PREDICTING WILDFIRES WITH CONVOLUTIONAL NEURAL NETWORKS (CNNS)

A THESIS SUBMITTED TO

THE FACULTY OF ACHITECTURE AND ENGINEERING

OF EPOKA UNIVERSITY

BY

JUNA FINDIKU

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR BACHELOR DEGREE

IN COMPUTER ENGINEERING

JUNE, 2024

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name:

Signature:

ABSTRACT

PREDICTING WILDFIRES WITH CONVOLUTIONAL NEURAL NETWORKS (CNNS)

Findiku, Juna

Department of Computer Engineering

Supervisor: M.Sc. Sabrina Begaj

Wildfires are considered a great danger to ecosystems, wildlife as well as human lives and infrastructure. This is the reason why the early detection of wildfires as well as their prediction is a crucial factor that may contribute to the area of effective wildfire management systems and their continuous improvement. The use of AI tools has modernized similar areas and fortunately, it has also made the prediction of such phenomena much easier, especially by using Convolutional Neural Networks (CNNs), which are well known for their powerful tools for extracting complex patterns from images, including here satellite images or weather data (mostly an area concerning weather prediction or climate prediction agencies). By deeply analyzing a variety of wildfire risk factors such as vegetation density combined with parameters such as temperature, humidity and wind speed, NN models can be adjusted for obtaining useful information about the detection of areas that possess a particularly high level of risk for outbreak of severe wildfires. In order to make the models more powerful and accurate, the training of CNNs on historical wildfire data and environmental factors is highly recommended, because in this way, the models can clearly identify concrete patterns that may indicate potential wildfire outbreaks.

ABSTRAKT

PARASHIKIMI I VATRAVE TË ZJARRIT DUKE PËRDORUR CONVOLUTIONAL NEURAL NETWORKS (CNN)

Findiku, Juna

Departamenti I Inxhinjerisë Kompjuterike

Supervizore: M.Sc. Sabrina Begaj

Zjarret konsiderohen si rreziqe serioze për ekosistemet, florën dhe faunën, si edhe për vetë aktivitetet e njeriut dhe infrastrukturën. Detektimi në kohë i pikave të zjarrit dhe parashikimi i zonave me rrezikshmëri të lartë për zjarre janë faktorë kyç që mund të kontribojnë në menaxhimin efektiv të vatrave të zjarrit. Përdorimi i inteligjencës artificiale ka ndikuar në modernizimin e disa fushave dhe e ka bërë parashikimin e fenomeve natyrore më të thjeshtë. Për shembull, përdorimi i Convolutional Network, një network ky i mirënjohur për detektimin e formave komplekse në imazhe, ka dhënë rezultate premtuese në imazhet satelitore apo te dhënat e motit. Duke analizuar një sërë faktorësh që ndikojnë në zhvillimin e vatrave të zjarrit si për shembull: densiteti i vegjetacionit kombinuar me një sërë parametrash të tjerë si temperatura, lagështia dhe shpejtësia e erës, modelet e Neural Network mund të pershtaten lehtësisht në të dhëna që ndihmojnë në parashikimin e zonave me risk për përhapjen e zjarreve. Për rritjen e saktësisë së këtyre modeleve, trajnimi i CNN duke përdorur të dhëna historike rreth vatrave të zjarrit dhe kombinimi i tyre me të dhëna mjedisore është një teknikë mjaft efektive, sepse në këtë mënyrë, modeli mund të përvetësojë aftësinë e detektimit të karakteristikave të imazheve të vendeve me potencial të lartë për shpërthime të mundshme të vatrave të zjarrit.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ACT	Activation
AI	Artificial intelligence
ANN	Artificial Neural Network
BN	Batch normalization
CNN	Convolutional Neural Network
CONV	Convolutional
DL	Deep Learning
DO	Dropout
FC	Fully-connected
ML	Machine Learning
NN	Neural Network
POOL	Pooling
RELU	Rectified Linear Unit
VGG	Visual Geometry Group

CHAPTER 1

1 INTRODUCTION

Wildfire risk has increased globally due to several factors compared to the past. An efficient response to wildfire zones plays a crucial role in the wildfire management system of every country. This work introduces several models that are part of the AI algorithms used for creating and training models able to categorize images. The models represented in this document are trained by using the images of a dataset containing a considerable number of satellite images. The size of the dataset is a key point in the rigorous and complete training of the models. The NNs and especially their subcategory, CNNs, have become the pivot in the area of image processing and many models have resulted successful in making predictions. The aim of this work is the completion of three models that should be capable of predicting the chance of a fire outbreak by finally categorizing the input image into one of the two classes: wildfire or no wildfire.

The combination of satellite imagery and ML techniques has been adopted successfully in predictions regarding variations in vegetation or outcomes of natural disasters. Since many years now, the DL techniques have been widely used in image classification and in the area of wildfire prediction and the results of DL have been accurate and very promising. Many governmental agencies already use AI tools for inspecting the images received from satellite images and they have started to use these tools in fields such as: weather prediction, the study of land usage, researches on the development of urban areas, etc..

1.1 Artificial Intelligence

1.1.1 Learning in Biology

To be able to understand and further explain the concept of deep learning, it is important to look at the origin of where the concept came from, namely Biology and nature itself. Learning is a fundamental aspect of life. It is a result of experience, which means that the influence of the environment and other external factors enable changing of behavior and thought process leading in growth and innovation.

The process of leaning starts in the brain, to be specific withing neurons, which are brains cells that are activated electrically. In the brain the learning process is local and incremental and can be explaining with the term Synaptic plasticity.

Synaptic plasticity describes the ability of the neurons to change the strength of the connections between them which is directly linked to the ability to learn and improve. F.e. when repeatedly doing the same task the synaptic plasticity increases which support and is crucial for the behavioral learning and the formation of memory.

This learning behavior was the foundation of artificial intelligence and with it related subtopics such as Machine Learning.

1.1.2 Machine Learning Introduction

Machine Learning describes artificial intelligence techniques, as it uses the biological concepts of learning, that give computers the skills to learn without being explicitly programmed to do so. Deep Learning is a subset of machine learning algorithms which analyze data with a mathematical model, which roughly depict how humans would draw conclusions (see Fig. 1). (Helmholtz, 2023)

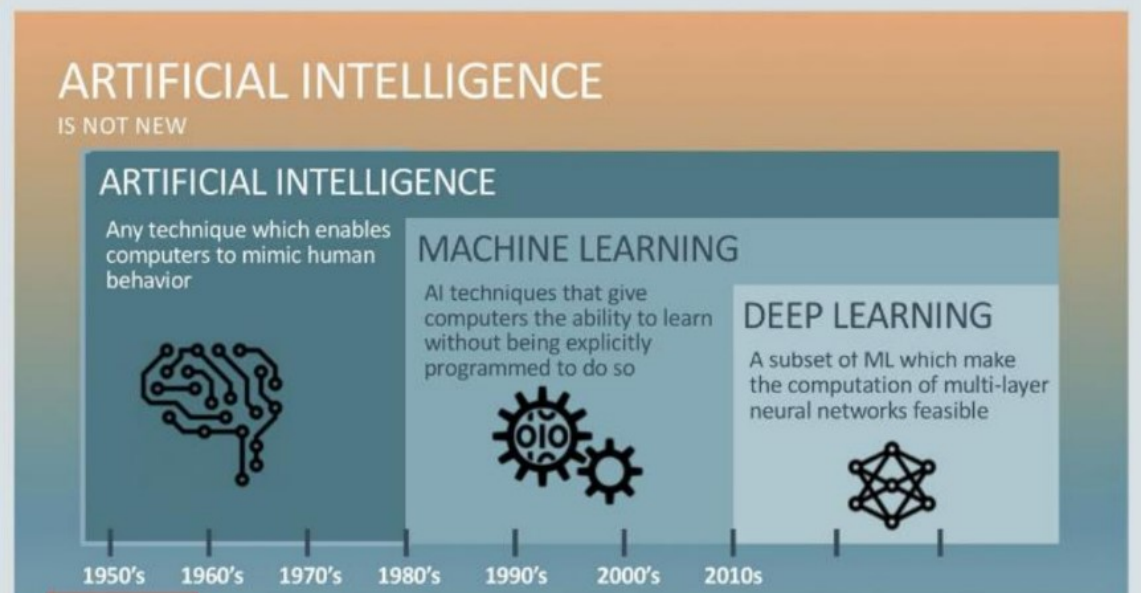

Figure 1: Artificial Intelligence and it's subtopics

A formalization model is used to identify the main components of learning and explain the differences between these learning methods.

The input is shown as set of environment states **S** and the lines represent a set of actions. The agent decides based on the current state what action **π** to take**. T** gives the transitions that change the environment state due to the currently action.

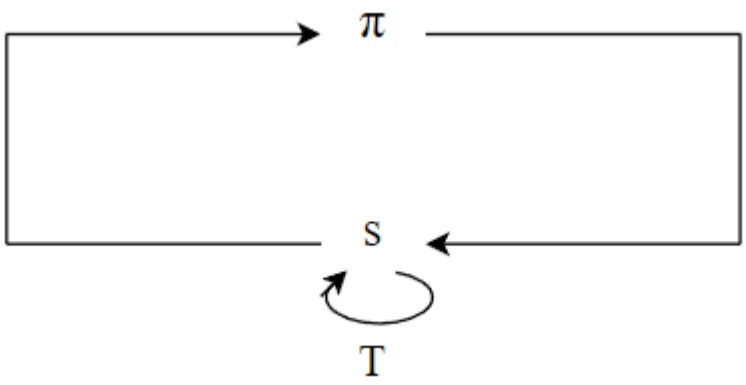

Figure 2: Formalization Model

The intelligent behavior is defined as an objective function with different parameters Θ_n. The actions a $\subset$ **A** are parametrizable functions (Neural Network) (see Fig. 2). (Brenna D. Argall, 2009)

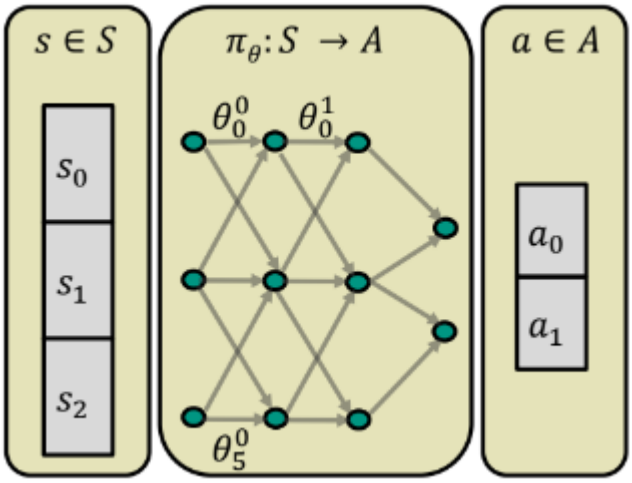

Figure 3: Neural Network model

1.1.2.1 Supervised Learning vs. Unsupervised

In case of supervised learning a labelled dataset of states and actions is available. The goal is to train the network to be able to learn a mapping $\pi_0 : S \dashrightarrow A$, which in other words mean finding the states S that lead to action A.

Referring to Fig. 4, this means that the actions are known to the model and the states π are to be defined.

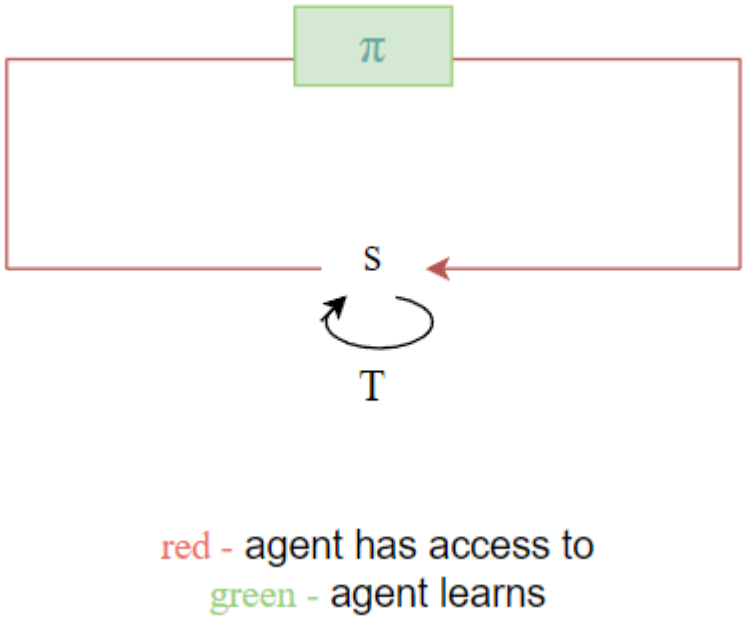

Figure 4: Supervised Learning formalized model (Helmholtz, 2023)

The supervisor is the training data itself and the agent is trained by feedback. The algorithm is reweighted everytime to allow the model learning by failure.

Unsupervised learning can be used to create models as no task is precisely defined and no objective function is available. This learning method is not part of this thesis and will not be described any further.

1.2 Basics of NNs

Neural Networks (NNs) are the main building blocks of deep learning systems, which are more complicated and introduce the concept of using several layers for optimizing

the results. In other words, NNs are simply learning algorithms, organized into layers of nodes and the layers are organized into 3 categories:

-input layer: feeds data into the network. The number of neurons in this layer is equal to the total number of features in the data.

-hidden layer: computations take place and patterns start to be identified. The hidden layer can have different numbers of neurons which are generally greater than the number of features.

The output from each layer is computed by matrix multiplication of the output of the previous layer with learnable weights of that layer and then by the addition of learnable biases followed by an activation function. The weights are decided probabilistically.

-output layer: enables the retrieval of the outcome.

This architecture is similar in essence to a labeled and directed graph structure, where each node is responsible for certain computations and is assigned a task. The connections of the NN transfer a signal from one node to another, labeled by a weight, that shows the extent to which the signal is amplified or diminished. Large, positive weights are an indicator that the signal is very important when making the classification and negative weights of the nodes are a sign that the output of the node is less important in the final classification. In a NN, the connection weights are modifiable using a learning algorithm.

NNs, otherwise known as Artificial Neural Networks (ANNs), try to mimic the neural connections in the nervous system of humans and its neurons are the main inspiration of NNs. A neuron firing is a binary operation, meaning that the results expected from the nodes after they are done with the respective computations are 0 or 1. The neuron will only fire if the total signal received exceeds a given threshold.

Deep learning is a subset of machine learning that uses multi-layered NNs, called deep neural networks, to simulate the complex decision-making power of the human brain.

1.3 Fundamentals of CNNs

Convolutional Neural Network is an extended or modified version of NN, that is heavily used in extracting features from grid-like matrix datasets. Since images are processed as matrices in the computers, CNNs are well known for their usage in interpreting image patterns and recognizing figures. Although this task is easy or even intuitive for humans,

it is difficult for computers to recognize certain patterns in images and CNNs have been very helpful in this area.

A CNN is composed of numerous layers, including the input layer, the convolutional layer, the pooling layer, and the fully connected layer. The Convolutional layer gathers information from the input image using filters, the Pooling layer reduces computation by downsampling the image, and the final prediction is generated by the fully connected layer. The network learns the best filters via backpropagation and gradient descent. A complete CNN architecture can be referred to as covnet. A covnet is a series of layers, where each layer transforms one volume to another using a differentiable function.

During training, the CNN automatically learns the values for the filters. In the context of image classification, the CNN goes through the following steps when analyzing the images of the dataset:

1-Detects edges from raw pixel data in the first layer,

2-uses these edges to detect shapes in the second layer,

3-uses the previously detected shapes to recognize higher-level features in the highest levels of the network.

The last layer of the CNN makes use of these features to make predictions about the content of the image. The introduction of several layers, each with specific functionalities leads to some main advantages of CNN, which are local invariance and compositionality.

The concept of local invariance allows the classification of an image as containing a particular object regardless of where in the image the object appears. Local invariance is strongly linked to the usage of POOL layers, that aid in identifying regions of input volume with a high response to a particular filter.

The second benefit is compositionality. Each filter composes a local patch of lower-level features into a higher-level representation. This composition allows the network to learn more rich features deeper in the network. The general steps of the network would be to first build edges from pixels, shapes from edges and then objects from shapes. After the full implementation of the CNN architecture is done and the training phase on the dataset has started, these mechanisms should be automated. The concept of building higher-level features from lower-level ones is exactly why CNNs are powerful in computer vision.

1.3.1 Understanding convolutions

A CNN can also be considered as a NN that uses convolution. Convolution is a mathematical operation that is interpreted as the merging of two sets of information. They are one of the most critical, fundamental building-blocks in computer vision and image processing. In terms of deep learning, an image convolution is an element-wise multiplication of two matrices followed by a sum. In the case of CNN, convolution is applied to the input data to filter the information and produce a feature map. The filter is known as kernel and it goes over the input image performing matrix-multiplication element after element. The kernel should stop at each (x,y) coordinate of the original image and examine the neighborhood of pixels located at the center of the image kernel. The entire neighborhood of pixels is convolved with the kernel and as a result, a single output value is attained. The result where the convolution took place will be written down to a feature map. The output value is stored in the output image at the (x,y) coordinates corresponding to the center of the kernel.

Padding and striding are two important concepts in convolution. Padding expands the input matrix by adding fake pixels to the borders of the matrix. The reason why the concept of padding comes into play is because the convolution basically reduces the size of the matrix, after the filter has gone through the image. Striding allows the network to skip some pixels and the number of pixels to be skipped depends on the value of stride. It leads to a reduction in spatial resolution and it makes the network more computationally efficient.

1.3.2 Layers of Convolutional Neural Network

There are many types of layers used to build Convolutional Neural Networks, but the ones most likely to be present in the architecture of a CNN include:

• Convolutional (CONV)

• Activation (ACT or RELU)

• Pooling (POOL)

• Fully-connected (FC)

• Batch normalization (BN)

• Dropout (DO)

Activation and dropout layers are not considered proper layers themselves, but are often part of network diagrams to make the structure explicitly clear. Pooling layers (POOL),

of equal importance as CONV and FC, are also included in network diagrams, as they have a substantial impact on the spatial dimensions of an image as it moves through a CNN. CONV, POOL, RELU, and FC are the most important when defining the actual network architecture.

1.3.2.1 Convolutional Layers

The CONV layer is the core building block of a CNN. The CONV layer parameters consist of a set of K learnable filters named kernels, where each filter has a width and a height, which are nearly always equal. These filters are small in terms of their spatial dimensions, but extend throughout the full depth of the volume. For inputs to CNN, the depth is the number of channels in the image. For volumes deeper in the network, the depth will be the number of filters applied in the previous layer. Every entry in the output volume is an output of a neuron that focuses on only a small region of the input. In this manner, the network learns filters that activate when they detect a specific type of feature at a given spatial location in the input volume. In lower layers of the network, filters may activate when they see edge-like or corner-like regions.

1.3.2.2 Activation Layers

Since no parameters or weights are learned within an activation layer, they are technically not considered proper layers. Sometimes, they are omitted from network architecture diagrams, as it is assumed that an activation immediately follows a convolution.

1.3.2.3 Pooling Layers

There are two methods to reduce the size of an input volume: CONV layers with a stride > 1 and POOL layers. The most common way to insert POOL layers is placing them in-between CONV layers. The main functionality of the POOL layer is to progressively reduce the spatial size of the input volume. This allows the reduction of the total number of parameters and computations involved in the network. Pooling is also effective in controlling overfitting. POOL layers operate on each of the depth slices of an input independently using either the max or average function. Max pooling is typically done in the middle of the CNN architecture to reduce spatial size, whereas average pooling is normally used as the final layer of the network, in order to avoid the usage of FC layers entirely.

1.3.2.4 Fully-connected Layers

FC layers are in the majority of cases present at the end of the network layer. As in feedforward neural networks , neurons in FC layers are fully-connected to all activations in the previous layer.

1.3.2.5 Batch Normalization

BN layers are utilized to stabilize the activations of a given input volume before passing it into the next layer in the CNN. Batch normalization has been shown to be extremely effective at reducing the number of epochs it takes to train a neural network. Batch normalization also has the additional advantage of helping stabilize training, allowing for a larger variety of learning rates and regularization strengths. BN is normally placed before the activation.

1.3.2.6 Dropout

Dropout is a regularization technique that prevents overfitting and also leads to an increase in testing accuracy. In many cases, the first batch of training samples influences the learning in a disproportionately high manner. This would prevent the model to continue to learn in a continuous manner the features of the dataset that appear in later samples or batches. Dropout can also be seen as a technique that allows the indirect training of the model with different architectures in parallel. Dropout simulates a sparse activation from a given layer, which in turn, encourages the network to learn a sparse representation as a side-effect. As such, it may be used as an alternative to activity regularization for uplifting sparse representations.

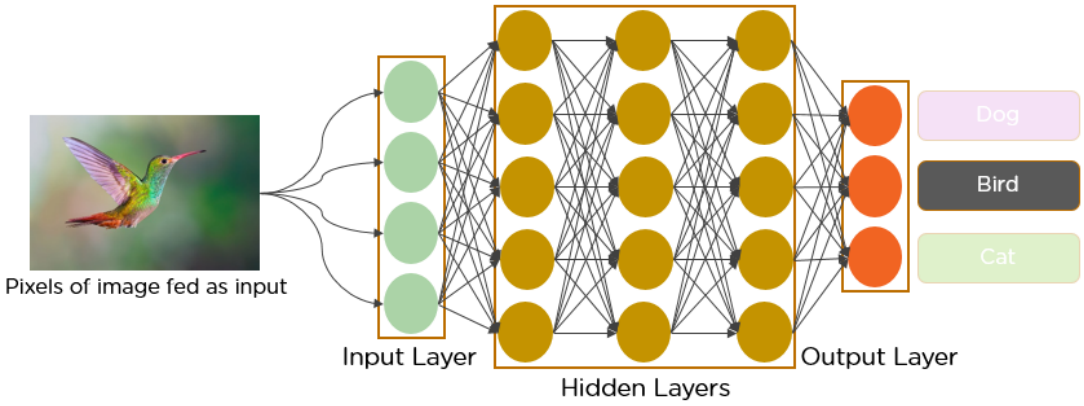

Figure 5:Typical convolutional neural network

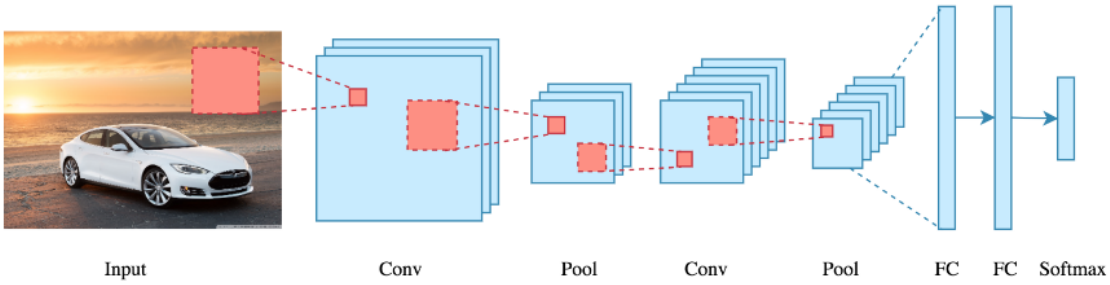

Figure 6: Layers of a convolutional neural network

1.4 Description of the dataset and general reviews about NNS and CNNs

1.4.1 Dataset organization

The steps to follow for the correct implementation of a DL model regarding the selected dataset for this project are:

1- The collection of images from trustworthy sites such as governmental agencies is the main part in the initiation phase. Considering the classification task of this thesis, the selected dataset should be a combination between images containing fire and images containing no fire. The distribution of the images of these two categories is also important, because a significant disproportionality between these two sets of images can directly lead to a biased training. Automatically, there would be an anomaly in the learning phase and this problem would become apparent in testing phase.

2- The next phase is linked to data preprocessing activities. The careful filtering of the input is always recommended before any further advancement in the model adjustments. The chosen dataset for this project is already filtered and most importantly, labeled. The training performed on unlabeled data is generally not a good practice and the results obtained from such practice end up in most of the times in unsatisfactory results, with very low accuracy values and numerous mistakes in final predictions.

3- In the end, after the operations mentioned above are completed, the loading of the filtered dataset should be done and the implementation of the DL architecture can start.

1.4.2 Characteristics of the dataset

The dataset selected for this project consists of satellite images with a resolution of 350 x 350 pixels, categorized into two classes: Wildfire and No Wildfire. The dataset contains a total of 42,850 images, with 22,710 images belonging to the Wildfire class and 20,140 images belonging to the No Wildfire class. To facilitate model training and evaluation, the dataset has been divided into three subsets: training, testing, and validation. The distribution of images among these subsets is as follows:

- **Training Set:** Approximately 70% of the dataset, used for training the model.
- **Testing Set:** Approximately 15% of the dataset, used for evaluating the model's performance.
- **Validation Set:** Approximately 15% of the dataset, used for fine-tuning hyperparameters.

Dividing the dataset in this way leads to a more accurate model that could make better predictions and classifications. It was created on this basis by the Canadian state and the link is attached in the references of this document. The original aim of the creation of such an image collection was initiated by the Canadian state to improve knowledge about fire regimes and to meet the designated needs of the available management plans following forest fires. The forest fire map that was used as the main reference when creating the dataset shows forest fires that occurred mostly in the territory of southern Quebec. The dataset and its deep analysis may also be considered eligible for research needs such as analyzing the impact of climate change, identifying possible patterns and modeling post-fire regeneration.

1.4.3 Dataset organization

The steps to follow for the correct implementation of a DL model regarding the selected dataset for this project are:

1- The collection of images from trustworthy sites such as governmental agencies is the main part in the initiation phase. Considering the classification task of this thesis, the selected dataset should be a combination between images containing fire and images containing no fire. The distribution of the images of these two categories is also important, because a significant disproportionality between these two sets of images can directly lead to a biased training. Automatically, there would be an anomaly in the learning phase and this problem would become apparent in testing phase.

2- The next phase is linked to data preprocessing activities. The careful filtering of the input is always recommended before any further advancement in the model adjustments. The chosen dataset for this project is already filtered and most importantly, labeled. The training performed on unlabeled data is generally not a good practice and the results obtained from such practice end up in most of the times in unsatisfactory results, with very low accuracy values and numerous mistakes in final predictions.

3- In the end, after the operations mentioned above are completed, the loading of the filtered dataset should be done and the implementation of the DL architecture can start.

CHAPTER 2

2 MATERIALS AND METHODS

2.1 Materials

The models will be trained and tested in Google Colab, after importing the libraries needed for loading and labeling the dataset, building the network architecture, setting the parameters for learning and afterwards, performing tests on the test dataset. Due to the probabilistic nature of the weights calculation and initialization in the training process of the model, some level of fluctuations in the results is expected. NN models make use of an optimization algorithm named stochastic gradient descent that updates the network weights incrementally in order to minimize the loss function. The final purpose of this mechanism is to find the right path that results in a set of weights that leads to the building of a useful model that makes accurate predictions. The dataset is loaded by using the API command Kaggle provides. The dataset was constructed by photos published from governmental agencies and the images had been already used for study purposes. During the implementation of the network, the following steps were followed:

1-Loading the dataset,

2- importing the libraries,

3-examining if the dataset was loaded successfully (Considering the fact that the dataset used contains a considerable number of images and errors in runtime are possible due to several reasons, it is highly recommendable to check if the dataset was loaded by either printing an image or a message after the steps above are executed.),

4-implementing the network chosen,

5-performing the training process and monitoring parameters such as: accuracy, validation accuracy, loss and validation loss continuously,

6-saving the model and reflecting on how the parameters vary from epoch to epoch in a graph that will be used later for analytical purposes,

7-testing the model on a validation dataset, that may provide valuable insights about possible ways to maximize the accuracy and minimize the mistakes the constructed model makes in the train and test data.

Considering the selected dataset for this classification task, the fact that the images, which are at the same time the input data, have all the same size and the quality of the photos is relatively good, the data is automatically considered preprocessed. After loading the dataset, it is also important to check if it was uploaded in the working environment correctly, so that the structure of the separation train-test-validation remains unchanged. The fact that all the images are organized in subfolders within the 3 folders (test, train and validation) is another important remark. By making use of Image Generator, the manual labeling of the images becomes unnecessary, because Image Generator has the capability to automatically label all the images according to their respective folder.

2.1.1 VGG-16

VGG-16 falls into the category of CNNs and it is considered as a very powerful architecture. It does not have a particularly large number of hyper parameters compared to more complex architectures, but it mainly focuses on having convolution layers of 3 x 3 filter with a stride 1 and it always contains a max pool layer of 2 x 2 filter of stride 2. It follows this arrangement of convolution and maxpool layers in a continuous manner throughout the entire architecture. In the end, it has 2 FC layers followed by a softmax layer for the output. The 16 in VGG-16 refers to the fact that it includes 16 layers that have weights.

The implementation of VGG-16 was achieved by utilizing a sequential model and the addition of layers was performed afterwards. As a preventative measure against the passing of negative values to the following layers, Rectified Linear Unit activation was used. In the end, the data was passed to the dense layer. Since the final goal consists of deciding in which of the 2 classes (wildfire or no wildfire) the image belongs to, there are only 2 dense unit layers. The softmax layer releases the final verdict about the categorization of the image. It is always advisable to also make a graphical representation of the network, which in this case was executed by using the command model.summary().

The next steps consist of compiling the model and passing data to the training model. There are certain parameters such as: the number of steps per epoch or the number of epochs that make a big difference in the overall performance of the model. Throughout every epoch, accuracy, validation accuracy, loss and validation loss were monitored. In

every epoch, as the model learns from the train dataset, these parameters will fluctuate, but in the end, only the best-performing version will be saved in Google Drive and will be further tested on the validation subfolder. After training finished and the model was tested on the test subfolder, a graph reflecting all the variations in accuracy, validation accuracy, loss and validation loss helps in deeper understanding of the performance of the model throughout the entire set of epochs. The training process will be conducted in 6 epochs and in each epoch, there will be a hundred phases. In the end, after the iteration of the 6th epoch has finished, several reruns of the training command are highly recommended, in order to choose the most accurate version between several ones. Loss and accuracy are the parameters that give insight on the performance of the model based on the training dataset and validation loss and validation accuracy are the parameters used to monitor the performance of the model based on the results obtained from test dataset.

Normally, it is expected to see an increase in validation accuracy and a decrease in validation loss, because as the training proceeds, the model must have gathered more information and enough data to identify the necessary patterns in the images that make the distinction between “wildfire” and “nowildfire”. In the early phases of training, a certain amount of undefitting is normal, because the model has still not gathered enough data to reach a good level of accuracy. As the training proceeds, accuracy normally increases, but in later phases, it may be accompanied by a continuous decrease in validation accuracy, as a result of overfitting. In many cases, the model learns in detail the characteristics of the train dataset and cannot generalize enough to make accurate predictions on previously unseen images. If the final model cannot adjust to photos of the same nature (in this case, satellite images showing territory containing or not wildfire), the usability of the model in real-life scenarios would be questioned.

```
Epoch 1/6
100/100 [==============================] - 90s 900ms/step - loss: 0.2412 - accuracy: 0.9081 - val_loss: 0.2453 - val_accuracy: 0.9250
Epoch 2/6
100/100 [==============================] - 89s 884ms/step - loss: 0.2389 - accuracy: 0.9078 - val_loss: 0.2219 - val_accuracy: 0.9156
Epoch 3/6
100/100 [==============================] - 88s 883ms/step - loss: 0.2206 - accuracy: 0.9112 - val_loss: 0.2042 - val_accuracy: 0.9250
Epoch 4/6
100/100 [==============================] - 88s 879ms/step - loss: 0.2668 - accuracy: 0.8984 - val_loss: 0.1812 - val_accuracy: 0.9312
Epoch 5/6
100/100 [==============================] - 90s 902ms/step - loss: 0.2408 - accuracy: 0.9062 - val_loss: 0.1803 - val_accuracy: 0.9375
Epoch 6/6
100/100 [==============================] - 88s 877ms/step - loss: 0.2120 - accuracy: 0.9212 - val_loss: 0.1661 - val_accuracy: 0.9312
```

Figure 7: Results of VGG-16 model with number of steps per epoch set to 100

After this first experiment, the code was rerun with a decreased number of steps by 50%. Since the number of epochs is six and it is considered a very small one for the size of the

dataset used for running this experiment, the previous number of steps equal to 100 epochs is more suitable, but observing the results after reducing it to 50 can be a good idea in order to prove that the previous choice was an optimal one for these conditions. Having only 6 epochs and 50 steps per epoch may not be practical and can lead to sudden increases or decreases as the model learns, because it is not a suitable data distribution overall.

```
Epoch 1/6
50/50 [==============================] - 91s 856ms/step - loss: 48.0259 - accuracy: 0.5263 - val_loss: 0.6888 - val_accuracy: 0.8719
Epoch 2/6
50/50 [==============================] - 46s 919ms/step - loss: 0.6850 - accuracy: 0.5631 - val_loss: 0.6124 - val_accuracy: 0.8969
Epoch 3/6
50/50 [==============================] - 43s 858ms/step - loss: 0.7861 - accuracy: 0.5719 - val_loss: 0.6842 - val_accuracy: 0.5531
Epoch 4/6
50/50 [==============================] - 44s 870ms/step - loss: 0.7021 - accuracy: 0.6025 - val_loss: 0.6748 - val_accuracy: 0.4563
Epoch 5/6
50/50 [==============================] - 44s 876ms/step - loss: 0.5487 - accuracy: 0.7294 - val_loss: 0.4098 - val_accuracy: 0.8531
Epoch 6/6
50/50 [==============================] - 44s 885ms/step - loss: 0.3651 - accuracy: 0.8544 - val_loss: 0.3583 - val_accuracy: 0.8438
```

Figure 2. Results of VGG-16 model with number of steps per epoch set to 50

As illustrated above, there are some abrupt changes from one epoch to another such as the sudden reduction of the validation accuracy from the second epoch to the third one, the reduction of the validation loss from the fourth epoch to the fifth one or the gigantic increase of validation accuracy from the fourth epoch to the fifth one. Because of these observations, only the first model was saved and it will be analyzed in more detail in later sections.

2.1.2 AlexNet

AlexNet is another type of CNN, that has proven to be successful and has reached satisfactory results in many cases. It achieves speedy results, mainly due to one of the main characteristics of AlexNet, that is the usage of ReLu activation function. ReLu-based deep convolutional networks are trained several times faster than tanh and sigmoid- based networks. The input size for AlexNet is generally 227 x 227 x 3. The architecture of this CNN consists of five convolutional layers, three max-pooling layers, two normalization layers, two fully connected layers and one softmax layer. Each convolutional layer contains a ReLu function and convolutional filters as well.

In the training process, the number of epochs as well as the number of steps per epoch were kept unchanged as in the first model using VGG-16 network, in order to be able to undergo an analysis and comparison between the models and to understand which one performs better. At this point, the preliminary prediction regarding the end results would be that VGG-16 will perform at least slightly better, mainly due to the fact that in the training process, it incorporates more convolutional layers compared to AlexNet. In the

case of AlexNet, since it is mostly used with smaller images, the clarification of the input size of the image is crucial for maximally avoiding any kind of error or mismatch in later phases of learning and testing.

```
100/100 [==============================] - 15s 49ms/step - loss: 0.9451 - accuracy: 0.7837 - val_loss: 3.8675 - val_accuracy: 0.6438
Epoch 2/6
100/100 [==============================] - 4s 41ms/step - loss: 0.5757 - accuracy: 0.8087 - val_loss: 0.4607 - val_accuracy: 0.8750
Epoch 3/6
100/100 [==============================] - 5s 55ms/step - loss: 0.5046 - accuracy: 0.8250 - val_loss: 1.1718 - val_accuracy: 0.8438
Epoch 4/6
100/100 [==============================] - 4s 39ms/step - loss: 0.5521 - accuracy: 0.8019 - val_loss: 0.5662 - val_accuracy: 0.7063
Epoch 5/6
100/100 [==============================] - 6s 58ms/step - loss: 0.4410 - accuracy: 0.8338 - val_loss: 0.9881 - val_accuracy: 0.8125
Epoch 6/6
100/100 [==============================] - 4s 44ms/step - loss: 0.4864 - accuracy: 0.8469 - val_loss: 0.5013 - val_accuracy: 0.8500
```

Figure 8: Results of AlexNet model with number of epochs set to 6

As shown in figure 3, the accuracy values for both the train and test dataset are similar and the same applies for the test dataset.

After this first trial, the same code was rerun with a higher number of epochs, which logically should lead to an improvement in validation accuracy and a lower validation loss. In the second trial, the number of epochs was set to 8. Since it is a very slight increase compared to the first version, only minimal increases are expected.

```
100/100 [==============================] - 9s 54ms/step - loss: 0.5720 - accuracy: 0.8131 - val_loss: 0.9768 - val_accuracy: 0.6250
Epoch 2/8
100/100 [==============================] - 5s 51ms/step - loss: 0.5232 - accuracy: 0.8131 - val_loss: 1.0168 - val_accuracy: 0.4500
Epoch 3/8
100/100 [==============================] - 6s 56ms/step - loss: 0.5350 - accuracy: 0.8056 - val_loss: 1.5424 - val_accuracy: 0.6250
Epoch 4/8
100/100 [==============================] - 4s 44ms/step - loss: 0.4084 - accuracy: 0.8494 - val_loss: 0.4917 - val_accuracy: 0.8813
Epoch 5/8
100/100 [==============================] - 6s 58ms/step - loss: 0.4271 - accuracy: 0.8487 - val_loss: 0.2547 - val_accuracy: 0.8938
Epoch 6/8
100/100 [==============================] - 4s 42ms/step - loss: 0.5143 - accuracy: 0.8188 - val_loss: 0.2752 - val_accuracy: 0.8875
Epoch 7/8
100/100 [==============================] - 6s 55ms/step - loss: 0.4102 - accuracy: 0.8462 - val_loss: 0.3474 - val_accuracy: 0.9000
Epoch 8/8
100/100 [==============================] - 5s 46ms/step - loss: 0.4498 - accuracy: 0.8344 - val_loss: 0.2217 - val_accuracy: 0.9125
```

Figure 9: Results of AlexNet model with number of epochs set to 8

Even though there is a distinct improvement in the testing parameters, only small changes can be noticed in the training parameters. This experiment shows the importance of the parameter of number of epochs in the success of the model. In the third section, only the first version of AlexNet model will be analyzed and compared to the other models. The reason is linked to general standards of comparisons, where all the parameters need to be the same for a more comprehensive and rigorous differentiation.

2.1.3 VGG-19

The VGG-19 architecture is identical to that of VGG-16, except for the fact that it contains 3 more convolutional layers. This makes it more complex and a deeper CNN compared to VGG-16. In most of the cases, VGG-19 has outperformed the VGG-16, but in essence, both VGG-16 and VGG-19 follow the same logic. The most significant improvement that VGGNet has represented is the reduction of the size of the convolution kernel and the increase in the number of convolution layers.

After constructing the architecture of the VGG-19 and printing the summary of it for a better understanding of all the convolutions, the training was started and it occurred in 6 epochs, each containing 100 steps, just like in the other models built in this thesis. Because VGG-19 contains even more layers than VGG-16 or AlexNet and since all CNNs are based on convolutions and the number of convolutions strongly influences the outcome of the model, the expectations are that VGG-19 will outperform the previous models.

```
Epoch 1/6
100/100 [==============================] - 177s 1s/step - loss: 36.3943 - accuracy: 0.6084 - val_loss: 0.3505 - val_accuracy: 0.8313
Epoch 2/6
100/100 [==============================] - 129s 1s/step - loss: 0.3732 - accuracy: 0.8441 - val_loss: 0.2202 - val_accuracy: 0.9125
Epoch 3/6
100/100 [==============================] - 128s 1s/step - loss: 0.3426 - accuracy: 0.8706 - val_loss: 0.3117 - val_accuracy: 0.8938
Epoch 4/6
100/100 [==============================] - 145s 1s/step - loss: 0.2721 - accuracy: 0.8965 - val_loss: 0.1775 - val_accuracy: 0.9406
Epoch 5/6
100/100 [==============================] - 129s 1s/step - loss: 0.2468 - accuracy: 0.9097 - val_loss: 0.1752 - val_accuracy: 0.9375
Epoch 6/6
100/100 [==============================] - 128s 1s/step - loss: 0.2277 - accuracy: 0.9150 - val_loss: 0.4043 - val_accuracy: 0.8500
```

Figure 10: Results of VGG-19

As shown above, the model has learned in a continuous manner and there are epochs where the parameters show a better classification compared to other iterations. In the next experiment, the learning rate was increased to 0.0015.

```
Epoch 1/6
100/100 [==============================] - 135s 1s/step - loss: 1017.1984 - accuracy: 0.4941 - val_loss: 1.0665 - val_accuracy: 0.4625
Epoch 2/6
100/100 [==============================] - 132s 1s/step - loss: 0.7424 - accuracy: 0.5291 - val_loss: 0.6832 - val_accuracy: 0.5719
Epoch 3/6
100/100 [==============================] - 133s 1s/step - loss: 0.7268 - accuracy: 0.5441 - val_loss: 0.6773 - val_accuracy: 0.5938
Epoch 4/6
100/100 [==============================] - 132s 1s/step - loss: 0.5849 - accuracy: 0.6520 - val_loss: 0.2084 - val_accuracy: 0.9312
Epoch 5/6
100/100 [==============================] - 135s 1s/step - loss: 0.2995 - accuracy: 0.8797 - val_loss: 0.1830 - val_accuracy: 0.9250
Epoch 6/6
100/100 [==============================] - 133s 1s/step - loss: 0.2773 - accuracy: 0.8978 - val_loss: 0.2557 - val_accuracy: 0.8906
```

Figure 11: Results of VGG-19 with an increased learning rate

The highest increase in validation accuracy as well as the highest decrease of the validation loss is detected in the fourth epoch. The increase in accuracy has been more

gradual compared to validation accuracy and the rates of the decrease of loss have been almost the same for the fourth, fifth and sixth epoch.

CHAPTER 3

3 RESULTS AND DISCUSSION

3.1 Performance of VGG-16

The model constructed using VGG-16 resulted in a quite accurate model, with very good end results. In the first epoch, the model has performed slightly better than in the second one, but no significant decrease or increase in parameters is observed. In the third iteration, a slight improvement in the accuracy and validation accuracy can be noted, but generally, based on the results obtained in the third epoch, it is clear that the model is obtaining very satisfactory results. In the fourth epoch, a decrease in accuracy can be spotted, accompanied by an increase in validation accuracy and a decrease in validation loss. This may be an indication that the model is performing quite good on the test dataset, but in order to prove this assumption, the monitoring of the next epochs is very important in order to identify the tendencies clearly. Some level of fluctuation is considered acceptable and only dramatic changes or continuous increasing or decreasing trends are the ones that are particularly concerning. In the fifth epoch, the training parameters continue to improve and some slight positive changes are observed in training parameters as well. In the last epoch, the validation loss has decreased a little, the loss has reduced, and the accuracy has climbed. The accuracy and validation accuracy are over 0.9, indicating that the first model using the VGG-16 network resulted successful in fulfilling the task of classifying the data into two categories.

In order to check the overall flow of the training process, the reflection of the six epochs in a graph is represented.

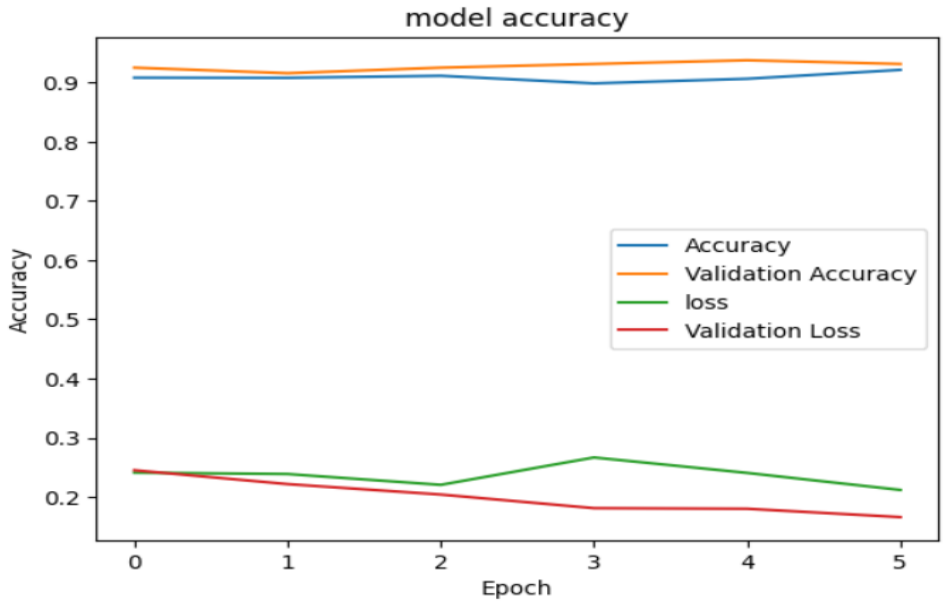

Figure 12: Performance of VGG-16 model

As observed in the graph, no concerning tendency of overfitting, underfitting or any other meaningful phenomenon is noticed. The model can be further applied to any imagery dataset for wildfire detection.

3.2 Performance of AlexNet

For evaluating the performance of AlexNet, the figures in the previous section will be used to analyze the results obtained after each epoch.

In the first epoch, the training parameters (validation loss and validation accuracy) have not reached the same highs compared to VGG-16. Taking here into consideration the size of the dataset chosen for this classification task, it would be highly preferred to have the technical capabilities to have much more epochs (possibly hundreds), but since working with Google Colab limits the number of experiments, the observations and conclusions about the behavior of the model will be only based on 6 epochs. In the second epoch, both validation accuracy and accuracy have increased and this may show that the model has started to slowly understand the patterns needed to perform the classification task and the learning process is going smoothly up to this point. In the third epoch, there is a good decrease in validation loss, but the validation accuracy has decreased as well. The loss has dropped and accuracy has increased. These two last changes may show that the learning process is going very well and that the model is continuing to learn from the train dataset, but up to this point, results for the test subfolder are not optimistic, since the number of mistakes in the classification remains

very high. The performance of the model on the test dataset is an important factor, because it shows how well the model performs on unknown data. If the model can classify poorly in the test dataset, this means that the constructed model is unable to generalize and its usability and implementation in a real-life scenario would be suspended.

The fifth epoch is the one representing important changes in both the train and test dataset. There is a decrease in loss, an increase in accuracy, a sharp increase in validation loss and a slight improvement in validation accuracy. This can be explained by the problem of overfitting, which prevents the model from generalizing well. Fighting the overfitting problem is complex and the application of regularization techniques is crucial, in order to intervene in a timely manner and prevent further escalation of the problem. More precisely, if the reason behind these changes in values in the fifth epoch is overfitting, it may come as a result of an inappropriate value of learning rate or it may indicate that the dropout is not working well. The dropout layer is employed to fight the problem of overfitting but in this case, if in the last epoch no improvement is detected and the tendency continues, it shows that an update of the learning rate or the checking of the dropout layer may be necessary. Another possible cause may be that the model is getting confused because of outliers or noise and instead of capturing the features and patterns needed for the classification task, it is not learning properly and the values in validation loss and validation accuracy are not consistent. In the sixth epoch, there is no significant change in the parameters of the training, but there is a good decrease in validation loss and the validation accuracy has risen by some units. The values obtained in the last epoch are the decisive ones that show that the values obtained in the fifth epoch were not related to the overfitting problem, but a possible cause could have been a set of noise or outliers not encountered before, that disrupted the learning process. Making assumptions or suggestions regarding the variation in parameters is challenging, because the way the model reads the train dataset and updates the weights is probabilistic (random) and because of the probabilistic nature of the process, the conductor of the experiment may simply share some ideas about these kinds of fluctuations, that are completely normal if encountered in a limited number of epochs.

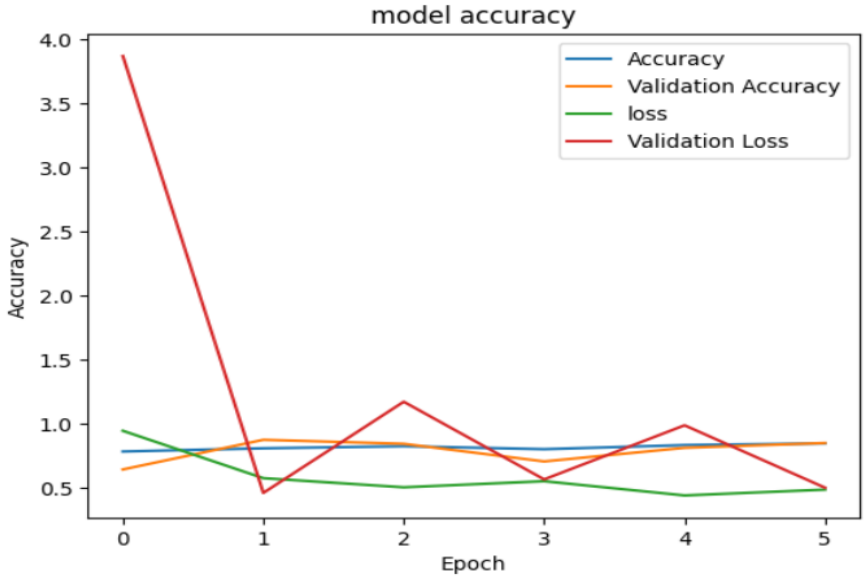

Figure 13: Performance of AlexNet model

There is a dramatic shrink in the validation loss from the first epoch to the second one, but the validation accuracy does not represent such a sharp increase. Still, this can be interpreted as a promising result that shows that the model learned quite well in this epoch, but further iterations were needed to stabilize the parameters and improve the validation accuracy.

3.3 Performance of VGG-19

In the first epoch, quite optimistic results are obtained in the parameters of the train dataset, but it is still early to conclude if the model is accurate or not. In the second epoch, there is a large decrease of the loss and accuracy has jumped by some units. The validation accuracy has jumped and a shrinking in validation loss is observed. In the third epoch, the parameters of the train dataset continue to show the same tendencies, but we see new patterns developing in the train dataset. The validation loss has improved and the validation loss has decreased. This may be an indication of overfitting, but in order to prove this preliminary hypothesis, it is necessary to carefully check the parameters in the following epochs. Generally, if the cause is overfitting, in the next epochs, an improvement in the train parameters and the worsening of the test parameters will likely occur.

In the fourth epoch, both the loss and the validation loss have decreased and accuracy and validation accuracy have increased. In the fifth epoch, there are only slight improvements in the train and test parameters. There are some variations in the four

parameters throughout the six epochs, but such fluctuations are considered normal and up to this point, there is not enough evidence of overfitting. In the last epoch, based on the metrics of the train dataset, it is almost sure that the training process is progressing smoothly and the model is learning progressively. It has an accuracy of 0.915 and the increasing tendency has been seen from the first epoch to the last one, without any concern of decrease or sudden drop. On the other hand, the validation loss has jumped by quite a few units and it has reached the highest point compared to all other values taken from the previous epochs. Meanwhile, the validation accuracy has decreased by 0.0875 and is close to the initial value of validation accuracy obtained in the epoch number one.

Considering the architecture of VGG-19 and the fact that it can be considered a very deep NN, the need for much more epochs and experiments is identified. The results observed in the last epoch show that the model has not converged yet and further training is needed in order to achieve optimal results. The graph regarding the variation in parameters also leaves room for discussion about other techniques that can be utilized in order to maximize accuracy, validation accuracy and minimize the loss and validation loss at the same time. One suggestion would be trying another learning rate or changing the number of steps per epoch.

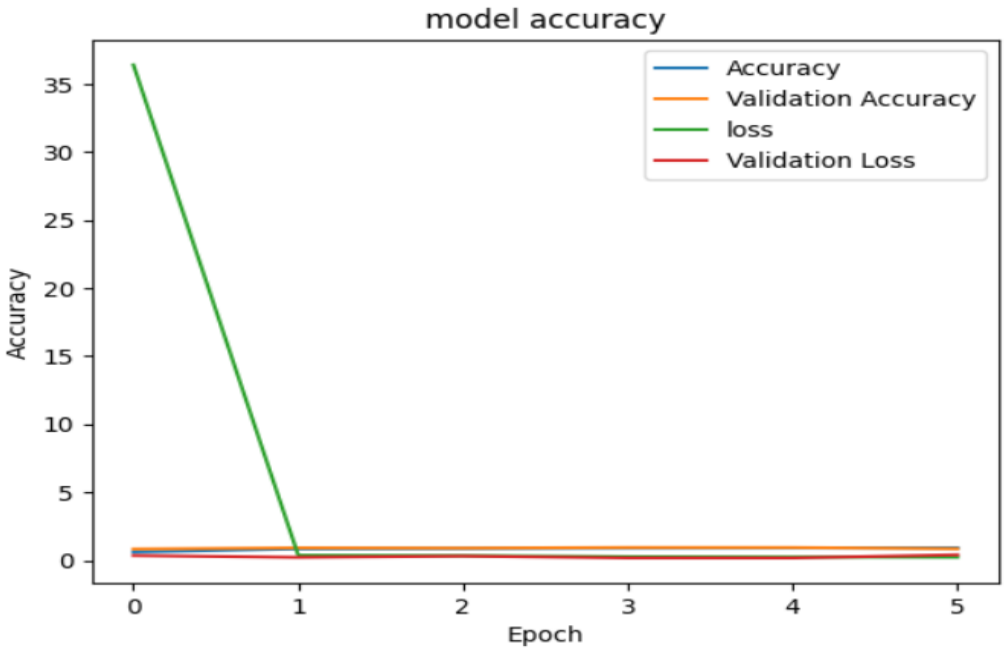

Figure 14: Performance of VGG-19 model

In figure 8, there is a dramatic drop of loss from the first epoch to the next one, which is a good indication that the model has started to learn and that the number of mistakes in the training data has been reduced. The reduction tendency in loss has continued throughout all the other epochs, but at a much lower rate compared to the first decrease seen between the first and the second drop. The graph also illustrates the fluctuations in the validation accuracy. The final value could have been higher if the model was trained using a much higher number of epochs, considering here other factors such as the size of the dataset and the architecture of the VGG-19 itself. Except for these remarks, there are only small variations from one iteration to another and these kinds of adjustments are normal, taking here into account that the values of the weights are decided probabilistically.

CHAPTER 4

4 CONCLUSIONS

This chapter compares and summarizes the results of the three models covered so far. Only the versions with 6 epochs and 100 steps per epoch were considered in this section in order to avoid confusion and provide a transparent comparison. By keeping the values of the parameters constant in all of the three models, it is easier to identify the network most suitable for performing the classification task at stake. There is a good chance that by modifying some of the parameters, the accuracy or loss outcomes may differ.

	VGG-16	AlexNet	VGG-19
Accuracy	0.9212	0.8469	0.9150
Validation accuracy	0.9312	0.85	0.85

Table 1: Summary of final accuracy values for the three models

From the results above, it is clear that in both the train and test dataset, VGG-16 has the highest performance followed by VGG-19 and AlexNet. Validation accuracy is the same for the VGG-19 model as well as for the AlexNet model. The fact that VGG-16 outperformed VGG-19 is particularly intriguing, considering the architecture of VGG-19 and the advantages it presents against VGG-16. Still, there was an epoch where VGG-19 reached a higher validation accuracy compared to VGG-16, but due to various reasons, the validation accuracy dropped in the following steps. It may be due to the fact that the model needed more iterations to converge, but the addition of the epochs was omitted due to the limitations of the platform used for running the experiments. On the other hand, the increase in the number of epochs can increase the risk of overfitting, meaning that the need for other mechanisms to prevent overfitting would arise.

A last test was performed on the validation dataset using the three models.

```
100/100 [==============================] - 26s 257ms/step - loss: 0.2029 - accuracy: 0.9197
Validation Loss: 0.202900692820549
Validation Accuracy: 0.9196875095367432
```

Figure 15: Results of VGG16 applied to the validation dataset

```
Found 6300 images belonging to 2 classes.
100/100 [==============================] - 25s 246ms/step - loss: 0.2262 - accuracy: 0.9166
Validation Loss: 0.22624337673187256
Validation Accuracy: 0.9165624976158142
```

Figure 16: Figure 10: Results of AlexNet applied to the validation dataset

```
Found 6300 images belonging to 2 classes.
100/100 [==============================] - 25s 248ms/step - loss: 0.2283 - accuracy: 0.9175
Validation Loss: 0.2282581776380539
Validation Accuracy: 0.9175000190734863
```

Figure 17: Figure 10: Results of VGG19 applied to the validation dataset

5 REFERENCES

Association, K. I.-T. (2023). *Lecture "Biologisch Motivierte Roboter"* . Karlsruhe.

Brenna D. Argall, S. C. (2009). *A survey of robot learning from demonstration,*, 469-483.

Ekman, M. (2021). *Learning Deep Learning: Theory and Practice of Neural Networks, Computer Vision, Natural Language Processing, and Transformers Using TensorFlow.*

Gavrilova, Y. (2021). *Convolutional Neural Networks for Beginners*.

Mandal, M. (2024). *Introduction to Convolutional Neural Networks (CNN)*.

Mohit Sewak, M. R. (2018). *Practical Convolutional Neural Networks.*

Munir, A. /. (2023). *Accelerators for Convolutional Neural Networks.*

Prince, S. J. (2012). *Computer Vision.*

Rosebrock, A. (2017). *Deep Learning for Computer Vision with Python.*

Journals:

https://serokell.io/blog/introduction-to-convolutional-neural-networks

Websites:

https://medium.com/@priyanshnagar015/step-by-step-vgg16-implementation-in-keras-for-beginners-100-understanding-89c6789eb4b4 (lastly visited on 19 June 2020).

https://medium.com/@owaisr814/how-to-implement-vgg19-vgg16-using-keras-and-pytorch-part-1-4ae1a858b466 (lastly visited on 14 September 2022).

https://medium.com/swlh/alexnet-with-tensorflow-46f366559ce8 (lastly visited on 19 January 2021).

https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53 (lastly visited on 15 December 2018).

https://towardsdatascience.com/train-validation-and-test-sets-72cb40cba9e7(lastly visited on 6 December 2016)

https://open.canada.ca/data/en/dataset/9d8f219c-4df0-4481-926f-8a2a532ca003

https://www.kaggle.com/datasets/abdelghaniaaba/wildfire-prediction-dataset (the link of the dataset in Kaggle)

6 APPENDIX A

6.1 Illustration of VGG-16 architecture

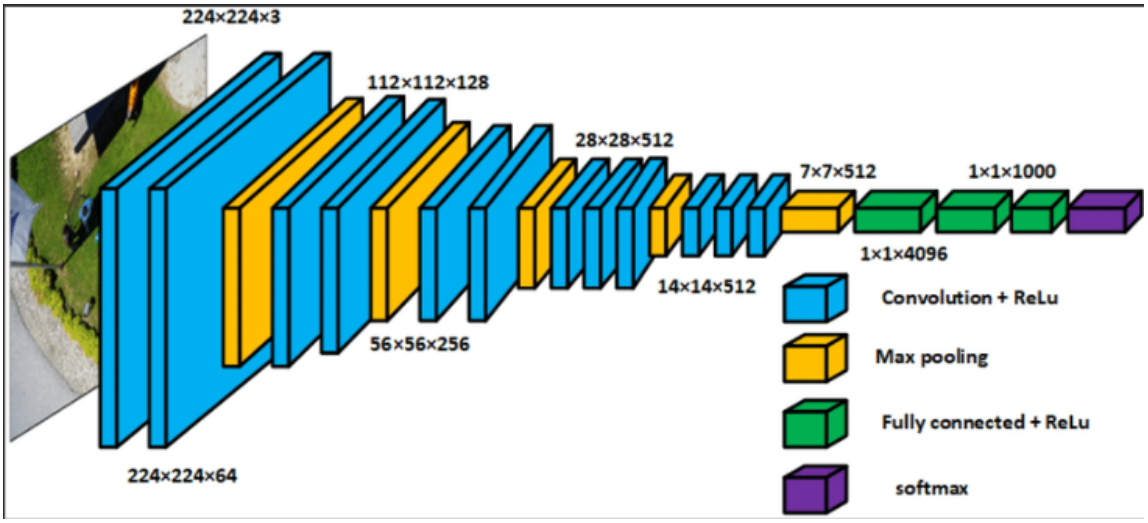

Figure 18: VGG-16 architectutre

6.2 Illustration of AlexNet architecture

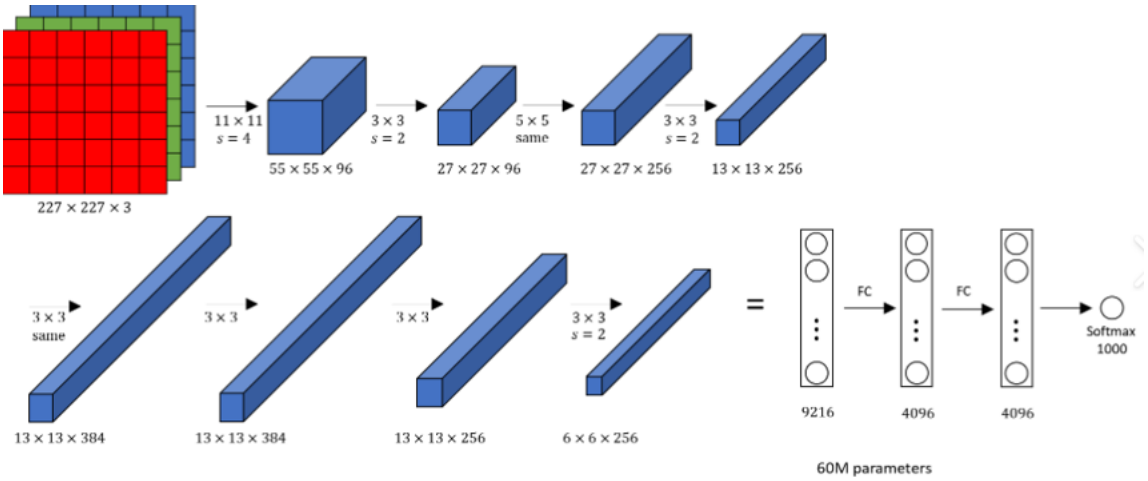

Figure 19: AlexNet architecture

6.3 Illustration of VGG-19 architecture

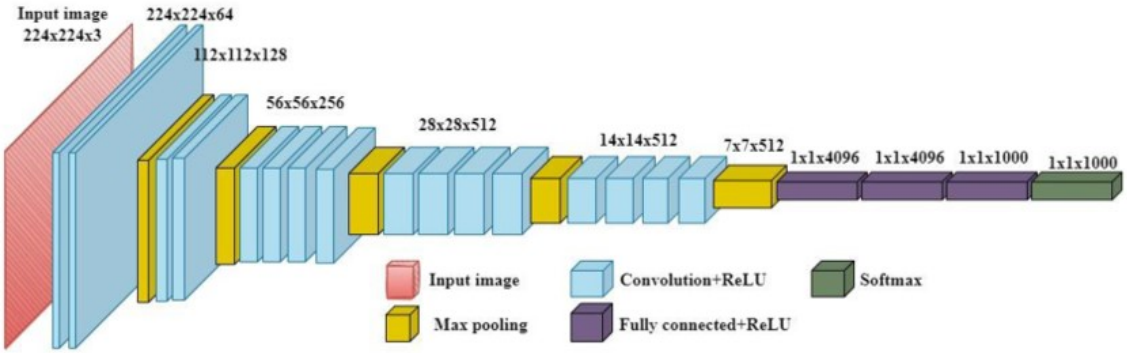

Figure 20: VGG-19 architecture