/gamerule doFireTick true
A binary classifier for recognizing fire

Leonardo Biason (*2045751*) Lorenzo Marinelli (*2043092*) Oscar Michele Norelli (*2046721*)

***Abstract*—This article contains the report for the Deep Learning course challenge, which asked to the course participants to build a binary classifier which should recognize whether there is some fire in a picture. We here present a possible implementation of this classifier, which reached an astonishing accuracy of 98.78%**

Sapienza, ACSAI, CNN, ResNet, Deep Learning, Computer Vision

Check our repository on GitHub
`ElBi21/doFireTick`

I. Introduction

CNNs have been widely adopted in many uses as of today, either for accessibility features or as useful detection tools. Many examples of medical emplyments exist, and that's only one of the many possible use cases. However, this kind of technology can also be useful for detecting dangers and perils, and for automating checking routines. We may consider CNNs as a type of sensors, which react to certain visual events. With this paper, we present a possible use case for CNNs, in particular, one tied to fire prevention.

Fire prevention is usually ensured thanks to smoke sensors, which cover small areas and allow for a granular control of reduced ambients. However, this is only possible when the ambient is a closed one, and it's hard to apply this kind of techniques on open spaces. That is where CNNs may well fit: by using a visual sensor on open areas, it becomes easier to control larger areas for fire hazards, and to alert any competent autority, should a fire break.

This is ultimately the scope of this paper: to present a working model which can be employed for simple fire detection tasks. We will here explain the structure used, and the performance reached by the model.

In order to perform the given task, we decided to use an ensemble of three ResNet [1] models, and to train them with a dataset which has been previously augmented by manipulating all the images.

`label: 0` `label: 1`

Fig. 1. Example of images belonging to the dataset with the corresponding label

II. Structure of the dataset

For this task, we used a dataset composed of 10,926 images; we later applied some steps of data augmentation so that we could achieve better results by making the model generalize enough.

The dataset is separated in two folders (for the two classes), where each folder represents the label of each image:

- folder `0` contains the images that do not contain fire;
- folder `1` contains the images that contain fire;

A. Data augmentation

Data augmentation is a common step which is usually made to enhance the number of items in a dataset by using items already belonging to said dataset. We applied the following operations to our dataset:

- **Random rotation**: to each image, a random rotation of x degrees could've been applied, where $x \in [0, 15]$. This ensures that the fire would be recognized even if it was not "parallel" with respect to the y axis of the image;
- **Color jitter**: a random color jitter was also applied to all the images. The parameters passed to the jitter function are listed on Table I. By

TABLE I
PARAMETERS OF THE COLOR JITTERING FUNCTION

Parameter	Value
`brightness`	0.2
`contrast`	0.2
`saturation`	0.2
`hue`	0.1

applying a color jitter, we make sure that the model would recognize a fire also when light conditions were not optimal in the photo (for instance if the image is either too dark or too bright);
- **Brightness noise**: for each image also some noise was added, and this was applied through a lambda function. For each image, some noise of equal size would be generated, and then through clamping the noise got applied to the image. This kind of noise allowed us to train the model to recognize a fire also in some harsh conditions, and helps avoiding overfitting.

After augmenting the whole dataset, the amount of samples went from 10,926 to 21,852, doubling the size of the dataset and allowing the models to train and reach more efficient results.

III. STRUCTURE OF THE ENSEMBLE

The ensemble is composed of three models, based on a ResNet 152, a ResNet 18 and a ResNet 50. The complete specifications of the three models can be found in Table II. Some small modifications have been made to each model, in order to achieve different results. We here list all the modifications that have been made to each model, alongside an explanation of why they have been applied.

For the performance evaluation process, the predictions from the three models could be combined in many ways:

- **Average**: in average mode, the outputs from the three models are averaged together, and this result represents the output of the ensemble. This is the method that we used;
- **Weighted average**: the outputs from the three models are averaged based on some given weights. This can be useful in case we consider the "opinion" of some models more meaningful than the others';
- **Vote**: the class considered to be the "most likely" by the majority of the outputs of the three models will be the class returned by the ensemble.

Model 1 • ResNet 152

In order to perform this classification, we started with a solid network, which originally was going to be the only network. After a throughout evaluation, we ultimately decided to create a model ensemble.

The ResNet 152 model also got some modifications on our end, such as the addition of two linear layers at the end of the network. The fore-last layer is connected to the output of the ResNet with a dropout of 0.5, and uses the ReLU as an activation function; the last layer is connected to the previous layer with a dropout of 0.5 and has two final neurons, one for each class (so one which stands for class `0`: `no fire` and one for class `1`: `fire`).

This model employs the focal loss function (implemented manually, in the Jupyter notebook present in the code repository) and the AdamW optimizer. Moreover, it was trained on the augmented dataset.

The reason why we decided to employ a ResNet 152 is that it's a stable, valid network: the ResNet model proved to be an important network, that could be easily adapted to our needs. This was a key point of our work: by using a pre-trained model and adding a few layers, we were able to construct a model which could fit very well our use case and that performs better than other architectures, by needing less training time;

Models 2 & 3 • ResNet 50 and ResNet 18

The other two models for the ensemble are a ResNet 50 and a ResNet 18, without any additional layer. There are several reasons for this choice:

- **Feasible**: using shallower models than ResNet 152 makes the training process faster;
- **Performance**: both models still provide a very good accuracy score on the validation set;
- **Meaningful**: the structural difference between this two models and ResNet 152 is crucial to have meaningful changes in the feature space, which are reflected in the output of the models;
- **Dataset differences**: we didn't apply data augmentation for ResNet 50: the change of the used datasets for training does not affect negatively the accuracy score and improves the difference between the two models behavior, in order to have a reasonable ensemble.

TABLE II
STRUCTURE OF THE ENSEMBLE

Model	Layer	Dimension	Loss function	Optimizer	Data augmentation?
ResNet 152	Backbone of the ResNet 152		Focal Loss	AdamW	✓
	Dropout	0.5			
	Linear (ReLU)	(1024×512)			
	Dropout	0.5			
	Linear	(512×2)			
ResNet 50	Backbone of the ResNet 50		Cross Entropy Loss	SGD	×
ResNet 18	Backbone of the ResNet 18		Focal Loss	SGD	✓

✓: Yes ×: No

IV. PERFORMANCE EVALUATION

The performance for the ensemble and the singular models that we described in this paper can be found in Table III.

Unfortunately, the difference in accuracy between the singular models and the ensemble is not as large as desired, but it's still an improvement over the single-model approach.

TABLE III
PERFORMANCE OF THE ENSEMBLE AND THE THREE MODELS

Model	Accuracy
ResNet 152	98.46%
ResNet 18	98.49%
ResNet 50	98.75%
Average of the ensemble	98.81%

The items have been listed in ascending order of accuracy

REFERENCES

[1] "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90.