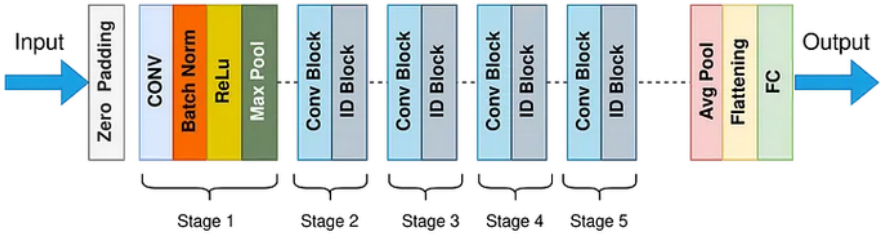


Figure 2.11: Structure of the ResNet50 [52]

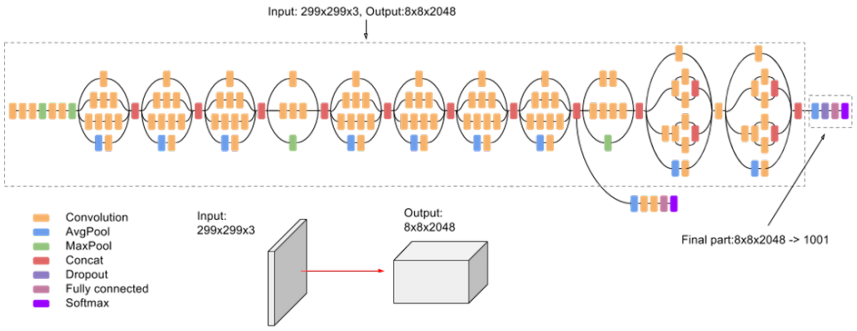


Figure 2.12: Structure of the InceptionV3 [53]

# Chapter 3

# Materials and Methods

## 3.1 GDC Data Portal NIH

The Cancer Genome Atlas, also known as TCGA, is a project initiated in 2005 and overseen by the National Cancer Institute (NCI) and the National Human Genome Research Institute (NHGRI), whose purpose is to catalog biologically important molecular changes responsible for the development of cancer using genomic sequencing and bioinformatics.[54]

Genomic Data Commons (GDC) is a data platform developed by the NCI to house and distribute the vast amounts of genomic data generated by TCGA and other related projects. GDC serves as a central repository for sharing and accessing genomic data, including DNA sequencing data, RNA expression data, and clinical information from various cancer studies. It comprises an extensive repository of more than 20,000 digital slide images encompassing 28 different tumor types, coupled with comprehensive clinical, genomic, and radiomic data. [55]. GDC provides researchers with a standardized and user-friendly interface to query, visualize, and analyze the collected data. The Figure 3.1 displays various types of cancer cases within GDC.

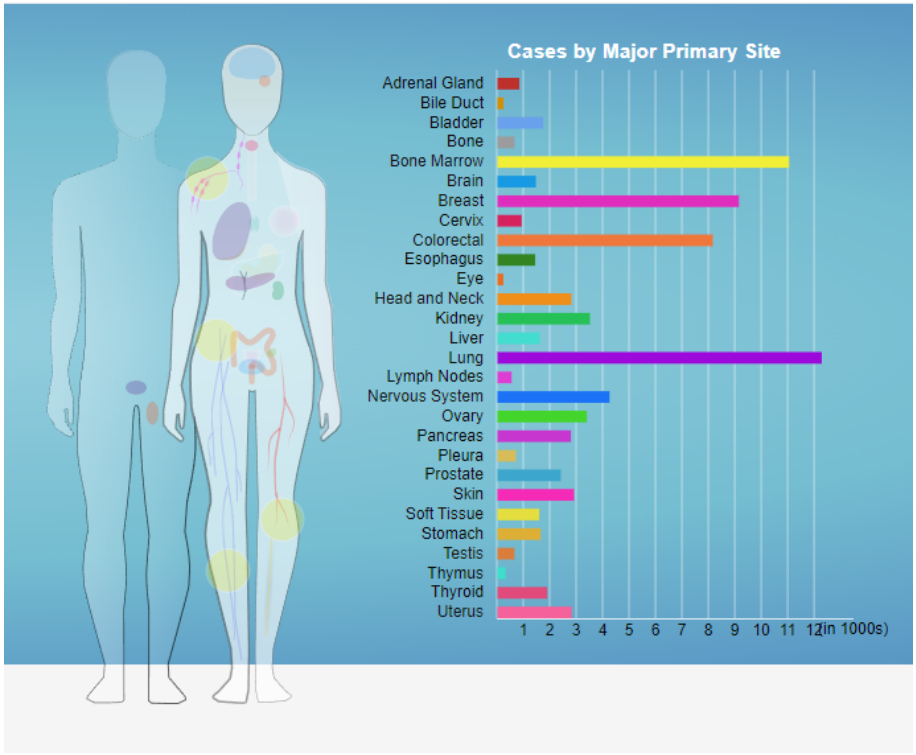


Figure 3.1: Cancer cases by Major Primary Site[9]

The tissues obtained and used in the GDC Data Portal typically come from cancer patients. These tissues are collected through medical procedures such as biopsies or surgeries. Researchers collect samples of tumor tissue as well as solid tissue normal adjacent to the tumor from the same patient, when available. These samples are then processed, sequenced, and analyzed to understand the genetic and molecular characteristics of the tumors and the underlying genetic factors contributing to cancer development and progression. [56]

The cancer histopathology images were obtained in the SVS format from the GDC Data Portal NIH. The SVS format allows us to work with high-resolution WSI. To obtain SVS images from the GDC platform, we followed these steps:

1. **Select the Cancer Type**: We began by choosing the specific type of cancer to be studied. This can be accomplished by selecting one of the bars in Figure 3.1 displayed on the platform. In our case we selected for one experiment **lung** and **breast** for the other one.

2. **Choose "TCGA Project"**: After we selected the cancer type, we proceed to opt for the "TCGA project."

3. **Save the Selection**: Following the selection of cancer type and TCGA project, we saved the chosen criteria. This step is essential to ensure that the criteria are retained for later reference.

4. **Access the Saved Set**: Upon saving the selection, we accessed the created set. We selected "Slide Image" as Data Type. This is critical as it signifies the intent to retrieve images in SVS format, commonly used for pathology slides.

5. **Add All Files to Cart**: To commence the download process, we clicked on "Add All Files to Cart." This action will populate our download cart with all SVS image files corresponding to your selected criteria.

6. **Utilize Data Transfer Tool (optional but highly recommended)**: Given that SVS image files can be substantial in size, we employed the Data Transfer Tool for the downloading process. This tool is specifically designed to facilitate efficient transfer of large datasets.

Breast and lung cancers have been chosen as research subjects due to the widespread utilization of imaging detection methods in these instances. Furthermore, the availability of extensive datasets within the GDC repository provides a wealth of diverse data sources for conducting experiments. From the documentation of the GDC documentation of the GDC, it is known that tumor types range from 01 - 09, normal types from 10 - 19 and control samples from 20 - 29. In this work, the terms "control" and "normal" will be used to refer to the same concept, a control tissue adjacent to the tumor from the subject. Therefore, images meeting the criteria with the following codes will be chosen:

- **01A**: The tissue is of tumor type, i.e., the class of that data is Primary Tumor.

- **11A**: The tissue of the specimen is control tissue, i.e. the class of that data is Solid Tissue Normal.

### 3.1.1 Breast data

Breast cancer ranks as the most prevalent form of cancer among American women, second only to skin cancer. Annually, there are approximately 230,000 new breast cancer diagnoses and tragically, it claims the lives of 40,000 individuals [57]. BRCA Breast invasive carcinoma, also known as invasive breast cancer, is a type of cancer that begins in the cells of the breast and has the potential to invade surrounding tissues and spread to other parts of the body. It is the most common form of breast cancer.[58]

In this scenario, all available images from GDC have been systematically downloaded. Following this, 100 random images from each category, distinguished by their respective codes for tumor and control, were randomly selected. These selected images have been organized into two separate folders, labeled "tumor" and "control," each containing precisely 100 images, identified by the codes 01A and 11A, respectively.

### 3.1.2 Lung data

In the United States, approximately 220,000 cases of lung cancer are diagnosed annually, surpassing the combined death toll of the next three most prevalent cancer types—colon, breast, and pancreatic cancer. At any given moment, nearly 400,000 Americans live under the pervasive threat of lung cancer [57]. Adenocarcinoma (LUAD) and squamous cell carcinoma (LUSC) are the most prevalent subtypes of lung cancer, and their distinction requires visual inspection by an experienced pathologist. [59] LUSC typically presents with a less favorable prognosis, often appearing as a tumor located in the upper part of the bronchial tree, and is closely linked to tobacco smoking. In contrast, LUAD tends to be found in peripheral areas of the lungs and is more commonly seen in individuals who do not smoke. [60]

In this particular scenario, we first acquired all available images. Subsequently, a preprocessing procedure was executed to distinguish between images depicting healthy tissue and those featuring tumor tissue, in accordance with the specific project requirements. To clarify, from the GDC, there is the option to select both LUSC and LUAD projects. In each one of these project, a combination of healthy tissue samples and tumor tissue samples is encountered. Accordingly, for the LUAD project, a selection of 100 tumor tissue images and 50 healthy tissue images will be chosen. Likewise, for the LUSC project, the same procedure will be applied, resulting in a total of 100 control images, 100 LUSC images, and 100 LUAD images.

The aim here is to assess whether the neural network we employ can not only discern whether an image exhibits signs of health or tumor presence but also differentiate between various carcinoma types, thus constituting a multi-classification problem.

## 3.2 Streamlit

**Streamlit** is an open-source Python library that is designed to create interactive web applications for data science, machine learning, and visualization purposes. It allows developers to transform data scripts into shareable and interactive web applications without requiring advanced web development skills. Streamlit achieves this by providing a simple and intuitive way to convert Python scripts into web apps, making it an attractive choice for the interactive tool that has been developed within this work.

Streamlit's simplicity, integration with data libraries (Matplotlib, Pandas, Scikit-Learn, OpenSlide, h5py, etc.), rapid prototyping capabilities, and easy deployment make it an excellent choice for creating interactive web applications to showcase research findings and data visualizations as part of a master's thesis project.

## 3.3 Design of *FPathai* and how to use it

The open-source application created in this project is accessible via GitHub [19]. Accessing *FPathai* is a straightforward process. The process that the user should follow is the next one:

1. **Clone the repository**: Start by cloning the GitHub repository to your local machine using Git. You can do this by running the following command in your terminal or command prompt: `git clone https://github.com/FernandoPC25/FPathai.git`

2. **Create a Virtual Environment (Optional but Recommended)**: It's a good practice to create a virtual environment to isolate your project's dependencies. This step helps prevent conflicts with other Python projects. You can create a virtual environment using the command: `python -m venv your_env_name`.

3. **Activate the virtual environment** You can activate the created virtual environment with `.\your_env_name\Scripts\activate` in Windows or in Ubuntu with `source your_env_name/bin/activate`.

4. **Install dependencies** Navigate to the cloned repository's directory and execute: `pip install -r requirements.txt`. Note: In Windows devices, OpenSlide has to be downloaded here and then follow this installation guide for importing the library correctly.

After the successfull installation, one can utilize *FPathai* by executing the command `streamlit run Homepage.py` from the terminal where the file *Homepage.py* is located. Figure 3.5 illustrates the result that users can expect when executing this particular line of code.

*FPathai*'s design prioritizes user-friendliness, requiring no programming skills from the user. To ensure optimal functionality, a simple preparatory step is necessary: creating specific folders. Within these folders, the corresponding svs images for each class should be organized. For instance, consider a classification challenge involving $n$ labels. It is essential to arrange them within their respective folders accordingly. This means that the user is responsible for manually establishing the folders and placing the images inside them. *FPathai* will then handle all subsequent processes. Figure 3.2 illustrates a basic diagram depicting the operational process of the application. Note: the user is encouraged to assign a name that accurately

represents the content of each folder. For instance, suitable names might include "control," "tumor," and "metastasis" for a three labeled case.

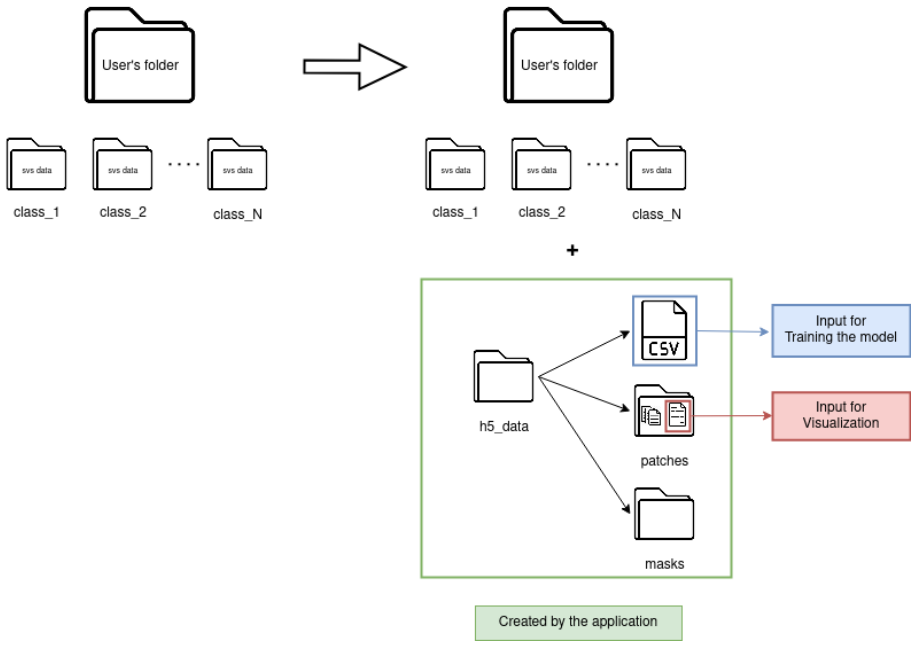


Figure 3.2: Scheme of *FPathai*

The design flow of the FPathhai application features built-in error control mechanisms to address situations where users fail to include the requested information. Some illustrative instances of this can be found in Figure 3.10.

Errors highlighted in red indicate instances where the user has not entered information in the correct format, whereas errors marked in yellow signify that the user has inputted the correct data but it does not meet the specified conditions. The latter scenario, for instance, may arise when the user enters a valid route, but the application cannot locate the requested information within that route.

### 3.3.1 Preprocessing

One of the primary challenges in histopathology images is separating the background from the tissue regions. We utilized the Otsu thresholding technique to automatically generate masks that delineate the tissue areas within the images. Otsu's method optimally classifies the image's intensity levels into foreground and background based on the variance of pixel intensities. [61]. An example of the mask created with the Otsu method of the Figure 1.2 can be seen in the Figure 3.3.

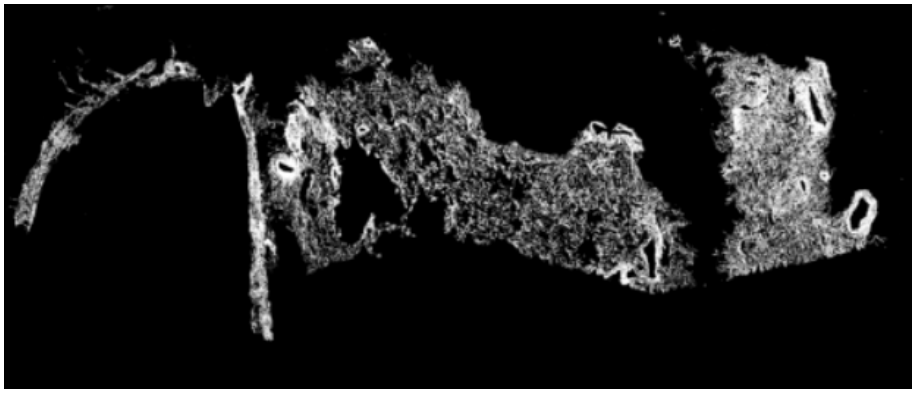

Figure 3.3: Mask created with the Otsu method of the Figure 1.2.

To enable efficient analysis, we extracted patches from the masked regions of the images. This step involved setting a threshold on the size of the connected components within the masks. Patches were then generated by cropping the image around these regions of interest as can be seen in Figure 3.4. This approach helps to focus the analysis on relevant tissue regions while discarding irrelevant or empty areas. Given the large volume of patches generated from the histopathology images, an efficient storage format is crucial. We chose the .h5 (HDF5) format for its ability to handle large datasets and hierarchical structures. Every single patch generated from the svs image is stored within its respective h5 file. Both the process of generating patches and saving them in the h5 format have been derived from the groundwork laid by previous work from the research group [12].

After completing the initial setup just explained, the user's interaction with the "Preprocessing" screen is remarkably straightforward. All that is required is to input the root directory containing these organized folders (referred to as "User's folder" in 3.2) in *FPathai* in the cell designated for directory input, as indicated in 3.6.

When incorporating a directory that fulfills the criteria outlined in Figure 3.2 , FPathai will present a maximum of 5 images per class, enabling users to verify the suitability of their data uploads. This functionality is depicted

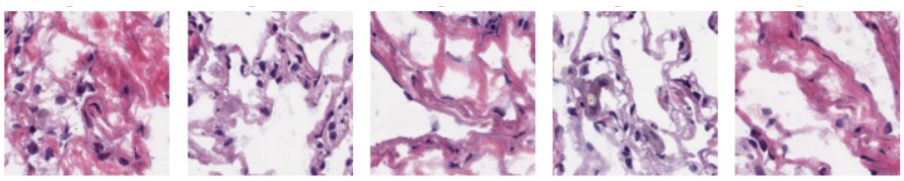

Figure 3.4: Example of 5 created patches created of the Figure 1.2.

in Figure 3.7.

After presenting the images, the user will be asked to specify two parameters: the **maximum number of patches to generate per image** and the **patch size**, as can be seen in the Figure 3.8. The patch creation is constrained to a maximum limit of 2000 to prevent memory overload, while the patch size is capped at 1024 to ensure that the patches created are not the same size as the original image.

After the user has selected both the maximum number of patches and the patch size, *FPathai* is primed to effortlessly generate and securely store them in the h5 format. To guarantee the tool's performance, it is important to abstain from adding any extra folders that has not been automatically generated by *FPathai*. This is because *FPathai* exclusively processes folders within its designated scope. If it encounters a folder outside this scope, it will terminate its operation.

Upon initiating the patch creation process, a directory named *h5_data* will be automatically generated to serve as the centralized repository for all pertinent data related to the generated patches. Within this *h5_data* directory, two subdirectories are established: *mask* and *patchesN* with *N* representing the chosen patch size.

*FPathai* streamlines the process of generating multiple image sets, each with varying patch sizes, according to user preferences. To optimize efficiency, it establishes the *mask* folder for preserving the initial mask generated during the analysis of the svs image, thus obviating the need for redundant computations when generating new patches from the same image set. Each new set of patches is systematically stored in distinct folders, as illustrated in Figure 3.2.

In addition to the comprehensive data within *h5_data*, an accompanying CSV file is generated. This CSV file plays a pivotal role in the subsequent "Training Model" phase, housing essential details required by the model. These details include the absolute path to the patch locations, the associated patient ID, the associated label (extracted from the corresponding folder), and the count of patches generated for the respective image. This step is shown in 3.9

It is important to note that the number of patches produced for each

image may exhibit variability. This variability is attributable to the image's content quality, particularly the presence of discernible features beyond the background. Images with minimal non-background content will naturally yield fewer patches.

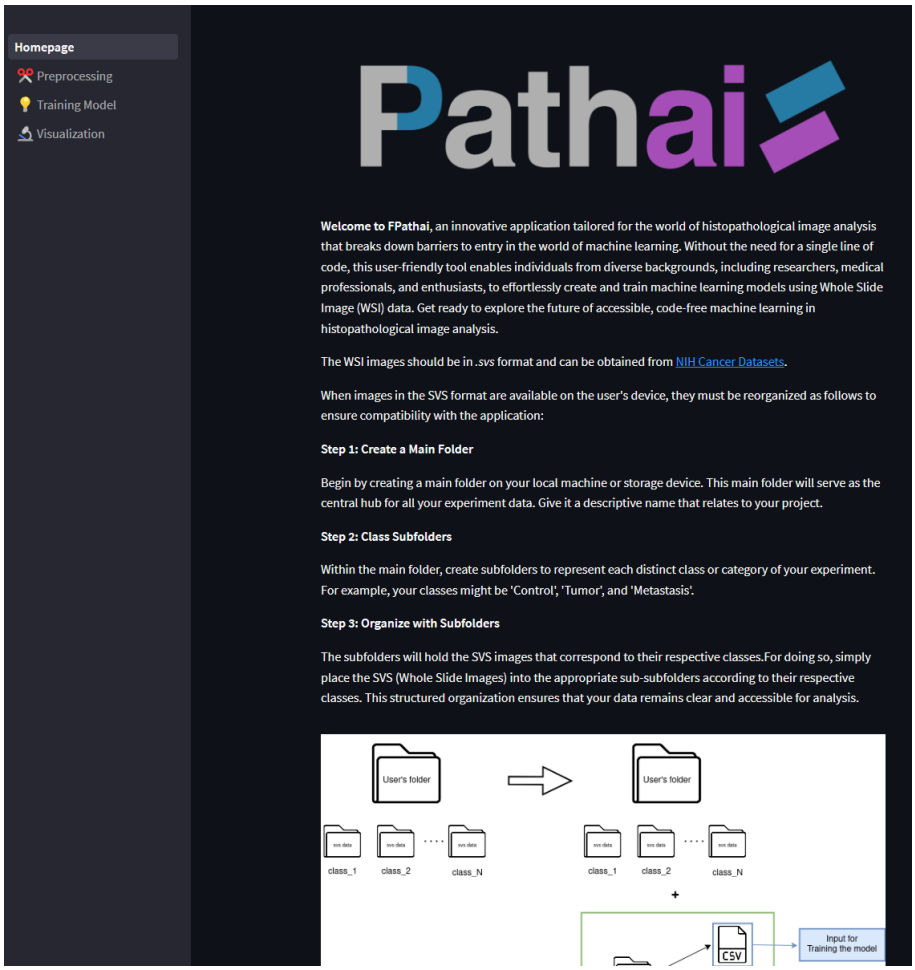


Figure 3.5: Homepage of *FPathai*

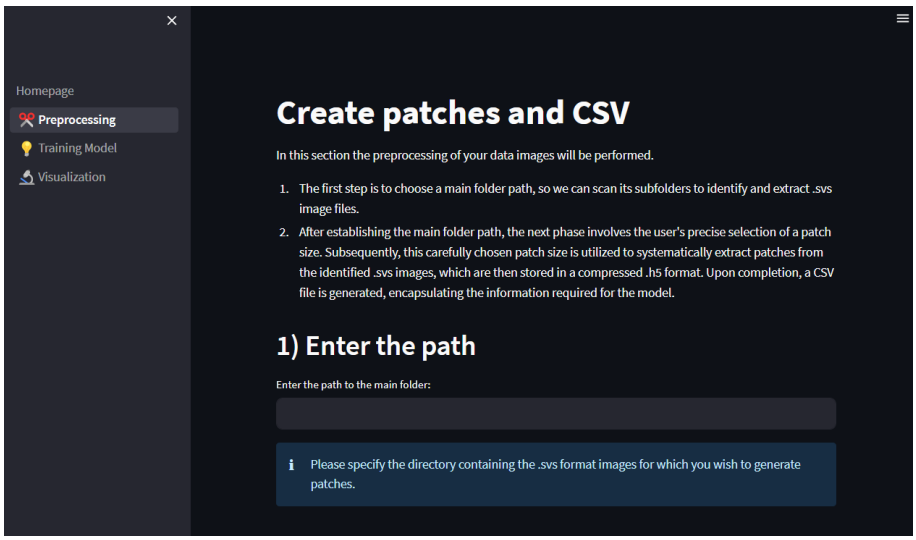


Figure 3.6: Preprocessing home page of *FPathai*

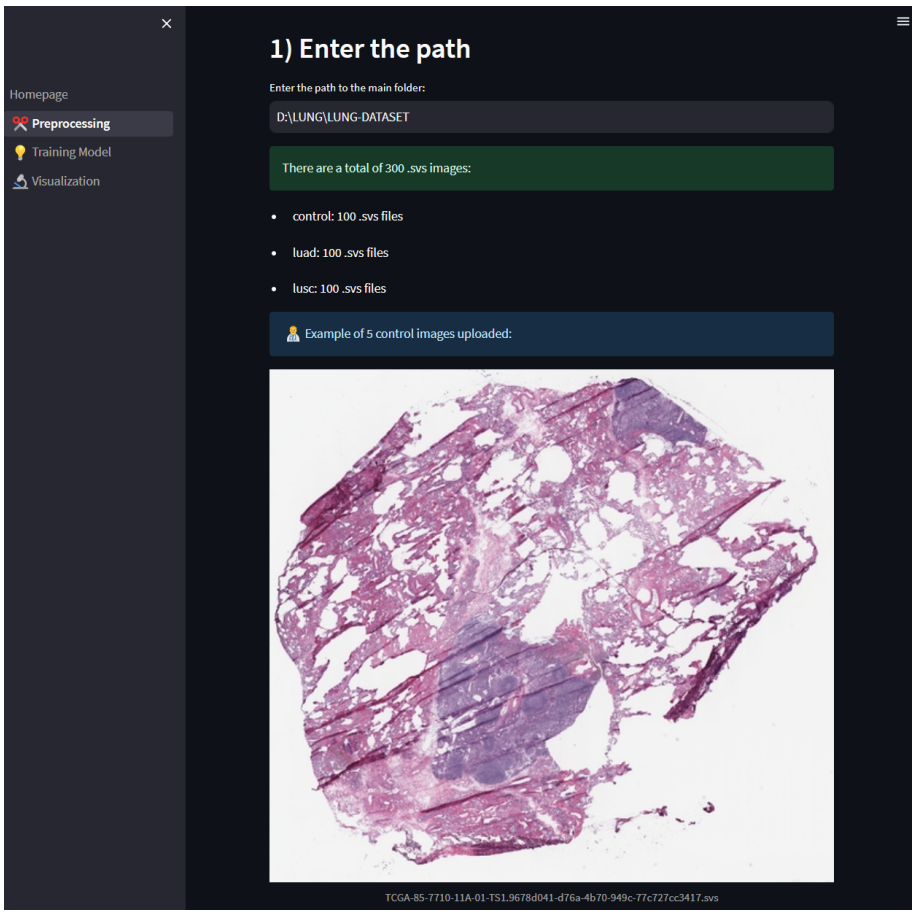


Figure 3.7: Showing the data before preprocessing in *FPathai*

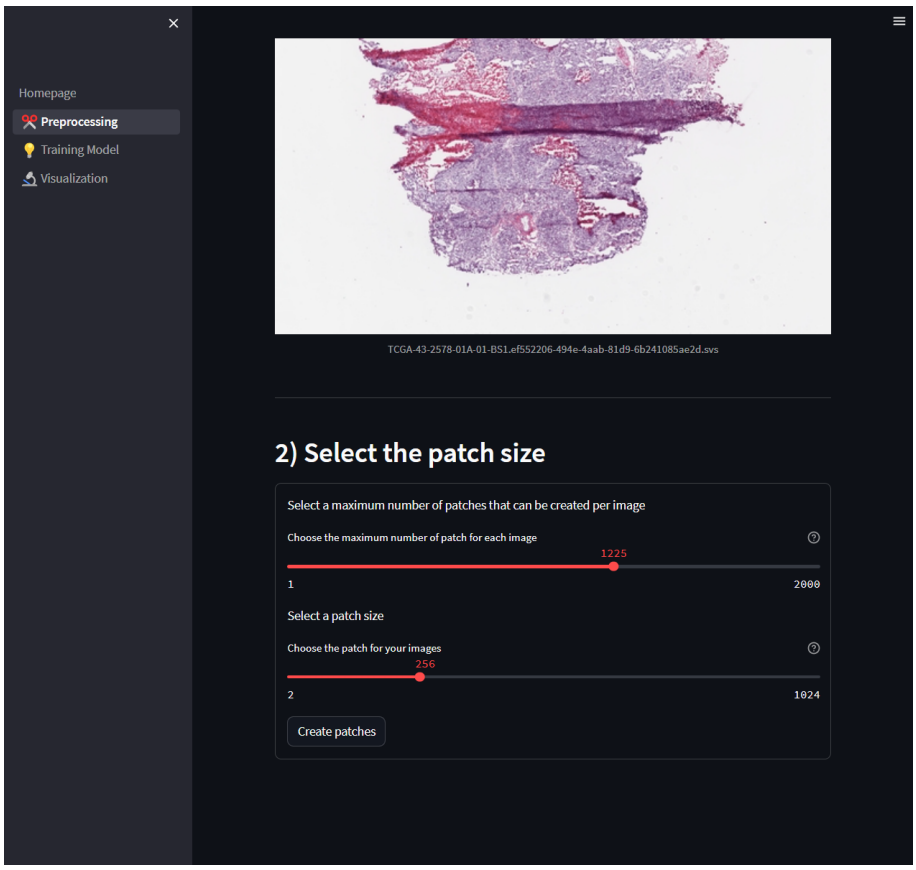


Figure 3.8: Selecting maximum number of patch and the patch size in *FPathai*

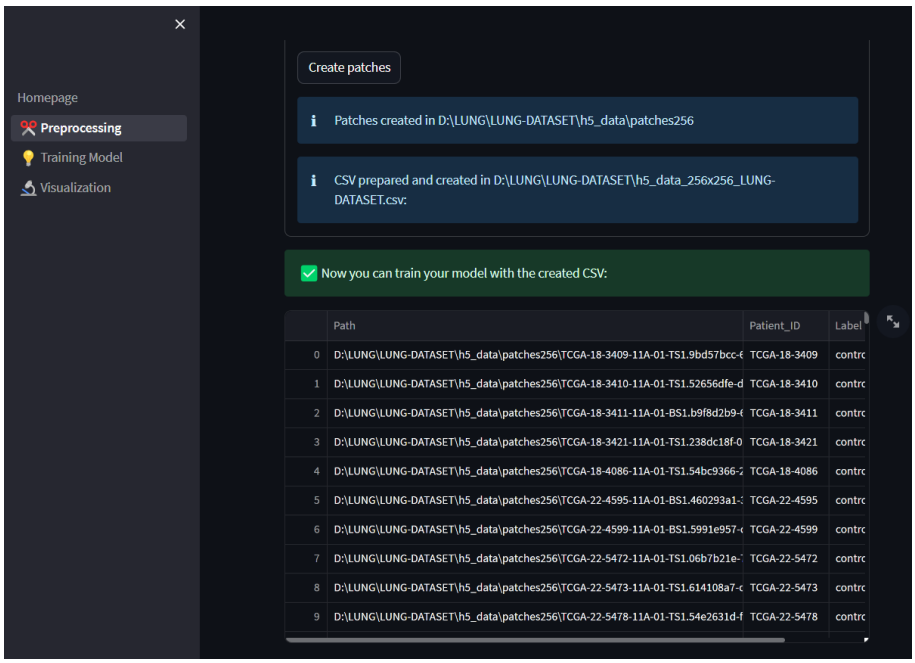


Figure 3.9: CSV checked and downloaded with *FPathai*

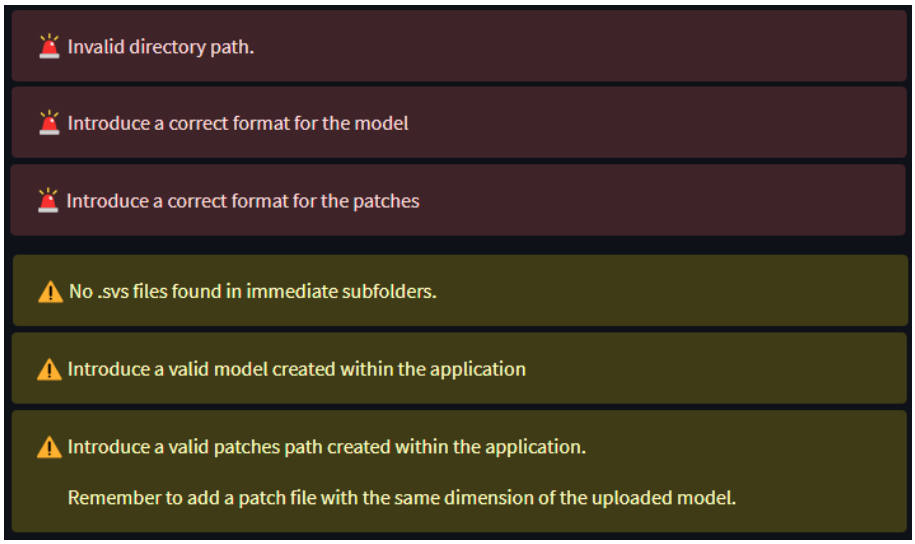


Figure 3.10: Example of errors controlled by *FPathai*

### 3.3.2 Training Model

The initial step for training the model entails the user's input of the created CSV that has been prepared in the prior preprocessing section, as illustrated in the Figure 3.12.

After successfully uploading the CSV, The user gains the flexibility to customize both the maximum number of patches to be used per image and the overall model configuration as can be seen in the Figure 3.13

When it comes to data loading within *FPathai*, a significant problem was found. As previously mentioned, the utilization of the .h5 files employ a more efficient storage format. However, this format presents a challenge during the loading phase as it necessitates accessing specific keys associated with each patch. Attempting to load all patches into memory simultaneously and then executing training in batches is infeasible due to the risk of memory overload.

To address the issue, the following strategy has been employed: For each image, the application loads all available keys, and subsequently, a random selection of keys is made, with the maximum selection size determined by the user's preference, corresponding to the maximum number of patches. If the user selects a number greater than the total number of patches available for the image, all available patches are included. Subsequently, a CSV file is generated in memory, with each row containing information about the .h5 file directory and the associated patch key. This process is visually depicted in Table 3.1. In the course of the training procedure, the initial $n$ rows are loaded into memory, with $n$ being determined by the chosen batch size, thus guaranteeing an efficient and easily manageable training experience. To mitigate overfitting, the dataset is shuffled at the commencement of each epoch. This method optimizes the data loading process, simplifying it by enabling the iteration through the rows of this CSV and the subsequent loading of the associated images.

| Path | Key | Patient_ID | Label |
|---|---|---|---|
| Path to .h5 file 1 | 78 | TCGA-22-4595 | tumor |
| Path to .h5 file 1 | 123 | TCGA-22-4595 | tumor |
| Path to .h5 file 1 | 8 | TCGA-22-4595 | tumor |
| Path to .h5 file 1 | 25 | TCGA-22-4595 | tumor |
| ... | ... | ... | ... |
| Path to .h5 file n | 499 | TCGA-57-9997 | control |
| Path to .h5 file n | 7 | TCGA-57-9997 | control |
| Path to .h5 file n | 417 | TCGA-57-9997 | control |
| Path to .h5 file n | 250 | TCGA-57-9997 | control |

Table 3.1: CSV loaded in memory

The complete training procedure relies on TensorFlow[62]. This encom-

passes the employment of transfer learning models, the ability to adjust hyperparameters, and the execution of the model training process.

Regarding model configuration, the user will be tasked with customizing the neural network. The primary choice in this regard is selecting the preferred transfer learning model that aligns with the user's intentions. Regardless of the transfer learning model selected, it is initialized with pre-trained weights from Imagenet. Subsequently, all network weights except the last two layers are frozen. Following this, a batch normalization layer is introduced, and the output is flattened. The final classification layer comprises 128 neurons with ReLU activation function and incorporates a *dropout* technique with a rate of 0.1. The activation function is configured as *softmax*, as it aligns with the use of *Categorical Crossentropy* for the task.

The transfer learning models outlined in this paper, in conjunction with the fully connected layer at the last layer, possess a set of parameters that can be seen in the table 3.2.

| **Model** | **Total params** | **Trainable** | **Non-trainable** |
|---|---|---|---|
| VGG16 | 18,911,426 | 6,555,522 | 12,355,904 |
| MobileNetV2 | 12,749,250 | 10,491,266 | 2,257,984 |
| ResNet50 | 40,373,506 | 16,781,698 | 23,591,808 |
| InceptionV3 | 31,248,546 | 9,441,666 | 21,806,880 |

Table 3.2: Number of Parameters in Different Models for an input shape of (256,256,3)

Among the models featured in this work, the ResNet model demands the most computational resources, closely trailed by InceptionV3, VGG16, and the most resource-efficient of them all, MobileNetV2[63]. This differentiation can exert a significant influence on decision-making regarding training and selection. In our particular case, the number of trainable parameters experiences a significant influence as a result of the configuration of the final layers in these models. This influence stems from the decision to unfreeze the last two layers. In this context, VGG16 emerges as a potentially favorable choice due to its balanced distribution of trainable and non-trainable parameters. This model is particularly robust, while under this configuration, it offers the fewest adjustable parameters, thereby expediting the training process. *FPathai* offers a range of customizable settings to fine-tune your machine learning model training process.

- **Batch size** refers to the number of data samples processed by a machine learning model during a single forward and backward pass. Small batch sizes lead to faster convergence but may increase the risk of overfitting due to frequent updates. Large batch sizes result in slower convergence but typically offer better generalization and stability, though they require more computational resources. In our results, sizes around

16, 32 and 64 have been selected.

- An **epoch** signifies the complete passage of the dataset through the neural network. In our experiments, we limit it to a maximum of 10 epochs, but users can select up to 100 iterations in *FPathai*.
- The **optimizer** is responsible for modifying the neural network's weights as it undergoes training. In *FPathai*, the user can select *Adam*, *SGD*, *RMSProp* and *Adagrad*.
- The **learning rate** the only parameter of the optimizers that can be modified. The remaining hyperparameters of each optimizer are configured with the default values provided by TensorFlow.

Furthermore, users are presented with the choice to employ data for validation, enabling real-time monitoring to gauge the presence of overfitting tendencies. It is really recommended to enable it and we have done it to show our results.

In the context of metric analysis, the application distinguishes between training and test metrics. For **training data**, loss and accuracy is calculated. To compute them, they accumulate throughout the epoch and are subsequently divided by the total number of batches processed. This approach yields a comprehensive assessment of the overall loss and accuracy for the entire epoch. For **testing data** *FPathai* offers a range of metrics, including loss, accuracy, F1-score, precision, recall, and AUC. They are calculated with specific functions of TensorFlow, but they can are computed as follows:

$$\begin{aligned}
\text{Accuracy} &= \frac{TP + TN}{TP + TN + FP + FN} \\
\text{Precision} &= \frac{TP}{TP + FP} \\
\text{Recall} &= \frac{TP}{TP + FN} \\
\text{F1-Score} &= 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}
\end{aligned}$$

The Recall is also known as sensitivity or True Positive Rate (TPR). The specifity (that is not calculated within the platform) is also known as The True Negative Rate ($TNR = \frac{TN}{TN+TP}$). The False Positive Rate is calculated by $\text{FPR} = 1 - TNR$. Whereas TPR quantities the rate at which true positives are idenfied, FPR the rate at which the model incorrectly classifies negative example as positive.

To generate and interpret the **Receiver Operating Characteristic (ROC)** curve, it is essential to have TPR and FPR at disposal since the

ROC curve materializes by graphing the TPR against the FPR while adjusting the classification threshold at various settings. The ROC graph (see 3.11) encapsulates the performance across different thresholds, consolidating the information from multiple confusion matrices. By computing the area under the ROC curve, we obtain the AUC (Area Under the Curve) metric. AUC is the conventional metric for comparing and evaluating different models against one another.

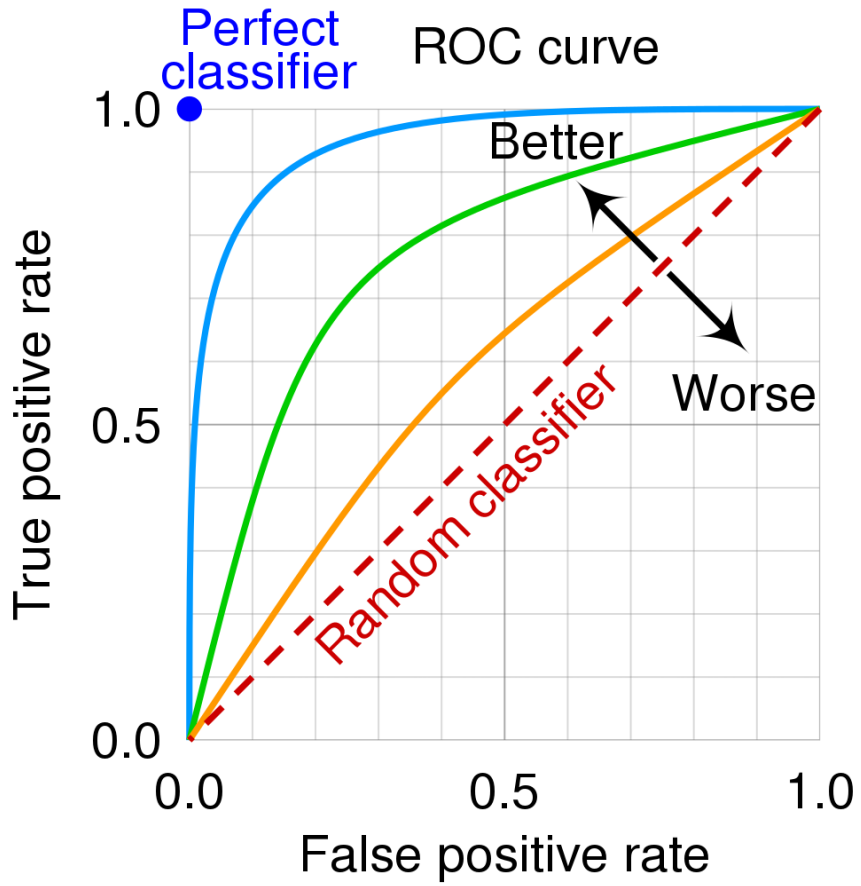


Figure 3.11: Example of ROC Curve [64] The scenario is optimal when the AUC equals 1, indicating the model's flawless ability to differentiate between all classes. Conversely, when the AUC equals 0.5, it represents the worst-case scenario, as the model cannot distinguish between classes and performs akin to a random classifier.)

After the platform displays these metrics, the model is downloaded to the same folder as `Homepage.py`. The model will be assigned a name to encapsulate its key information. This name will include details such as the

selected transfer learning model, a UUID, patch size, maximum patches used, batch size, epochs, and optimizer. This nomenclature serves the purpose of facilitating model identification, particularly when managing a large number of mode The models will be saved in TensorFlow's .h5 format and can be utilized in the subsequent visualization section.

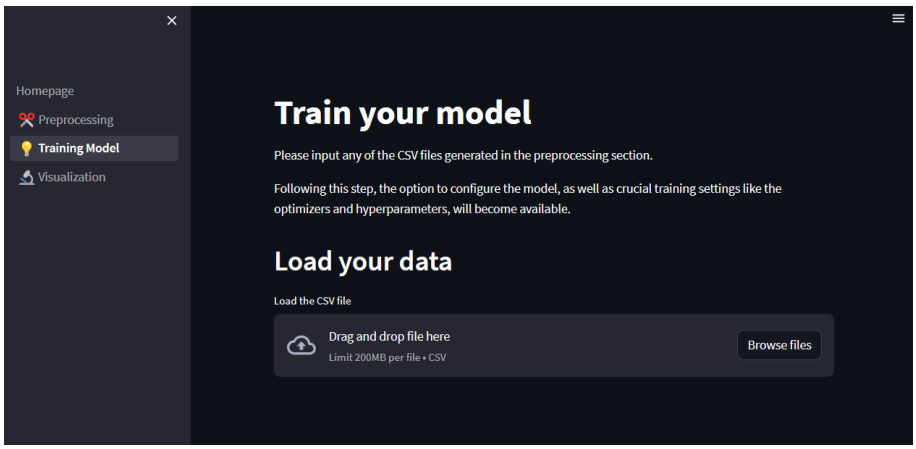


Figure 3.12: Training model homepage in *FPathai*

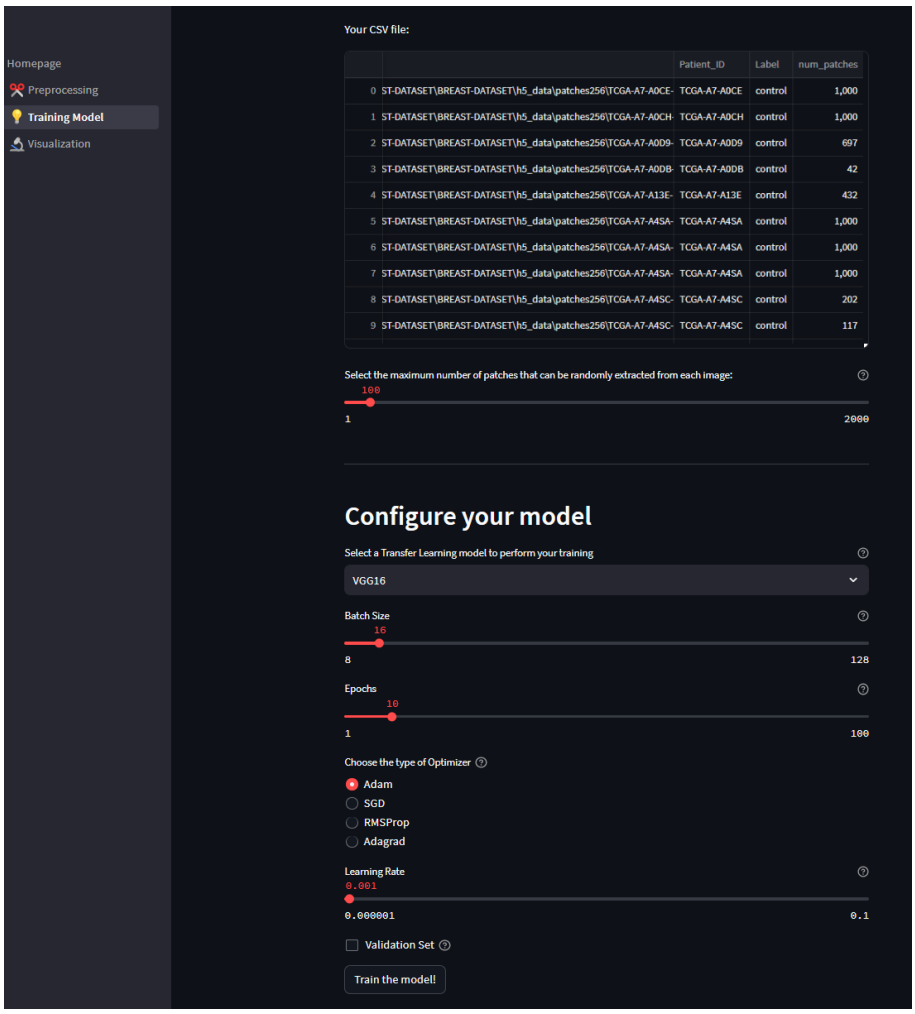


Figure 3.13: Select maximum number of patches per image to use and configure the training paremeters in *FPathai*

### 3.3.3 Visualization

The purpose of this feature is to provide a visual representation of the model's predictions within the patch. In other words, it intends to help the user comprehend which specific element within the patch the neural network is focusing on when making a prediction. Traditionally, this task requires an extensive clinical evaluation, radiological imaging, and clinical correlation. [65] To enhance the pathologist's understanding, we've considered integrating GRAD-CAM into the model. This addition aims to provide interpretability to the model, taking a step toward the realm of explainable artificial intelligence (XAI).

Grad-CAM relies on the gradients of the network's final prediction with respect to the activations of the last convolutional layer. These gradients indicate how much each feature map in the last convolutional layer contributes to the final prediction. [66] The last convolutional layer is crucial for Grad-CAM because it contains the feature maps that capture high-level, spatially localized information that is used to generate the heatmap highlighting the important regions of the input image. The choice of the VGG16 model is reaffirmed, primarily driven by the presence of a convolutional layer within one of its last two layers, leading to an improvement in visualization quality through GradCAM. In our scenario, we will employ a "jet" heat map to emphasize points in red that the convolutional layer has considered significant for its prediction, while retaining points in blue to signify those it has assigned little importance to. An example of a patch along with its Grad-CAM prediction within *FPathai* can be seen in the Figure 3.14.

To initiate this process, the user is required to upload the CSV file they used for training the model that needs testing. This step serves the purpose of enabling *FPathai* to identify the labels to be predicted and present them in an organized format, matching the folder names where the user initially stored the SVS images. Following this, the user is tasked with incorporating the model they trained, which can be obtained from the *Training Model* section, along with an associated .h5 file obtained from the **Preprocessing** section. The visualization homepage can be seen in Figure 3.16.

A comprehensive guide on how to utilize *FPathai*, serving as a summary of all the details elucidated in this chapter, is visually depicted in the Figure 3.15. It serves to provide a concise overview of the process. In essence: The process begins with image preprocessing, where a set of images is transformed into patches. These patches are then used to train a neural network model. Subsequently, by applying this trained model to the patches, an activation map is generated. The activation map is a visual aid intended to assist pathologists in comprehending the neural network's focal points for classification.

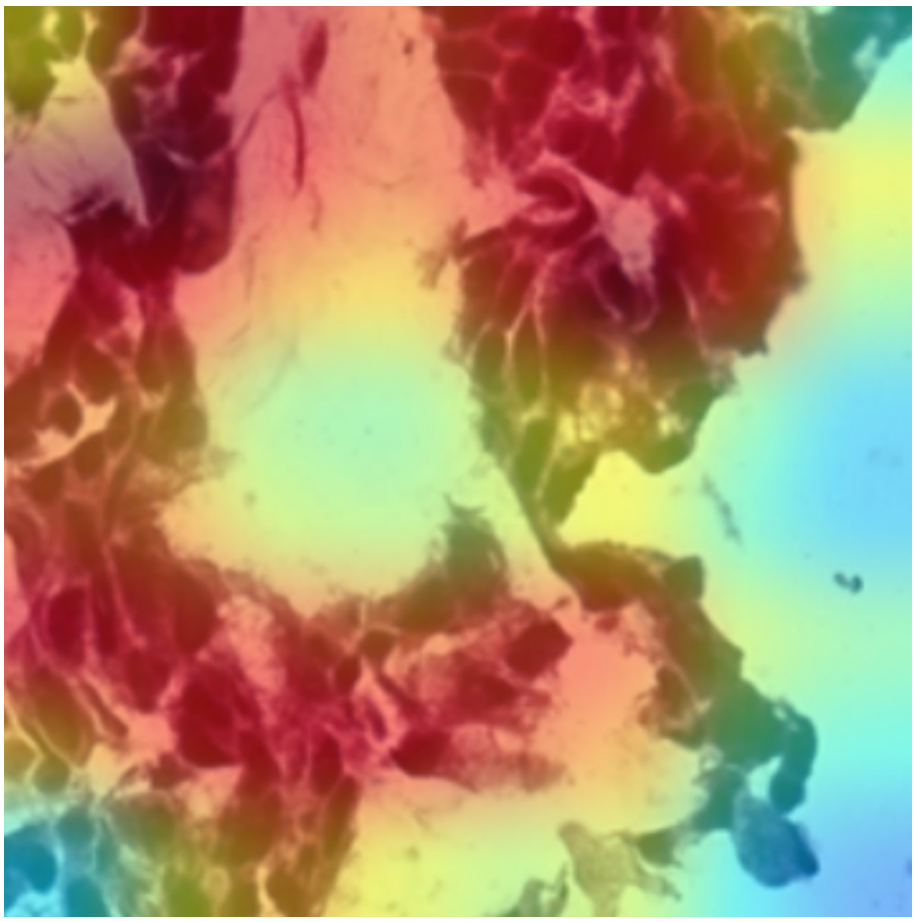

Figure 3.14: Grad-CAM visualization

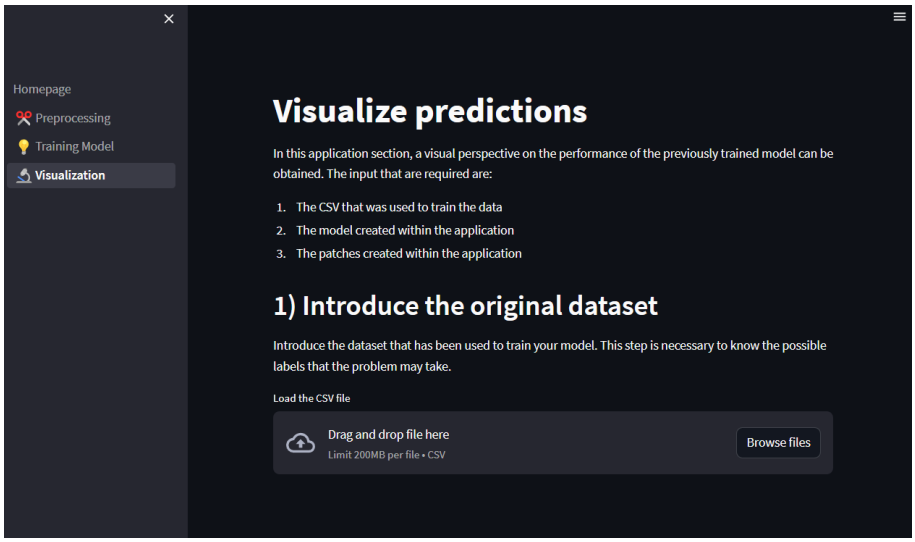


Figure 3.16: Visualization homepage in *FPathai*

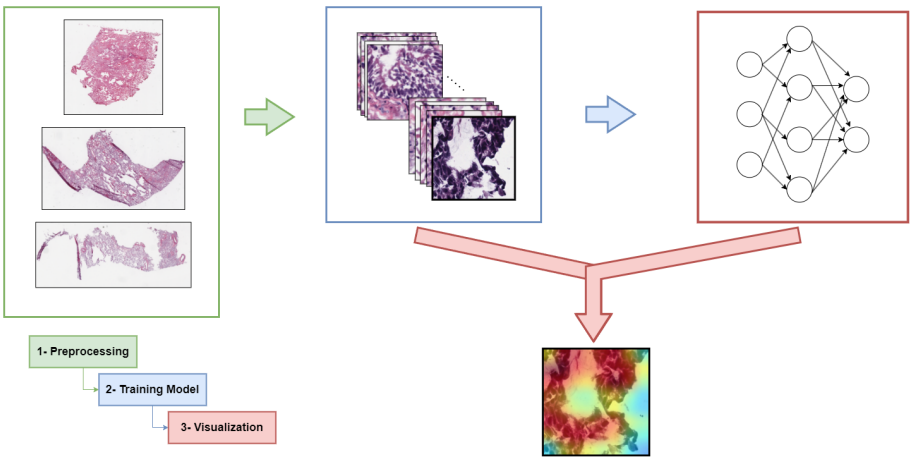


Figure 3.15: Procedure for utilizing *FPathai*

# Chapter 4

# Results and discussion

It is important to emphasize that our focus in this study is to highlight the **utility** of the created application rather than striving for exceptionally precise outcomes. Consequently, in the results showcased herein, we have deliberately limited the number of patches and epochs employed. Nevertheless, *FPathai* has been optimized to accommodate a more extensive range of images and epochs as needed, provided it remains within the memory constraints of the device. For these reasons, the focus here is on presenting the test that produced the most accurate results, without delving into the details of every conducted test. As a result, we decided to modify our approach. For the breast cancer dataset, we opted for a batch size of 32, and for the lung cancer dataset, we chose a batch size of 64. Additionally, we adjusted the patch sizes to 64 for breast cancer scenarios and 128 for lung cancer scenarios. We opted for a larger batch size in the case of lung cancer due to the presence of three distinct classes. This choice ensures that each batch contains a substantial representation of all three classes, promoting a balanced distribution of data within the batches.

## 4.1 Breast cancer detection: binary experiment

### 4.1.1 Preprocessing

For this experiment, our chosen image preprocessing approach involved generating 1000 random patches from each of the 200 images (comprising 100 control and 100 tumor samples), with each patch having dimensions of **64x64** pixels. However, when it came to uploading this data into *FPathai*, we decided to select **500 random patches** from each image instead, as can be seen in 4.1. This decision was made to avoid excessively long training times associated with handling all 200000 patches. Consequently, we formed a total dataset of 100000 patches.

It is worth noting that the reason we generated 1000 patches per image initially was due to the small patch size. This detail is important to clarify

because if we were to increase the patch size, the preprocessing might not generate as many patches as the specified number, as the mask percentage might not surpass a certain threshold.

Given the utilization of a total of 100,000 patches, we allocated 20% for testing, equating to 20000 patches designated for this purpose. The remaining 80% is dedicated to training, with an additional 20% of this training data set aside for validation. Consequently, our data set comprises **64000 patches for training**, **16000 patches for validation**, and the aforementioned **20000 patches for testing**.

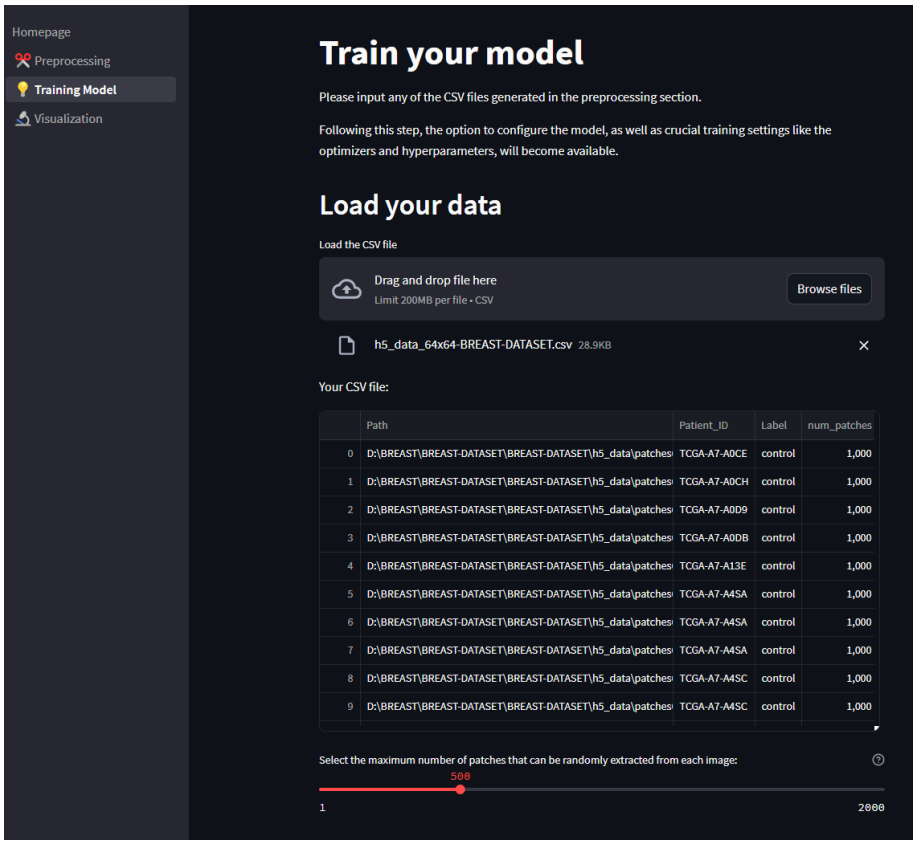


Figure 4.1: Loading the breast dataset

### 4.1.2 Training model

The configuration in this binary classification problem has been set up as depicted in Figure 4.2. Specifically, we've selected the VGG16 model with a batch size of 32 for 10 epochs. The chosen optimizer in this scenario is Adam, utilizing a learning rate of $10^{-3}$. Additionally, the validation set has been incorporated to monitor potential overfitting.

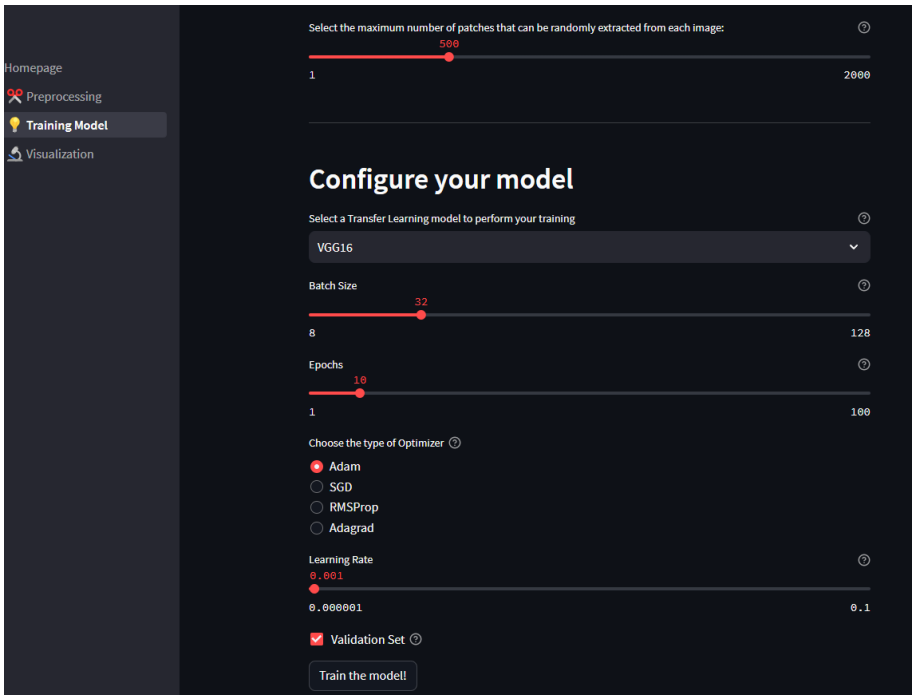


Figure 4.2: Setting the model for the breast cancer detection experiment within *FPathai*

Once all the model configurations are in place, we pressed the *Train the Model!* button to initiate its execution. As illustrated in Figure 4.3, we can observe the real-time updates of accuracy and loss metrics after each epoch.

To provide users with a more visually informative experience of the accuracy and loss metric, *FPathai* offers downloadable graphical representations, denoted as Figure 4.4 and Figure 4.5 respectively. These figures graphically depict the data presented in Figure 4.3, enhancing user comprehension.

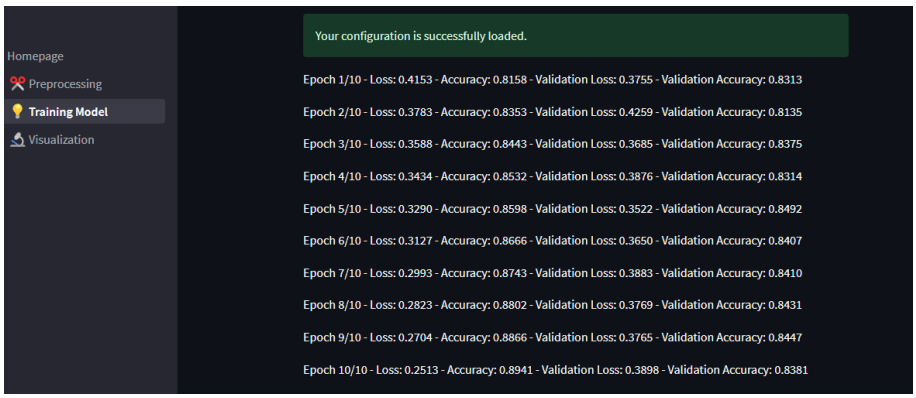


Figure 4.3: Loss, Accuracy, Validation Loss and Validation Accuracy for the breast cancer detection experiment within *FPathai*

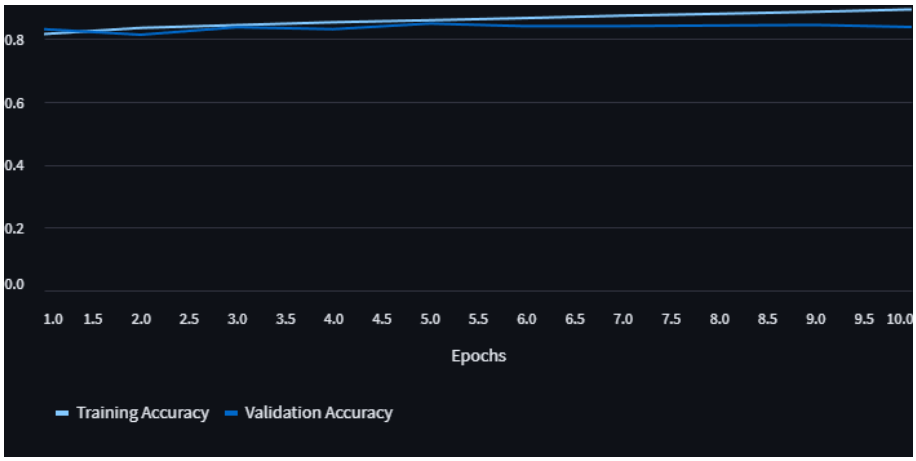


Figure 4.4: Evolution of Accuracy for the breast cancer experiment per epoch as shown in *FPathai*

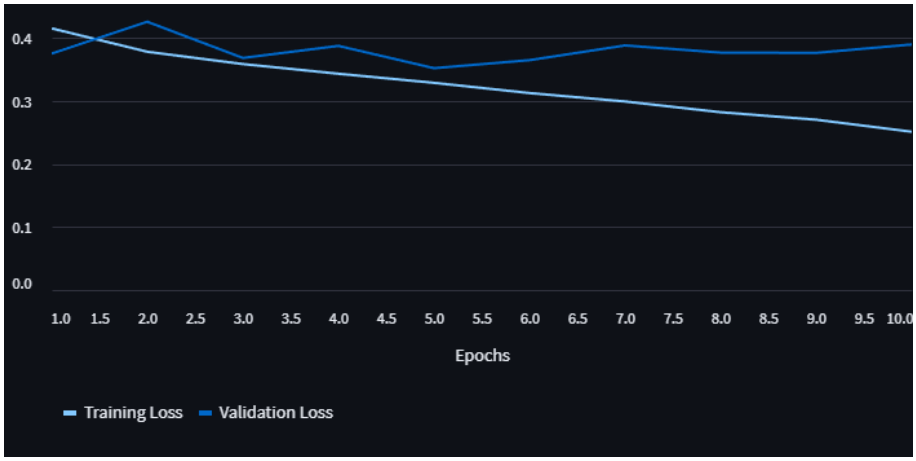


Figure 4.5: Evolution of Loss for the breast cancer experiment per epoch as shown in *FPathai*

These graphs shows that the training accuracy metrics exhibit gradual but consistent improvement, albeit at a slow pace, and the training loss metrics follow suit. However, the situation differs for the validation dataset. Notably, the validation loss remains nearly constant across the epochs. This observation raises the possibility that, in this configuration, the neural network may start memorizing the data, potentially achieving satisfactory results after just one iteration.

Upon completing the training phase, the model undergoes testing using the reserved training data. The initial visual representation is provided by the confusion matrix, displayed in Figure 4.6. This matrix serves as a means to quantify the model's correct and erroneous predictions.

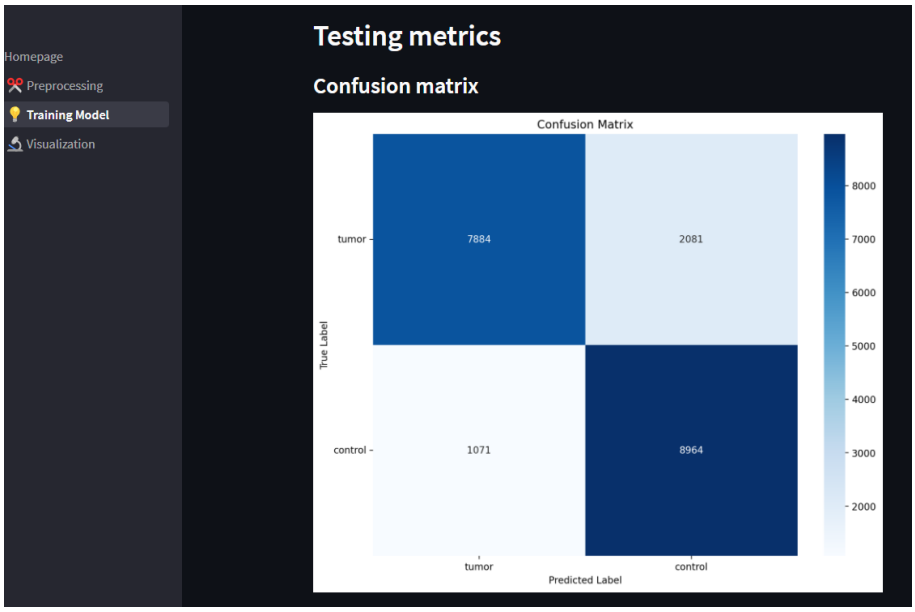


Figure 4.6: Confusion Matrix for the breast cancer experiment showed in *FPathai*

In this context, the model correctly identifies 7884 control patches and 8964 tumor patches. However, it erroneously categorizes 2081 tumor patches as control patches. The latter holds paramount importance within the confusion matrix, demanding minimization. Regrettably, the model falls short in this aspect. Ideally, it is preferable to incur a higher number of false positives over false negatives, with a positive case being the detection of tumor patches. It is important to clarify that even though we have a total of 20000 patches to test, it does not necessarily mean there are exactly 10000 from control and 10000 from tumors. This is due to the random selection process, and in this specific scenario, we have a total of 9965 patches for tumor and 10035 patches for control.

After scrutinizing the confusion matrix, we can glean valuable insights by examining its information across various thresholds through the creation of a ROC curve, depicted in Figure 4.7.

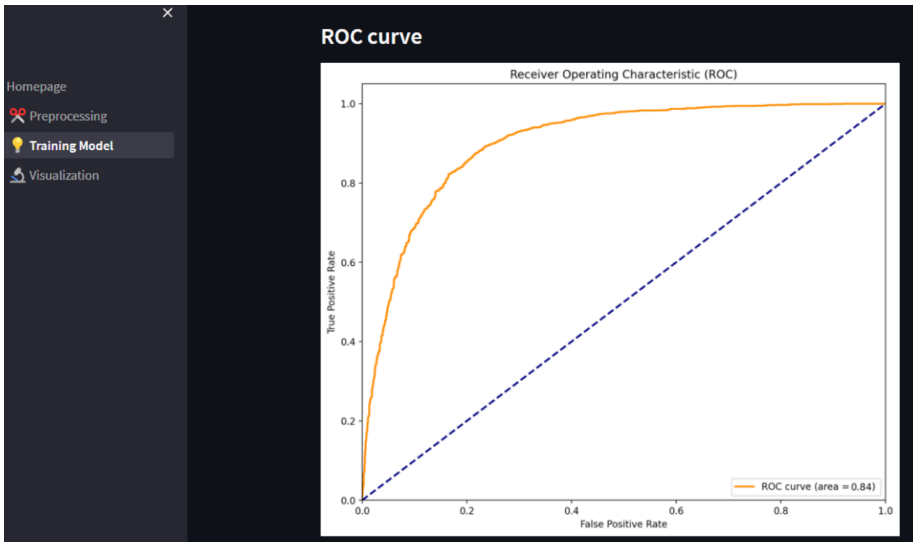


Figure 4.7: ROC for the brest cancer model showed in *FPathai*

Furthermore, *FPathai* provides comprehensive metrics, including loss, accuracy, F1-Score, Precision, and Recall, evaluated on the test dataset, as visually represented in Figure 4.8. These metrics are complemented by the AUC score. These results collectively demonstrate a success rate of approximately 85%. This achievement is particularly notable given the model's simplicity and the few amount of patches taken. It could be argued that the limited number of epochs contributes to the inability to achieve higher precision. However, it is important to note that signs of overfitting were already emerging in the model. Consequently, while increasing the number of epochs might potentially enhance performance, we remain skeptical that this alone would yield substantial improvements. Furthermore, techniques such as data augmentation can also be used to reduce the overfitting, although it has not been applied in this work.

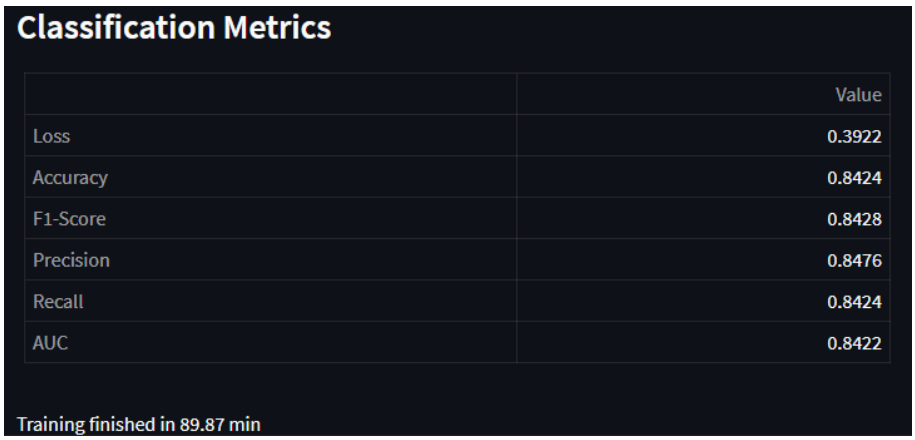


Figure 4.8: Testing metrics for the breast cancer model shown in *FPathai*

### 4.1.3 Visualization

After we have reviewed the testing metrics, we proceed to the visualization section. Here, we delved into understanding what the model considered when classifying a patch as one category or another. This exploration holds significance regardless of whether the model's prediction is accurate or not. If the model predicts correctly, it helps users take note of the relevant factors. Conversely, if the prediction is incorrect, it aids users in realizing that the illuminated value may not be a crucial determinant for classifying it as the specified category. Once we have input the directory path for the trained model and the directory path for the .h5 file containing the patches to be analyzed, the patches were presented alongside their respective GRAD-CAM visualizations. We will commence by demonstrating the display of patches generated from a tumor image as shown in 4.9.

All three tumor patches have been accurately classified with a class prediction accuracy of 100%. When examining the GradCAM map, it becomes evident that the activation is particularly pronounced within the darker regions of the image, corresponding to the basophilic structures shaded in blue. While this conclusion may not be as nuanced as that of an expert pathologist, it remains both intriguing and highly significant for our study.

In the context of the three control patches, we observe in the Figure 4.10 a mixed outcome: one is correctly classified as 100% control, another erroneously as 100% tumor, and a third correctly as 99.786% control. Analyzing the activation maps in this figure, we find that drawing definitive conclusions is more challenging compared to the tumor case. This discrepancy may arise from the patches' less defined shapes, causing the neural network to struggle in pinpointing specific structures. Even in the second patch containing a distinct nucleotide circle, the network doesn't seem to

focus on that expected element. It is worth noting that the neural network's attention seems directed toward aspects that might not appear initially important to the human eye, a factor to consider both in its misclassification and correct identifications.

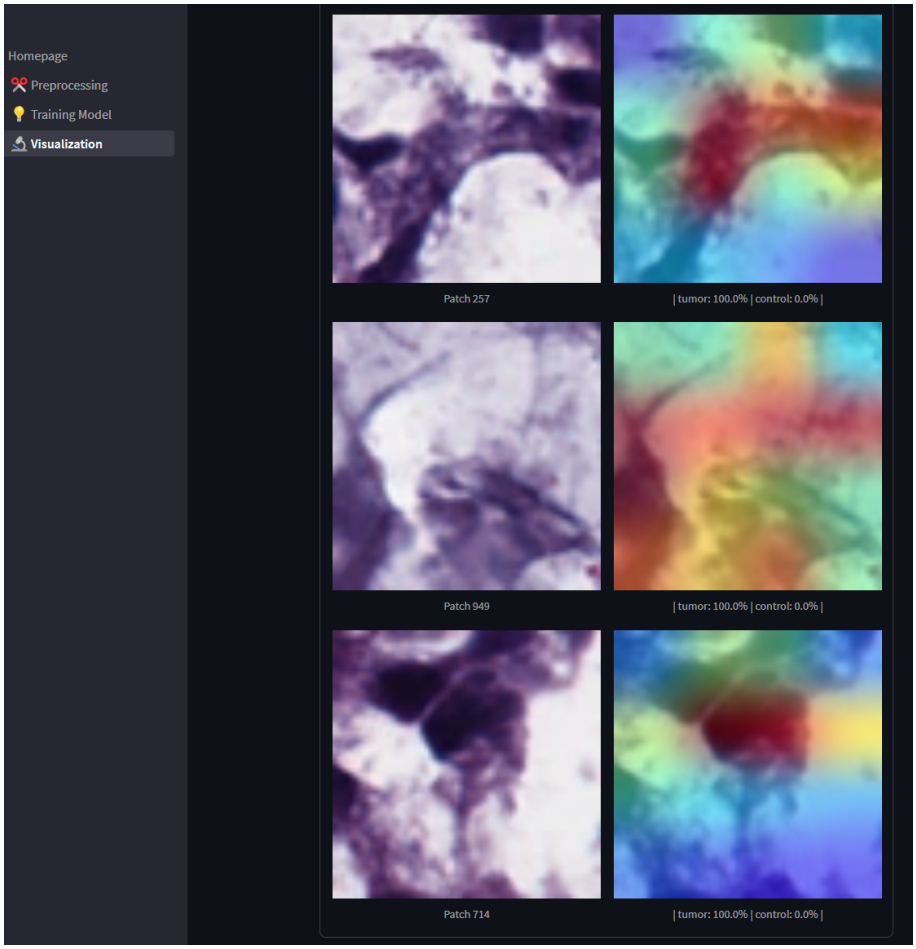


Figure 4.9: Visualization of GradCAM for patches of a breast tumor patient shown in *FPathai*

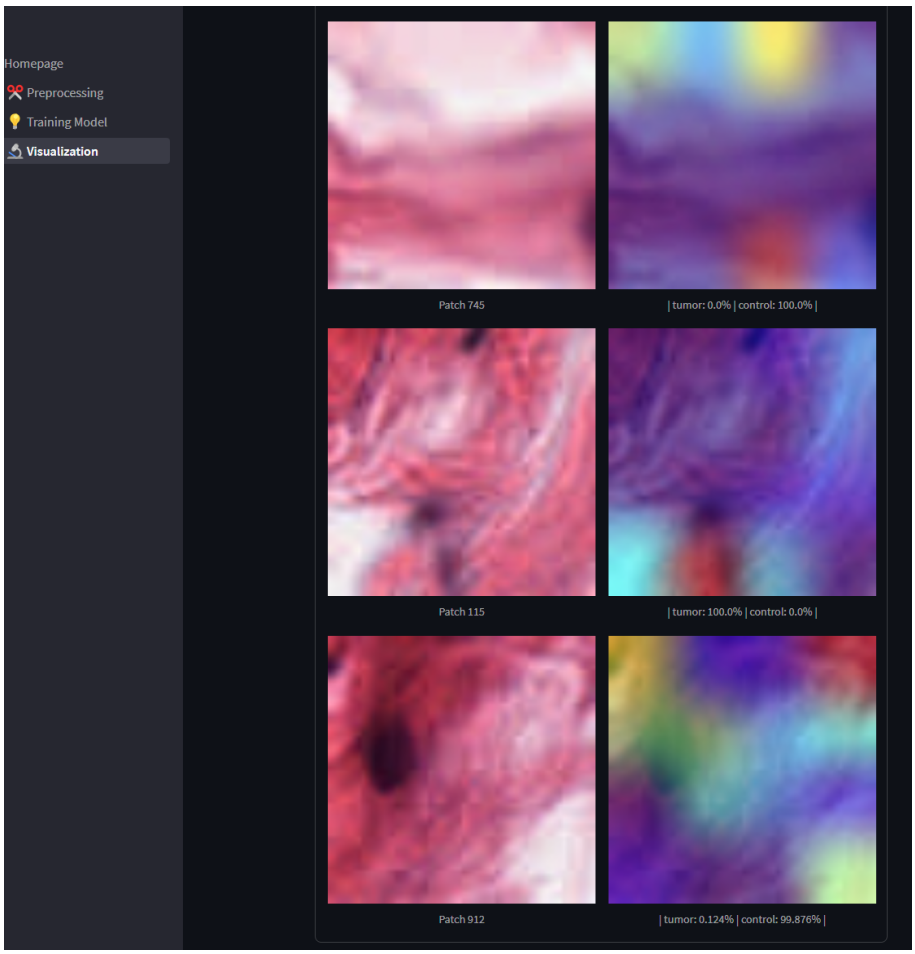


Figure 4.10: Visualization of GradCAM for patches of a breast control patient shown in *FPathai*

## 4.2 Lung cancer detection: multiclassification experiment

### 4.2.1 Preprocessing

For this experiment, our chosen image preprocessing approach involved generating 500 random patches from each of the 300 images (comprising 100 control, 100 LUAD tumor samples and 100 LUSC tumor samples), with each patch having dimensions of **128x128** pixels. However, when it came to uploading this data into *FPathai*, we decided to select **300 random patches** from each image instead, as can be seen in 4.11. This decision was made to avoid excessively long training times associated with handling all 300,000 patches. Consequently, we formed a total dataset of 90,000 patches.

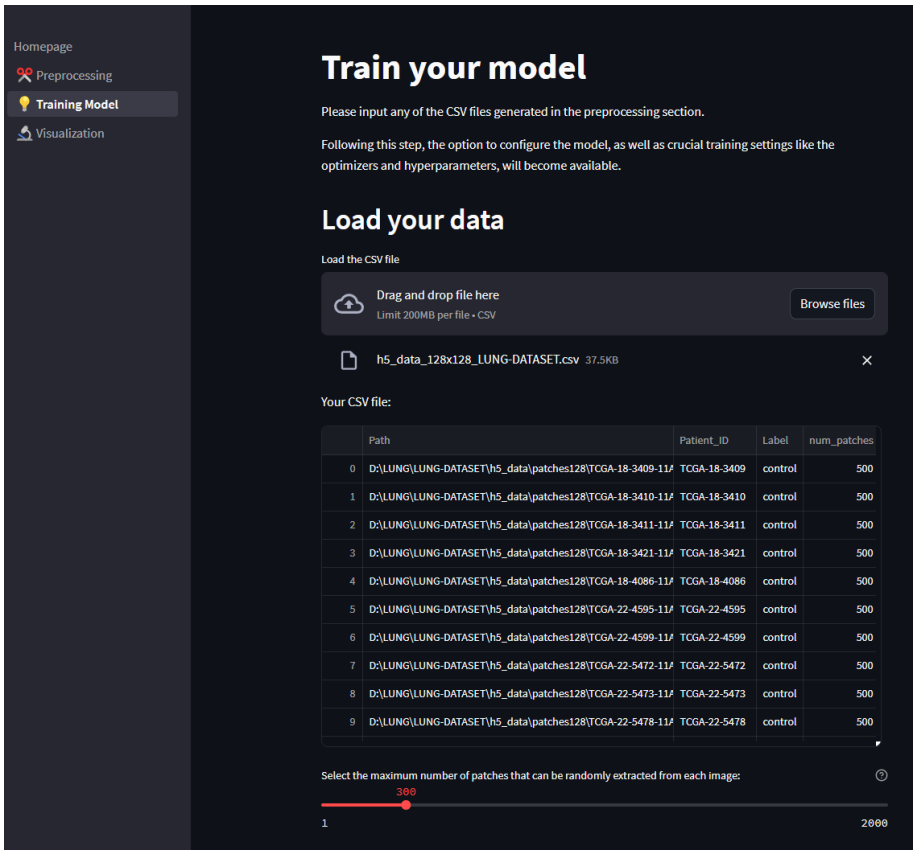


Figure 4.11: Loading the lung dataset

In this scenario, we generated the maximum achievable number of patches per file, which, in this case, was capped at 500. This is attributed to the relatively small dimension of the chosen patch. We noticed that despite en-

larging the patch size to 128, we continued to generate the desired maximum number of patches. This fact is attributed to our predefined limit of 500. If we had set a higher limit, the maximum number of patches per image might not have been created.

Given the utilization of a total of 90000 patches, we allocate 20% for testing, equating to 18000 patches designated for this purpose. The remaining 80% is dedicated to training, with an additional 20% of this training data set aside for validation. Consequently, our data set comprises **57,600 patches for training**, **14400 patches for validation**, and the aforementioned **20,000 patches for testing**.

### 4.2.2 Training model

The configuration in this multi classification problem has been set up as depicted in Figure 4.12. Specifically, we've selected the VGG16 model with a batch size of 64 for 10 epochs. In the context of this multi-classification problem, the decision was made to select a batch size of 64. This decision is driven by the inclusion of an additional cancer type within the dataset, necessitating a sufficient representation of each class within every batch to optimize the model's generalization capability. The chosen optimizer in this scenario is SGD, utilizing a learning rate of $3 \times 10^{-4}$. Additionally, the validation set has been incorporated to monitor potential overfitting.

Once all the model configurations are in place, we pressed the *Train the Model!* button to initiate its execution. As illustrated in Figure 4.13, we can observe the real-time updates of accuracy and loss metrics after each epoch.

To provide users with a more visually informative experience of the accuracy and loss metric, *FPathai* offers downloadable graphical representations, denoted as Figure 4.14 and Figure 4.15 respectively. These figures graphically depict the data presented in Figure 4.13, enhancing user comprehension.

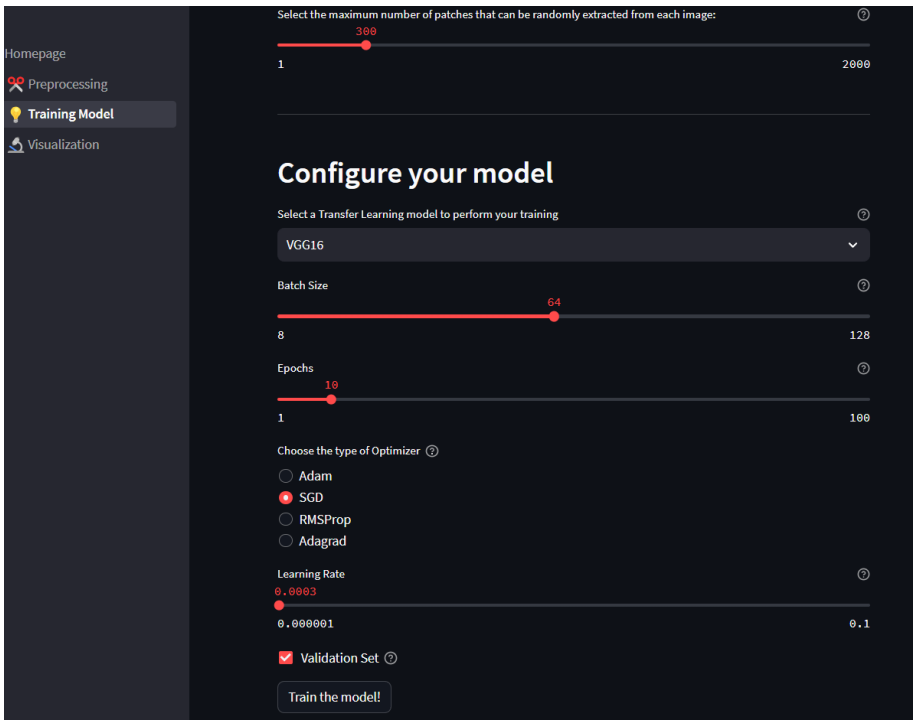


Figure 4.12: Setting the model for the lung cancer detection experiment within *FPathai*

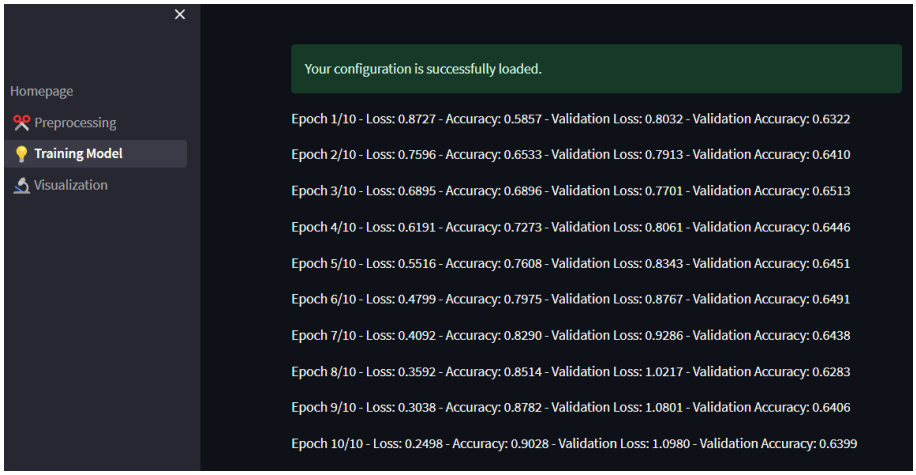


Figure 4.13: Loss, Accuracy, Validation Loss and Validation Accuracy for the lung cancer detection experiment within *FPathai*

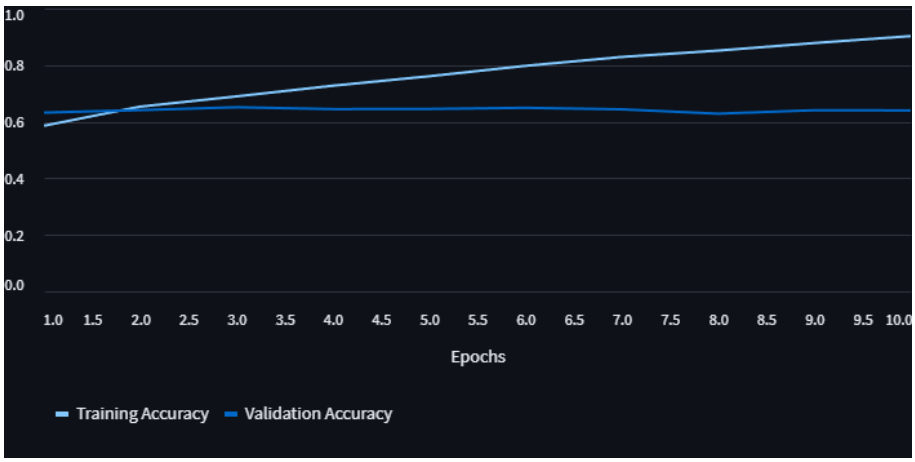


Figure 4.14: Evolution of Accuracy per epoch for the lung cancer experiment as shown in *FPathai*

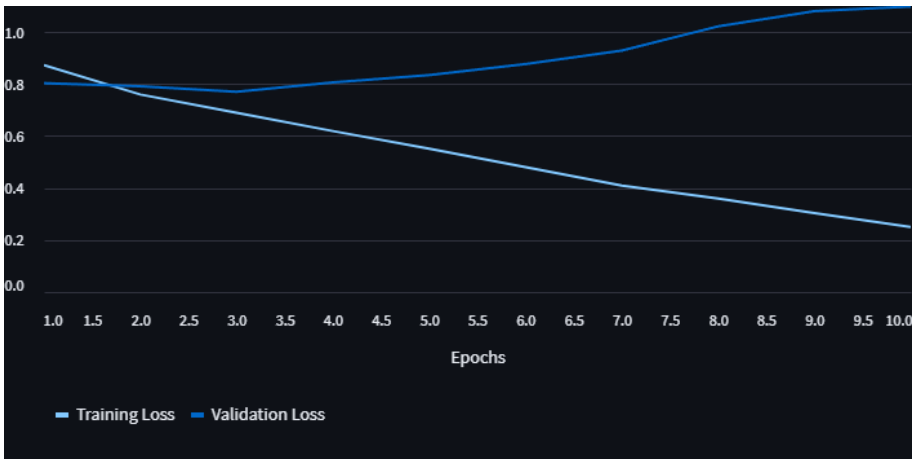


Figure 4.15: Evolution of Loss per epoch for the lung cancer as shown in *FPathai*

In this scenario, we encountered a clear case of overfitting. This is evident from the training loss and accuracy metrics consistently improving with each epoch, while the validation metrics show a different pattern. Specifically, the validation loss tends to increase as the number of epochs progresses, indicating that the neural network requires additional regularization to enhance its ability to generalize when presented with new data. This graph unmistakably underscores the vital significance of utilizing the validation set.

Upon completing the training phase, the model undergoes testing using

the reserved training data. The initial visual representation is provided by the confusion matrix, displayed in Figure 4.16. This matrix serves as a means to quantify the model's correct and erroneous predictions.

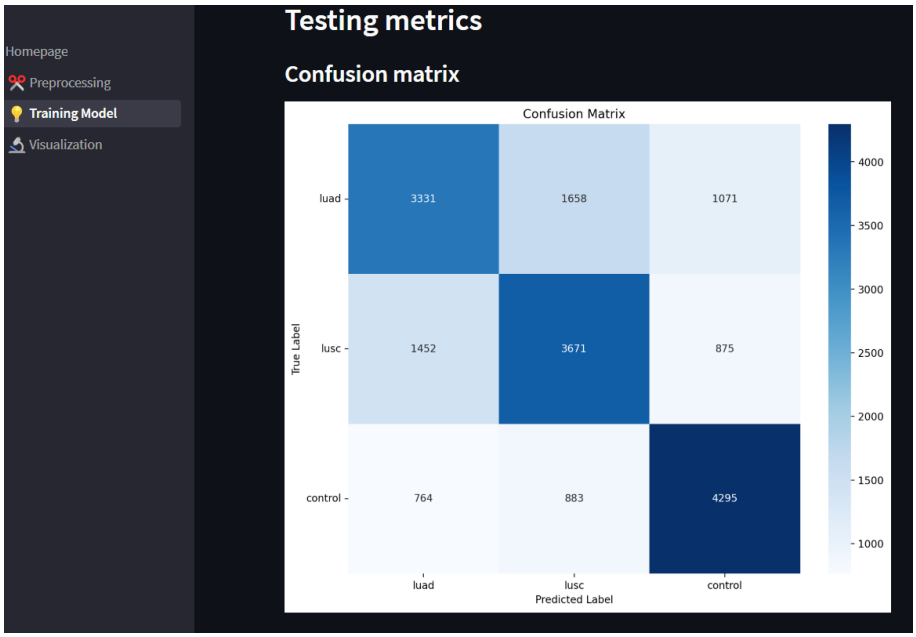


Figure 4.16: Confusion Matrix for the lung cancer experiment showed in *FPathai*

- Referring to the **LUAD** label, the model correctly identifies 3331 patches but misclassifies 1658 as LUSC and 1071 as control.
- Referring to the **LUSC** label, the model correctly identifies 3671 patches but misclassifies 1452 as LUAD and 875 as control.
- Referring to the **control** label, the model correctly identifies 4295 patches but misclassifies 764 as LUAD and 883 as LUSC.

It is important to clarify that even though we have a total of 18000 patches to test, it does not necessarily mean there are exactly 6000 for control and 10,000 for tumors. This is due to the random selection process, and in this specific scenario, we have a total of 6060 patches for LUAD, 5998 for LUSC and 5942 patches for control.

This results suggests that the model frequently exhibits confusion when differentiating between various types of tumors. Nevertheless, it demonstrates proficiency in accurately discerning between different tumor types when compared to control patches.

After scrutinizing the confusion matrix, we can glean valuable insights by examining its information across various thresholds through the creation of a ROC curve, depicted in Figure 4.17.

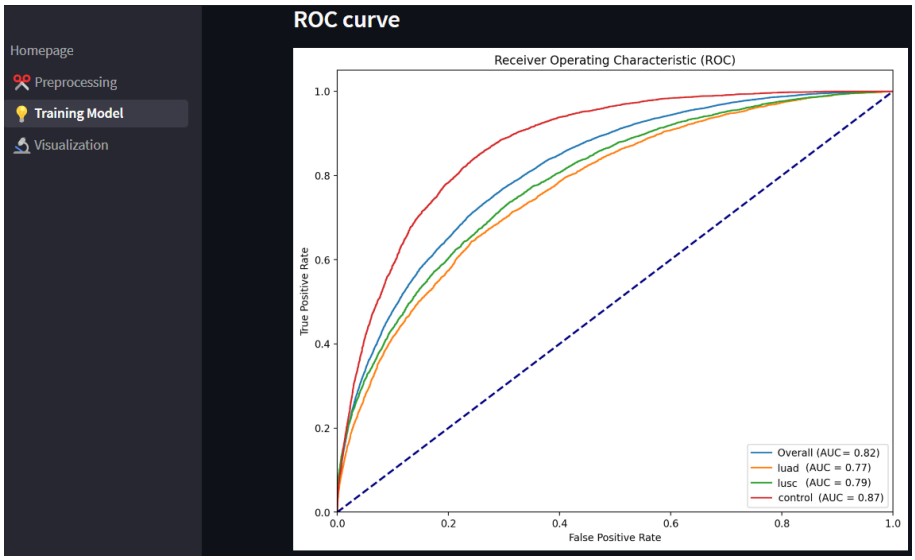


Figure 4.17: ROC for the lung cancer model showed in *FPathai*

This graph holds significant interest as it presents a One Versus All analysis for each label, accompanied by the overall AUC for the global model, as depicted in the Figure 4.18. Notably, when we focus solely on the comparison of control versus cancer data (combining LUSC and LUAD), the model performs well enough with an AUC of 0.87. However, when we dissect the analysis into individual assessments of LUSC and LUAD, the AUC values deteriorate significantly, reaching 0.79 and 0.77, respectively.

Furthermore, *FPathai* provides comprehensive metrics, including loss, accuracy, F1-Score, Precision, and Recall, evaluated on the test dataset, as visually represented in Figure 4.18, complemented by the explained AUC score.

Considering these metrics, it becomes evident that the model has struggled to achieve significant generalization. This limitation can be attributed to the relatively small dataset used and the inherent simplicity of the model. However, it is crucial to bear in mind that this is a classification problem involving three variables, and achieving values near 0.63 is distinct from a binary case, where such a result would indicate nearly random classification. In this scenario, a random classification would yield values approximately around 0.33. Therefore, although these results may be considered modest, they remain acceptable given the constraints explained.

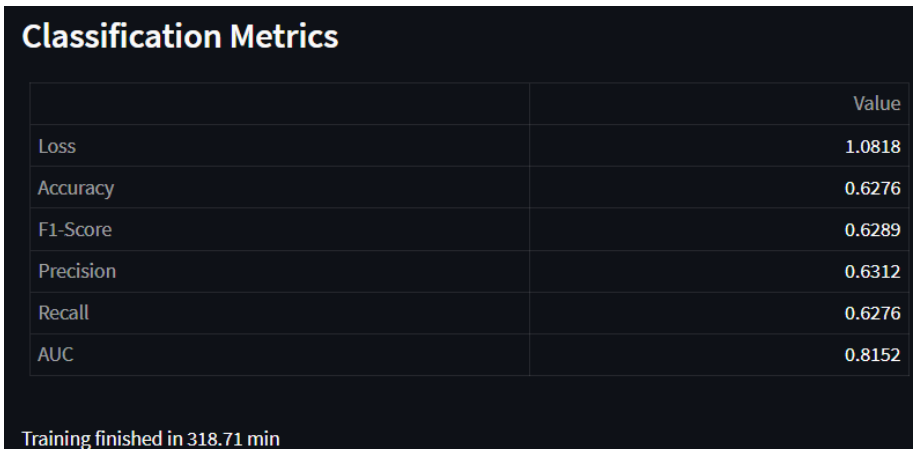


Figure 4.18: Testing metrics for the lung cancer model shown in *FPathai*

### 4.2.3 Visualization

The rationale behind its significance mirrors the explanation provided in the lung section's visualization. Once we have input the directory path for the trained model and the directory path for the .h5 file containing the patches to be analyzed, the patches were presented alongside their respective GRAD-CAM visualizations.

We will commence by demonstrating the display of patches generated from a **LUAD** tumor image as shown in 4.19. In this scenario, out of the three LUAD patches examined, one is correctly classified while the other two are misclassified as LUSC. This outcome aligns with the expectations derived from the analysis conducted using the confusion matrix. Notably, the neural network appears to place emphasis on the blue regions, corresponding to the nucleic acids within cells, much like the observation made in the lung section. The fact that the activation occurs within these bluish regions suggests that the neural network might be grasping what it should emphasize.

In the Figure 4.20, we present three correctly classified **LUSC** patches in an attempt to gain insight into the reasons behind their accurate classification. Interestingly, the red circle areas does not appear to correspond to discernible structures visible to the naked eye. We can suggest, though not with absolute certainty, that the model might be concentrating on the bluer regions once more; in other words, it is directing its attention towards the cell nuclei again.

In the Figure 4.21, we observe three **control** patches accurately classified as "control" with a 100% prediction accuracy. What stands out is that, in this particular case, the neural network seems to focus on the white regions within the pink tissue. These pale or colorless regions typically represent the background, devoid of any distinct coloring. While this deduction may

currently lack firm empirical validation, it may hold potential utility for pathologists. The predictions in this instance consistently exhibit either a 100% or 0% accuracy rate. This pattern suggests a potential issue of overfitting, wherein the model may have essentially memorized the patch structures rather than truly learning their underlying patterns.

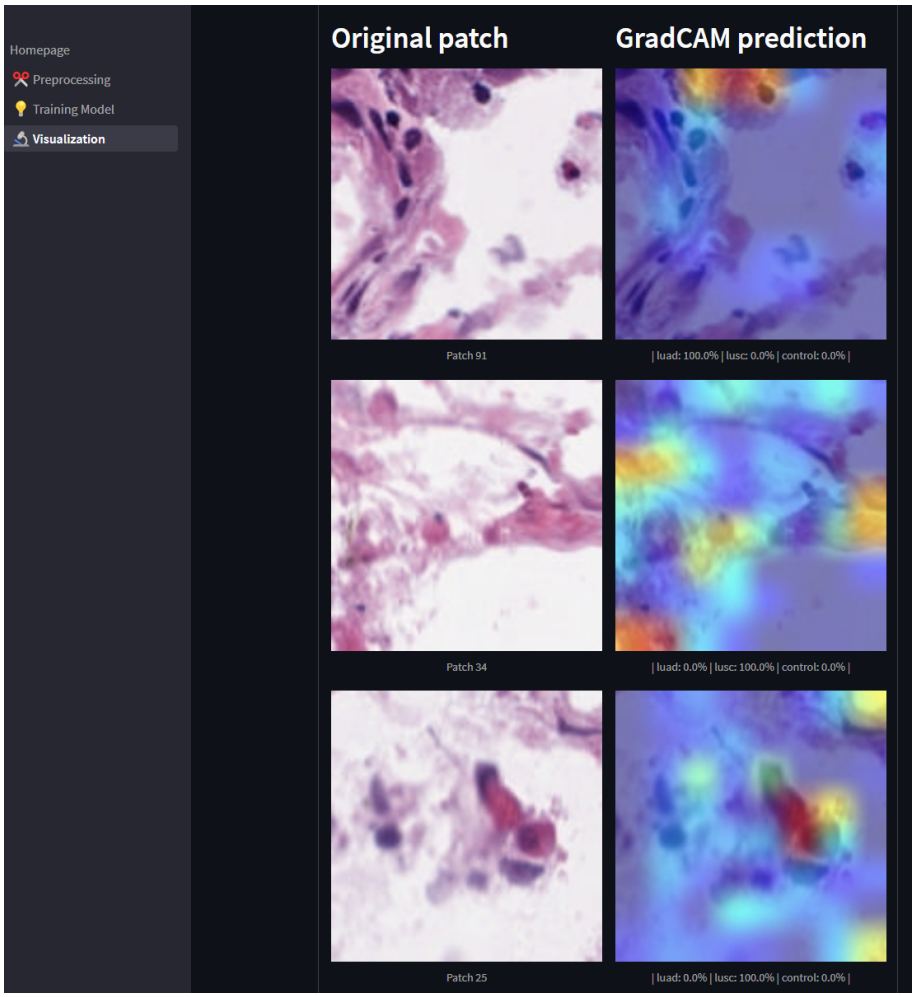


Figure 4.19: Visualization of GradCAM for patches of a LUAD patient

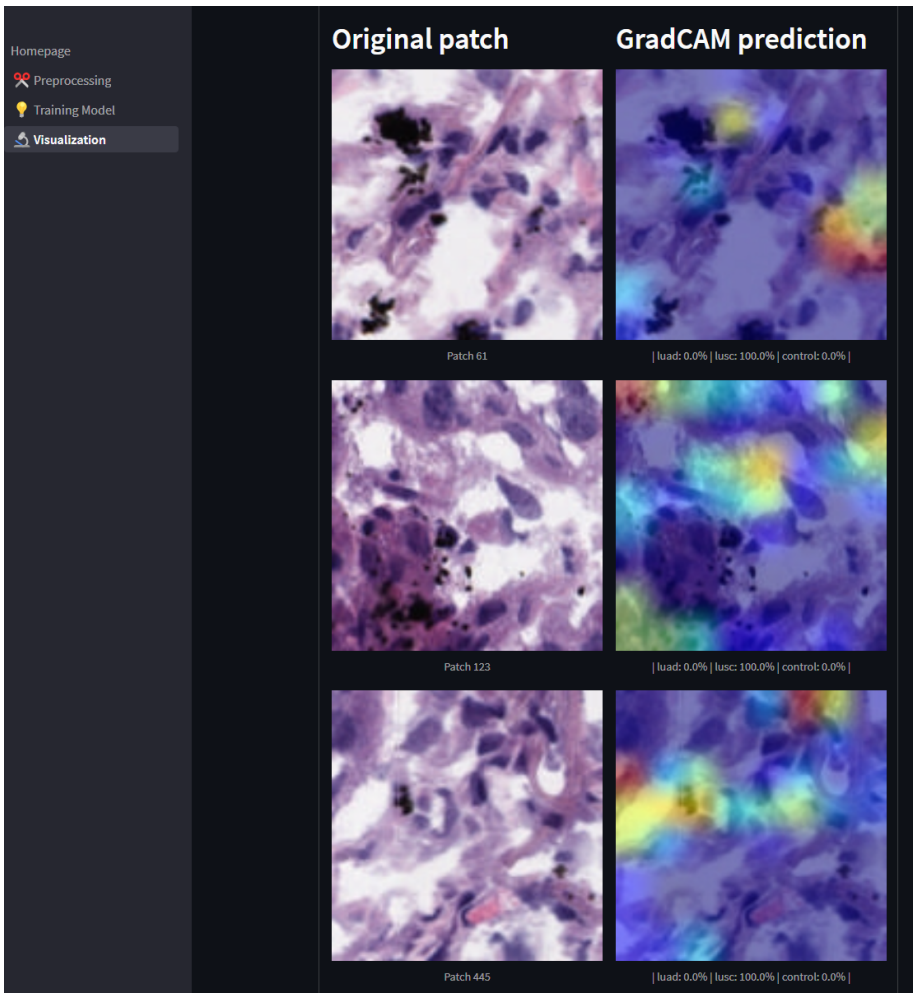


Figure 4.20: Visualization of GradCAM for patches of a LUSC patient

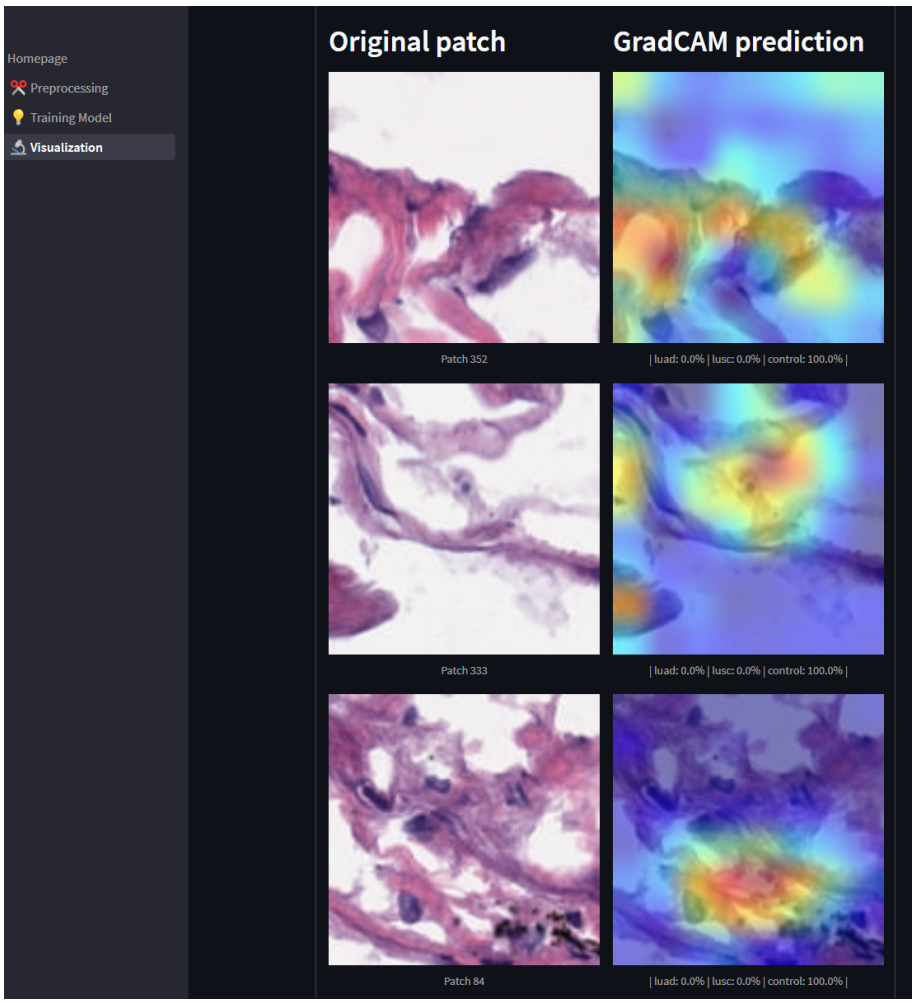


Figure 4.21: Visualization of GradCAM for patches of a lung control patient

# Chapter 5

# Conclusions and future work

## 5.1 Conclusions

One of the fundamental principles that guided the development of this tool was its potential to serve as a stepping stone for individuals aspiring to enter the domain of digital pathology. The intention was not only to create an effective and user-friendly application but also to craft a comprehensive master's thesis that could serve as a foundational resource for those seeking to explore this dynamic field.

Throughout the methodology and results sections, we have systematically elucidated the conceptualization and development process of *FPathai*. We've also delved into the challenges we encountered during the application's creation, while providing illustrative screenshots that guide users on its usage. These demonstrations were performed through two distinct experiments, one focused on breast cancer, a binary classification and the other on lung cancer, a multilabel classification.

Through the use these screenshots, we have successfully confirmed the application's functionality. The preprocessing component efficiently and accurately generates patches from WSI, saving them in a .h5 format and generating a CSV file that will be utilized during the model training phase. Our model training process has also undergone rigorous testing and proven to be effective. While the results obtained could be improved, it is noteworthy that we achieved our objectives without needing to use a single line of code within the application. Furthermore, the Grad-CAM visualization is functioning as intended. It is worth noting that in certain patches, the activation results may appear less meaningful. This can be attributed to our training process, which was conducted with limited data and relied on a nearly frozen transfer learning model.

Undeniably, AI has become an integral part of our daily existence. Despite its complexity and novelty, it is incumbent upon us as scientists to ensure its accessibility to a wider audience. Thus, *FPathai*, consistently

prioritize user-friendliness and simplicity. In light of the remarkable strides made in the field of AI, it is imperative for clinicians to acquire proficiency in harnessing this invaluable tool. AI models have emerged as powerful aides in augmenting clinical practices, yet it is essential to emphasize that their role is **complementary rather than substitutive** to that of clinicians. Nevertheless, given the undeniable trajectory of AI influence in healthcare, there is a pressing need to equip clinicians with the knowledge and skills required to leverage these tools effectively. *FPathai* serves as a valuable resource to assist in achieving this goal.

## 5.2 Future work

Streamlit, while being a valuable tool for simplifying application development without requiring extensive knowledge of backend or frontend concepts, does have its constraints. These constraints include limitations in the way of accessing local data and the structure for presenting the entire application process within a single screen. We had to consider these limitations when building *FPathai*. Others platforms could be investigated, following more traditional web development frameworks.

In this study, we have refrained from implementing any preprocessing steps on the original images, except for the creation of corresponding patches. It would be highly intriguing to explore a research avenue where various preprocessing techniques are systematically applied and how they would improve the performance of a model trained without the same preprocessed images. This exploration could commence by **delving into the libraries outlined in Chapter** 3, conducting a comprehensive review of their functionalities, gaining a deep understanding of their mechanisms, and extracting patches from them as part of the process.

With that being said, in terms of preprocessing, as a potential improvement for this master's thesis and for future research, I have contemplated integrating **stain normalization** as a feature. Stain normalization is a technique employed to mitigate potential biases arising from batch effects, which stem from variations in staining colors and intensities across different slides[67]. There are different methods of stain normalization like Reinhard, Macenko, and Vahadane [15] that can be applied that can contribute to a more comprehensive and robust research framework. [68, 69].

*FPathai* currently supports WSI analysis exclusively in the SVS format, which is commonly found in the GDC dataset. It would be advantageous to expand its compatibility to encompass a **broader range of WSI image formats** that OpenSlide is able to open[70].

During the training process, we relied solely on a single test set, which may not accurately represent the broader dataset. To address this issue, incorporating **K-Fold cross-validation** into our workflow would be a valu-

able addition. While it's worth noting that this approach might extend the already substantial training time, it has the potential to yield models with superior generalization capabilities. Furthermore, in our pursuit of improved results, conducting a **grid search for hyperparameter tuning** is another avenue to explore. This involves systematically testing a range of hyperparameter values to discover the optimal combination. It is essential to acknowledge that this, too, can significantly prolong the training duration.

In terms of the metrics presented for multiclassification problems, *FPathai* can be completed by incorporating additional metrics for the OVA scenarios. This approach would provide a clearer understanding of how each label is classified individually. However, it is important to note that in this current work, such metrics were intentionally omitted to avoid any potential confusion for the user. The primary objective in presenting classification metrics was to convey the overall performance of the model.

Achieving a flawless interpretability of the model through the precise application of **Grad-CAM** would mark the zenith of our research efforts. This accomplishment holds paramount significance as it directly serves the primary goal of aiding pathologists. Hence, delving into a dedicated research direction aimed at crafting a model tailored to histopathological images, ensuring the utmost accuracy in its Grad-CAM outputs, stands as an exceptionally compelling and worthwhile avenue to explore.

# Bibliography


[1] Michael H Ross and Wojciech Pawlina. *Histology*. Lippincott Williams & Wilkins, 2006.

[2] Stephan W Jahn, Markus Plass, and Farid Moinfar. Digital pathology: advantages, limitations and emerging perspectives. *Journal of clinical medicine*, 9(11):3697, 2020.

[3] Andrew H Fischer, Kenneth A Jacobson, Jack Rose, and Rolf Zeller. Hematoxylin and eosin staining of tissue and cell sections. *Cold spring harbor protocols*, 2008(5):pdb–prot4986, 2008.

[4] M.D. Mikael Häggström. Main types of staining seen on he stain., 2022. `https://en.wikipedia.org/wiki/H%26E_stain#/media/File:Eosinophilic,_basophilic,_chromophobic_and_amphophilic_staining.png` [Online; accessed August 23, 2023].

[5] Jeroen Van der Laak, Geert Litjens, and Francesco Ciompi. Deep learning in histopathology: the path to the clinic. *Nature medicine*, 27(5):775–784, 2021.

[6] Richard Colling, Helen Pitman, Karin Oien, Nasir Rajpoot, Philip Macklin, CM-Path AI in Histopathology Working Group, Velicia Bachtiar, Richard Booth, Alyson Bryant, Joshua Bull, et al. Artificial intelligence in digital pathology: a roadmap to routine use in clinical practice. *The Journal of pathology*, 249(2):143–150, 2019.

[7] Navid Farahani, Anil V Parwani, and Liron Pantanowitz. Whole slide imaging in pathology: advantages, limitations, and emerging perspectives. *Pathology and Laboratory Medicine International*, pages 23–33, 2015.

[8] Mark D Zarella, Douglas Bowman, Famke Aeffner, Navid Farahani, Albert Xthona, Syeda Fatima Absar, Anil Parwani, Marilyn Bui, and Douglas J Hartman. A practical guide to whole slide imaging: a white paper from the digital pathology association. *Archives of pathology & laboratory medicine*, 143(2):222–234, 2019.

[9] NIH. Genomic data commons data portal, 2023. `https://portal.gdc.cancer.gov/` [Accessed: August 30, 2023].

[10] Metin N Gurcan, Laura E Boucheron, Ali Can, Anant Madabhushi, Nasir M Rajpoot, and Bulent Yener. Histopathological image analysis: A review. *IEEE reviews in biomedical engineering*, 2:147–171, 2009.

[11] Muhammad Khalid Khan Niazi, Anil V Parwani, and Metin N Gurcan. Digital pathology and artificial intelligence. *The lancet oncology*, 20(5):e253–e261, 2019.

[12] Francisco Carrillo-Perez, Francisco M Ortuno, Alejandro Börjesson, Ignacio Rojas, and Luis Javier Herrera. Performance comparison between multi-center histopathology datasets of a weakly-supervised deep learning model for pancreatic ductal adenocarcinoma detection. *Cancer Imaging*, 23(1):1–11, 2023.

[13] Alessia Marcolini, Nicole Bussola, Ernesto Arbitrio, Mohamed Amgad, Giuseppe Jurman, and Cesare Furlanello. histolab: A python library for reproducible digital pathology preprocessing with automated testing. *SoftwareX*, 20:101237, 2022.

[14] Peter Bankhead, Maurice B Loughrey, José A Fernández, Yvonne Dombrowski, Darragh G McArt, Philip D Dunne, Stephen McQuaid, Ronan T Gray, Liam J Murray, Helen G Coleman, et al. Qupath: Open source software for digital pathology image analysis. *Scientific reports*, 7(1):1–7, 2017.

[15] James M Dolezal, Sara Kochanny, Emma Dyer, Andrew Srisuwananukorn, Matteo Sacco, Frederick M Howard, Anran Li, Prajval Mohan, and Alexander T Pearson. Slideflow: Deep learning for digital histopathology with real-time whole-slide visualization. *arXiv preprint arXiv:2304.04142*, 2023.

[16] Adam G Berman, William R Orchard, Marcel Gehrung, and Florian Markowetz. Pathml: a unified framework for whole-slide image analysis with deep learning. *medRxiv*, pages 2021–07, 2021.

[17] James M Dolezal. Slideflow documentation, 2023. `https://slideflow.dev/overview/` [Accessed: August 30, 2023].

[18] Dana-Farber Cancer Institute and Weill Cornell Medicine. Pathml documentation, 2023. `https://pathml.readthedocs.io/en/latest/overview.html` [Accessed: August 30, 2023].

[19] Fernando Palomino Cobo. Application for histopathological image analysis, 2023. `https://github.com/FernandoPC25/TFM` [Accessed: September 6, 2023].

[20] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[21] T. Hastie, R. Tibshirani, and J.H. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer series in statistics. Springer, 2009.

[22] Yunjie He, Hong Zhao, and Stephen TC Wong. Deep learning powers cancer diagnosis in digital pathology. *Computerized Medical Imaging and Graphics*, 88:101820, 2021.

[23] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[24] Jose Fumo. A gentle introduction to neural networks series — part, 2017. https://towardsdatascience.com/a-gentle-introduction-to-neural-networks-series-part-1-2b90b87795bc [Accessed: September 1, 2023].

[25] Marvin Minsky and Seymour Papert. An introduction to computational geometry. *Cambridge tiass., HIT*, 479(480):104, 1969.

[26] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[27] IBM. Deep neural network, 2022. https://www.ibm.com/content/dam/connectedassets-adobe-cms/worldwide-content/cdp/cf/ul/g/3a/b8/ICLH_Diagram_Batch_01_03-DeepNeuralNetwork.png [Accessed: August 31, 2023].

[28] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[29] Kamil Krzyk. Coding deep learning for beginners — linear regression (part 3): Training with gradient descent, 2018. https://towardsdatascience.com/coding-deep-learning-for-beginners-linear-regression-gradient-descent-fcd5e0fc077d [Accessed: September 1, 2023].

[30] Ning Qian. On the momentum term in gradient descent learning algorithms. *Neural networks*, 12(1):145–151, 1999.

[31] Agnes Lydia and Sagayaraj Francis. Adagrad—an optimizer for stochastic gradient descent. *Int. J. Inf. Comput. Sci*, 6(5):566–568, 2019.

[32] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[33] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323. JMLR Workshop and Conference Proceedings, 2011.

[34] Sagar Sharma. Activation functions in neural networks, 2017. `https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6` [Accessed: September 1, 2023].

[35] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr, 2015.

[36] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[37] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[38] Md Zahangir Alom, Tarek M Taha, Christopher Yakopcic, Stefan Westberg, Paheding Sidike, Mst Shamima Nasrin, Brian C Van Esesn, Abdul A S Awwal, and Vijayan K Asari. The history began from alexnet: A comprehensive survey on deep learning approaches. *arXiv preprint arXiv:1803.01164*, 2018.

[39] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[40] Arden Dertat. Towards data science. applied deep learning part 4., 2017. `https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2` [Accessed: August 31, 2023].

[41] Firelord Phoenix. Pictorial example of max-pooling, 2018. `https://computersciencewiki.org/images/8/8a/MaxpoolSample2.png` [Accessed: August 30, 2023].

[42] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

[43] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[44] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[45] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[46] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[47] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[48] Andrej Karpathy. t-sne visualization of cnn codes, 2012. `https://cs.stanford.edu/people/karpathy/cnnembed/cnn_embed_full_1k.jpg` [Accessed: August 30, 2023].

[49] Standford. Deep learning lectures, 2017. `http://cs231n.stanford.edu/slides/2017/` [Accessed: September 1, 2023].

[50] Research Gate. The architecture of vgg16, 2021. `https://miro.medium.com/v2/resize:fit:720/format:webp/1*NNifzsJ7tD2kAfBXt3AzEg.png` [Accessed: September 1, 2023].

[51] Research Gate. The architecture of the mobilenetv2 network., 2021. `https://www.researchgate.net/figure/The-architecture-of-the-MobileNetv2-network_fig3_342856036` [Accessed: September 1, 2023].

[52] Gorlapraveen123. Resnet50, 2021. `https://upload.wikimedia.org/wikipedia/commons/9/98/ResNet50.png` [Accessed: September 1, 2023].

[53] Papers with code. Inception-v3, 2015. `https://production-media.paperswithcode.com/methods/inceptionv3onc--oview_vjAbOfw.png` [Accessed: September 1, 2023].

[54] NCICCGenomics@mail.nih.gov. The cancer genome atlas program (tcga), 2023. `https://www.cancer.gov/ccg/research/genome-sequencing/tcga` [Accessed: September 5, 2023].

[55] David A Gutman, Jake Cobb, Dhananjaya Somanna, Yuna Park, Fusheng Wang, Tahsin Kurc, Joel H Saltz, Daniel J Brat, Lee AD Cooper, and Jun Kong. Cancer digital slide archive: an informatics resource to support integrated in silico analysis of tcga pathology data. *Journal of the American Medical Informatics Association*, 20(6):1091–1098, 2013.

[56] Walter C Bell, Katherine C Sexton, and William E Grizzle. How to efficiently obtain human tissues to support specific biomedical research projects. *Cancer Epidemiology Biomarkers & Prevention*, 18(6):1676–1679, 2009.

[57] Michael Greger. Comer para no morir. *Editorial Paidós*, 2018.

[58] Jamie DePolo. Invasive ductal carcinoma (idc), 2023. `https://www.breastcancer.org/types/invasive-ductal-carcinoma` [Accessed: September 4, 2023].

[59] Nicolas Coudray, Paolo Santiago Ocampo, Theodore Sakellaropoulos, Navneet Narula, Matija Snuderl, David Fenyö, Andre L Moreira, Narges Razavian, and Aristotelis Tsirigos. Classification and mutation prediction from non–small cell lung cancer histopathology images using deep learning. *Nature medicine*, 24(10):1559–1567, 2018.

[60] Dorota Anusewicz, Magdalena Orzechowska, and Andrzej K Bednarek. Lung squamous cell carcinoma and lung adenocarcinoma differential gene expression regulation through pathways of notch, hedgehog, wnt, and erbb signalling. *Scientific reports*, 10(1):21128, 2020.

[61] Nobuyuki Otsu. A threshold selection method from gray-level histograms. *IEEE transactions on systems, man, and cybernetics*, 9(1):62–66, 1979.

[62] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu,

and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[63] Muhammad Umair, Muhammad Shahbaz Khan, Fawad Ahmed, Fatmah Baothman, Fehaid Alqahtani, Muhammad Alian, and Jawad Ahmad. Detection of covid-19 using transfer learning and grad-cam visualization on indigenously collected x-ray dataset. *Sensors*, 21(17):5813, 2021.

[64] Wikipedia. Receiver operating characteristic, 2023. `https://upload.wikimedia.org/wikipedia/commons/1/13/Roc_curve.svg` [Accessed: September 5, 2023].

[65] Ming Y Lu, Tiffany Y Chen, Drew FK Williamson, Melissa Zhao, Maha Shady, Jana Lipkova, and Faisal Mahmood. Ai-based pathology predicts origins for cancers of unknown primary. *Nature*, 594(7861):106–110, 2021.

[66] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

[67] Frederick M Howard, James Dolezal, Sara Kochanny, Jefree Schulte, Heather Chen, Lara Heij, Dezheng Huo, Rita Nanda, Olufunmilayo I Olopade, Jakob N Kather, et al. The impact of site-specific digital histology signatures on deep learning model accuracy and bias. *Nature communications*, 12(1):4423, 2021.

[68] Marc Macenko, Marc Niethammer, James S Marron, David Borland, John T Woosley, Xiaojun Guan, Charles Schmitt, and Nancy E Thomas. A method for normalizing histology slides for quantitative analysis. In *2009 IEEE international symposium on biomedical imaging: from nano to macro*, pages 1107–1110. IEEE, 2009.

[69] Abhishek Vahadane, Tingying Peng, Amit Sethi, Shadi Albarqouni, Lichao Wang, Maximilian Baust, Katja Steiger, Anna Melissa Schlitter, Irene Esposito, and Nassir Navab. Structure-preserving color normalization and sparse stain separation for histological images. *IEEE transactions on medical imaging*, 35(8):1962–1971, 2016.

[70] Adam Goode, Benjamin Gilbert, Jan Harkes, Drazen Jukic, and Mahadev Satyanarayanan. Openslide: A vendor-neutral software foundation for digital pathology. *Journal of pathology informatics*, 4:27, 09 2013.