Wildfire Detection Using Self-Supervised Learning

Sara Alexandra Curado Fernandes
Instituto Superior Técnico
sara.a.fernandes@tecnico.ulisboa.pt

***Abstract*—Wildfires are one of the most challenging disasters to control and are responsible for thousands of hectares burned, infrastructure destroyed, and lives lost every year all over the world. To develop robust systems capable of detecting and locating wildfires with high efficiency through supervised learning, it is necessary to acquire extensive labelled datasets. However, the number of publicly available images with associated annotations is low. Moreover, there are thousands of images related to forest fires available online, but without annotation. Furthermore, generating all the required labels can be time-consuming and costly. Therefore, this thesis hopes to take advantage of this unlabelled data by proposing a system that uses self-supervised learning to achieve the final goal of fire classification. The proposed methodology is divided into two phases. First, a network is trained to solve some tasks, different to the final task, using the unlabelled dataset. Afterwards, by combining part of the learned model with a small, labelled dataset, the final classifier is achieved. When comparing the models only trained with the small labelled dataset, the proposed methodology achieved a better performance, proving that there are advantages to using the unlabelled data available online.**

***Index Terms*—Wildfire, Self-supervised learning, Deep learning, Convolutional neural networks, UAVs.**

I. Introduction

WILDFIRES are responsible for thousands of burned hectares, infrastructure destroyed and human lives lost, year after year. In mainland Portugal, wildfires were responsible for more than 861,000 hectares burned between 2016 and 2020, according to ICNF[1].

The Firefront project[2], which this thesis is part of, aims to assist firefighting teams by developing a system that can automate the detection, monitoring and fighting of wildfires and potential re-ignitions. The idea is to implement this system through aerial vehicles equipped with an RGB camera to capture real-time images of the forest and, eventually, fires.

To develop robust systems capable of detecting and locating wildfires with high efficiency through supervised learning, it is necessary to acquire extensive datasets. Unfortunately, those datasets are not publicly available and, even though there are thousands of unlabelled images available online, the process of labelling all those images can be very expensive and highly time-consuming.

Therefore, we propose a methodology that can take advantage of this huge number of unlabelled images without having to manually generate the respective labels. As illustrated in Fig. 1, first, a network is trained to solve some tasks, different to the final task, using only the unlabelled data. Afterwards, using part of the network obtained and a small, labelled dataset, the final classifier is submitted to training, in order to identify whether the image includes fire.

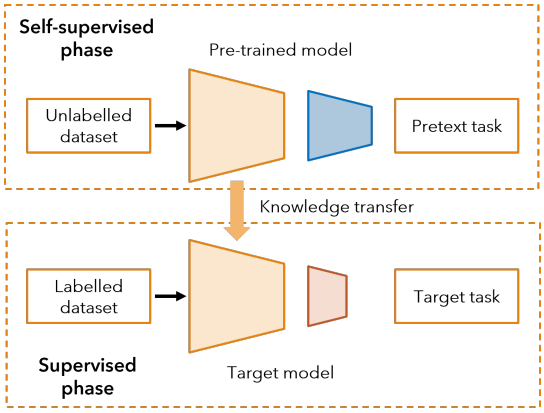


Fig. 1. Overview of self-supervised learning methodology.

II. State Of The Art

Over recent decades, several methods have been developed to identify fires in the early stages using image processing, computer vision and deep learning.

A. Traditional methods

The first methods to appear that used image recognition for fire detection were based on colour, movement and spatial and time positioning, since these are the most relevant aspects of the fire. Most of the traditional methods take advantage of the colour histogram to identify the pixels as fire or non-fire. However, the colour space used can differ [1], [2], [3]. These methods require the definition of some thresholds and the pixel is considered fire if it falls within such thresholds.

In order to improve this type of methodology, some methods propose the use of background subtraction, which involves the subtraction between a current image and a part of the scene (background). The elements that do not form part of the background are segmented and, with the help of some thresholds, it is determined whether they correspond to fire. In particular, the CICLOPE method, proposed in [4], was implemented in Portugal and monitors 1,300,000 hectares of Portuguese forest. Nonetheless, the cameras are fixed in surveillance towers which means that not every part of the forest is captured.

Finally, the implementation of a fire detection system in Unmanned Aerial Vehicles (UAVs) using colour-based methods [5]. However, in areas surrounding the fire, it's common

[1]https://www.pordata.pt/Portugal/Inc%c3%aandios+rurais+e+%c3%a1rea+ardida+%e2%80%93+Continente-1192

[2]http://www.firefront.pt/

to occur false positives, as the tonality is similar to the fire colours. However, in areas surrounding the fire, it is common for false positives to occur, as the tonality is similar to that of the fire colours.

Overall, colour-based methods achieve high accuracy, but they strongly depend on the information provided by the authors, such as camera parameters and thresholds. Furthermore, they have a high false positive rate, most likely due to environmental conditions and camera parameter changes.

More recently, deep learning has risen appeared and become helpful in solving several tasks related to image recognition in many different areas, by using neural networks, more specifically CNN [6].

B. Deep learning methods

Unlike traditional methods, deep learning methods automatically extract the most relevant features of objects which leads to improved performance. Most of these methods rely on supervised learning to train the networks, which means human-annotated images are required to train the networks.

In accordance with [7], several architectures, such as AlexNet [8], VGG16 [9] and SqueezeNet [10], were trained using 160,000 labelled images to classify fire in an image.

Another supervised method [11] suggests the use of two classifiers, one global and one on the patch level sharing the initial layers of the AlexNet architecture allowing the network to identify small regions of the fire.

The authors of [12] recommend image pre-processing operations such as histogram equalization and smooth low-pass filtering to eliminate irrelevant aspects of the image and highlight the relevant elements of the fire.

The main disadvantage of these methods is the fact that they need a vast labelled dataset to train the networks and even though there are thousands of images available online, the proportion of labelled images is very small. Furthermore, depending on the type of task, the labels can be image level, for classification, or pixel level, for segmentation, in which case more time is required to generate the labels.

In [13],the authors developed a methodology that can segment an image using only image-level labels, outperforming the methods that use pixel-level labels to segment the fire regions in images. The method consists of two networks, the first determines whether the image has fire or not, and if it has, the image is used as input to a second network to perform the segmentation of the fire regions of the image. However, a significant number of images labelled on an image-level is still required to train the classification network.

Therefore, this thesis aims to develop a methodology that significantly reduces the number of labelled images needed to train a classifier, by developing a system that takes advantage of unlabelled images available online. The proposed method uses self-supervised learning to pre-train the network and thus reduce the number of labelled images required to train the classifier. Additionally, this methodology does not require techniques such as background subtracting and can therefore be implemented both on aerial vehicles and fixed systems such as surveillance towers.

III. Self-supervised Learning

Self-supervised learning is a subset of unsupervised learning as the data used for training is not human-annotated. However, in this type of learning the network does not focus on clustering the data, as is the case in most unsupervised methods. Instead, it must learn the visual features from the unlabelled data by solving one or more pretext tasks [14]. In order to solve these tasks, labels are automatically generated from data itself allowing the network to learn in a supervised manner without requiring human annotation and therefore, drastically reduces the time and money needed to generate the labels. Once the training has finished, part of the learned network is used to solve the target task, for example classification, detection or segmentation. There are two ways to use the weights extracted from the pre-trained network. On one hand, transfer learning consists of using a previously trained network for some tasks to help solve a different, but related problem, in this case, fire classification. In contrast, fine-tuning involves using the previously learned weights exclusively as initialisation of the new network, retraining it for the final task. For the target tasks, labelled data is needed but in a much smaller quantity.

According to [15], self-supervised pretext tasks can be divided into three categories, as illustrated in Fig. 2 depending on the objective function and architectures.

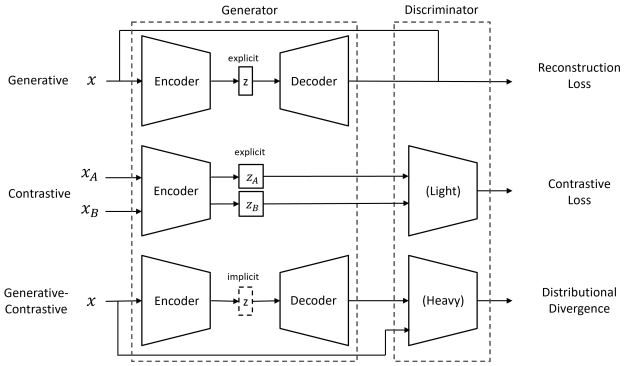


Fig. 2. Differences between the three self-supervised task categories [15].

1) Generative
   The main idea behind these methods is to reconstruct the input. Typically, an encoder is used to obtain the representations, z, in the latent space from the input x and then the decoder tries to reconstruct x from z obtaining $\hat{x}$. Finally, using a reconstruction loss, $\hat{x}$ is compared to x and the weights of the network are updated.One simple example of these methods is the use of an autoencoder [16] to reconstruct the input image x without applying any distortion. In this case, the label is the input image x. Alternatively, random noise can be applied to the input image and in this case, the network reconstructs the image without the noise [17].

2) Contrastive
   As the name suggests, the learning process is achieved by contrasting two elements. The contrast can be context-instance where the goal is to find a relation

between local features and the global representation. For example, the authors of [18] suggested applying a rotation multiple of 90º and the output of the network should be the rotation applied with the aim of forcing the network to detect the salient objects, recognise their relative orientation and compare them with the dominant orientation of all the objects. Another type of contrastive method is instance-instance where the idea is to compare the global representation of two inputs. In this case, some transformations are applied to the training images so that each initial image has two versions in the final training set. Most of these methods, such as [19], [20], try to maximise the similarity between positive pairs, where positive pairs are two transformed versions of the same image and the rest are treated as negative pairs.

3) Generative-Contrastive
This category is also called Adversarial Learning. In this case, the network training process is divided into two steps. First, fake samples are generated by the generator component and according to the real data. Then the discriminator has to compare the real samples with the fake ones. In order to generate the samples, the network architecture corresponds to an autoencoder, facilitating the generative task but making the contrastive task more complex.

IV. Methodology

The proposed system is intended to be implemented in aerial vehicles equipped with an RGB camera to capture real-time images of the forest and, eventually, the fire. The methodology suggested is divided into two phases, the first is responsible for pre-training the network by solving some pretext tasks using unlabelled images related to forest and wildfires. The second stage corresponds to the target task, in this case, fire classification.

A. Self-supervised phase

For the self-supervised stage, three pretext tasks were developed to train the model to extract the fire representations in three different ways. In all the tasks only images without human annotations were used.

First, a generative task was implemented referred to as image reconstruction and as the name suggests, the idea is to reconstruct the input image, x, using an autoencoder [16]. With this architecture, the image is first compressed using the encoder component to obtain the representation vector in the latent space. Then, using the decoder, the image is reconstructed from the low dimension representation, as suggested in Fig. 3, resulting in $\hat{x}$. The encoder input can be both original images or transformed images. The purpose of applying transformations to the images is to augment the number of total images and allow a more diverse dataset. Therefore, before training the transformations are applied and the final training dataset has all the original images as well as one or more transformed versions. In both cases, the output must be the same as the input and not necessarily the same as the original image. The transformations applied were colour-based and geometric and the details are described in Section IV-C. The architecture of the autoencoder used

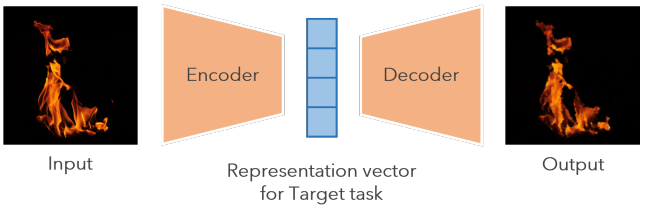


Fig. 3. Autoencoder components to solve the image reconstruction pretext task.

is described in Table I where Conv. stands for convolutional and Avg. stands for average. For network training, the Mean Squared Error given by

$$\mathcal{L}_{MSE} = \frac{1}{N}\sum_{i=1}^{N}(x_i - \hat{x}_i)^2, \tag{1}$$

was used as loss function, the number of epochs was 60 and for batch size 75 was used.

TABLE I
AUTOENCODER ARCHITECTURE FOR IMAGE RECONSTRUCTION PRETEXT TASK.

Component	Layer	Kernel	Activation function	Output
Encoder	Conv.	3x3	ReLU	224x224x32
	Avg. Pooling	2x2	-	112x112x32
	Conv.	3x3	ReLU	112x112x32
	Avg. Pooling	2x2	-	56x56x32
	Conv.	3x3	ReLU	56x56x64
	Avg. Pooling	2x2	-	28x28x64
	Conv.	3x3	ReLU	28x28x64
	Avg. Pooling	2x2	-	14x14x64
	Conv.	3x3	Tanh	14x14x64
	Avg. Pooling	2x2	-	7x7x64
Decoder	Transpose Conv.	3x3	ReLU	14x14x64
	Transpose Conv.	3x3	ReLU	28x28x64
	Transpose Conv.	3x3	ReLU	56x56x64
	Transpose Conv.	3x3	ReLU	112x112x32
	Transpose Conv.	3x3	ReLU	224x224x32
	Conv.	3x3	ReLU	224x224x3

Afterwards, a contrastive task, referred to as SimCLR [19], was implemented which can be divided into four steps, as shown in Fig. 4.

First, it is applied to each image of the batch two sets of random transformations to achieve two different versions of the same original image in the final batch. Versions obtained from the same original image are considered a positive pair while two versions generated from different original images correspond to a negative pair. Consequently, each image of the final batch has 1 positive pair and $2(N-1)$ negative pairs, so that in total there are $2N^2$ pairs since the pair x_i and x_j is considered different from the pair x_j and x_i. Moreover, the transformations are applied in each epoch and therefore during the network training the transformations are not constant resulting in different pairs along different epochs. The transformations applied are described in Section IV-C.

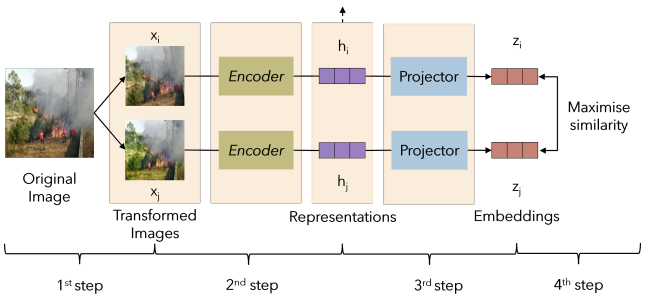


Fig. 4. Steps of SimCLR methodology [19]. The transformation applied to x_i was rotation and to x_j was colour manipulation. (Adapted image [21])

The second step converts the transformed images x_i and x_j into the respective representations, h_i and h_j, using an encoder. These representation vectors are used later for the target task.

The next step uses a non-linear projector to obtain the embeddings, z_i and z_j. According to the authors, the introduction of the non-linear projector allows accuracy to be increased by 10%.

Finally, the last step corresponds to the loss function referred to as Normalized Temperature-Scaled Cross-Entropy Loss (NT-Xent). To determine the loss it is first necessary to calculate the similarity between two embeddings of the same batch according to

$$\text{sim}(z_i, z_j) = \frac{z_i^T z_j}{\|z_i\| \, \|z_j\|}. \tag{2}$$

Ideally, the value for the similarity of a positive pair will be high and in contrast, a negative pair should result in lower values. With the similarity function defined, the NT-Xent loss function is given by

$$\ell_{i,j} = -\text{sim}(z_i, z_j) + \log\left(\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp\left(\frac{\text{sim}(z_i, z_j)}{\tau}\right)\right) \tag{3}$$

where i and j are a positive pair and τ is the temperature parameter with a value of 0.1 as this achieves the best results The NT-Xent function has two terms, the first relates to the similarity maximisation and the second is the contrastive term.

The main disadvantage is that this loss function needs a huge batch size to increase network performance. According to the authors of SimCLR, the best results were achieved when batches with 8,192 images were used, as the bigger the batch size, the more negative pairs there will be, easing the algorithm convergence. On another hand, the loss function benefits from embeddings with lower dimensions. Finally, when more epochs were used to train the network, the performance gap between models trained with a smaller batch size decreased.

Considering the number of images in the dataset is reduced, the batch size is 128 images. To compensate for the small dataset, 200 epochs were used. Lastly, the architecture used consists of an encoder and a projector. The encoder component is identical to that shown in Table I and the projector is a combination of four layers. The first is a Flatten layer, responsible for converting the encoder output to a single vector, the second layer is a Dense layer with 64 neurons, followed by an ReLU activation layer and finally another Dense layer with 64 neurons.

The third and last pretext task is the Barlow Twin task proposed by [20]. This method is very similar to SimCLR since the goal is the same, that is, to maximise the similarity between positive pairs and minimise the agreement between negative pairs using similar architecture. Furthermore, positive and negative pairs are obtained the same way, i.e. with the same transformations. The pipeline of Barlow Twins' method is shown in Fig. 5. As with SimCLR, for each original image,

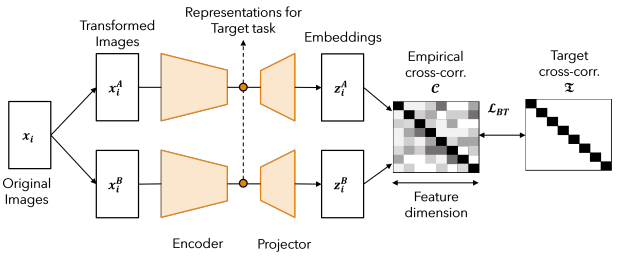


Fig. 5. Steps of Barlow Twins methodology [20]. (Adapted image)

x_i, of the batch, two sets of transformations are applied obtaining two new versions, x_i^A and x_i^B. Moreover, the positive pairs are divided into two sets, A and B, so that each image in set A has its positive pair in set B and the remaining images of set B are its negative pairs. Likewise, each image of the set B has a single positive pair in set A and the remaining images of set A are its negative pairs. In this case, the authors consider the pair x_i^A and x_i^B to be identical to the pair x_i^B and x_i^A and therefore is only counted once. So, for a batch with N images, each image has one positive pair an $N-1$ negative pairs, resulting in a total of N^2. New transformed versions of each image are obtained in each epoch which means new pairs are obtained for each epochs.

Using an encoder the representations of each version is determined and will be used for the target task after the self-supervised phase is complete. Once the representations are obtained they will serve as input to a projector and the embeddings, z_i^A and z_i^B, are computed.

Finally, the loss can be calculated using the proposed loss function in [20]. This loss function is the main difference between the two methods, SimCLR and Barlow Twins, since this function allows small batch size to be used without compromising network performance. To compute the loss function, it is first necessary to calculate the cross-correlation matrix, $\mathcal{C}$, between the embeddings using

$$\mathcal{C}_{ij} \triangleq \frac{\sum_b z_{b,i}^A z_{b,j}^B}{\sqrt{\sum_b (z_{b,i}^A)^2}\sqrt{\sum_b (z_{b,j}^B)^2}}. \tag{4}$$

where b is the batch index and, unlike SimCLR, i and j can be a positive ($i = j$) or negative ($i \neq j$) pair. The main idea of the proposed loss function is to force $\mathcal{C}$ to be equal to the identity matrix, and as such, the representations of the positive pairs must be similar and minimise the redundancy between these two vectors. The function is divided into two terms, the first is the invariance term that ensures that the network is invariant to

the distortions applied and the second term is the redundancy reduction responsible to force independence between negative pairs. The Barlow Twins loss function is given by

$$\mathcal{L}_{BT} \triangleq \sum_i (1 - \mathcal{C}_{ii})^2 + \lambda \sum_i \sum_{j \neq i} \mathcal{C}_{ij}^2 \quad (5)$$

where λ is a positive constant that allows a compromise between the two terms of the loss function. According to the authors, the value that provides best performance is 0.005 accordingly this was the value used in this thesis.

The architecture used to solve this pretext task is similar to SimCLR since the encoder is also identical to that shown in Table I, and it also has a projector component. However, in contrast to SimCLR, the Barlow Twins loss function takes advantage of embeddings with higher dimensions. The projector starts with a Flatten layer followed by three Dense layers with 1024 nodes and after each of the two first Dense layers comes a batch normalisation layer and an ReLU activation layer. The batch size and the number of epochs used during the training was 128 and 70, respectively.

B. Supervised phase

The supervised task is classification and consists of identifying fire in an image. In this case, a small dataset with image-level annotations was used but unlike self-supervised pretext tasks, no transformation was applied to the dataset. The classifier was a combination of part of the network trained with one of the self-supervised pretext tasks and two new layers. From the self-supervised pre-train network, the encoder component was extracted, the architecture for which is illustrated in Table I. A further two layers were appended to the encoder, first a Global Average Pooling and then a Dense layer with one neuron, and sigmoid as the activation function.

C. Dataset

In order to develop the proposed model, it is crucial to create an image dataset as big and as diverse as possible for each of the abovementioned phases. For this thesis, images were gathered from different sources in order to have enough to train and evaluate the models during both phases. In total, 2,500 images were collected, of which 692 came from the Portuguese Firefighters website [22] and 1,808 from a dataset created by one of the works from the Firefront project [13]. Table II shows the image distribution for each of the phases.

TABLE II
IMAGE DISTRIBUTION FOR EACH PHASE.

Name	Phase	# Images	Fire [%]	
A	Self-supervised	2000	-	
B	Supervised	500	Positive	70
			Negative	30

For network training during the self-supervised phase, the dataset A is split into two parts, one with 90% for training and the other 10% for validation. This partition allows the evolution of the loss to be monitored and ensures that the network is not overfitting. For the supervised phase, the cross-validation technique was used since the dataset available for this phase is reduced. Cross-validation requires the division of the dataset into multiple folders. Ten folders were defined and, as such, each iteration had 450 images to train (90%) and 50 for testing (10%). This procedure allowed us to determine whether the classifier was overfitting and to compare the performance of the different classifiers.

During the self-supervised phase, the application of several transformations to the dataset was explored with the purpose of augmenting the number of images to train the network. The studied transformations were colour-based and geometric.

- Colour-based transformations: The colour-based transformation considered was the manipulation of the Hue, H, and Saturation, S, channels by converting the RGB images to the HSV colour-space and randomly modifying the channels. Even though the network cannot learn to identify fire exclusively through the colour histogram, colour is one of the most important features of fire and therefore the variation of the channels must be moderate, especially in the hue channel. The variation of the S channel provides wider diversity regarding the luminosity of the images, while the manipulation of the H channel enables the network to learn the shapes and edges of the objects.
- Geometric transformations: This type of transformation is particularly useful when the network has to learn without relying on the object position in the image. In the case of forest images captured with an aerial vehicle, fire, as well as other objects, may not occupy the same position and orientation in the image. In this way, the network must be exposed to different perspectives so that it is not based solely on these two aspects. Firstly, the impact of applying rotations to the image was investigated, however, such rotations should be small to avoid providing the network with unrealistic examples. Then, other transformations were explored such as horizontal flip, and crop and resize.

Table III shows all the different transformations studied during this thesis and Fig. 6 provides an example of each transformation applied to the images.

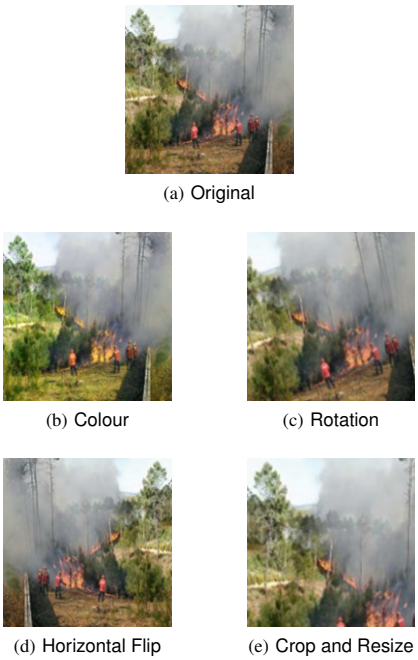
TABLE III
LIST OF POSSIBLE TRANSFORMATIONS APPLIED TO THE IMAGES DURING THE SELF-SUPERVISED PHASE.

Transformation	Description
Colour	Random variation of H channel up to 20 %; Random variation of S channel up to 30 %.
Rotation	Random rotation between 1 and 20º.
Horizontal Flip	Horizontal flip.
Crop and Resize	Cropping a square of the image with random dimension between 100 and 200 pixels followed by resizing to original dimension of the image.

In addition to studying the application of these transformations alone, combinations of transformations in the same

dataset were also analysed. The groups of transformations used were:

i) Colour
ii) Rotation
iii) Horizontal Flip, Crop and Resize
iv) Colour or Rotation
v) Colour + Rotation
vi) Crop and Resize, Horizontal Flip and Rotation
vii) All



(a) Original (b) Colour (c) Rotation (d) Horizontal Flip (e) Crop and Resize

Fig. 6. Examples of transformations applied to images during self-supervised phase.

V. Tools Used

For the development of the proposed system, the code was compiled through the Google Colab components [23]. The processor used was the Intel(R) Xeon(R) CPU @ 2.30GHz, the GPU component was NVIDIA-SMI 470.74 and the maximum RAM available was 12 Gb.

Furthermore, the environment where the system was developed used the Python version 3.7.12 and the Tensorflow [24] and Keras [25] libraries.

The use of this work environment revealed some limitations in the system development. It proved to be impossible to use more than 5,000 images to train the models or increase the batch size. By incrementing either of these parameters, the session would close, and progress was lost.

VI. Experimental Results

So that it is possible to determine the usefulness of using self-supervised learning to pre-train a classifier, it is essential to determine the baseline. In this case, the baseline is achieved when the classifier is trained only in a supervised manner. Consequently, using only set B from the Table II and without applying any type of transformation, the classifier is randomly initialised and trained. Table IV shows the average results of the accuracy and F1 score.

TABLE IV
Baseline classifier performance.

Average Accuracy [%]	Average F1 Score [%]
82.8 ($\pm$3.9)	87.0 ($\pm$3.6)

A. Comparison between different pretext task and baseline

In order to determine whether there is an advantage to using self-supervised learning for network pre-training, the best models of each pretext task studied are compared with the baseline.

For the image reconstruction (IR) task, the classifier with the best performance was obtained when pre-trained using Colour transformation to increase the training set. In this case, Colour transformation was applied to the initial set containing 1,000 images, so that in the final training set, each image had exactly two versions, the original and one transformed image. In this task, the transformed versions of each original images were obtained before the network training, therefore the images are constant in all epochs.

For the two contrastive tasks, SimCLR and Barlow Twins, each image of the initial set is required to have two versions to train the network, as shown in Fig. 4 and Fig. 5. For both tasks, the classifier with the best performance was acquired in a similar manner, i.e. by using the same group of transformations. Thus, to obtain the two versions, Colour and Rotation transformation was simultaneously applied. In the end, half of the versions were the original image and the other half corresponded to the transformed images. For both tasks, the 2,000 images from set A in Table II.

In both contrastive methods, the batches contain 128 images which means it's requires 16 iterations to cover the initial set of images. Unlike image reconstruction pretext task, the transformations are applied to each batch and not to the entire dataset directly. The total number of final samples is equal to the sum of all the pairs in each epoch, therefore, in SimCLR there are $2 \cdot 128^2 \cdot 16 = 524.288$ final samples and for Barlow Twins there are $128^2 \cdot 16 = 262.144$ final samples in each epoch.

After the self-supervised phase, the encoder component of each resultant network was extracted and used to initialise the respective classifiers. To train each classifier, the 500 images from dataset B in Table II were used without any transformation being applied.

The results shown in Table V suggest that the classifiers obtained from image reconstruction and Barlow Twins pretext tasks can achieve better performances compared to the baseline. However, it is essential to execute a statistical test to be able to infer that these tasks provides better performance when compared to the baseline.

The F1 Scores from each iteration of cross-validation provided the data used to calculate the statistical test t-Test. Through the t-test results, it is only possible to state that the pretext task image reconstruction allowed a classifier with better performance than the baseline to be achieved, since only in this case is the p-value less than 5%. Generally, the contrastive tasks score better results when compared to

TABLE V
PERFORMANCE OF THE BASELINE AND THE BEST CLASSIFIERS OBTAINED FOR EACH OF THE PRETEXT TASKS.

Name	Pretext task # Initial images	Transf.	Average Accuracy [%]	Average F1 Score [%]
Baseline	-	-	82.8 (±3.9)	87.0 (±3.6)
IR	1,000	Colour	87.2 (±4.5)	90.9 (±3.1)
SimCLR	2,000	Colour + Rotation	83.0 (±5.6)	86.9 (±5.2)
Barlow Twins	2,000	Colour + Rotation	86.0 (±5.7)	89.5 (±5.0)

TABLE VI
PERFORMANCE OF THE BEST CLASSIFIERS FOR EACH OF THE PRETEXT TASKS WHEN TRANSFER LEARNING AND FINE-TUNING IS CARRIED OUT.

Name	Pretext task # Initial images	Transf.	Transfer Learning F1 Score [%]	Fine-tuning F1 Score [%]
IR	1,000	Colour	70.5 (±4.1)	90.9 (±3.1)
SimCLR	2,000	Colour + Rotation	66.9 (±7.2)	86.9 (±5.2)
Barlow Twins	2,000	Colour + Rotation	74.1 (±4.7)	88.4 (±5.0)

generative tasks. However, considering that the data available for training the networks is reduced, it becomes complicated to generate so many negative pairs and consequently, the network has more difficulty in meeting the goal. Furthermore, the images used for the training are similar given that the existent objects, such as trees, smoke, and fire, are common in many of the images.

B. Transfer-learning vs. Fine-tuning

With this experiment, it is expected that the results between two pre-trained classifiers with the same pretext task can be compared, but one is obtained by fine-tuning and the other with transfer learning. In this experiment, the pre-trained networks that achieved the best performances described in Section VI-A were used.

The results presented in Table VI allow us to conclude that by using fine-tuning instead of transfer learning, it is possible for classifiers to achieve a better performance. This suggests that, although the previously learned network helps the initialisation of the classifiers, the learned weights are not fully adequate for the classification task and require some adjustment.

C. Number of images during self-supervised phase

In this experiment, two ways of incrementing the dataset were considered to train the network during the self-supervised phase. First, the aim was to verify whether the classifiers could achieve better performance when the number of original images used during the supervised phase increased. Furthermore, given that contrastive tasks require a large number of images and as 2,000 is a reduced set, the impact of further reducing the number of images used was not analysed. Thus, for this experiment, only the image reconstruction pretext task was used. The number of images used for training ranged between 500, 1,000 and 2,000, all belonging to set A in Table II, with no transformation being applied to the images. Once again, from the networks obtained, the encoder was extracted and the respective weights were used to initialise the classifier. To train and evaluate the performance of each classifier, the 500 images from set B were used without applying any transformation.

From the results presented in Table VII, it is not possible to conclude that increasing the number of original images during the self-supervised phase is beneficial or harmful when initialising the encoder weights of the classifier. Comparing the results of the three classifiers with the baseline, the performance of the pre-trained classifiers for the different number of images appear to surpass the baseline. However, to confirm this assertion, it is essential to perform the t-Test.

TABLE VII
CLASSIFIER PERFORMANCES AS A FUNCTION OF THE NUMBER OF IMAGES USED FOR TRAINING DURING THE SELF-SUPERVISED PHASE.

# Images during self-supervised phase	Average Accuracy [%]	Average F1 Score [%]
Baseline	82.8 (±3.9)	87.0 (±3.6)
IR with 500	84.0 (±4.5)	88.1 (±3.2)
IR with 1,000	84.2 (±2.6)	88.3 (±2.4)
IR with 2,000	84.2 (±4.3)	88.5 (±3.0)

After calculating the t-Test between the baseline and each pre-trained classifier for the different pretext tasks, none of the classifiers presented a p-value lower than 5%. Thus, it is not possible to state that increasing the number of original images necessarily results in a classifier with better performance.

The second approach to increase the dataset was to use data augmentation. In this case, each image in the initial set A has at least one version in the final set, one being the original version of the image and possibly transformed versions. As a starting point, 1,000 original images from set A were used and to which some transformations were applied. For the first transformation groups, that is, Colour, Rotation, Colour or Rotation and Colour + Rotation, the ratio of transformed images was altered between 50% and 100% obtaining 1,500 and 2,000 total images, respectively.

Observing Fig. 7, the situations in which colour transformation was applied to all images, that is, in the case of the Colour and Colour + Rotation groups,performance

improvement can be seen when comparing the cases where the classifier was pre-trained with 1,000 original images and 2,000 images (1,000 original and 1,000 transformations). On the contrary, the use of non-colour transformations does not contribute to an improvement in performance of the resulting classifiers, exhibiting similar performance to the pre-trained classifier with just the initial 1,000 images.

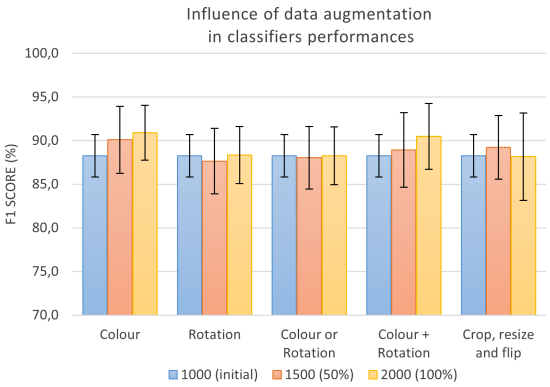


Fig. 7. Performance of pre-trained classifiers as a function of increasing the total number of images by applying transformations to 50% and 100% of the images.

4,000 and 5,000 images. In the case of the 4,000 images, the Colour, Rotation and Colour + Rotation transformations were applied so that the final training set had 1,000 original images, 1,000 with colour transformation, 1,000 with rotation transformation and 1,000 simultaneously transformed with colour and rotation. To obtain the 5,000 images, in addition to the transformations described above, it was also applied crop, resizing and horizontal flip were also , simultaneously applied , to the 1,000 initial images. It should be noted that despite reaching a training dataset with more images, it is not guaranteed that augmenting the dataset using multiple transformations results in better performance [26]. In the case of reduced datasets, applying several transformations, such as rotation, colour manipulation and cropping may not prevent the model from overfitting.

Table VIII shows the performances of the classifiers for each of the situations. In this case, despite overfitting not being seen, it was also not possible to achieve a performance improvement compared to the pre-trained classifier with the Colour grouping. The best option is to apply exclusively Colour transformation to duplicate the set since better results are obtained and less memory, GPU and time is consumed.

TABLE VIII
CLASSIFIER PERFORMANCES WHEN USING 1000, 2000, 4000 AND 5000 IMAGES WITH DIFFERENT GROUPS OF TRANSFORMATIONS DURING SELF-SUPERVISED PHASE.

Transformation Groups	**# Images**	**Average Accuracy [%]**	**Average F1 Score [%]**
Original	1000	84.2 (±2.6)	88.1 (±3.2)
Colour	2000	87.2 (±4.5)	90.9 (±3.1)
Colour, Rotation and Original	4000	86.0 (±3.9)	89.6 (±3.8)
All	5000	86.6 (±3.9)	90.2 (±3.2)

VII. CONCLUSIONS AND FUTURE WORK

For fire detection, one of the main challenges identified is the lack of labelled image datasets to train the models for the supervised tasks, such as classification or segmentation. To make up for the scarcity of these datasets a solution was developed that would take advantage of thousands of publicly available images without annotations.

The main purpose of this thesis was to determine if self-supervised learning could overcome methods that rely exclusively on supervised learning when there is a small labelled dataset for training and evaluating models. Hence, three different pretext tasks were used to determine the impact that self-supervised learning could have on the classification network. For the generative pretext task proposed, that is, image reconstruction, the classifier outperformed the baseline, which was trained using only supervised learning. In particular, an improvement of 3.9% for F1 Score was observed. Therefore, it is possible to conclude that the use of self-supervised learning to initialise a classifier is worthwhile. In the case of contrastive tasks, although there is no improvement over the baseline, the Barlow Twins task achieves a similar performance. The main explanation for not surpassing the baseline performance is related to the fact that the networks need more images to solve the contrastive tasks proposed in this thesis.

Initially, the idea was to use a larger unlabelled dataset. However, this was not possible since the virtual environment where the system was developed did not allow the dataset to be increased, at the expense of ending the session without saving the progress achieved.

An aspect that proved to be relevant when verifying the usefulness of self-supervised learning was the introduction of transformations. The influence of applying different transformations was analysed, first when used individually and then mixed with others. The transformation that was seen to have more impact was the Colour transformation which, when used individually, surpassed the baseline performance.

Additionally, at an early stage of wildfires, the fire is often hidden from aerial view due to vegetation. Nevertheless, the smoke produced by the fire quickly becomes visible in an aerial perspective thus its detection can play a crucial role in monitoring and detecting forest fires. In the future, the classification and segmentation of smoke can be studied using the proposed methodology.

Moreover, it is hoped that this work will serve as a motivation to further study methodologies that use self-supervised learning, as it has been proven that it can surpass supervised methods when there is limited labelled data available. Furthermore, self-supervised learning is an emerging solution and, in this thesis, only a small set of methods was addressed. The literature is constantly evolving, so it may be beneficial to study other techniques.

Finally, with this dissertation, we hope to have developed a methodology that will contribute to the Firefront project, assisting firefighting forces in monitoring fires.

REFERENCES

[1] T. Çelik and H. Demirel, "Fire detection in video sequences using a generic color model," *Fire Safety Journal*, vol. 44, no. 2, pp. 147 – 158, 2009.

[2] B. U. Toreyin and A. E. Cetin, "Online detection of fire in video," in *2007 IEEE Conference on Computer Vision and Pattern Recognition*, 2007, pp. 1–5.

[3] C. Yuan, Z. Liu, and Y. Zhang, "Aerial images-based forest fire detection for firefighting using optical remote sensing techniques and unmanned aerial vehicles," *Journal of Intelligent & Robotic Systems*, vol. 88, no. 2-4, pp. 635–654, 2017.

[4] M. Batista, B. Oliveira, P. Chaves, J. C. Ferreira, and T. Brandao, "Improved real-time wildfire detection using a surveillance system," in *IAENG*, 2019, pp. 520–526.

[5] H. Cruz, M. Eckert, J. Meneses, and J.-F. Martínez, "Efficient forest fire detection index for application in unmanned aerial systems (uass)," *Sensors*, vol. 16, no. 6, p. 893, 2016.

[6] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–44, 05 2015.

[7] J. Lehr, C. Gerson, M. Ajami, and J. Krüger, "Development of a fire detection based on the analysis of video data by means of convolutional neural networks," in *Iberian Conference on Pattern Recognition and Image Analysis*. Springer, 2019, pp. 497–507.

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[9] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015.

[10] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <0.5 MB model size," 2016.

[11] Q. Zhang, J. Xu, L. Xu, and H. Guo, "Deep convolutional neural networks for forest fire detection," in *Proceedings of the 2016 International Forum on Management, Education and Information Technology Application*. Atlantis Press, 2016, pp. 568–575.

[12] Y. Chen, Y. Zhang, J. Xin, G. Wang, L. Mu, Y. Yi, H. Liu, and D. Liu, "Uav image-based forest fire detection approach using convolutional neural network," in *14th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, 2019, pp. 2118–2123.

[13] B. Amaral, "Fire and smoke detection with weakly supervised methods," Master's thesis, Instituto Superior Técnico, 2021.

[14] L. Jing and Y. Tian, "Self-supervised visual feature learning with deep neural networks: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 4037 – 4058, 2020.

[15] X. Liu, F. Zhang, Z. Hou, L. Mian, Z. Wang, J. Zhang, and J. Tang, "Self-supervised learning: Generative or contrastive," *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–1, 2021.

[16] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *science*, vol. 313, no. 5786, pp. 504–507, 2006.

[17] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th international conference on Machine learning*, 2008, pp. 1096–1103.

[18] N. Komodakis and S. Gidaris, "Unsupervised representation learning by predicting image rotations," in *International Conference on Learning Representations (ICLR)*, 2018.

[19] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *International conference on machine learning*. PMLR, 2020, pp. 1597–1607.

[20] J. Zbontar, L. Jing, I. Misra, Y. LeCun, and S. Deny, "Barlow twins: Self-supervised learning via redundancy reduction," 2021.

[21] A. Chaudhary, "The illustrated simclr framework," 2020, Último acesso: 17.09.2021. [Online]. Available: https://amitness.com/2020/03/illustrated-simclr

[22] A. A. Bombeiros distrito guarda. Bombeiros portugueses. Last accessed: 29.09.2021. [Online]. Available: http://www.bombeiros.pt/galeria/index.php

[23] E. Bisong, *Google Colaboratory*. Berkeley, CA: Apress, 2019, pp. 59–64.

[24] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[25] F. Chollet *et al.* (2015) Keras. [Online]. Available: https://github.com/fchollet/keras

[26] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, no. 1, pp. 1–48, 2019.