Report IA323 - Fire Detection

Antoine Domingues, Clément Laroudie, Maxence Leguéry, Billy Weynans

1 Introduction

Fire detection is a critical task to prevent disasters and ensure public safety. Traditional methods rely on sensors or rule-based image analysis, which can be prone to false alarms and limited in real-world conditions. Deep learning and computer vision offer a promising alternative to automatically identify fire in images with high accuracy.

However, one of the main challenges in training such models is the limited availability of annotated datasets. Fire incidents are relatively rare, and manually labeling large-scale datasets is both time-consuming and costly. This scarcity of labeled data can lead to overfitting and poor generalization in deep learning models.

2 Dataset

We will be using the Wildfire Prediction Dataset [1] which is a collection of satellite images. This is a *balanced* dataset with 2 classes : *nowildfire* and *wildfire*. Here are the original splits of the dataset :

split	**train**	**validation**	**test**
samples	30 250	6 300	6 300

However, the main objective of the project is to be in a low-labelling setting. We have to impose ourselves a constraint, forbidding access to the labels in the original training set. Therefore, in order to obtain labelled samples to train our model, we split the original validation set into a new training set (70%) and a new validation set (30%). The number of samples per split is summarized as follow :

split	**train (unlabeled)**	**train**	**validation**	**test**
samples	30 250	4 410	1 890	6 300

3 Experiments

We performed experiments with different architectures and various methods to improve our results, starting from a simple baseline. We progressively tried to incorporate the unlabelled data in the training process to push further the results.

3.1 Simple CNN

We first simply tried to train a basic CNN. The main advantages are efficiency and speed. It didn't take too long to reach 92.4% accuracy on the test set. (From now on, every accuracy that will be mentioned is an accuracy over the test set.)

However, we can't be satisfied with that : we didn't use any of the images from the training set. Moreover, recall is an important factor when we want to detect important events as wildfire. As we can see, with the current recall of 0.871, 13% of actual wildfires were missed, which represents 365 samples. The confusion matrix and additional metrics are presented in table 1. The main objective will be to improve our model using the unlabelled training images in a clever way.

1. https://www.kaggle.com/datasets/abdelghaniaaba/wildfire-prediction-dataset/data

Metric	Value
Accuracy	92.4%
Precision	95.6%
Recall	87.1%

(a) Performance metrics

		Prediction: n	Prediction: p
Actual	n	3 367	113
	p	365	2 455

(b) Confusion matrix. n = *no wildfire*, p = *wildfire*

Table 1 – Results for the CNN (threshold 0.5)

3.2 Auto encoder with MLP

In the previous method, we trained a CNN with a random initialization. We wanted to leverage the unlabeled actual train set. To do so, we built an auto-encoder which can be trained in an unsupervised way using the Mean Square Error between the initial image and the reconstructed one. The chosen encoder corresponds to the convolutional part of the previous architecture (CNN). They both share the same architecture.

When the auto encoder has been trained on the train set we reuse the encoder and initialize the convolutional part of the previous architecture with it. Hence, we can do the same steps as with the CNN and train the encoder and MLP head with the labeled data. With this modified architecture, we obtained an accuracy of 93.3%.

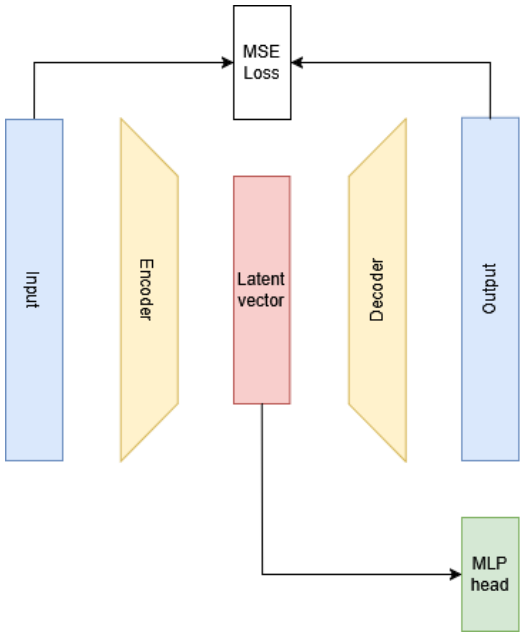

Figure 1 – Auto-encoder with MLP head

3.3 Transfer learning : Pretrained Resnet50

To enhance our baseline model, we used a ResNet-50 architecture, which benefits from residual connections to facilitate gradient flow and improve training stability. By initializing the network with pretrained ImageNet weights [2], we enable the model to transfer learned feature representations from a large-scale dataset, accelerating convergence and improving generalization. With this approach, we achieve an accuracy of 98.4%, demonstrating the effectiveness of transfer learning in our task.

3.4 Auto-encoder and transfer learning

In the previous section, the use of a pretrained Resnet50 led to better results than the Auto-encoder with the MLP. Another setting that has been tested is to mix both architectures : the Auto-encoder with MLP and the pretrained Resnet50. Hence, the pretrained Resnet50 was used as an encoder (without the feature classification layer) and fine-tuned in an unsupervised way as in the previous method in Section 3.2. The goal was the same : we wanted to leverage the unlabeled training set.

We sequentially did two successive fine-tuning, one with the auto encoder in an unsupervised way, and the other with the labeled training data.

With this approach, we obtained an accuracy of 98.6% which is better than the previous two methods alone.

3.5 Deep Ensemble

A costly but effective improvement is the Deep Ensemble approach. By training multiple models independently and aggregating their predictions, we enhance robustness and reduce variance, leading to a more reliable model. Despite the increased computational cost, this method boosts accuracy to 98.8%, demonstrating the benefits of ensemble learning in improving generalization.

3.6 Pseudo-labelling

To further improve performance, we performed pseudo-labeling using our best model so far (Deep Ensemble, 98.8% accuracy). We assign pseudo-labels to the unlabeled data in the training set, effectively expanding our dataset. For simplicity, we used a labelling threshold of 0.5 which corresponds to the most likely label predicted by our network. The model is then fine-tuned on this augmented dataset, benefiting from additional supervision. This approach boosts accuracy to 99.2%, demonstrating the power of self-training in semi-supervised learning.

3.7 Vision transformer

Until now, we only used convolutional neural network. Let's explore the Vision Transformer (ViT).

3.7.1 Simple classification training

We first tried a simple training from scratch using a classification head. We reach 90.9% accuracy. This is way less than the previous methods. In fact, because we are using a transformer architecture, it needs a large amount of data to be properly trained. Here, we are even limited regarding our training data which are mostly unlabeled.

3.7.2 SimMIM pretraining

Similarly to what we did with the auto encoder, we pre-trained our ViT on the unlabeled data using SimMIM [3]. The idea is to mask some parts of the image and learn to reconstruct it. We used the L1 reconstruction loss on each masked patch.

2. https://www.image-net.org/
3. https://arxiv.org/abs/2111.09886

With this pre-training, we learned intermediate feature representations specific to this dataset. We, then, hope to gain performance by fine-tuning this model with a classification task. We got 91.9%.

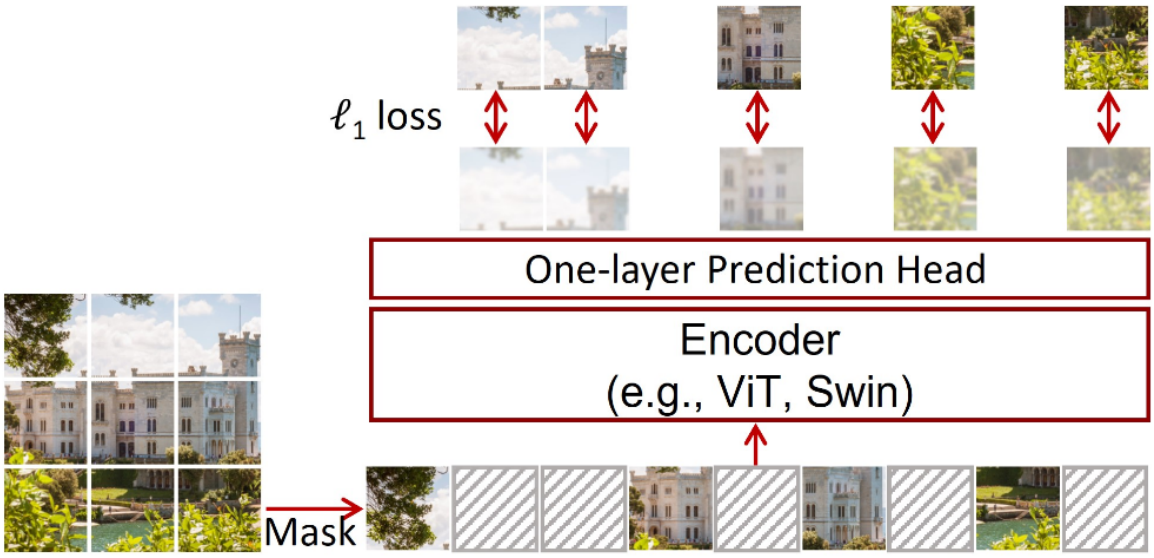

Figure 2 – SimMIM principle

4 Results

We summarized our results in table 2.

Method	**Accuracy** (%)	**AUROC**
Simple CNN	92.4	0.926
AE with MLP	93.3	0.980
Resnet50	98.4	0.997
AE with ResNet50	98.6	**0.998**
DE resnet50	98.8	**0.998**
DE resnet50 pseudo labels	**99.2**	**0.998**
ViT	90.9	0.963
ViT SimMIM	91.9	0.966

Table 2 – Results of our methods on the test set.

The accuracy reported in this table is computed with a threshold of 0.5 (meaning the predicted label is the one with the maximum probability after the softmax function). To put these results into perspective, we also reported the AUROC (Area Under the ROC curve) as it does not depend on the threshold chosen. Indeed, a different threshold may lead to a better accuracy. In an industrial context, choosing this threshold is critical because a decision has to be taken. For example, ones may want to choose a threshold to reduce the false positive rate (despite lower accuracy). In our research context, the AUROC allows us to evaluate the overall classification ability of the different models.

The code is available on GitHub [4]. See README.md for more information. The weights are available on HuggingFace [5].

4. https://github.com/maxenceleguery/fire-detection
5. https://huggingface.co/collections/Maxenceleguery/fire-detection-67ab264d354af57d29299ed1

5 Conclusion

In this work, we explored various deep learning architectures and training strategies to improve classification performance in a low-labelling setting. Starting with a simple CNN baseline, we progressively enhanced our models using transfer learning, deep ensembles, and pseudo-labeling... Additionally, we evaluated transformer-based architectures (ViT) which, despite their potential, did not outperform our CNN-based approaches in this setting.

Our results highlight the effectiveness of transfer learning, ensemble methods, and semi-supervised learning in improving classification performance. Future work could explore self-supervised learning techniques, data augmentation strategies, and hybrid CNN-Transformer models to further enhance generalization and robustness.