UNIVERSIDADE
DE PERNAMBUCO

Universidade de Pernambuco
Escola Politécnica de Pernambuco
Programa de Pós-Graduação Acadêmica em Engenharia de Computação

Angel Antonio Ayala Maldonado

KutralNext: An Efficient Multi-label Fire and Smoke Image Recognition Model

Master Thesis

Recife, March 2021

Universidade de Pernambuco
Escola Politécnica de Pernambuco
Programa de Pós-Graduação Acadêmica em Engenharia de Computação

Angel Antonio Ayala Maldonado

KutralNext: An Efficient Multi-label Fire and Smoke Image Recognition Model

Master Thesis

Thesis submitted to the Post-graduate Program in COMPUTER ENGINEERING at the Universidade de Pernambuco as partial requirement to earn the title of Master in Computer Engineering.

Prof. Dr. Bruno Jose Torres Fernandes
Orientador
Dr. Francisco Javier Cruz Naranjo
Coorientador

Recife, March 2021

Esta tesis va a dedicada a mi familia que siempre me han apoyado en mis decisiones y fomentado las ganas de estudiar. A mi polola Alejandra que también me ha ayudado a completar mis metas. A mis amigos que siempre me han brindado sus buenos momentos. Finalmente, pero no menos importante a mis orientadores Bruno y Francisco que me han brindado la oportunidad de desarrollar un buen trabajo así como oportunidades para crecer personalmente.

"We can only see a short distance ahead, but we can see plenty there that needs to be done" -
Alan Turing

Abstract

Early alert fire and smoke detection systems are crucial for management decision making as daily and security operations. One of the new approaches to the problem is the use of images to perform the detection. Fire and smoke recognition from visual scenes is a demanding task due to the high variance of color and texture. In recent years, several fire-recognition approaches based on deep learning methods have been proposed to overcome this problem. Nevertheless, many developments have been focused on surpassing previous state-of-the-art model's accuracy, regardless of the computational resources needed to execute the model. In this work, is studied the trade-off between accuracy and complexity of the inverted residual block and the octave convolution techniques, which reduces the model's size and computation requirements. The literature suggests that those techniques work well by themselves. Furthermore, in this research was demonstrated that combined, it achieves a better trade-off. Efficient models are required for hardware constrained systems, such as mobile devices, embedded systems, and robotics, achieving high performance at low-power consumption. This work proposed the KutralNext architecture, an efficient model with reduced number of layers and computacional resources for single- and multi-label fire and smoke recognition tasks. Additionally, a more efficient KutralNext+ model improved with novel techniques, achieved an 84.36% average test accuracy in FireNet, FiSmo, and FiSmoA fire datasets. For the KutralSmoke and FiSmo fire and smoke datasets attained an 81.53% average test accuracy. Furthermore, state-of-the-art fire and smoke recognition model considered, FireDetection, KutralNext uses 59% fewer parameters, and KutralNext+ requires 97% fewer flops and is 4x faster.

Contents

List of Figures

List of Tables

List of abbreviations and acronyms

DL	Deep Leaning
NN	Neural Network
CNN	Convolutional Neural Network
SGD	Stochastic Gradient Descent
RGB	Red, Green, and Blue colorspace
CCTV	Closed Circuit TV
ILSVRC	ImageNet Large Scale Visual Recogntion Challenge
MB	Megabyte
GB	Gigabyte
FL	Focal Loss
CB	Class Balanced
GPU	Graphical Process Unit
ROC	Receiver Operating Characteristics curve
AUROC	Area Unde the Receiver Operator Characteristics curve
flops	Floating-point operations
α	Learning rate, for the optimizer algorithm case
α	The octave feature representation hyper-parameter in the octave convolution
$\mathbb{R}$	Real numeric domain
$\in$	Belongs

Acknowledgements

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001, Fundação de Amparo a Ciência e Tecnologia do Estado de Pernambuco (FACEPE), and Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) - Brazilian research agencies.

Chapter 1

Introduction

The presence of fire in some environments is capable of causing massive losses; hence, the early recognition for this kind of accident is primordial. Early recognition of fire can be translated in a quick response to manage the accident, and therefore, high accuracy of fire recognition is also essential. In this regard, a system capable of triggering an alarm with high accuracy is crucial for the response team in charge of monitoring this kind of accident.

Fire accidents can be present in many environments, e.g., open-air, private, or community use spaces, among others, and can be originated because of human intervention, piece of machinery malfunction, unstable state of some structures, or in many other cases as a consequence of other natural disasters. Uncontrolled fire, or blaze, can affect in economic, social, and environmental way principally. This damage could be restored or not. In case it could be restored, considerable effort and consequently, resources are required. A common type of fire accident is the forest fire, which can significantly damage the environment [1] and increase its severity if it spreads.

In Latin America, the forest fires are mainly present in the Amazonia [2] and Chile [3], and have economic and environmental consequences such as mentioned by Urzua et al. [4]. Chile, just in 2014, had more than 8000 fires, which affected 130000ha. After the forest fire, the soil remains damaged [5], and it is difficult for the vegetation to grow again. When this type of accident occurs in the environment, all plants and animal life disappear from the affected zone due to the environment's perturbation. The fires' problem is that they are unpredictable, in the way of when or where they will occur, especially for forest fires. Hence, an early alert system would help to manage these accidents or natural disasters.

This work proposes an efficient deep learning model to recognize fire and smoke as a multi-label classification task specialized for embedded devices such as CCTV devices, mobile and robotic systems. The model's architecture development focused on low computing power devices with high accuracy in acquiring fire and smoke features. In order to obtain a suitable model, different architectures were proposed inspired by generic- and specific-purpose deep learning models and trained with previously used datasets. Hereof, a final efficient architecture was developed after checking different efficient techniques such as convolve methods and convolutional blocks with a specific setup.

1.1 Motivation

The fire alarm systems are a combination of sensors and machine learning algorithms to identify patterns of warning. Usually, the sensor system is composed of: (i) heat or temperature detectors, which typically are not early warning devices; (ii) flame detectors that habitually are built-on optical, UV, and IR sensors; and (iii) smoke detectors, frequently using photoelectric, ionization, or a combination of both. The use of images to fire recognition is a new promising approach [6], based on the excellent results that deep learning models obtain in image processing applications [7], avoiding the use of special sensors to perform the recognition.

The Deep Learning (DL) approach [8] has proved to be suitable for automating the feature acquisition from complex data in machine learning tasks. In such a way, DL works in multiple levels of abstraction for data representation. Considering that, the use of computer vision to fire recognition reduces specific sensors' necessity, being suitable for the inlay to portable, remote, and mobile devices. However, DL approaches have some challenges: (i) the required computational resources; (ii) the model's computation complexity and size; and (iii) the quantity of data needed for its training, among others. The previous challenges can be tackled by focusing on the development of deep learning models for mobile devices [9], which has been a less explored area in DL literature.

Many benefits can be obtained from developing an efficient portable DL model. In the first place, this kind of algorithm can process more images during a determined amount of time, hence, recognize fire in real-time at a high frame rate. In the second place, it can be executed parallelly in the same machine to process data from different sources simultaneously. These algorithms can be used in robotics or autonomous mobile systems, capable of using reduced processing power, storage size, and energy consumption in terms of implementation.

1.2 Objectives

1.2.1 General objective

Study, develop and compare efficient deep learning techniques to build a lightweight and computationally inexpensive model for fire recognition in still images.

1.2.2 Specific objectives

- Study about deep learning and current efficient processing image techniques.

- Define image dataset to benchmark state-of-the-art and developed fire recognition models.
- Develop a baseline lightweight approach and a more efficient one for fire recognition.
- Perform training and testing of each proposed model.
- Compare the state-of-the-art and developed models' benchmark results.

1.3 Methodology

In order to fulfill the objective of this work to develop a lightweight and efficient model for fire recognition, the following methodology was implemented. In the first place, a literature review of current deep learning methods used to recognize fire with a lightweight scope. In the second place, a literature review of current techniques to improve deep learning tasks such as classification, considering convolutions, activation, and loss function, among others. Once the state-of-the-art was found, a dataset selection follows to benchmark each model's performance in the fire and smoke recognition task. In this way, previously used datasets were reviewed and relabeled to use in this work, obtaining training, validation, and testing subsets. The first experimentation stage consists of optimizing a model to recognize fire in a single-label approach. A multi-label approach is later addressed to recognize fire and smoke as an extension from the single-label approach, using the best-proposed architectures. Those models first learned more complex features in a challenging dataset to be optimized in the selected datasets for this fire and smoke recognition task. More details about this are mentioned in chapter 3 and illustrated in Figure 1.

The state-of-the-art models were replicated from their original works by the respective authors to compare the proposed approaches. For the proposal, a lightweight baseline architecture was defined to develop portable models, with efficient techniques replacing some baseline blocks. Each technique improvement level was tested separately and combined. Furthermore, a fine-tunning and transfer learning methods were used to improve the proposals' performance. The training and testing stages were executed with each dataset under a cross-validation and cross-testing setup.

With the models' parameters optimized, the validation and testing accuracy, the receiver operating characteristic (ROC) curve, the area under the ROC curve, the number of parameters, and floating operation points were measured to analyze the suitability to be implemented in low-resource hardware and run at a high frame rate, proving to be computationally inexpensive with high accuracy.

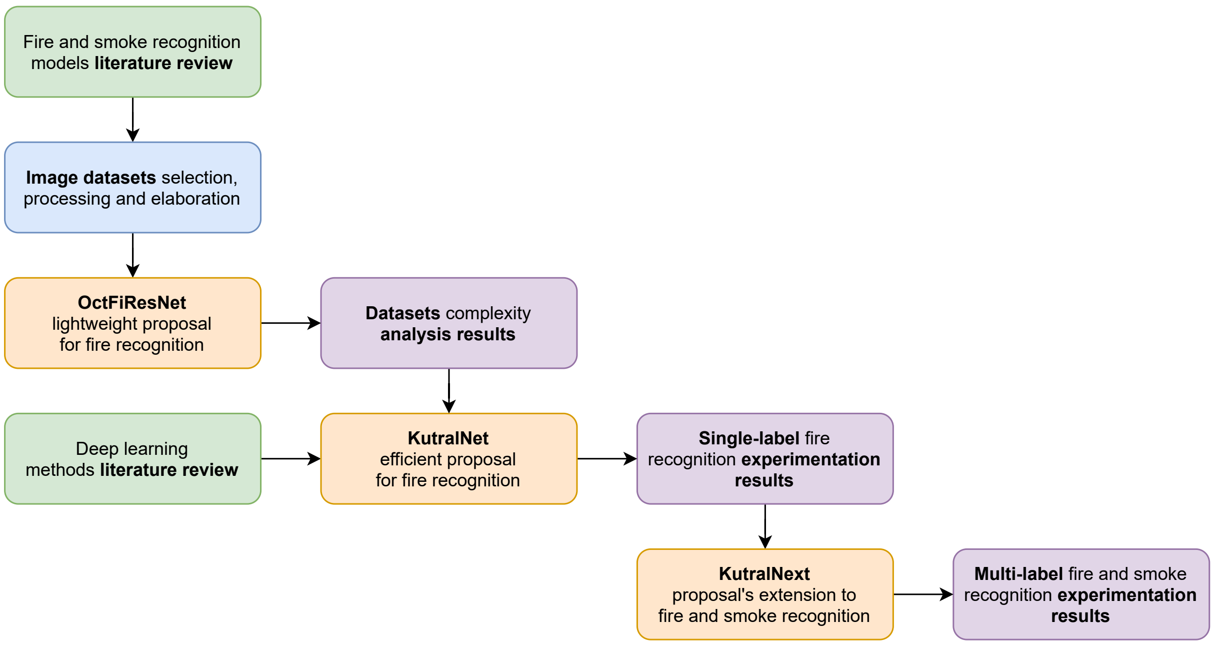

Fire and smoke recognition models literature review
Image datasets selection, processing and elaboration
OctFiResNet lightweight proposal for fire recognition
Datasets complexity analysis results
Deep learning methods literature review
KutralNet efficient proposal for fire recognition
Single-label fire recognition experimentation results
KutralNext proposal's extension to fire and smoke recognition
Multi-label fire and smoke recognition experimentation results

Figure 1 – An overview of the proposed methology. Green squares represent the literature review stages. The orange squares are each proposal of this project, and the purple ones are the experimentation for each proposal. The blue square represent the dataset selection, processing and elaboration for all the images used in this research.

1.4 Obtained results and contribution

During this work, two contributions were achieved for the fire and smoke recognition problem. The first one is a suitable dataset for training models to recognize images with only fire or fire and smoke in a classification task. The FiSmo dataset was previously compiled from internet images search engines and other datasets with images labeled with fire, smoke, and none. In this work, FiSmo was relabeled and augmented to be used in a multi-label approach for fire and smoke recognition. A second benchmark dataset was also compiled from previous datasets and combined to create training and testing subsets under the same multi-label approach.

The second contribution is the efficient KutralNext baseline architecture to recognize fire and smoke in images, with a reduced number of layers, hence, storage size and computational resources. Thus, a more efficient KutralNext+ architecture was developed from the baseline with novel convolution methods, capable of achieving a good generalization and acquiring useful features to recognize fire and smoke in still images with an 81.53% of average test accuracy in the KutralSmoke and FiSmo fire and smoke datasets. Moreover, KutralNext+ obtained a 93.40%, and 94.22% AUROC and precision values in acquiring fire features, and an 89.59% and 56.27%, respectively acquiring smoke features. Outstanding results were achieved by KutralNext+, presenting 97% fewer flops and executed 4x faster than FireDetection, a previous fire and smoke recognition model.

1.5 Document structure

- In chapter 2 are mentioned the previous approaches which tackling the fire recognition problem, being the most recent using deep learning methods with some novel convolution methods to process a high dimensional input. Additionally, the literature review of deep learning techniques for fire recognition are also discussed.
- In chapter 3 is detailed each contribution of this work, presenting the implemented technique, how they were built, the developed models, and the training strategies used.
- In chapter 4, are detailed the datasets used in this work to further compare each proposed model with previous state-of-the-art and the metrics used for this purpose.
- In chapter 5 are the benchmark results of each proposal with previous models, starting from the OctFiResNet approach to the following KutralNet architectures to finally get the KutralNext+ model, demonstrating its effectiveness and suitability for being used in hardware-constrained systems.
- In chapter 6 are the final remarks summarizing the works achievements and future works proposals.

Chapter 2

Background

In the last years, multiple methods to automate fire recognition were proposed, most of them for video surveillance systems, such as Closed Circuit TV systems. Some surveillance equipment uses low-resolution cameras at a flat frame rate and others more sophisticated ones with proper image resolution cameras. For a machine to have the ability to recognize fire from still images is a highly demanding task due to recognition of texture, color, and the fire's phenomenon representation by itself [10].

The first approaches to fire recognition in computer vision were addressed using the color space. Yoon-Ho et al. [11] presented a work that uses the RGB space information to detect the foreground of fire-like objects in video sequences for a CCTV fire detection system. Another color-based technique was presented by Nikos et al. [12], where the authors process the spectral color space from optical and infrared cameras as a remote fire surveillance system. Another technique is presented by Dimitropoulos et al. [13], where the authors developed a texture recognition algorithm employing color probabilities and contour irregularity features. In their following work [14], the same authors implemented a dynamic analysis to take advantage of the fire's features obtained using linear models. Finally, in color-based algorithms, a Spatio-temporal approach was addressed by Barmpoutis et al. [10], which analyzes frame by frame, searching for flickering and color-probability.

The most recent method has been addressed using a deep learning approach through a convolutional neural network (CNN). The CNN process an input data image through each convolutional layer to finally infer which label must correspond to the input image. More details are discussed in the following section.

2.1 Deep learning

Machine learning algorithms have been processing raw input data to solve many problems through a pattern-recognition construction so long ago. Deep learning (DL) [8] is a technique used to learn data representation from a high dimensional raw input over multiple processing layers. This technique discovers the underlying structure in large datasets by optimizing its parameters with the backpropagation algorithm. Generally speaking, the more processing layers used, the more complex structure can be discovered.

In this work, we will focus on the supervised learning paradigm [15], which in simple words can be defined as a mapping function that must be found from annotated training data. Each data instance is labeled with the class where it belongs, i.e., an RGB image labeled with a car, cat, or dog class. This mapping function is composed of parameters that must be optimized to achieve a high score for each category label at the function's exit. In this regard, an objective function must be settled to measure the error between the function output and the corresponding label to further properly adjust the trainable weights to reduce this error. The learning algorithm optimizes the parameters by computing a gradient vector for each weight, determining how much it must variate, being updated in the opposite direction to the gradient vector. These errors and gradient values are computed over all the training examples; however, it is not always feasible to load the entire data on memory, being introduced by batches of N data elements that will be loaded and processed by the function or weighted model.

The model's error reduction can be made by a simple stochastic gradient descent (SGD) procedure. This procedure consists of feeding the model with a few examples, inferring the outputs, computing the errors, computing the average gradient, and adjusting the weights properly. The procedure is repeated over all the data batches until the average of the objective function stops decreasing. When the model's stack of layers is bigger, the gradient computation becomes tricky. However, multi-layer gradient computation has already been solved by the backpropagation algorithm [16], investigated by several different groups during the 1970s and 1980s. Backpropagation's main idea is applying the chain rule for derivatives (or gradients), calculating the gradient of a layer backward from the output of the same layer or the same as the input of the subsequent layer. In this regard, the gradients are propagated from the top, or exit, back to the bottom, or input of the model.

2.1.1 Convolutional neural networks

These neural networks (NNs) were designed to process data represented in multiple arrays like 1D sequence or audio signal, 2D image data with color space data, or 3D volumetric images in videos. Additionally, they are biologically inspired by the classic notion of visual neuroscience cells [17] and the visual cortex ventral pathway [18]. Four key ideas are denoted from the physiological properties, local connections, shared weights, pooling, and layers' stack.

A model architecture is commonly made of a stack of convolutional blocks, containing a convolutional layer, a non-linear activation function, and a pooling layer. The convolutional layers are organized in feature maps and connected to local patches in the feature maps of previous layers after being processed by a bank's weighted kernel filter. The mathematical approach for the feature map is a discrete convolution, hence the name. As image data is often highly correlated between pixels, becomes essential a weighted kernel to keep the feature map's correlation and detect an specific pattern in different data parts. Between one layer and another, a non-linear activation function passes the signal from the current layer to the next, distorting the input for the categories become linearly separable by the last layer. Pooling layers can detect local feature conjunctions from the previous layer, merging semantically similar features into one done by coarse-graining the position of each feature, reducing the dimension at the output.

2.2 Efficient deep learning methods

With the success of deep convolutional neural networks, efficient techniques had appeared with newly proposed models. The first known used technique is the residual connection [19], which formally can be defined as $\mathcal{H}(x) = \mathcal{F}(x) + x$ where x is the input signal or the identity connection, and $\mathcal{F}(x)$ is the convoluted input signal. This strategy reduces the overfitting during the training of deeper architectures, improving a gradient's ability to propagate across multiples layers requiring almost the same number of operations.

The second widely used technique is the depthwise separable convolution, which separates the convolution in a channel-wise spatial correlation mapping, followed by a cross-channel mapping with a 1x1 convolution, also called pointwise convolution. Chollet et al. [7] proposed the Xception model, where the authors used depthwise separable convolutions and residual connections in the architecture. Depthwise convolution was first presented by Sifre et al. [20], and more deeply proven its efficiency by Chollet [7] with its Xception model, which uses depthwise separable convolution and residual connections in almost all the architecture. The computational cost of a vanilla convolution is given by

$$\mathcal{C}_v = D_k * D_k * M * N * D_f * D_f,$$

where D_k is the kernel size, assumed square, M is the number of input channels, N is the number of output channels, and D_f is the feature map size. In comparison, the depthwise convolution's computational cost is given by

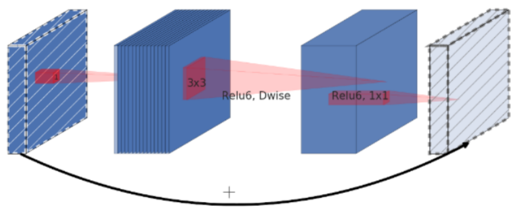
$$\mathcal{C}_{dw} = D_k * D_k * M * D_f * D_f.$$

In this regard, the depthwise convolution is more efficient than the vanilla convolution because it breaks the relationship between the number of output channels and the kernel size. The addition of the 1x1 convolution to the depthwise convolution was proposed by Sifre et al. [20] and the final computational cost

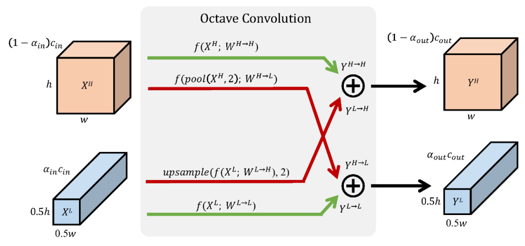
$$\mathcal{C}_{sdw} = D_k * D_k * M * D_f * D_f + M * N * D_f * D_f,$$

obtaining a reduction of

$$\mathcal{W}_{sdw} = \frac{D_k * D_k * M * D_f * D_f + M * N * D_f * D_f}{D_k * D_k * M * N * D_f * D_f} = \frac{1}{N} + \frac{1}{D_k^2}. \tag{2.1}$$



(a)



(b)

Figure 2 – Main DL techniques used in this work. (a) The inverted residual block. Diagonally hatched layers do not use non-linearities. The thickness of each block is used to indicate its relative number of channels. The inverted residuals connect the bottlenecks. Adapted from [21]. (b) Detailed design of the octave convolution. Green arrows correspond to information updates, while red arrows facilitate information exchange between the two frequencies. Adapted from [22]

A third most commonly used technique is the inverted residual block [21], which is composed of depthwise separable convolution and residual connections. The peculiarity of this convolutional block is the presence of the shortcut in the bottleneck, between the layers with a low number of channels as can be observed in Figure 2a. Additionally, it presents an expansion layer that increases the number of channels processed between the bottleneck using a depthwise convolution, before and after pointwise convolution. This convolutional block presents a computational cost of which depends on an expansion rate t

$$\mathcal{C}_{irb} = D_f * D_f * M * t(M + D_k^2 + N). \tag{2.2}$$

Another new technique for efficient model design is the octave convolution [22], which decomposes the input signal in a high-spatial frequency to describe the rapidly changing details and in a low-spatial frequency to describe the smoothly changing structure. The authors have demonstrated that use the octave convolution in popular DL models like ResNet [19] consistently improves the results, reducing the flops and model's size. Formally, let X be the input image $\in \mathbb{R}^{M*D_f*D_f}$, where D_f is the spatial dimension considered squared, and M the number of channels. X is factorized into $X = \{X^H, X^L\}$, considering $X^H \in \mathbb{R}^{(1-\alpha)M*D_f*D_f}$ the high-frequency feature maps of fine details, and $X^L \in \mathbb{R}^{\alpha M*\frac{D_f}{2}*\frac{D_f}{2}}$ the low-frequency feature maps of general characteristics. Here $\alpha \in [0, 1]$ is a hyper-parameter denoting the ratio of channels allocated in the low-frequency part. In this regard, the computational cost is reduced by two components given by

$$\mathcal{C}_o = D_f * D_f * M * N * (1-\alpha) + \frac{D_f}{2} * \frac{D_f}{2} * M * N * \alpha, \tag{2.3}$$

where N is the number of output channels. Each side of the equation's addition represents the convolutional cost for the high- and low-frequency. After processing the signal, each frequency feature map's information is exchanged between them, as shown in Figure 2b.

2.3 Fire and smoke recognition

The most recent methods using DL approaches have tackled the problem of fire and smoke recognition through a convolutional neural network (CNN), as a single-label classification task, where the CNN process an input data image by each convolutional layer, reducing its dimensionality into meaningful features. The features acquired by a CNN have been proven to be related to the network's depth. Early layers can obtain simple features like colors and shapes, and final layers process complex features [23]. After rich features were obtained from the input, this data representation is processed by a classifier, which usually is a linear regressor. A few fully connected layers with a considerable amount of hidden units can also be used as a linear regressor to infer which label corresponds to the image. The most recent methods are detailed as follows.

Sharma et al. [24] developed a custom fire classification model based on VGG16 [23] and ResNet50 [19], two generic-purpose DL models, where the authors just modified the classification stage, adding one fully connected layer at the top of the network implementing transfer learning and fine-tuning methods. Additionally, the authors created their dataset with 651 images, considering 549 unbalanced images for training with 490 non-fire images, assuming that the probability of fire occurrence is relatively small. However, the testing subset is balanced with 102 images presenting 51 images for each situation. Muhammad et al. [25] had proposed a SqueezeNet based-model, where the authors present a custom framework to process the input signal, to classify and locate the fire in a single-label approach. Namozov et al. [26] presented a VGG16 inspired approach with 12 convolutional layers and the adaptative piecewise linear activation [27] function instead of traditional rectified linear units [28]. Their proposal was trained with their dataset with 2440 images labeled as fire and smoke equally balanced. Additionally, the authors implemented data augmentation using Generative Adversarial Networks to create three subsets from the original one. Gotthans et al. [29] proposed the Fire Detection model to fire and smoke recognition trained with two datasets to compare it against AlexNet [30] and SqueezeNet [31]. The model received an input image of 224x244 pixels with RGB channels, normalized with mean values of (0.485, 0.456, 0.406) and standard deviation of (0.229,0.224, 0.225) for each channel. The authors proposed to achieve a lightweight model capable of recognizing just fire, and fire and smoke in still images. Additionally, they tested the model's execution in the Jatson Nano platform, obtaining the same results. The Fire Detection model reduced in 27% the execution time compared to AlexNet, with only 1% less accuracy.

A lightweight model was proposed by Jadon et al. [32], capable of processing images of 64x64 pixels on RGB channels. The architecture comprises three consecutive convolution blocks that contain a convolution layer, an average pooling layer, a dropout layer, and three fully connected layers as the classifier. Additionally, the authors presented training and testing datasets with 2,425 and 871 images each, achieving good performance. The proposed approach was focused on being used in an IoT embedded fire alarm system. Oh et al. [33], used the

EfficientNet-B0 [34] generic-purpose model to recognize a fire emergency from images. The model was optimized using transfer-learning and fine-tunning methods from a pretrained version. A custom dataset was also collected from various image search engines, using an automated downloading algorithm for cloud, snow, rural, fire, wave, and waterfall labeled images. Next, they made a manual cleanup operation obtaining a total of 14,741 images. In this case, the fire-labeled images contained an open-air environment with the presence of fire, smoke, or both.

2.4 Discussion

To summarizing, many fire and smoke recognition algorithms were proposed using general-purpose deep learning models, focused on surpassing previous results mainly. All previous approaches address the fire and smoke classification problem under a single-label approach, leaving out that the fire and smoke labels can be present separately, together, or not present. The fire recognition task has been proven to require just a few layers to achieve a good performance in acquire fire's features. In this regard, a lightweight and efficient architecture must be affordable to be implemented in hardware constrained systems to monitoring or fire control. For example, the development of this kind of fire recognition DL model can work in the mobile vehicle system for fire detection proposed by Madhevan et al. [35].

Chapter 3

Efficient fire and smoke recognition model

As fire images present characteristics hardly to be hand-craft extracted, a DL approach is used for this purpose in this work. Three different DL architectures were developed in this work's aims, focused on reducing the size and computational cost to achieve fire and smoke recognition in still images. As mentioned in section 2.2, experiments with different novel techniques were carried out, checking the feasibility of obtaining small and efficient models suitable for low computing power. In this regard, the residual connection [19] and the octave convolution [22] were tested on the first lightweight approach inspired in the ResNet architecture to recognize fire in still images, called OctFiResNet. Next, a custom specific-purpose lightweight DL architecture is proposed with five convolutional blocks and a residual connection before the classifier with two exits, one for the fire label and the other for the non-fire label. The reduced architecture is named KutralNet and used as a baseline to develop efficient portable models using the octave convolution, the depthwise convolution [20], and the inverted residual block [21] separated and combined. Moreover, as KutralNet extension, the best efficient models were pretrained using the ILSVRC 2012 dataset and learning complex features to use transfer-learning and fine-tuning methods to recognize fire and smoke in images. A single-label approach was addressed in all the previous proposals to classify the fire and smoke presence in images. However, a multi-label approach brings out a more specific fire severity recognition level, inferring in different non-exclusive units the fire or smoke presence in an image. This KutralNet extension was named KutralNext, implementing a different objective function to solve the unbalanced labels data distribution present in datasets.

3.1 ResNet based fire recognition model

The first proposed model is based on the ResNet architecture, which comprises levels of blocks with convolutional layers, followed by a batch normalization layer and an activation layer. Furthermore, each block connects its input independently with the block's output, known as residual. Therefore, the next block obtains as input the processed and raw signal from the current block. Each level's first block of the ResNet architecture, unlike the middle blocks, process the residue by convolution and batch-normalization layers. To connect each block uses an activation layer that processes the previous block signal to the next one.

To obtain a lightweight model capable of reaching high performance on portable hardware, it needs to have as few parameters as possible. In order to reduce the parameters model number, this work proposed a low deep level architecture. This low-depth model with just a few layers is suitable for recognizing fire, demonstrated by Jadon et al. [32], to work with a constrained-hardware system. This first OctFiResNet approach, to work with the minimum hardware requirements as possible, the octave convolution [22] replaces the vanilla convolutions. The octave convolution processes the signal in two different channels, one for high-frequencies to acquire more detailed features and the other for low-frequencies to more general features. This technique allows the model to work with less memory and fewer flops compared to a vanilla convolution layer.

Combining the ResNet-like architecture with octave convolutions reduces the model's size and computational cost. The input for the network is a 96x96 pixels image at the RGB channel. This architecture is composed of 2 ResNet levels with 4 and 2 blocks, respectively. It has a global average pooling on top of the network, followed by a fully-connected layer with two exits and a softmax activation. The α ratio value for the octave convolution is 0.25 and with 64 initial filters. This configuration turns out a model with 956,226 trainable parameters with an on-disk size ~12MB. The model's development was using the PyTorch library. A simplified version of the architecture can be seen in Figure 3, and additional implementation details are in the project's repository[1].

3.2 Lightweight efficient deep learning model

The following proposal for fire recognition sets a baseline model to develop portable versions focused on reducing the model's complexity in processing the input image. The KutralNet[2] model was developed as a suitable option for limited hardware devices and built other efficient versions using the octave convolution and the inverted residual block to test each efficient technique by themselves and combined. Hereof, three portable models were obtained from this baseline using efficient deep learning techniques. The octave and depthwise convolution [22, 21] demonstrated excellent performance with a sharp reduction of operations and parameters required, resulting in more efficient models. This reduction is resulting from convolutions with low kernel dimensions for both cases.

For the case of the separable depthwise convolution in the inverted residual block [21], it increases the number of parameters and reduces the flops efficiently, as demonstrated in

[1] OctFiResNet's public repository <https://github.com/angel-ayala/fire_recognition>
[2] The name took inspiration from Mapuche language or Mapudungun where kütral means fire.

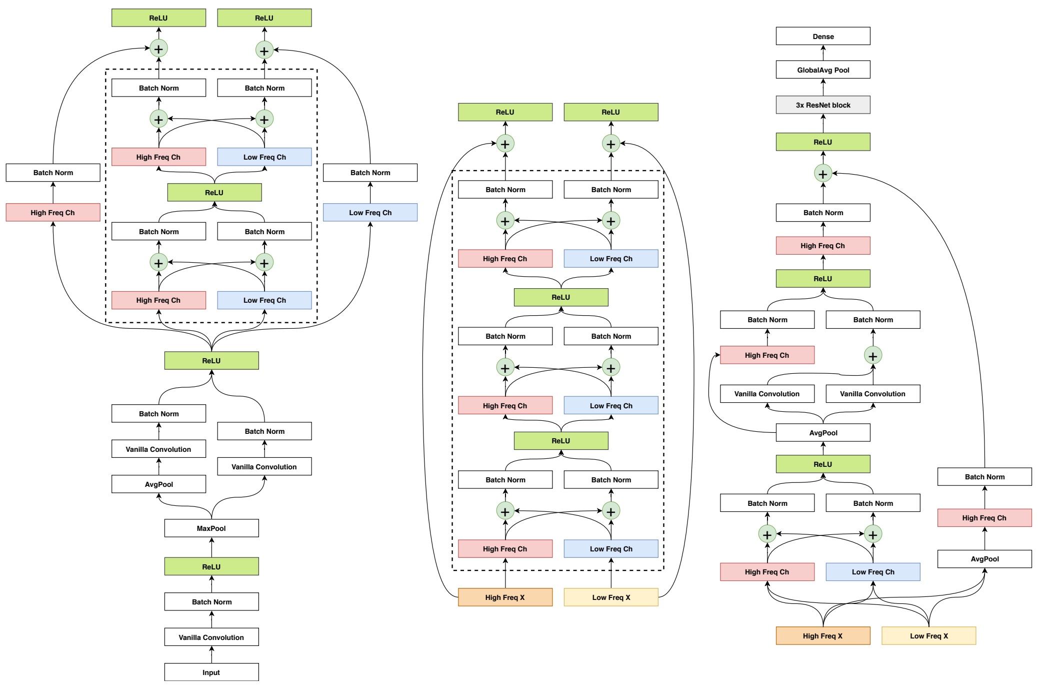


Figure 3 – Simplified architecture of the proposed model blocks of ResNet with octave convolution. To the left is the initial block, where the residual passes through a convolution and batch normalization. In the middle, it is the consecutive block with no processed residual, which is repeated 3 times with 2 levels. Finally, to the right is encountered the top layers which merge the octave convolution into one. After this, the signal is processed by a ReLU activation consecutive with a ResNet block of 3 levels with vanilla convolution, batch normalization, and ReLU activation.

Equation 2.1. Given the grouping way to process the convolution channels denoted as $groups = C_{in}$ and $out\_channels = C_{in} * K$, in which the output filters are K times the input filters, reducing the mathematical complexity of the operation formally expressed in Equation 2.2. For the octave convolution case, a reduction in both parameters and flops is achieved due to the separate way of processing the filters on high and low frequency, computing the parameters information W into two components $W = [W_H, W_L]$ and exchanging the information between them, expressed in Equation 2.3. Additionally, these convolution techniques, used in different deep learning model architectures, and various tasks such as classification, object detection, and semantic segmentation, achieve a model's size reduction, less computational requirements, and improved performance in some cases. This second proposal combines these techniques, presenting a new convolution type, achieving a valuable trade-off between accuracy, model size, and computational cost. Additional details of the implementations are in the project's repository[3].

3.2.1 Baseline model's architecture

The KutralNet model's baseline was inspired by OctFiResNet and FireNet models, mixing between a deep model and a lightweight one, capable of processing 84x84 pixels

[3] KutralNet's public repository <https://github.com/angel-ayala/kutralnet>

images in RGB channels. The KutralNet architecture comprises three kinds of convolutional blocks, named KutralBlockN (KBN), where N corresponds to the number of output channels, KutralBlockP (KBP), and KutralBlockO (KBO). KBN block was built with a convolution layer with N channels as output, a batch-normalization layer, a LeakyReLU activation, and a max-pooling layer to size-down the output. Next, the KBP block comprises two convolution layers and a batch-normalization layer. Finally, the KBO block possesses a LeakyReLU activation, a global average pooling layer, and a fully-connected layer with two exits, one for fire and the other for non-fire labels. This architecture was defined for processing low-dimension images in a lightweight configuration. Each block details are shown in Figure 4a. As shown in Figure 4b, the architecture consists of three KBN blocks, one KBP block, and finally, a KBO output block. A max-pooling and batch-normalization layers, as a shortcut, process the signal from the KB64 block to the final KBO output block. This setup was followed because it has been proved that just a few layers can acquire enough features for a fire classification task to improve the inference time [32]. Additionally, using a shortcut and batch-normalization layers avoids overfitting the model [19]. Also, the LeakyReLU was chosen since a non-zero slope for the negative part improves the results [36] and presents a low-cost implementation.

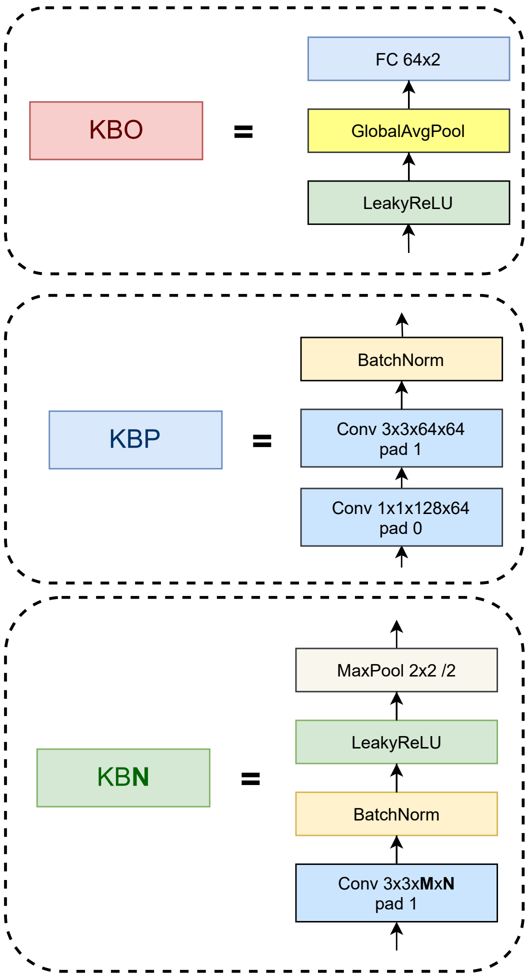
3.2.2 KutralNet Mobile

With those convolution methods, previously mentioned in section 2.2, the first portable model, **KutralNet Mobile** was developed by implementing the inverted residual block named KutralMobileBlockN-E (KMBN-E) in replacement of the KBN blocks from the baseline architecture. The N-E refers to the number of output channels and the expansion rate t, respectively. The KMBN-E block comprises a pointwise followed by a separable depthwise convolution, with a batch-normalization layer and a ReLU6 activation in between. Additionally, a batch-normalization layer is stacked at the end of the block. A representation of the KMBN-E block is shown in Figure 5a.

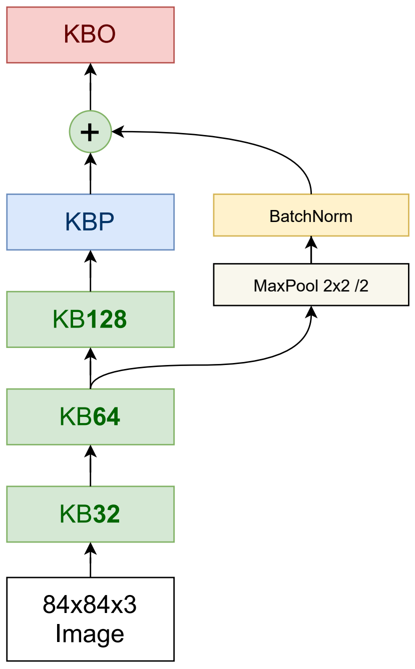
KutralNet Mobile's architecture structure contains a KBN block, followed by three KBMN-E blocks and the KBO block at the end. A shortcut connection process the signal from the first KMB64-4 block through a max-pooling layer and a batch-normalization layer to the final KBO block, as shown in Figure 5b.

3.2.3 KutralNet Octave

The second portable **KutralNet Octave** model replaces all the vanilla convolution from the baseline's architecture with the octave convolution [22]. In this case, the octave convolution separates the processing into the *octave feature representation* or low-frequency signal for the most general features and its counterpart or high-frequency signal for the most fine-grained features. A hyper-parameter α gives each convolution's size, processing the signal separately, and exchanging the information between them at the end. This convolution was named OctConvN,



(a)



(b)

Figure 4 – (a) The KutralNet main blocks. The KutralBlockN (KBN) where N refers to the output channels number, KutralBlockP (KBP), and the KutralBlockO (KBO). (b) The baseline KutralNet model with three KBN blocks, a KBP block, a shortcut connection, and a KBO block with two exits.

where N denotes the number of output channels. The α parameter for the OctConvN was settled to 0.5 in all the cases. Additionally, the main convolutional block was named KutralOctaveBlockN (KOBN), where N is the number of output channels passed to the OctConvN, followed by twins of batch-normalization layers, twins of LeakyReLU activation and, twins max-pooling layers. One of each twin-layer member is for the high- and low-frequency convolution. An overview is shown in Figure 6a.

KutralNet Octave is composed by two KOBN blocks, a KBP block merged with a shortcut connection from the KOB64 block, processing the signal through an OctConv128 block, to the final KBO block, as presented in Figure 6b.

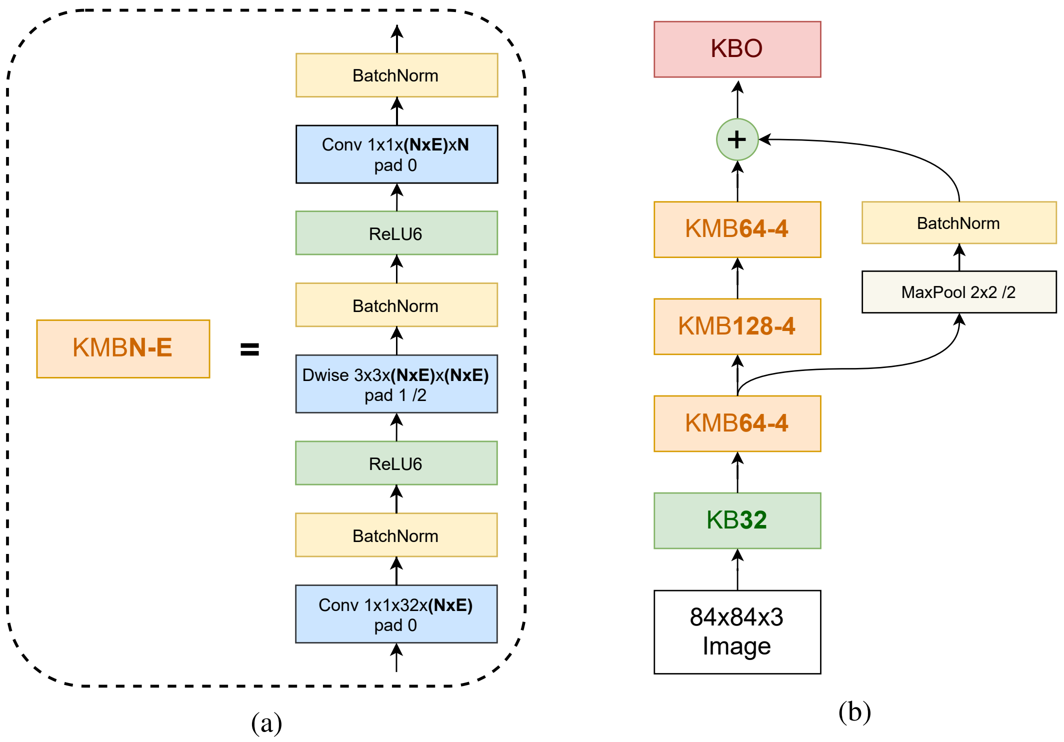
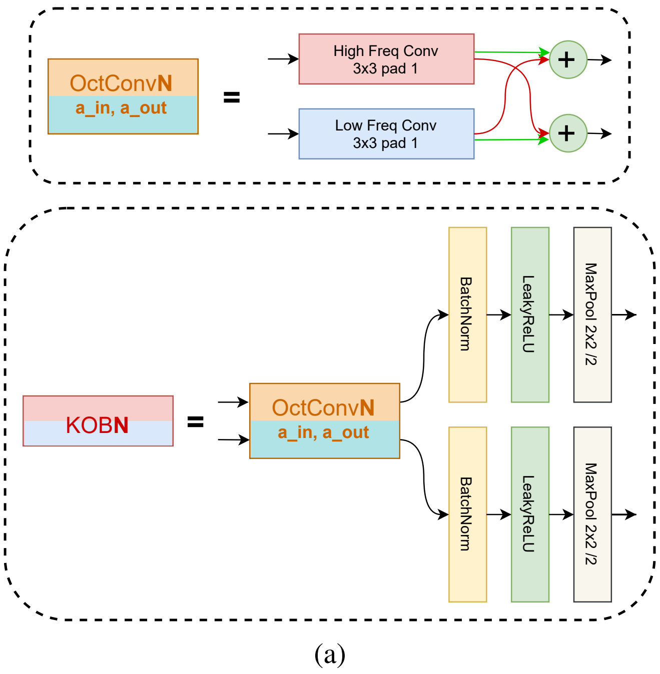


Figure 5 – (a) The KutralNet Mobile main block, the KutralMobileBlockN-E (KMBN-E), where N refers to the output channels number and E to the t value of expansion rate. (b) The KutralNet Mobile model with a KB32 block, three KMBN-E blocks, a shortcut connection, and a KBO block as the exit.

3.2.4 KutralNet Mobile Octave

The third and final portable model combines the previous two models, using the inverted residual block structure [21] with the octave convolution [22] strategy, and was named **KutralNet Mobile Octave**. Like the KMBN-E block uses separable depthwise convolution, the developed octave version replaces the kernel sizes, resulting in new blocks named OctConvPN for the pointwise 1x1 convolution and OctConvDN for the depthwise convolution. The KMBN-E block modified version is named KutralMobileOctaveBlockN-E (KMOBN-E), where N-E refers to the output channels number and the expansion rate t, respectively. It comprises an OctConvPN block, twins of batch-normalization, twins of LeakyReLU activation, OctConvDN block, twins of batch-normalization, twins of LeakyReLU activation, another OctConvPN, and finally twins batch-normalization layers. For the first OctConvPN and OctConvDN, the N number of output channels is given by multiplying N output number channels by E expansion rate value from the KMOBN-E block. These main blocks are shown in Figure 7a.

KutralNet Mobile Octave constitutes a KBN block, followed by three KMOBN-E blocks and a KBO block. Additionally, a shortcut connection process the model's signal from the first KMOB64-4 block through twins max-pooling layers, twins bath-normalization layers and, an OctConvPN block to the final KBO block.



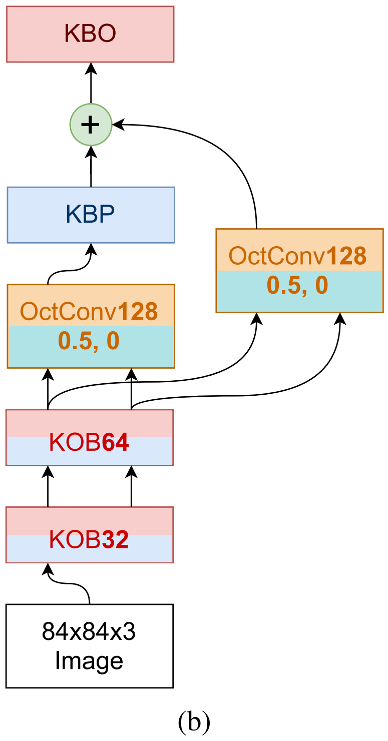
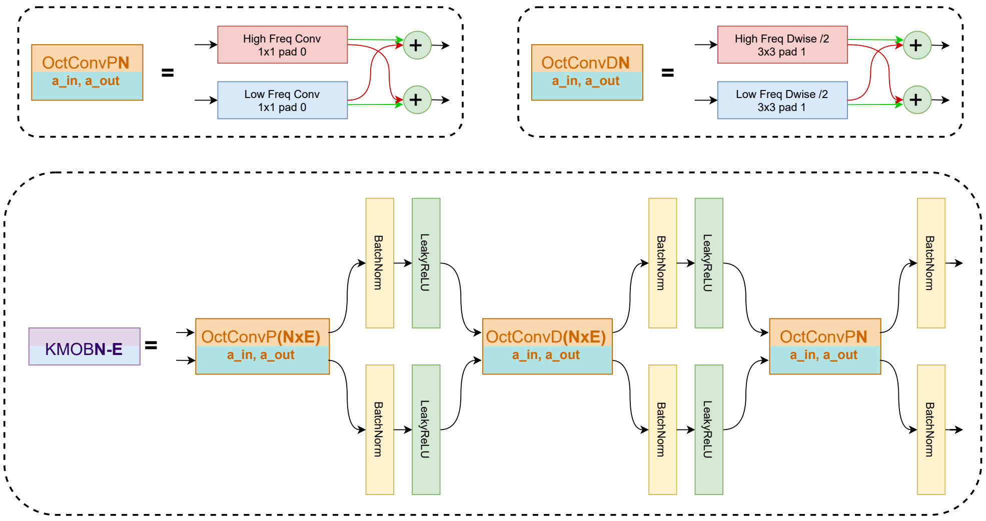


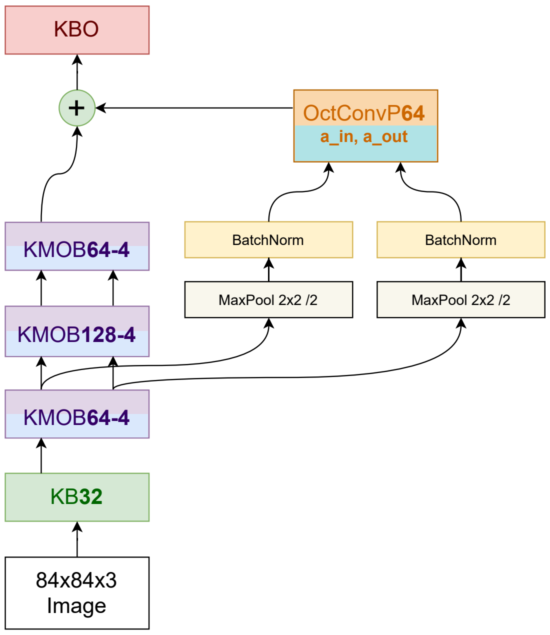
Figure 6 – (a) The KutralNet Octave main blocks, the upper block, is a simplified version of the Figure 2b, named OctConvN. The lower block is the KutralOctaveBlockN (KOBN), where N refers to the number of output channels passed to the OctConvN. (b) The KutralNet Octave model with two KOBN blocks, two OctConv128, a KBP block, a shortcut connection, and a KBO block as the exit. The input and output for the OctConvN block, when $\alpha = 0$, just the high-frequency convolution is used, represented with the left-side arrow from one block to another.

3.3 KutralNext: Multi-label fire and smoke recognition model

All of the previous methods were considered using a single-label fire-flame classification task, indicating if there is a fire presence in the images or not. In this third and final proposal, a multi-label fire and smoke recognition task extends the KutralNet proposal called KutralNext. In terms of architecture, no changes were made in this proposal, and the main changes rely on the classifier exits of the KBO block. For KutralNet, one exit was used for the positive case and the other for the negative case of fire presence, being mutually exclusive. For KutralNext, the first exit indicates fire presence in the image, and the second exit indicates if there is smoke present in the image, being complementary. Additionally, KutralNext uses a pretrained version of KutralNet, and KutralNet Mobile Octave with ImageNet explained next. Those models were named KutralNext and KutralNext+, both chosen from previously obtained results, demonstrating good performance in fire recognition trained from scratch. The models were adjusted for fire and smoke recognition using the Class Balanced loss function, explained later. Experiments have demonstrated that the multi-label approach, in addition to recognizing smoke in the image, it also improves the model's capability to acquire fire's features.



(a)



(b)

Figure 7 – (a) The KutralNet Mobile Octave main blocks, the KutralMobileOctaveBlockN-E (KMOBN-E), where N refers to the number of output channels and E for the t value of expansion rate. OctConvPN and OctConvDN are the octave convolution version for the separable depthwise convolution layers. (b) The KutralNet Mobile Octave model with a KBN block, three KMOBN-E blocks, a shortcut connection, and a KBO block as the exit. The left-side connection from one block to another represents the $\alpha = 0$ value for the input or output of octave convolution.

3.3.1 ImageNet Pretraining

One of the challenges in deep learning model developments is the huge amount of data required for training. In this regard, using pretrained models over a challenging dataset with a considerable quantity of instances and labels improves the results using transfer learning and fine-tuning, reducing the data required to learn filter kernels to acquire valuable information from a high dimensional input.

For this purpose, we use the ImageNet ILSVRC 2012 dataset [37], which comprises 1.3 million instances with 1,000 classes, designed for a classification and detection competition, being widely used as a models' performance benchmark. Many classical DL models such as ResNet and EfficientNet have been trained with ImageNet and are publicly available in different repositories to be used by the community. We use the ImageNet dataset to training the baseline, and the efficient architectures for later use in the fire and smoke classification task.

3.3.2 Class Balanced Loss

As a dataset grows, focused on obtaining more instances of those classes of interest, it is much more likely to have a long-tailed distribution with many underrepresented classes. A novel framework is implemented in our proposal to deal with this class imbalance issue, which uses the effective number of samples or expected volume of samples to define each class's impact on the loss value. This method is named class balanced loss [38], and defines the effective number of samples as $(1-\beta^n)/(1-\beta)$, where n is the number of samples and β an hyper-parameter $\in [0,1]$ which control how fast the effective number of samples grows as n increases. This loss function's main idea is to introduce a class weighting factor inversely proportional to the effective number of samples to balance the output loss value as a model- and loss-agnostic method, formulated as

$$\text{CB}(p,y) = \frac{1-\beta}{1-\beta^{n_y}}\mathcal{L}(p,y), \tag{3.1}$$

where n_y is the number of samples for the class y, $\mathcal{L}(p,y)$ is the loss function for the predicted class probability p.

In our proposal, the $\mathcal{L}(p,y)$ loss function is replaced by the focal loss (FL) [39], which is an α-weighted method to address the class imbalance issue, defining each class impact in the loss value with $\alpha \in [0,1]$ for the target class y, and $1-\alpha$ for the other classes, defined as follows

$$\text{FL}(p_y) = -(1-p_y)^\gamma \log(p_y), \tag{3.2}$$

where p_y is the probability of the y class, $(1-p_y)^\gamma$ is a modulating factor with a $\gamma \geq 0$ hyper-parameter to determine how smoothly it affects the loss function, focusing in difficult samples. Each p_y class probability at the exit of the models is represented by the sigmoid cross-entropy loss denoted by

$$p_y = \frac{1}{(1+\exp -z_y)}.$$

In this regard, our implementation includes the base sigmoid cross-entropy loss, with the datasets classes weighted by the focal loss, and defining each class impact by the class balanced loss, formulated in next

$$\mathrm{CB}_{\text{focal}}(z, y) = -\frac{1-\beta}{1-\beta^{n_y}}(1-p_y)^{\gamma}\log(p_y), \tag{3.3}$$

where z is the model's predicted class probability.

Chapter 4

Experimental Setup

The environment used to train and test each model was an online open cloud platform for machine learning algorithms. This online platform provides a ready-to-use ecosystem with libraries for data manipulation, data visualization, and the training process, among others. The environment is available through a virtual machine configured with up to 13GB of memory, an Intel Xeon@2.30GHz, and an NVIDIA GPU with 12GB of memory.

4.1 Datasets

Due to the fire and smoke recognition task's complexity, mainly because of some similarities with fire-like objects, a suitable dataset is required for training the proposed models. In this regard, a fair trade-off between images with and without fire or smoke presence is needed to carry out a good training process. Moreover, it has to be considered images with fire-like color objects labeled as no-fire. Five datasets have been selected from the literature review to check the model's training process's suitability. Four of the datasets have been used in previous works, and another one has been recently compiled. Custom names have been given to the five previously used datasets, FireSense[4] [14], CairFire[5] [24], FireNet[6] [32], FiSmo[7] [40], and FireSmoke[8] [41]. All datasets were previously used and tested in their corresponding works, except for FiSmo, which was only published and detailed as a fire and smoke labeled dataset. More details of each image dataset will be discussed below.

The FireSense dataset is a video compilation that contains 27 videos for fire detection and 22 videos for smoke detection. From the fire detection videos are 11 videos with fire presence and 16 videos without it. Moreover, the smoke detection videos are 13 videos with smoke presence and 9 videos without it. From this compilation, just the fire detection videos were used.

[4] Online available at <https://zenodo.org/record/836749> [Accessed: July, 2019]

[5] Online available at <https://github.com/cair/Fire-Detection-Image-Dataset> [Accessed: July, 2019]

[6] Online available at <https://github.com/arpit-jadon/FireNet-LightWeight-Network-for-Fire-Detection>, [Accessed: July, 2019]

[7] Online available at <https://github.com/mtcazzolato/dsw2017> [Accessed: July, 2019]

[8] Online available at <https://github.com/DeepQuestAI/Fire-Smoke-Dataset> [Accessed: July, 2020]

Therefore, frame extraction has been performed for the training of the model. Just one frame per second was obtained from these videos, getting a total of 906 frames.

For the CairFire dataset, the authors generated the dataset by selecting images from the internet. They present images with different fire scenarios, indoor and outdoor, and different illumination types as fire-like colors. The dataset is highly unbalanced and contains 110 images with fire presence and 541 images without it.

The FireNet dataset is a recent compilation of challenging images with and without fire presence. The authors complement the datasets used in previous works with internet images to make them more diverse. They produce a dataset summarizing a total of 2,425 images for training and 871 images for testing purposes.

In the FiSmo dataset, the authors also created a compilation of images from other datasets obtaining 6,063 images. The source datasets used for FiSmo are in the context of the RESCUER Project[9]. One of the subsets is called Flickr-FireSmoke, which has 5,556 images with fire and smoke labels in total. Another subset is Flickr-Fire, which present balanced quantities of images between fire and no-fire images from Flickr-FireSmoke, adding 281 other images with the presence of fire. Additionally, from the BoWFire dataset [42], which is included in the compilation, just the testing subset with 226 images is used as part of the dataset. This selection is made because the training subset is meant to training a pixel-value fire recognition algorithm. From FiSmo also, the contained subset of FiSmo (FiSmoB), which comprises 1968 images equally balanced between the fire and no-fire label, was used independently. An augmented version of FiSmo (FiSmoA) is also used, adding 485 black images labeled as no-fire to check out the models' response to this kind of augmentation. In addition to the balanced FiSmo version, we have also used an augmented version of this subset (FiSmoBA), which replaces 98 no-fire images for black images.

The FireSmoke dataset presents 3,000 internet compiled images for fire, smoke, and neutral classes with 1,000 images each. The authors propose the use of 900 for training and 100 for testing of each class. Nevertheless, in this work, the entire dataset was used. The labels details for each dataset are present in Table 1.

As all the datasets were compiled for a single label approach, the FireNet, FiSmo, and FireSmoke datasets were selected to convert each label's instance to a multi-label approach given the total number of images and the complexity in the data representation for both fire and smoke labels. In this regard, 16,140 images were reviewed and relabeled with four kinds of labels: a) none, for those images with no fire neither smoke presence; b) fire, for those images with only fire presence; c) smoke, for those images with only smoke presence; d) fire and smoke, for those images with the presence of both fire and smoke. After images' label checking, the FireNet and FireSmoke datasets were highly unbalanced, merged into a new one named

[9] Project FP7-ICT-2013-EU-Brazil - "RESCUER - Reliable and SmartCrowdsourcing Solution for Emergency and Crisis Management"

Table 1 – Quantity of images present in each dataset for fire recognition task.

Dataset	Use	Fire	No-Fire	Total
FireSense	training	329	577	906
CairFire	training	110	541	651
FireNet	training	1,124	1301	2,425
FireNet	testing	593	278	871
FiSmo	training	2,004	4,059	6,063
FiSmoA	training	2,004	4,544	6,548
FiSmoB	training	984	984	1,968
FiSmoBA	training	984	984	1,968
Total		4,441	6,070	10,511

Note: the fire and no-fire totals row differs in addition to each row value given the FiSmo subset variations, from which a few images were added, as explained in section 4.1

Table 2 – Quantity of images present in each dataset used for fire and smoke recognition task.

Dataset	Set	Fire & Smoke	Fire	Smoke	None	Total
FireNet	training	750	352	46	1,277	2,425
FireNet	testing	55	537	1	278	871
FireSmoke	training	677	247	862	914	2,700
FireSmoke	testing	64	39	93	104	300
KutralSmoke	training	1,427	599	908	2,191	5,125
KutralSmoke	testing	119	576	94	382	1,171
FiSmo	training	795	1,267	384	3,617	6,063
FiSmoA	training	795	1,267	384	4,102	6,548
Total		2,341	2,442	1386	6,675	12,844

The FireNet, FireSmoke, and FiSmoA datasets are not considered in the total row because FireNet and FireSmoke are contained in the KutralSmoke dataset, and FiSmoA is contained in the FiSmo dataset, except for the none label's difference.

KutralSmoke, summarizing 5,125 training and 1,171 testing images. Therefore, in this fire and smoke recognition task, the FiSmo relabeled and KutralSmoke datasets were used in this task with a total of 12,359 images. The label distribution for each checked and used dataset are shown in Table 2, and image samples are shown in Figure 8.

4.2 Performance evaluation

Six different proposals were developed in this work, OctFiResNet, KutralNet, and three portable versions, KutralNext and KutralNext+. Each proposal was developed for a different purpose, along with this work. OctFiResNet was developed as the first lightweight ResNet-inspired model to check the datasets' complexity for use in the fire recognition task. KutralNet

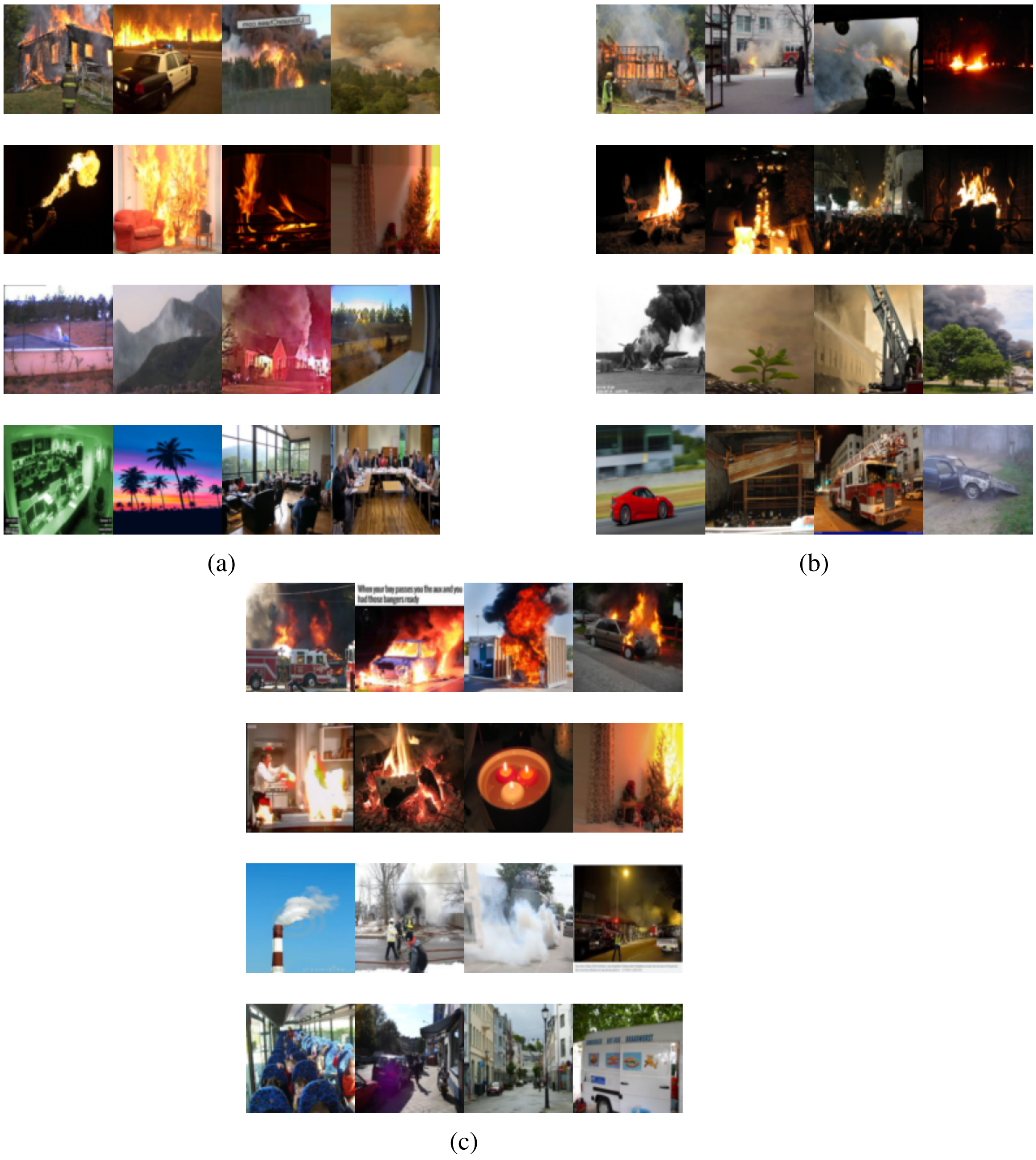


Figure 8 – Data samples for each dataset used in this work. (a) FireNet image samples, (b) FiSmo image samples, (c) KutralSmoke image samples. In the first row are the Fire and Smoke labeled images, the second row are the fire labeled images, the third row are the smoke labeled image, and in the bottom row are the none labeled images.

was developed as a low-complexity model to recognize fire and later build more efficient models using deep learning techniques as the octave and depthwise convolutions, with the inverted residual block. Finally, the KutralNext and KutralNext+ proposals extend the KutralNet and KutralNet Mobile Octave models, respectively, trained over the ImageNet dataset to be optimized in a fire and smoke multi-label recognition task. All of the models were developed with the Python programming language.

4.2.1 OctFiResNet proposal

For the OctFiResNet model's training, the FireSense, CairFire, FireNet, and FiSmo datasets were used. This model was implemented using the Keras framework with TensorFlow as a backend for the neural network approach. The images were normalized to make their values $\in [0, 1]$ before being processed by the model. The model was trained during 100 epochs with Adam optimizer with Nesterov momentum with a learning rate of $\alpha = 0.0001$.

Each dataset was used for training, validation, and testing the proposed model to achieve the best possible dataset comparison. For example, using cross-dataset validation, the entire FireSense dataset was used for training and, the entire Cairfire, FireNet, and FiSmo dataset were used separately for validation. Therefore, each dataset obtains four different training results, one by the split dataset for training and validation, and the other three using cross-dataset validation.

When the same dataset was used, it was split into two different subsets, one for training and the other for validation. For the FireSense and FiSmo datasets, the 80% of the images were used for training purposes and 20% for validation. For CairFire and FireNet datasets, the same divisions made by the authors at [24] and [32] were used, respectively. In this regard, for the CairFire dataset, 549 images were used for training and 102 images for validation. From the training subset, 59 images had fire presence, and 490 did not have. The validation subset was composed of 51 images with fire presence and 51 images without it. For the FireNet dataset, the 70% was used for training, while the other 30% of the images was used for validation. For statistical analysis, the algorithm has been executed ten times for each dataset.

4.2.2 KutralNet architectures

The first experiment with KutralNet aimed to define a baseline model and prove its effectiveness to fire recognition. For this purpose, three different models were implemented for comparing the baseline. The first model is a novel lightweight model for fire recognition, the FireNet model [32], which is implemented in a Raspberry Pi as part of a fire alarm system. A second model is a modified version of ResNet50 used with the transfer-learning technique for training the classifier on the top of the network with fire and no-fire images. In this proposal experimentation, OctFiResNet is also included to test its generalization capabilities, based on the ResNet model, with few layers and octave convolutions. The final implemented model is our proposal, KutralNet, to address a computationally efficient and lightweight deep neural network, balancing between the parameters and effectiveness. All of the models were trained and evaluated over the FireNet, FiSmo, FiSmoA, FiSmoB, and FiSmoBA datasets and developed with the PyTorch library. The trained and evaluated KutralNet is the baseline to build on the portable approaches.

With the defined baseline, the next set of experiments aimed to reduce the operations required for processing the images for fire recognition of the KutralNet model. To reduce the operations required, techniques such as the depth-wise convolution presented with an inverted residual block in [21] and the octave convolution presented by Chen et al. [22] were implemented separately at first and mixed later to check their compatibility and efficiency. This reduction results in three different models named: (i) **KutralNet Mobile** for the one where uses the inverted residual block, inspired in MobileNetV2; (ii) **KutralNet Octave** is called the model which uses the octave convolution with a reduction parameter $\alpha = 0.5$ combined with the depth-wise convolution, requiring fewer operations and space to store the model; and (iii) **KutralNet Mobile Octave**, which uses the inverted residual block in combination with the octave convolution using the depth-wise form.

All of the models' training was performed during 100 epochs to choose the model with the best validation accuracy. All of them were trained using cross-entropy loss and the Adam optimizer with default parameters, except for KutralNet, which during the training stage, presenting a learning rate variation from $\alpha = 10^{-4}$ to $\alpha = 10^{-5}$ on the epoch 85.

4.2.3 KutralNext methodology

The experimentation purpose of KutralNext is an extension of KutralNet where the baseline architecture and the Mobile Octave variation were used to be pretrained using the ImageNet dataset. Those models were chosen due to their obtained results in the fire recognition task, named KutralNext and KutralNext+ for the baseline and the best portable variation to be used in a fire and smoke recognition under a multi-label classification task. The JaukeNet proposals were compared against previously developed models FireNet and FireDetection, including the OctFiResNet proposal, using the relabeled version of FireNet, FiSmo, FiSmoA, and KutralSmoke datasets. All previous models were developed in a lightweight scope to be used in a single-label fire recognition task, with FireDetection as the only model previously used for both fire-only and fire and smoke recognition tasks.

The experimental setup compares the fire and smoke recognition performance of the KutralNext architectures against fire-specialized models. The first one compares the single-label fire classification task performance, where each model must inference the presence or not of fire in images. For this case, just the fire label is used from the image datasets, codifying the target label y into a two components vector $\in [0, 1]$. When the first component is equal to 1, it indicates no fire presence, and when the second component is 1, it indicates fire presence, being able to just one of them be equal to 1. The second experiment is the multi-label fire and smoke classification task, where each model must inference the presence of fire, smoke, or none in images. For this kind of task, the fire and smoke labels were used from each dataset, codifying the target label y as vector $\in [0, 1]$, with one component for each class. In this case, the target label can represent the fire and smoke presence with both components equals 1, and both components equal 0, to represent neither fire nor smoke presence. In both experiments, the behavior was checked under two different data representations and distribution using a cross-dataset test, proving each model's robustness.

All of the images were preprocessed with a resize transformation to fit each model's input size and normalized with values $\in [0, 1]$. For each model, a different loss function was used in order to follow their original implementation. For FireNet, OctFiResNet, and FireDetection, a cross-entropy loss was used with a softmax activation in the single-label experimentation and a binary cross-entropy loss with the sigmoid activation in the multi-label approach.

Chapter 5

Experimental results

The following sections are presented each experiment carried out to obtain efficient models to fire and smoke recognition. In the first place, different datasets were analyzed with OctFiResNet, a first lightweight model approach for fire recognition. A quantitative and qualitative analysis was performed in four previously used datasets, using the validation accuracy obtained during the training of OctFiResNet. FiSmo was a remarkable, challenging dataset due to its large number of instances, the contained subsets, and the smoke-labeled images. In the second place, an deep learning model aims to be reduced in size and effectively recognize fire in still images. In this regard, the KutralNet architectures were built as an efficient approach, with a baseline convolution layers stack to reduce the mathematical complexity later using state-of-the-art convolution techniques. Three portable models were developed from the combination of the inverted residual block, the separable depthwise convolution, and the octave convolution. The best portable model was named KutralNet Mobile Octave. The KutralNet architectures were extended, used for fire and smoke recognition under a multi-label approach in the last place. For this, the KutralNet and KutralNet Mobile Octave models were pretrained over ImageNet, naming the models as KurtalNext and KurtalNext+. In this experimentation, a new loss function was implemented to deal with the unbalanced datasets' labels. A multi-label approach was used given the existence probability of the fire or smoke in an image and because they are not mutually exclusive classes.

Overall all the models were measured using the validation and testing accuracy, the Receiver Operating Characteristic (ROC) curve, the area under the ROC curve, the floating-point operations (flops), the number of parameters, and the time required to process the entire test dataset used for each experiment. Those metrics were selected to compare each model generalization and acquisition of the fire and smoke features under the same and different data distributions. Each used model is represented in Table 3, by the computational cost in terms of flops and parameters.

Table 3 – The computational cost of each model used in this work represented with flops and parameters ordered by parameters number.

Model$_{InputSize}$	**Flops**	**Parameters**
KutralNext$_{84x84}$	76.85M	138.91K
KutralNext+$_{84x84}$	24.59M	185.25K
FireDetection$_{224x224}$ [29]	783.50M	335.53K
FireNet$_{64x64}$ [32]	8.94M	646.82K
OctFiResNet$_{96x96}$	928.95M	956.23K

Table 4 – Averaged OctFiResNet results with cross-dataset validation.

Dataset	**FireSense**	**CairFire**	**FireNet**	**FiSmo**
FireSense	**100%** $\pm$ **0.0%**	82.87% $\pm$ 0.89%	67.84% $\pm$ 2.87%	69.71% $\pm$ 3.21%
CairFire	83.43% $\pm$ 2.08%	**90.78%** $\pm$ **1.61%**	90.65% $\pm$ 2.08%	79.59% $\pm$ 1.21%
FireNet	85.43% $\pm$ 2.29%	100% $\pm$ 0.0%	**95.47%** $\pm$ **0.34%**	81.90% $\pm$ 0.69%
FiSmo	88.29% $\pm$ 2.58%	94.09% $\pm$ 0.50%	84.94% $\pm$ 0.81%	**87.44%** $\pm$ **0.42%**

Average results from 10 times of training execution obtained with cross-dataset validation. For the results obtained in FireSense and FiSmo with themselves, a dataset split of 80%/20% was used for training and validation, respectively. For the CairFire, 549 images were used for training and, 110 images for validation. For FireNet a 70%/30% split was used. In rows are listed the training used datasets, and in columns are the validation used datasets.

5.1 OctFiResNet: A first lightweight approach

The first obtained results are related to the OctFiResNet model and the FireSense, CairFire, FireNet, and FiSmo datasets. The cross-dataset validation methodology aimed to define the best dataset option to be used in fire recognition. An entire dataset was used for training and a second one for validation to measure the generalization capability during the model's optimization. Additionally, the training and validation stages were also implemented using a single dataset split for both stages to check the data representation's complexity under the same data distribution. The averaged results obtained by OctFiResNet on each dataset are shown in Table 4.

After training, the overall obtained validation results show excellent performance in each dataset, representing an interesting improvement by OctFiResNet, considering the dataset's original works' previously reported outcomes. In the case of FireSense, an accuracy of 95.27% has been previously reported by Dimistropoulos et al. [14] with their method analyzing the image through sliding windows of $8x8$ pixels to perform the classification. OctFiResNet was able to improve the reported results with a 100% of validation accuracy for the 20% of the extracted video frames. This remarkable accuracy can be originated from the high similarity between each frame. Using CairFire, a modified version of the ResNet50 [24] obtained a 92.15% validation accuracy against the 90.78% average validation accuracy of OctFiResNet. Concerning FireNet, Jadon et al. [32], and their lightweight model was able to obtain a 93.91% validation accuracy, and OctFiResNet achieved a 95.47% average validation accuracy. Finally, for FiSmo, the authors have only presented the dataset. To the best of our knowledge, no work has been presented yet using this dataset. Therefore, no comparison has been performed. Regardless, the OctFiResNet model obtains an average validation accuracy of 87.44% and a maximum of 88.38% for the 20% of the dataset.

5.1.1 Discussion

Given the obtained results, OctFiResnet achieved high accuracy on datasets with a reduced number of images. However, datasets with a greater variety of fire-like objects in the images are more difficult to generalize. Furthermore, the reduced number of parameters does not affect the overall performance negatively. From the addressed datasets, FiSmo presents a more challenging number of images compared to the others, being suitable to be used alongside the FireNet dataset for a fire classification task given its high volume data. Thus, the FiSmo dataset is suitable given to the previously labeled smoke data, which can be used as an extension from the fire recognition to a fire and smoke classification task.

5.2 KutralNet: An efficient fire recognition model

The following two subsections separate the KutralNet architectures' experiments to achieve a portable deep learning model for fire recognition. The first experimentation checks the proposed baseline model's feasibility to recognize fire, using few convolutional layers to process the images. Experimental comparisons of KutralNet were made against the FireNet, OctFiResNet, and a modified ResNet50 deep learning models. After check the baseline performance, the final experimentation allowed to optimize the computational cost of the model, exploring the benefits of different techniques presented in previous years as the inverted residual block, the depthwise, and octave convolution. The different portable proposals got almost the same accuracy as the baseline model with fewer parameters or flops, as shown in Table 3.

5.2.1 Baseline comparison

The baseline comparison was performed with three models to improve the proposal's results, focused on being lightweight and efficient. The first compared model was FireNet from Jadon et al. [32], which comprises just a few convolution layers as part of a fire alarm system. The second model was presented by Sharma et al. [24], where a pretrained ResNet50 makes the feature extraction to be later analyzed by a multilayer perceptron with 4096 hidden units and infer the corresponding label. Finally, the proposed OctFiResNet model is also used, as a reduced version of ResNet50, presenting a few layers and replacing the vanilla convolution by the octave convolution.

The first results were compared with the models trained over the FireNet dataset, obtaining a validation accuracy of 93.83%, 96.02%, 95.34%, and 98.22% for FireNet, KutralNet, OctFiResNet, and ResNet50, respectively. Correspondingly, the test accuracy results are 88.98%, 83.70%, 88.18%, and 89.44% for FireNet, KutralNet, OctFiResNet, and ResNet50. FiSmo dataset was used to check the models' generalization under a different data distribution during training and validation. In general, the validation and testing accuracy were lower for all the models, with KutralNet as the best model in both terms. Test results were obtained using the test purpose FireNet's subset.

A final experiment was to evaluate the models' prediction with a black image as input. All of the models trained with FireNet and FiSmo datasets miss-classified the black image with the fire label. The FiSmo dataset was augmented to deal with this miss-classification issue, adding 10% of the no-fire labeled images with black images. This kind of augmentation showed useful improvements in training and testing stages, as can be observed in Table 5 and Figure 9c. The most in-depth models outperform the FireNet model's results, except for OctFiResNet, which negatively affects test accuracy.

The test performances of each model trained with different datasets are shown in Figure 9. As shown, comparing Figure 9b and Figure 9c, the black images added into the FiSmo dataset showed a positive reaction for the ROC curve in most models. Regarding the AUROC index, KutralNet and ResNet50 models present an improvement; for FireNet, a diminishment and OctFiResNet remains almost the same. FireNet achieves the best value in all the datasets, but as presented in Table 5 for FiSmo and FiSmoA achieves a low test accuracy.

To visualize the models' comparison more straightforwardly, the bottom of Table 5 presents the validation accuracy, test accuracy, and AUROC index for each model trained with different datasets averaged. Overall, KutralNet presents a better generalization during testing than all the other models, achieving high performance with fewer parameters and operations.

The proposed KutralNet baseline accomplishes an interesting performance compared with previous deep models for fire recognition. This model presents a few convolution layers in order to acquire a feature representation of fire in images. A model with a few numbers of layers consequently present a reduced number of parameters and operations required for this task. The resultant baseline reduces 85% the parameters number and 92% the number of operations required compared to the OctFiResNet model to process an image signal of 84x84 pixels in RGB channels.

5.2.2 Portable versions

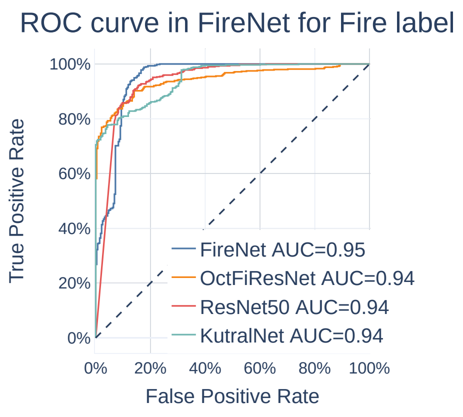
Once checked the KutralNet baseline architecture, the next experiment aims to reduce its computational cost. For this purpose, the convolutional layers were modified from the baseline, resulting in three different models to check the most efficient strategy. The first model,

Table 5 – KutralNet baseline comparison results against previous fire recognition models.

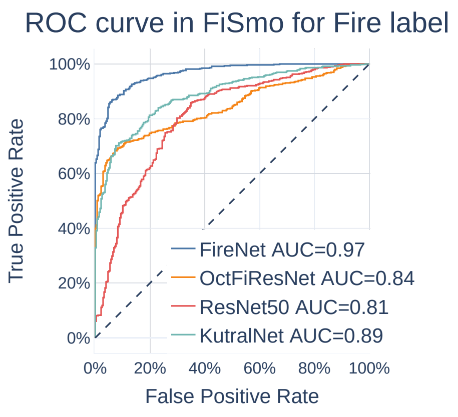
DS	Model	Val. acc.	Test acc.	AUROC
FireNet	FireNet [32]	93.83%	88.98%	**94.63%**
	KutralNet	96.02%	83.70%	94.49%
	OctFiResNet	95.34%	88.17%	94.31%
	ResNet50 [24]	**98.22%**	**89.44%**	93.62%
FiSmo	FireNet [32]	84.50%	58.90%	**96.60%**
	KutralNet	**88.62%**	**74.63%**	88.62%
	OctFiResNet	87.96%	72.79%	84.28%
	ResNet50 [24]	87.47%	40.53%	80.56%
FiSmoA	FireNet [32]	85.65%	59.93%	**95.25%**
	KutralNet	89.54%	76.46%	92.69%
	OctFiResNet	89.24%	66.93%	84.06%
	ResNet50 [24]	**92.06%**	**80.83%**	94.00%
Average	FireNet [32]	87.99%	69.27%	**95.49%**
	KutralNet	91.40%	**78.26%**	91.93%
	OctFiResNet	90.85%	75.97%	87.55%
	ResNet50 [24]	**92.58%**	70.26%	89.39%

KutralNet Mobile, replaces the baseline structure to get the inverted residual blocks with depthwise convolution, as proposed in [21], simplifying the processing operations required. The second model, **KutralNet Octave**, replaces the vanilla convolution from the baseline with the octave convolution [22] for signal processing. The third model, **KutralNet Mobile Octave**, implements a combination of previously mentioned techniques, the inverted residual block with the octave convolution to replacing the baseline structure.

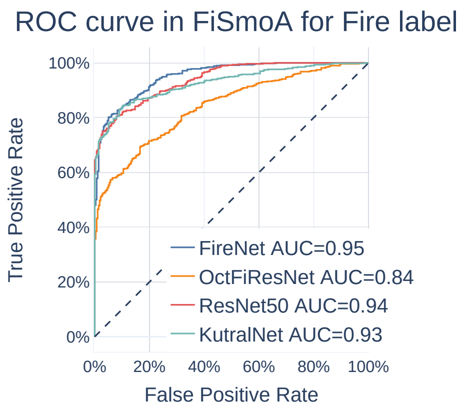
In the first place, the training was performed over FiSmo, achieving a validation accuracy of 88.62%, 85.99%, 87.55%, and 87.39% by KutralNet, KutralNet Mobile, KutralNet Octave, and KutralNet Mobile Octave, respectively. Additionally, the test accuracy obtained from the models over the test FireNet's subset was 74.63%, 67.28%, 72.33% and, 72.91%, respectively. All of the obtained results are shown in Table 6 for each model trained over all the datasets. As same that with the baseline, the black image test was carried out in order to check the reliability of features obtained from the input signal. For this purpose, the balanced augmented version of FiSmo, FiSmoBA dataset was used for training the model, getting the lowest miss-classification error in all the trained models. Additionally, the models got $\pm1\%$ of validation accuracy difference compared with the balanced FiSmoB dataset. For the black image test, the KutralNet Mobile with octave convolution got the lowest miss-classification rate with 10%, 30%, and 0% for the FiSmo, FiSmoB, and FiSmoBA datasets, respectively. Overall, as shown in Figure 10, the KutralNet Mobile Octave performs well in different variations of the FiSmo dataset. Additionally, the AUROC index is even better than the baseline with the balanced version of the dataset and the augmented balanced version. For the KutralNet Octave case, it performs better than the Mobile Octave version with FiSmo and its augmented balanced version. Considering the



(a)



(b)



(c)

Figure 9 – The models' test results with FireNet-Test, which comprises 871 images for fire classification. All of the models were trained under different datasets and tested over the test subset of FireNet. The augmented dataset, FiSmoA, presents better results than FiSmo for all the models, in exception for OctFiResNet. (a) ROC curve for the models trained over the FireNet dataset. (b) ROC curve for the models trained over the FiSmo dataset. (c) ROC curve for the models trained over the FiSmoA dataset. In overalll, FireNet presents the best AUROC value but with low test accuracy. KutralNet performs the second-best AUROC value achieving the best test accuracy, followed by OctFiResNet and ResNet50.

trade-off between parameter numbers and operations required for processing the image, the Kutralnet Octave presents a solution with fewer parameters than KutralNet Mobile Octave and, conversely, requires more operations.

5.2.3 Discussion

As has been demonstrated, the proposed efficient KutralNet baseline can achieve a competitive accuracy in both terms of validation and test accuracy than the modified version of ResNet50. However, the KutralNet proposal was trained from scratch compared to ResNet50, previously optimized with the ImageNet dataset to learn complex features, and then be used with transfer-learning to fire recognition task. The ResNet50 model is a generic-purpose deep

Table 6 – KutralNet portable versions results comparison.

DS	Model	Val. acc.	Test acc.	AUROC
FiSmo	KutralNet	**88.62%**	**74.63%**	88.62%
	KutralNet Octave	87.55%	72.33%	**93.20%**
	KutralNet Mobile	85.99%	67.28%	81.12%
	KutralNet Mobile Octave	87.39%	72.90%	85.87%
FiSmoB	KutralNet	**95.18%**	75.32%	83.12%
	KutralNet Octave	95.18%	67.74%	71.07%
	KutralNet Mobile	94.67%	74.28%	89.77%
	KutralNet Mobile Octave	93.65%	**84.62%**	**92.55%**
FiSmoBA	KutralNet	93.40%	78.07%	85.75%
	KutralNet Octave	92.64%	**81.63%**	**92.41%**
	KutralNet Mobile	93.91%	74.40%	84.02%
	KutralNet Mobile Octave	**93.91%**	80.94%	90.53%
Average	KutralNet	**92.40%**	76.00%	85.83%
	KutralNet Octave	91.79%	73.90%	85.56%
	KutralNet Mobile	91.52%	71.99%	84.97%
	KutralNet Mobile Octave	91.65%	**79.49%**	**89.65%**

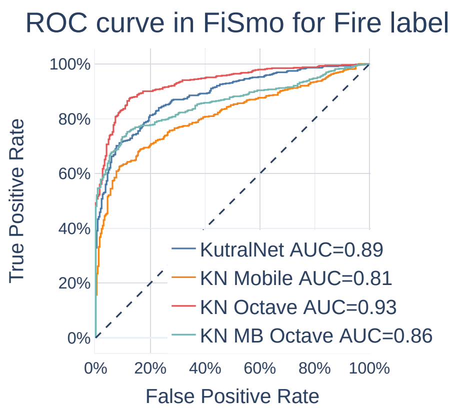
learning model and presents a huge computational cost to obtain $\pm 2\%$ difference in test accuracy. KutralNet has shown to be able to generalize better than ResNet under a different data distribution, surpassing the capability to obtain fire's features.

The Octave and Mobile Octave versions performed comparable results against the baseline in terms of validation accuracy and outperform test accuracy and AUROC values. It was demonstrated that octave convolution achieves rich fire features, but it is volatile to the variation in the input data due to the variation in AUROC terms between FiSmoB and FiSmoBA. Nevertheless, the black image augmentation affected the Mobile Octave version's generalization negatively. However, it surpassed the baseline under a balanced configuration, requiring no augmentation to perform well.

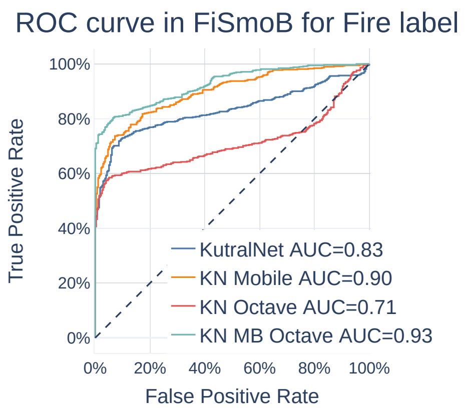
How it was proven, a previously optimized model use can increase the effectiveness in any specific-task, given the prior complex learned features from a challenging dataset as ImageNet. The next experiment addresses this pretrained model over KutralNet and KutralNet Mobile Octave, given the seized results under an efficient scope. Additionally, the balanced data issue is also addressed to take over a better training process.

5.3 KutralNext: Fire and smoke recognition

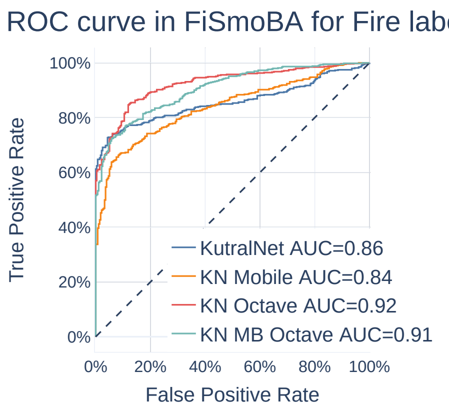
In the following section, a fire and smoke recognition problem is addressed under a multi-label approach. For this purpose, the KutralNet and KutralNet Mobile Octave architectures were pretrained over the ILSVRC 2012 dataset to learn useful filters and acquire images' features



(a)



(b)



(c)

Figure 10 – The models' test results with FireNet-Test, which comprises 871 images for fire classification. The models' results are from the training with different datasets. The FiSmo, FiSmoB, and the FiSmoBA with 98 no-fire images replaced with black images. The augmented dataset, FiSmoA, presents better results than FiSmo for all the models. (a) ROC curve for the models trained over the FiSmo dataset. (b) ROC curve for the models trained over FiSmo balanced dataset. (c) ROC curve for the models trained over the augment balanced FiSmo dataset. KutralNet performs the second-best AUROC value achieving a better test accuracy, followed by OctFiResNet and ResNet50.

to then be optimized against FiSmo, FireNet, and KutralSmoke datasets to conduct a comparison of their suitability for single- and multi-label classification tasks. The models were named KutralNext and KutralNext+, respectively. Different data distribution from more than one data source is affordable to check the models' generalization capability to acquire the features and recognize the fire or smoke presence in an image. In this way, and as aforementioned, two types of experiments were carried out. The first subsection explains the results obtained during the single-label fire recognition task. The second subsection describes each model's results in recognizing fire and smoke in still images as a multi-label classification task. A third subsection discusses how good were the models generalizing in both tasks and the benefits presented by the KutralNext+ model. All of the models were compared using the following metrics: the accuracy obtained during validation and testing, the receiver operating characteristic (ROC) curve, the area under the ROC curve (AUROC), the number of floating-point operations (flops) and, the time required to process all the images in the correspondingly testing dataset. In Table 3 are the cost of each model in terms of parameters and flops, Table 7 and Table 9 present the mean training results obtained during 5 executions with their standard deviation values for the single- and multi-label approaches, respectively.

5.3.1 Single-label classification: Fire recognition

The first experimentation set considers the single-label fire recognition task performance KutralNext compared against state-of-the-art fire recognition deep learning models. Table 7 shown the training and testing statistics results, and in Table 9 are the performance of each model over all the datasets with averaged and standard deviation values. Overall, for this task, the KutralNext+ proposal generalized the best, and KutralNext achieved the best performance acquiring fire's features, both against previous fire recognition models. The KutralNext proposals achieve the best results, being the most time inexpensive models in image processing.

From the results presented in Table 7, previous fire recognition models results are similar with OctFiResNet as the best accurate model during validation followed by FireDetection and FireNet, with FireDetection performing better than OctFiResNet in test accuracy. However, KutralNext has proven to achieve the best generalization trained over different data distributions as FiSmo and FiSmoA, where KutralNext+ obtain the best average validation and test accuracy. In terms of time, OctFiResNet is the model that requires about 3 seconds to process the test images, followed by FireDetection with 1.5 seconds, leaving KutralNext and KutralNext+ requiring 0.45 and 0.42 seconds, respectively. The FireNet model presents a lightweight approach found in the literature, and it processes the test dataset 29% faster than KutralNext+ with 0.30 seconds. However, it presents a difference of 4.96% and 14.15% in validation and test accuracy, respectively. All of the KutralNext models outperform all the previous fire recognition models in the single-label approach, demonstrating an efficient computational cost model to fire recognition. The best trained average KutralNext+ model is 0.29% and 5.51% higher than KutralNext, and is 1.83% and 2.65% higher than OctFiResNet in terms of validation and test accuracy, respectively.

Moreover, in Table 8, in terms of fire's feature acquisition for each model, the FireNet model obtain the best AUROC and precision among previous fire recognition models, followed by OctFiResNet and FireDetection. In this regard, it was demonstrated that a few layers acquire enough features to recognize fire. Nevertheless, they are not good enough in complex images where the fire is present. The KutralNext proposal performs the best in both AUROC and precision values for all the datasets, achieving 94.00% and 97.13%, respectively. KutralNext+ performs competitive results against KutralNext, with 93.64% and 97.03% for AUROC and precision, respectively. In this regard has been demonstrated the trade-off between the model's depth and efficiency to fire recognition. Additionally, KutralNext+ presents a similar result under a more efficient configuration, requiring less time to process an image.

Table 7 – KutralNext training comparison results during 5 executions in the fire recognition task.

DS	Model	Val. acc.	Test acc.	Test (ms)
FireNet	FireDetection [29]	94.99% ± 0.69%	86.59% ± 2.62%	1550 ± 26
	FireNet [32]	93.59% ± 0.45%	82.62% ± 4.43%	**301 ± 17**
	KutralNext	98.36% ± 0.38%	79.06% ± 4.61%	458 ± 27
	KutralNext+	**98.40% ± 0.22%**	**89.53% ± 2.66%**	414 ± 20
	OctFiResNet	97.04% ± 0.61%	82.73% ± 6.12%	3191 ± 39
FiSmo	FireDetection [29]	90.77% ± 0.21%	73.04% ± 3.67%	1543 ± 24
	FireNet [32]	87.63% ± 0.32%	62.69% ± 7.98%	**302 ± 20**
	KutralNext	92.03% ± 0.17%	77.43% ± 2.60%	450 ± 20
	KutralNext+	**92.44% ± 0.37%**	**80.62% ± 1.26%**	428 ± 14
	OctFiResNet	90.21% ± 0.49%	70.70% ± 5.05%	3156 ± 46
FiSmoA	FireDetection [29]	88.49% ± 0.20%	71.23% ± 5.22%	1531 ± 12
	FireNet [32]	85.44% ± 0.61%	64.20% ± 2.76%	**287 ± 21**
	KutralNext	90.72% ± 0.09%	78.05% ± 3.38%	443 ± 12
	KutralNext+	**90.72% ± 0.09%**	**82.92% ± 2.57%**	409 ± 11
	OctFiResNet	88.53% ± 0.47%	76.37% ± 8.02%	3177 ± 34
Average	FireDetection [29]	91.42% ± 0.37%	76.95% ± 3.84%	1541 ± 21
	FireNet [32]	88.89% ± 0.46%	69.84% ± 5.06%	**297 ± 19**
	KutralNext	93.70% ± 0.21%	78.18% ± 3.53%	450 ± 20
	KutralNext+	**93.85% ± 0.23%**	**84.36% ± 2.17%**	417 ± 15
	OctFiResNet	91.93% ± 0.52%	76.60% ± 6.40%	3175 ± 39

Interesting results were obtained over the black augmented FiSmo version, FiSmoA, which improves FireNet, KutralNext, KutralNext+, and remarkably OctFiResNet in test accuracy values. For the FireDetection model, the augmentation negatively impacts the performance reducing a 2% test accuracy and increasing its deviation value compared with FiSmo. By a counterpart, in the test performance results of Table 8, the AUROC and precision values of the models trained with FiSmoA and the black images augmentation increase the performance for all the models, resulting in a reduction of the standard deviation values.

A more detailed performance obtained over the datasets can be observed in Figure 11a for the models trained over FireNet where KutralNext+ obtain a 97.59%, followed by KutralNext with 94.18% AUROC index. It can be observed in the ROC curve that KutralNext performs well at a low false-positive rate. The use of the FireNet dataset proves the model's generalization over the same data distribution. A different data distribution models' behavior is achieved with FiSmo and FiSmoA shown in Figure 11b and Figure 11c. In FiSmo, KutralNext keeps first place with a 92.44% of AUROC index, followed by KutralNext+ with a 90.57%. An even higher AUROC value was obtained trained with the augmented version of FiSmo, where KutralNext achieves first place with 95.39%, followed by our KutralNext+ model with 92.79%. In both datasets, KutralNext achieves the best curve with low false-positive rates in terms of the ROC curve.

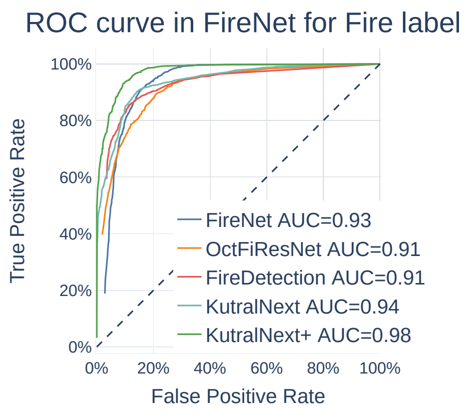
Table 8 – KutralNext performance comparison during 5 executions in the fire recognition task.

DS	Model	AUROC	Precision
FireNet	FireDetection [29]	$91.25\% \pm 2.50\%$	$93.79\% \pm 1.06\%$
	FireNet [32]	$92.90\% \pm 4.83\%$	$94.78\% \pm 3.66\%$
	KutralNext	$94.18\% \pm 2.61\%$	$95.60\% \pm 0.77\%$
	KutralNext+	**97.59% ± 1.05%**	**96.23% ± 1.02%**
	OctFiResNet	$91.07\% \pm 5.59\%$	$91.85\% \pm 3.32\%$
FiSmo	FireDetection [29]	$81.16\% \pm 3.81\%$	$92.55\% \pm 2.87\%$
	FireNet [32]	$86.44\% \pm 3.35\%$	$95.48\% \pm 4.10\%$
	KutralNext	**92.44% ± 1.49%**	**97.22% ± 1.11%**
	KutralNext+	$90.57\% \pm 2.00\%$	$97.25\% \pm 1.51\%$
	OctFiResNet	$80.03\% \pm 5.01\%$	$94.71\% \pm 1.87\%$
FiSmoA	FireDetection [29]	$83.02\% \pm 6.79\%$	$94.24\% \pm 3.68\%$
	FireNet [32]	$92.35\% \pm 1.37\%$	$97.79\% \pm 0.82\%$
	KutralNext	**95.39% ± 1.26%**	**98.57% ± 0.79%**
	KutralNext+	$92.79\% \pm 2.22\%$	$97.62\% \pm 1.68\%$
	OctFiResNet	$92.09\% \pm 3.22\%$	$97.24\% \pm 1.50\%$
Average	FireDetection [29]	$85.15\% \pm 4.37\%$	$93.53\% \pm 2.54\%$
	FireNet [32]	$90.56\% \pm 3.18\%$	$96.02\% \pm 2.86\%$
	KutralNext	**94.00% ± 1.79%**	**97.13% ± 0.89%**
	KutralNext+	$93.65\% \pm 1.76\%$	$97.03\% \pm 1.40\%$
	OctFiResNet	$87.73\% \pm 4.61\%$	$94.60\% \pm 2.23\%$

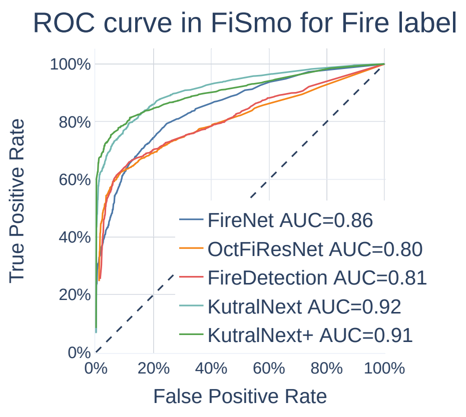
5.3.2 Multi-label classification: Fire and smoke recognition

In this second experiment, we check out the performance in the fire and smoke multi-label recognition task of our models' proposals with two datasets used for training and one dataset for testing. The training datasets were FiSmo and KutralSmoke, and the testing dataset was KutralSmoke Test. With those datasets, the models' were trained and compared with different data distribution of the corresponding labels and checking its generalization. This fire and smoke classification task analyzed each label separately under a multi-label approach due to the chance of appearing the fire or smoke class in the same image. Table 9 shown the statistics results of each model trained over all the datasets with averaged values for the validation and test accuracy, and test time and Table 10 presented each model's test performance. Our proposals are the best in recognize fire and smoke, being the most time inexpensive models. The classification was considered binary, considering fire, smoke, or both classes as a true label and none class as a false label.

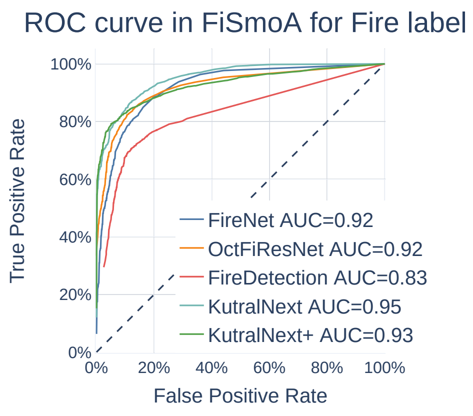
Table 9 shown the models' training performance, where it can be observed that in average validation accuracy, KutralNext performs the best with an 85.43%, followed by KutralNext+ with an 85.84% taking into consideration their deviation values. Different results were obtained during testing, where KutralNext+ performs the best with the same and different data distribution with an 81.53%, being better than KutralNext with 79.03%. In this regard, KutralNext+ surpasses



(a)



(b)



(c)

Figure 11 – Single-label classification ROC curve mean tests for 5 executions for all the models trained over FireNet, FiSmo and FiSmoA dataset. It can be observed that KutralNext+ performs the best under the same data distribution in the FireNet dataset (a), and being competitive against KutralNext in different data distributions as the FiSmo and FiSmoA datasets, (b) and (c), respectively.

the state-of-the-art fire recognition models, requiring less time in processing the test data images.

Now, in terms of time required to process the 1,171 testing images, OctFiResNet was the most time-consuming, taking over 2.0 seconds, followed by FireDetection with 1.87 seconds. For the KutralNext architectures, KutralNext+ is the model that requires more time with 0.61 seconds, leaving KutralNext as the model which requires less time with 0.41 seconds. FireNet is the model that requires less time to process the images; nevertheless, it also presents the lowest mean validation and test accuracy.

A general overview of each model in terms of AUROC and precision in the test dataset is shown in Table 10. In the first place, for the fire label, KutralNext has shown the best average AUROC value and OctFiResNet the best mean precision value in this multi-label test approach. Considering the mean AUROC between both datasets, the KutralNext model obtains a 94.47%, taking the first place, followed by KutralNext+ with 93.40%. In overall, all the models present a good performance to detect fire in this approach. However, for the smoke label, a lower outcome has been shown in AUROC and precision terms. KutralNext+ achieved a remarkable AUROC

Table 9 – KutralNext training results during 5 executions in the fire and smoke recognition task.

DS	Model	Val. acc.	Test acc.	Test (ms)
KutralSmoke	FireDetection [29]	$80.95\% \pm 0.92\%$	$77.59\% \pm 3.22\%$	1883 ± 81
	FireNet [32]	$78.85\% \pm 0.46\%$	$77.11\% \pm 3.60\%$	**339 ± 23**
	KutralNext	$89.66\% \pm 0.44\%$	$86.70\% \pm 2.02\%$	430 ± 21
	KutralNext+	**90.46% ± 0.48%**	**88.08% ± 0.69%**	603 ± 34
	OctFiResNet	$84.00\% \pm 0.63\%$	$79.03\% \pm 4.58\%$	2040 ± 11
FiSmo	FireDetection [29]	$77.18\% \pm 0.99\%$	$63.04\% \pm 8.60\%$	1856 ± 99
	FireNet [32]	$74.70\% \pm 0.66\%$	$56.89\% \pm 6.26\%$	**335 ± 22**
	KutralNext	**81.20% ± 0.74%**	$71.36\% \pm 2.31\%$	424 ± 21
	KutralNext+	$81.22\% \pm 3.18\%$	**74.98% ± 3.22%**	624 ± 33
	OctFiResNet	$78.04\% \pm 0.41\%$	$56.69\% \pm 2.38\%$	2046 ± 6
Average	FireDetection [29]	$79.07\% \pm 0.96\%$	$70.32\% \pm 5.91\%$	1870 ± 90
	FireNet [32]	$76.77\% \pm 0.56\%$	$67.00\% \pm 4.93\%$	**337 ± 23**
	KutralNext	**85.43% ± 0.59%**	$79.03\% \pm 2.16\%$	427 ± 21
	KutralNext+	$85.84\% \pm 1.83\%$	**81.53% ± 1.96%**	614 ± 33
	OctFiResNet	$81.02\% \pm 0.52\%$	$67.86\% \pm 3.48\%$	2043 ± 9

value with 89.59% and precision of 56.27%, followed by KutralNext with 87.00% and 46.92%, respectively, being the best model in comparison with previous extended models to acquire smoke features and recognize it under a multi-label approach. All of the models have been shown better outcomes trained over the same data distribution than a different data distribution.

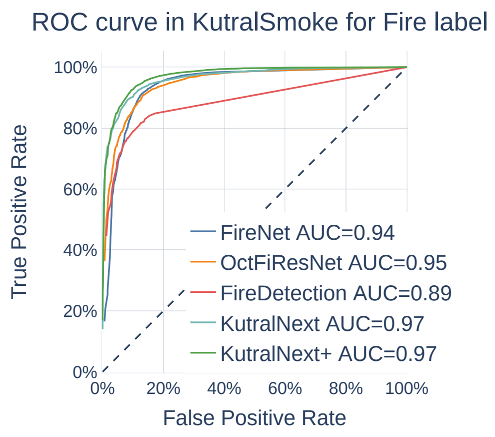
Figure 12 shows the mean ROC values obtained for the models trained over all the datasets to compare each models' performance in terms of feature acquisition for each model. The KutralNext proposals presented the best results for both classes from the used datasets, capable of acquiring features at a low false-positive rate. Remarkable results were obtained for the smoke label compared with previous models, as shown in Figure 12b and Figure 12d. Additionally, KutralNext and KutralNext+ obtained the best results under a different data distribution as the case for the FiSmo dataset. In this way, their implemented techniques efficiency has been demonstrated because the models' design was not meant to recognize smoke. Even so, it achieved the best results in smoke class features.

5.3.3 Discussion

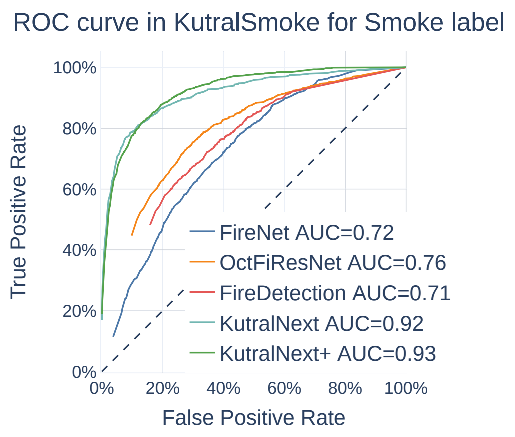
The KutralNext deep learning models proposal were capable of achieving a proper performance for fire and smoke recognition as a single- and multi-label approach, compared to previous deep learning models in the same approach. All of the models compared were designed under a single-label approach and adapted to be used under a multi-label approach for this research project. The FireDetection model was the only one designed to recognize fire and smoke from images. All of the models previously used were designed to recognize fire only. However, the output layer was successfully adapted for those models, demonstrated results obtained for

Table 10 – KutralNext performance during 5 executions in fire and smoke recognition task.

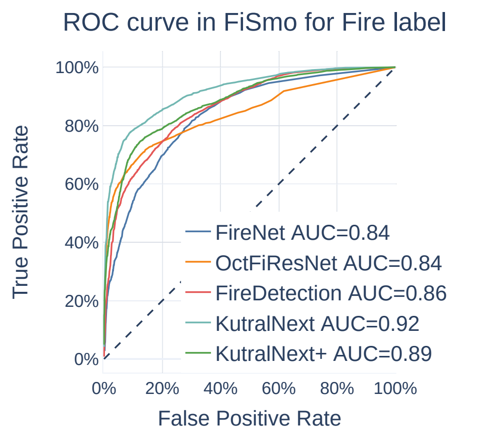
DS	Model	AUROC	Precision
		Fire Label	
KutralSmoke	FireDetection [29]	88.72% ± 4.04%	95.02% ± 1.87%
	FireNet [32]	94.18% ± 1.68%	94.07% ± 1.28%
	KutralNext	96.96% ± 0.49%	**97.12% ± 0.80%**
	KutralNext+	**97.46% ± 0.43%**	96.69% ± 1.21%
	OctFiResNet	94.84% ± 2.67%	94.74% ± 2.11%
FiSmo	FireDetection [29]	85.73% ± 7.74%	92.61% ± 4.77%
	FireNet [32]	83.66% ± 5.49%	90.74% ± 3.48%
	KutralNext	**91.98% ± 2.97%**	93.64% ± 4.11%
	KutralNext+	89.35% ± 2.03%	91.74% ± 3.48%
	OctFiResNet	84.25% ± 3.61%	**96.70% ± 2.03%**
Average	FireDetection [29]	87.23% ± 5.89%	93.82% ± 3.32%
	FireNet [32]	88.92% ± 3.59%	92.40% ± 2.38%
	KutralNext	**94.47% ± 1.73%**	95.38% ± 2.45%
	KutralNext+	93.40% ± 1.23%	94.22% ± 2.35%
	OctFiResNet	89.54% ± 3.14%	**95.72% ± 2.07%**
		Smoke Label	
KutralSmoke	FireDetection [29]	70.78% ± 3.84%	29.30% ± 2.59%
	FireNet [32]	72.22% ± 1.55%	28.00% ± 1.73%
	KutralNext	91.74% ± 1.23%	**52.91% ± 3.82%**
	KutralNext+	**92.59% ± 1.77%**	52.19% ± 6.55%
	OctFiResNet	76.42% ± 6.25%	31.49% ± 5.90%
FiSmo	FireDetection [29]	67.38% ± 3.92%	33.06% ± 3.21%
	FireNet [32]	67.79% ± 5.35%	35.01% ± 6.30%
	KutralNext	82.27% ± 1.19%	40.93% ± 2.41%
	KutralNext+	**86.59% ± 3.22%**	**60.35% ± 12.66%**
	OctFiResNet	66.95% ± 4.95%	29.75% ± 3.68%
Average	FireDetection [29]	69.08% ± 3.88%	31.18% ± 2.90%
	FireNet [32]	70.00% ± 3.45%	31.51% ± 4.02%
	KutralNext	87.00% ± 1.21%	46.92% ± 3.11%
	KutralNext+	**89.59% ± 2.50%**	**56.27% ± 9.60%**
	OctFiResNet	71.69% ± 5.60%	30.62% ± 4.79%



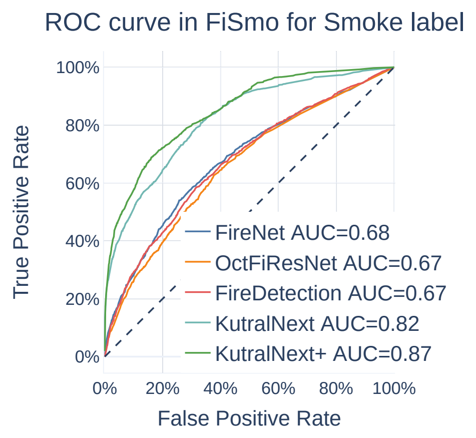
(a)



(b)



(c)



(d)

Figure 12 – Multi-label ROC curve mean performance for each model trained over different data distribution datasets. (a) and (b) present the models' performance trained over the KutralSmoke dataset. (c) and (d) present the models' performance trained over the FiSmo dataset. (a) and (c) show the fire label. (b) and (d) show the smoke label. The JN initials correspond to KutralNext. In (a) and (b) the KutralNext+ model achieves the best performance under low false positive rate. In (c) and (d) the KutralNext and KutralNext Mobile models, respectively achieves the best performance at low false positive rate.

the fire label. The central aspect of addressing a multi-label approach in still images is that fire or smoke can be present separately, together, or not to be in the image. In this regard, the multi-label approach could be suitable for an early alert system to measure the fire's magnitude. This magnitude could be translated from the inference of each label present in the image. When only the smoke label was detected, it could be an early stage of the fire. When just the fire label was detected, it could be a fire of minor intensity. Alternatively, if both fire and smoke labels were present, a more extensive fire was detected.

Each proposed and implemented model was compared in terms of validation, test accuracy, ROC curve, area under the ROC curve, precision, number of parameters and flops, and time required to process the test images. With those metrics, the generalization, feature acquisition capability, mathematical complexity, and storage size required were analyzed by each model. Hereof, the final optimized proposal to fire and smoke recognition achieved an efficient design with 138.9K parameters, and 76.9M flops for KutralNext, and a 68% flops reduction achieved by KutralNext+ with 24.6M flops. KutralNext+ performed the best generalization with an 84.36% average test accuracy in the single-label fire recognition task. Thus, it achieves the second-best in AUROC and precision with 93.65% and 97.03%, leaving KutralNext as the best acquiring fire features with 94.00% of AUROC and 97.13% of precision. Again, KutralNext+ achieve the best generalization, but this time in the multi-label fire and smoke recognition task with an 81.53% average test accuracy. In terms of fire feature acquisition, KutralNext got the best with 94.47% of AUROC index followed by KutralNext+ with 93.40%, in precision, KutralNext and KutralNext+ obtained a 95.38% and 94.22%, respectively. OctFiResNet achieved the best precision with 95.72% but presented a low generalization with 67.86% average test accuracy. KutralNext+ got the best AUROC and precision values for the smoke label with 89.59% and 56.27%, respectively, followed by KutralNext. In this way, KutralNext+ achieved the best test metrics, acquiring both fire and smoke features in a better way, being suitable for portable device implementations.

The results obtained highlight the importance of using a pre-trained model with ILSVRC 2012 [37] over one trained from-scratch. This benefit was demonstrated by the KutralNext+ model's performance, from which the pre-trained versions perform significantly better than the one from-scratch version in the single-label fire classification task with a 5.10% more in an average test accuracy. Additionally, present 4.05% less mean standard deviation. During the efficient models' training over the ILSVRC 2012 dataset, the validation accuracy was not appropriate to classify the 1000 contained classes. Nevertheless, the optimized parameters obtained during the training of our efficient models, KutralNext and KutralNext+, were enough to obtain better performance for both fire and smoke recognition task-specific model architecture design. These from-scratch results were not included in this research. However, this aspect has been widely demonstrated [43]. Additionally, the portable version with the inverted residual block and the octave convolution methods reduced the model's flops. It improved accuracy in single- and multi-label fire and smoke recognition tasks, suitable for a portable device at a high frame rate. Thus, our portable proposal is suitable for a fire detection vision-based system for incidents with fire or smoke presence.

The only concern encountered in this research, is the unrelated values of flops and the time required by each model. This issue is not related to the model or its techniques, but it is related to the PyTorch library[10]. Another considerable aspect of time is in the multi-label problem with the label preparation, which requires a few steps of encoding the classes before being processed by the model. Thus, the time issue can be solved for a final portable detection system migrating the library and implementing a more specific label codification system.

[10] Some users reported the slow implementation in the depthwise convolution using CUDA 32 bits floating-point operations to the PyTorch repository <https://github.com/pytorch/pytorch/issues/18631>

A brief qualitative analysis from true-positive, false-positive, true-negative, and false-negative cases obtained from the KutralNext+ model was carried out for both fire and smoke labels.

From a total of 16 samples illustrated in Figure 13, consider the four different cases for both labels. from top to down and from left to right, the cases are TP, FP, TN, FN. In the rows are the fire cases and in the columns the smoke cases.

In the fire cases, the model can recognize the fire blurred by the smoke with different densities. Darken scenarios mean no much difference to detect fire features. In terms of smoke features, it can be observed that the model recognizes smoke texture. However, darken scenarios and color reflection over the smoke remains a challenge.

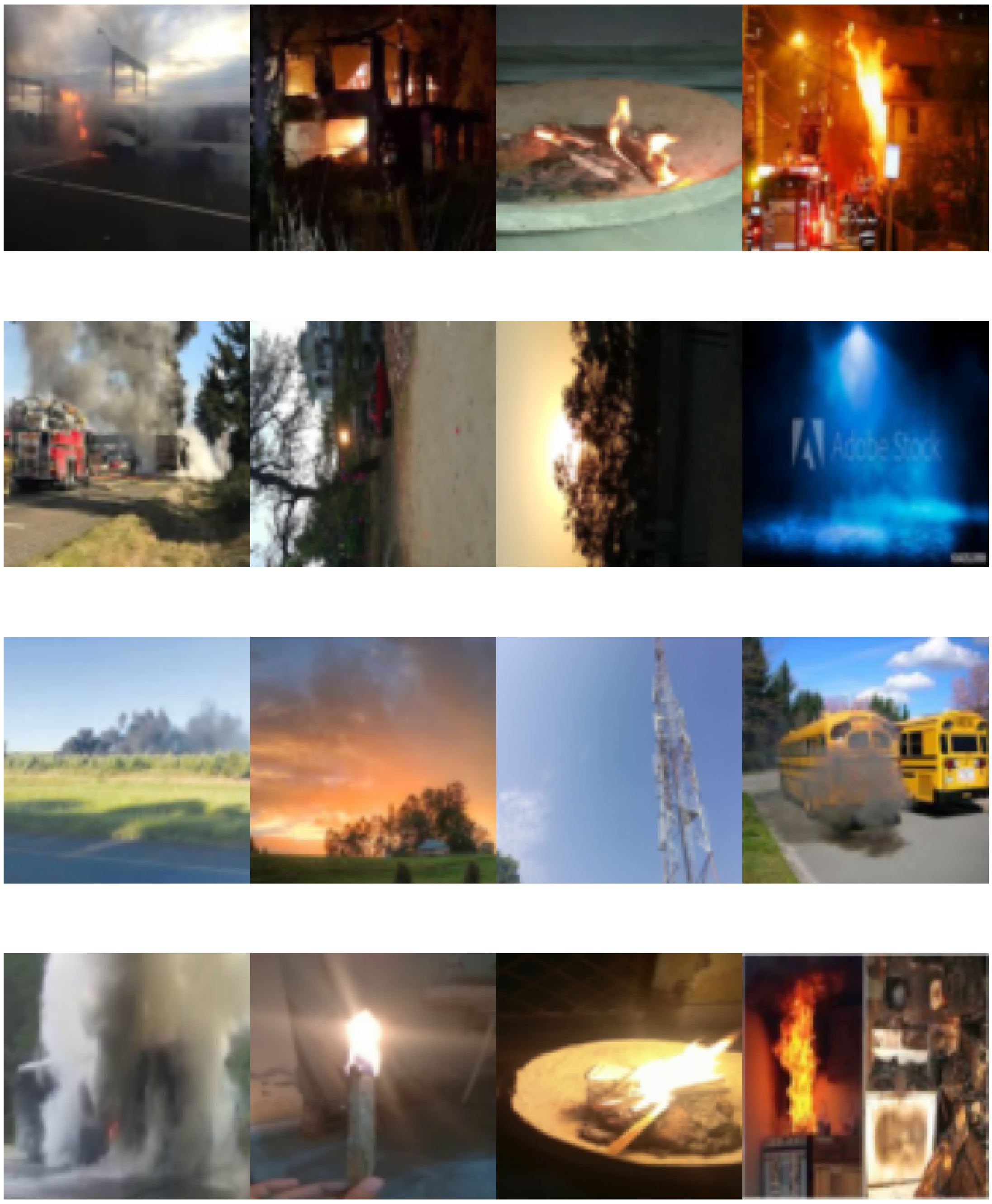


Figure 13 – Here are shown predicted KutralSmoke test sample images processed by KutralNext+ trained over FiSmo. From top to down and from left to right are the true-positive, false-positive, true-negative, and false-negative cases for each label. Each row is each fire case, and each column each smoke case.

Chapter 6

Conclusion

Fire disasters may lead to massive losses affecting in an environmental, social, and economic way, caused by natural or human causes. Hereof, an early detection system is affordable to manage this kind of accident, reducing the blazes' affected area. In this regard, a deep learning model for fire and smoke recognition under an efficient and reduced size scope was developed in this work. The model's development aims to be used on constrained-hardware devices such as alarm, video surveillance, or even to add fire recognition capability to robotic systems or any other mobile devices. Thus requiring less storage space, an efficient model can also be executed under a low computational resource setup. Ensure a real-time processing algorithm with high accuracy is challenging for deep learning due to previous state-of-the-art generic-purpose models that are mathematically complex designed to recognize a higher amount of classes.

For this work purpose, a study about the current fire and smoke recognition algorithm and novel techniques that optimize the models' performance, such as octave and separable depthwise convolution and the inverted residual block. Once reviewed the literature, a dataset analysis was carried out, proposing the first lightweight approach for fire recognition named OctFiResNet. The CairFire, FireSense, FireNet, and FiSmo datasets were checked using cross-dataset validation. OctFiResNet performed better results than previous approaches using those datasets. In this way, it demonstrated that FiSmo and FireNet datasets are suitable options to be used in a fire and smoke classification task given the challenging fire scenarios and the smoke labeled images. Those datasets allowed the fire recognition model's main architecture development, which fits the defined scope of reduced size and computational cost for this work. The KutralNet proposal was able to perform better than previous models, proving the effectiveness in recognizing fire. Furthermore, it was optimized with novel deep learning convolution methods, with KutralNet Mobile Octave as the best portable model in this task. A final proposal with those models was completed, named KutralNext and KutralNext+. Both models were trained under more complex representation data features in the ImageNet dataset to be optimized with the fire and smoke labeled datasets. A novel approach for fire and smoke recognition was proposed with 138.9K parameters, and 76.9M flops, with an efficient model developed in this work. KutralNext+

considerably reduces the number of flops to 24.6M, achieving the best performance with 84.36%, and 81.53% mean test accuracy in the fire, and fire and smoke recognition tasks, respectively. Additionally, it comprises 97% fewer flops, being 16% more accurate during testing in the fire and smoke recognition than FireDetection hence is executed 4x faster with better generalization.

6.1 Future works

As future works, a variety of further improvements can be addressed. In terms of datasets, use augmentation techniques for training a more robust model capable of generalizing better in challenging scenarios, even more for the smoke label. A Generative Adversarial Network approach [44] could be implemented to address this, or a custom learned augmentation technique proposed by Cubuk et al. named Autoaugment [45]. In terms of the architecture, it can be improved using different activation or loss functions. An additional test must be addressed, considering the real-time execution performance in an embedded system such as a Raspberry Pi without GPU acceleration. Furthermore, an extension of this work is to be used in detection using a bounding box approach.

Bibliography

[1] MORENO, J.; ARIANOUTSOU, M.; GONZáLEZ-CABáN, A.; MOUILLOT, F.; OECHEL, W.; SPANO, D.; THONICKE, K.; VALLEJO, V.; VéLEZ, R. ed.). Forest fires under climate, social and economic changes in europe, the mediterranean and other fire-affected areas of the world. *FUME : lessons learned and outlook*, 1 2014.

[2] BARLOW, J.; PERES, C. A. Avifaunal responses to single and recurrent wildfires in amazonian forests. *Ecological Applications*, v. 14, n. 5, p. 1358–1373, 2004.

[3] ÚBEDA, X.; SARRICOLEA, P. Wildfires in chile: A review. *Global and Planetary Change*, v. 146, p. 152 – 161, 2016. ISSN 0921-8181.

[4] VALENZUELA, N. V. U.; BUENO, M. F. C. Incendios forestales: principales consecuencias económicas y ambientales en chile. *Revista Interamericana de Ambiente y Turismo - RIAT*, v. 7, n. 1, p. 18 – 24, 2011.

[5] PÉREZ, V. Q. Perturbaciones a la vegetación nativa por grandes fuegos de 50 años atrás, en bosques nordpatagónicos. caso de estudio en chile meridional. *Anales de Geografía de la Universidad Complutense*, v. 28, n. 1, p. 85 – 104, 5 2008.

[6] ALVES, J.; SOARES, C.; TORRES, J. M.; SOBRAL, P.; MOREIRA, R. S. Automatic forest fire detection based on a machine learning and image analysis pipeline. In: *New Knowledge in Information Systems and Technologies*. [S.l.]: Springer International Publishing, 2019. p. 240–251. ISBN 978-3-030-16184-2.

[7] CHOLLET, F. Xception: Deep learning with depthwise separable convolutions. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2017.

[8] LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. *Nature*, Nature Publishing Group, v. 521, n. 7553, p. 436–444, may 2015. ISSN 0028-0836.

[9] DENG, Y. Deep learning on mobile devices: A review. In: AGAIAN, S. S.; ASARI, V. K.; DELMARCO, S. P. (Ed.). *Mobile Multimedia/Image Processing, Security, and Applications 2019*. [S.l.]: SPIE, 2019. v. 10993, p. 52 – 66.

[10] BARMPOUTIS, P.; DIMITROPOULOS, K.; GRAMMALIDIS, N. Real time video fire detection using spatio-temporal consistency energy. In: *2013 10th IEEE International Conference on Advanced Video and Signal Based Surveillance*. [S.l.: s.n.], 2013. p. 365–370.

[11] KIM, Y.-H.; KIM, A.; JEONG, H.-Y. Rgb color model based the fire detection algorithm in video sequences on wireless sensor network. *International Journal of Distributed Sensor Networks*, v. 10, n. 4, p. 923609, 2014.

[12] NIKOS, G.; CETIN, A.; DIMITROPOULOS, K.; TSALAKANIDOU, F.; KOSE, K.; GUNAY, O.; GOUVERNEUR, B.; TORRI, D.; KURUOGLU, E.; TOZZI, S.; BENAZZA-BENYAHIA, A.; CHAABANE, F.; KOSUCU, B.; ERSOY, C. A multi-sensor network for the protection of cultural heritage. In: . [S.l.: s.n.], 2011.

[13] DIMITROPOULOS, K.; TSALAKANIDOU, F.; GRAMMALIDIS, N. Flame detection for video-based early fire warning systems and 3d visualization of fire propagation. In: . [S.l.: s.n.], 2012. p. 18–20.

[14] DIMITROPOULOS, K.; BARMPOUTIS, P.; GRAMMALIDIS, N. Spatio-temporal flame modeling and dynamic texture analysis for automatic video-based fire detection. *IEEE Transactions on Circuits and Systems for Video Technology*, v. 25, n. 2, p. 339–351, 2 2015. ISSN 1051-8215.

[15] CUNNINGHAM, P.; CORD, M.; DELANY, S. J. Supervised learning. In: ____. *Machine Learning Techniques for Multimedia: Case Studies on Organization and Retrieval*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008. p. 21–49. ISBN 978-3-540-75171-7.

[16] WERBOS, P. J. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, v. 78, n. 10, p. 1550–1560, 1990.

[17] HUBEL, D. H.; WIESEL, T. N. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of Physiology*, v. 160, n. 1, p. 106–154, 1962.

[18] GOLDMAN-RAKIC, P. S.; RAKIC, P. Preface: Cerebral Cortex Has Come of Age. *Cerebral Cortex*, v. 1, n. 1, p. 1–1, 01 1991. ISSN 1047-3211.

[19] HE, K.; ZHANG, X.; REN, S.; SUN, J. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[20] SIFRE, L.; MALLAT, S. Rigid-motion scattering for texture classification. *arXiv preprint arXiv:1403.1687*, 2014.

[21] SANDLER, M.; HOWARD, A.; ZHU, M.; ZHMOGINOV, A.; CHEN, L. MobileNetV2: Inverted residuals and linear bottlenecks. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2018. p. 4510–4520. ISSN 1063-6919.

[22] CHEN, Y.; FAN, H.; XU, B.; YAN, Z.; KALANTIDIS, Y.; ROHRBACH, M.; YAN, S.; FENG, J. Drop an octave: Reducing spatial redundancy in convolutional neural networks with octave convolution. *arXiv preprint arXiv:1904.05049*, 2019.

[23] SIMONYAN, K.; ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[24] SHARMA, J.; GRANMO, O.; GOODWIN, M.; FIDJE, J. T. Deep convolutional neural networks for fire detection in images. In: *Engineering Applications of Neural Networks*. Cham: Springer International Publishing, 2017. p. 183–193. ISBN 978-3-319-65172-9.

[25] MUHAMMAD, K.; AHMAD, J.; LV, Z.; BELLAVISTA, P.; YANG, P.; BAIK, S. W. Efficient deep CNN-based fire detection and localization in video surveillance applications. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, Institute of Electrical and Electronics Engineers Inc., v. 49, n. 7, p. 1419–1434, 2018. ISSN 21682232.

[26] NAMOZOV, A.; CHO, Y. I. An efficient deep learning algorithm for fire and smoke detection with limited data. *Advances in Electrical and Computer Engineering*, v. 18, p. 121–128, 2018.

[27] AGOSTINELLI, F.; HOFFMAN, M.; SADOWSKI, P.; BALDI, P. Learning activation functions to improve deep neural networks. *arXiv preprint arXiv:1412.6830*, 2014.

[28] AGARAP, A. F. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*, 2018.

[29] GOTTHANS, J.; GOTTHANS, T.; MARSALEK, R. Deep convolutional neural network for fire detection. In: *2020 30th International Conference Radioelektronika (RADIOELEKTRONIKA)*. [S.l.: s.n.], 2020. p. 1–6.

[30] KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. In: PEREIRA, F.; BURGES, C. J. C.; BOTTOU, L.; WEINBERGER, K. Q. (Ed.). *Advances in Neural Information Processing Systems*. [S.l.]: Curran Associates, Inc., 2012. v. 25, p. 1097–1105.

[31] IANDOLA, F. N.; HAN, S.; MOSKEWICZ, M. W.; ASHRAF, K.; DALLY, W. J.; KEUTZER, K. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

[32] JADON, A.; OMAMA, M.; VARSHNEY, A.; ANSARI, M. S.; SHARMA, R. FireNet: A specialized lightweight fire & smoke detection model for real-time iot applications. *CoRR*, abs/1905.11922, 2019.

[33] OH, S. H.; GHYME, S. W.; JUNG, S. K.; KIM, G.-W. Early wildfire detection using convolutional neural network. In: OHYAMA, W.; JUNG, S. K. (Ed.). *Frontiers of Computer Vision*. Singapore: Springer Singapore, 2020. p. 18–30. ISBN 978-981-15-4818-5.

[34] TAN, M.; LE, Q. Efficientnet: Rethinking model scaling for convolutional neural networks. In: *International Conference on Machine Learning*. [S.l.: s.n.], 2019. p. 6105–6114.

[35] MADHEVAN, B.; SAKKARAVARTHI, R.; Mandeep Singh, G.; DIYA, R.; JHA, D. K. Modelling, simulation and mechatronics design of a wireless automatic fire fighting surveillance robot. *Defence Science Journal*, Defense Scientific Information and Documentation Centre, v. 67, n. 5, p. 572–580, 9 2017. ISSN 0976464X.

[36] XU, B.; WANG, N.; CHEN, T.; LI, M. Empirical evaluation of rectified activations in convolutional network. *CoRR*, abs/1505.00853, 2015.

[37] RUSSAKOVSKY, O.; DENG, J.; SU, H.; KRAUSE, J.; SATHEESH, S.; MA, S.; HUANG, Z.; KARPATHY, A.; KHOSLA, A.; BERNSTEIN, M.; BERG, A. C.; FEI-FEI, L. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, v. 115, n. 3, p. 211–252, 2015.

[38] CUI, Y.; JIA, M.; LIN, T.-Y.; SONG, Y.; BELONGIE, S. Class-balanced loss based on effective number of samples. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2019. p. 9268–9277.

[39] LIN, T.-Y.; GOYAL, P.; GIRSHICK, R.; HE, K.; DOLLÁR, P. Focal loss for dense object detection. In: *Proceedings of the IEEE international conference on computer vision*. [S.l.: s.n.], 2017. p. 2980–2988.

[40] CAZZOLATO, M. T.; AVALHAIS, L. P.; CHINO, D. Y.; RAMOS, J. S.; SOUZA, J. A. de; RODRIGUES-JR, J. F.; TRAINA, A. Fismo: A compilation of datasets from emergency situations for fire and smoke analysis. In: *Brazilian Symposium on Databases - SBBD*. [S.l.: s.n.], 2017. p. 213–223.

[41] ANTZOULATOS, G.; GIANNAKERIS, P.; KOULALIS, I.; KARAKOSTAS, A.; VROCHIDIS, S.; KOMPATSIARIS, I. *A Multi-Layer Fusion Approach For Real-Time Fire Severity Assessment Based on Multimedia Incidents*. [S.l.]: Zenodo, 2020.

[42] CHINO, D. Y. T.; AVALHAIS, L. P. S.; JR., J. F. R.; TRAINA, A. J. M. Bowfire: Detection of fire in still images by integrating pixel color and texture analysis. *CoRR*, abs/1506.03495, 2015.

[43] Studer, L.; Alberti, M.; Pondenkandath, V.; Goktepe, P.; Kolonko, T.; Fischer, A.; Liwicki, M.; Ingold, R. A comprehensive study of imagenet pre-training for historical document image analysis. In: *2019 International Conference on Document Analysis and Recognition (ICDAR)*. [S.l.: s.n.], 2019. p. 720–725.

[44] BOWLES, C.; CHEN, L.; GUERRERO, R.; BENTLEY, P.; GUNN, R.; HAMMERS, A.; DICKIE, D. A.; HERNÁNDEZ, M. V.; WARDLAW, J.; RUECKERT, D. Gan augmentation: Augmenting training data using generative adversarial networks. *arXiv preprint arXiv:1810.10863*, 2018.

[45] CUBUK, E. D.; ZOPH, B.; MANE, D.; VASUDEVAN, V.; LE, Q. V. Autoaugment: Learning augmentation strategies from data. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2019. p. 113–123.