UNIVERSITÀ DEGLI STUDI DI SALERNO

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE
ED ELETTRICA E MATEMATICA APPLICATA

Machine Learning

GROUP 1

Intonti Giovanni 0622701994

Squitieri Beniamino 0622702021

Vitale Antonio 0622701988

Problem aims and goals

The objective of the project is to develop a fire or smoke recognition algorithm using machine learning techniques, utilizing the knowledge gained during the Machine Learning course of 2022/2023, the Google Colab platform, and the A100 graphics card provided by the MIVIA laboratory for training the model. The expected outcome is to train a model capable of quickly identifying the presence of fire, thereby replacing human agents in monitoring video surveillance environments captured during both day and night, indoors and outdoors, and from varying distances.

Network chosen

Due to the nature of the problem, we have chosen to adopt a 3D CNN network introduced during the course: SlowFast_R50. This model utilizes two different pathways for each video: slow pathway and fast pathway. The slow pathway operates at a reduced frame rate, in this particular experiment, 32 frames of the video to be analyzed, while the fast pathway operates at a higher frame rate and is capable of capturing temporal information for action recognition in videos.

One pathway is designed to capture semantic information derived from a few sparse frames with a slow update rate. On the other hand, the other pathway is responsible for capturing rapidly evolving information, operating at a high update rate and high temporal resolution. This particular network has been trained on datasets such as Kinetics, Charades, and AVA, which are highly focused on human action recognition. However, the choice to use the aforementioned network was driven by three main reasons:

- It was presented during the course in a final exercise: the network was introduced and demonstrated as part of the course;
- Preliminary tests on the provided videos showed that the network exhibited the ability to recognize certain relevant classes, such as "extinguish fire," "cooking on a campfire," and "juggling fire." Despite being trained for human action recognition, these initial results indicated its potential for fire-related recognition;
- Being a 3D CNN network, it was deemed as an ideal choice for fire or smoke recognition due to its ability to differentiate between slow and fast pathways. Rather

than solely relying on static image analysis, this network leverages temporal and spatial dynamics in videos, making it well-suited for detecting fire or smoke occurrences.

The mentioned network is indeed much more complex compared to a regular CNN, and this complexity is reflected in the training times. Before obtaining the A100 graphics card, training was challenging due to the limited computational power provided by the Colab platform and the limited daily GPU usage time allocated by the platform. Additionally, the locally available computational resources were also insufficient for efficient training.

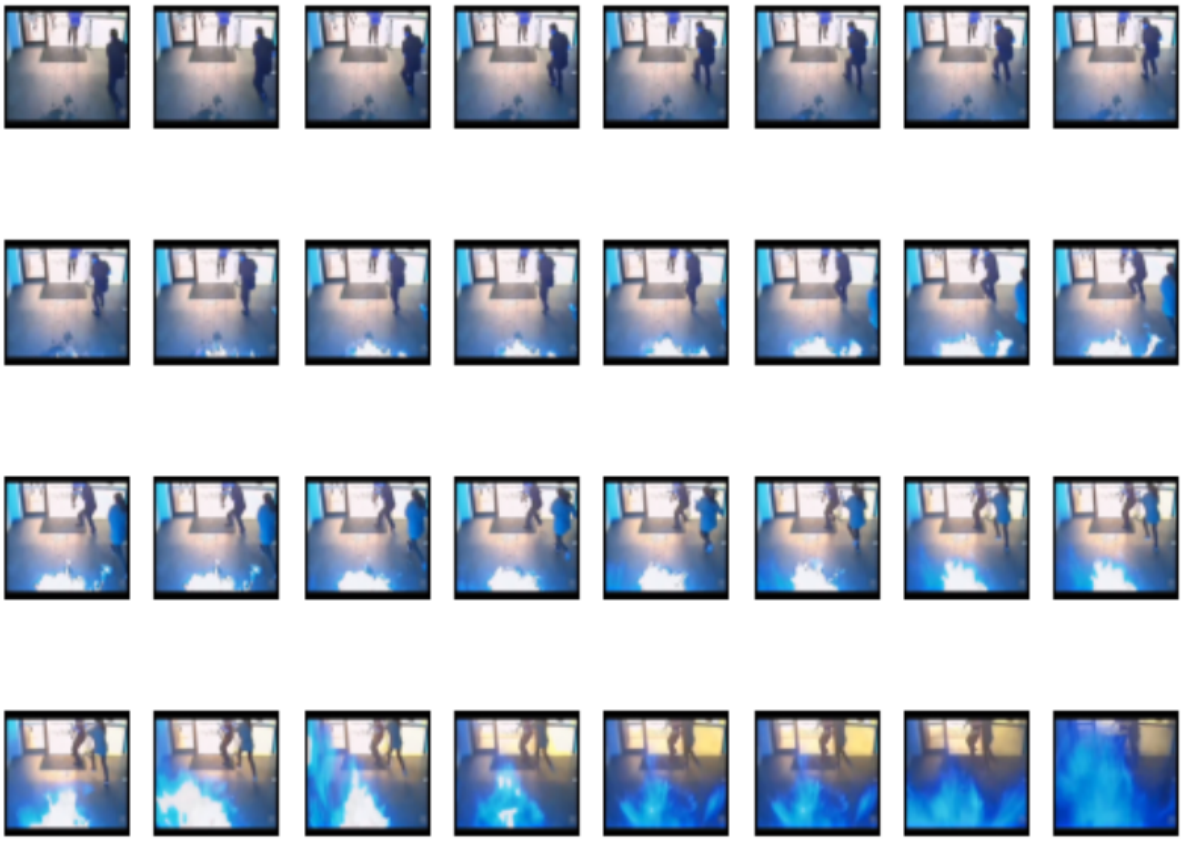
Data Visualization

The main idea of a SlowFast network is to have two separate data flows, one "slow" and one "fast," to capture both spatial and temporal information in videos. The slow flow is a tensor characterized by the following elements:

- **batch_size**: it represents the number of video samples processed simultaneously by the neural network.
- **num_frames** : It is the number of frames considered in the slow flow, with the standard value being 8.
- **channels**: it represents the number of color channels present in each frame, with the standard value being 3 (red, green, blue).
- **height**: It represents the height of the video frames in pixels, with the standard value being 224.
- **width**: It represents the width of the video frames in pixels, with the standard value being 224.

Since the selected model handles data in two distinct pathways, the standard network transformation was implemented in combination with image frame preprocessing to obtain a list of two tensors representing the slow and fast paths of the video retrieved from the data loader. At this point, the data was visualized to verify that the transformations were applied correctly and that the data was suitable for our problem. (For visualization purposes, the network's normalization step was omitted to display the images accurately.)

Format [8x3x224x224] , Path Slow



Format [32x3x224x224] , Path Fast

From the data visualization, it was observed that even relatively small or distant flames in the background are still visible enough for the network when resized to 224x224. Increasing the input image size would have significantly increased the processing time. Additionally, since the original network was trained on an input size of 224x224, it was decided to maintain this value.

Furthermore, it was noticed that there were many similar videos in the provided dataset. Initially, these videos were included in the early stages of training. However, later on, they were reprocessed as described in the "Dataset Augmentation" section.

Network transformations

Three variables were used to manage various data transformations:

- **transform**: this transformation is necessary to make the input frames compatible with the SlowFast_R50 network.

- **preprocessing**: This transformation is applied to each individual frame during the loading process by the DataLoader.
- **augmentation**: This transformation is used to perform data augmentation on the images loaded by the DataLoader.

transform

The standard transform of the network includes several operations:

- **UniformTemporalSubsample**: this operation uniformly subsamples the frames of the video so that the total number of frames matches the required input size.
- **Lambda**: this operation applies a lambda function to each pixel of the video, dividing the pixel value by 255. This is typically done to normalize the pixel values between 0 and 1.
- **NormalizeVideo**: this operation normalizes the video by using the mean and standard deviation provided by the network. It ensures that the pixel values are standardized to improve the training process.
- **ShortSideScale**: this operation resizes the video proportionally to its shorter side while maintaining the aspect ratio.
- **CenterCropVideo**: This operation crops the video at the center using a specified input dimension.
- **PackPathway**: this operation converts the video frames into a list of two tensors, one for the slow pathway and one for the fast pathway. This step separates the frames into their respective pathways, allowing the network to process them independently.

By applying these operations in sequence, the standard transform prepares the video frames for input to the network, ensuring proper subsampling, normalization, resizing, cropping, and pathway separation.
Later on, the Lambda, ShortSideScale, and CenterCropVideo transformations were removed. The Lambda transformation was removed because it is already performed during the image-to-tensor conversion in the ImglistOrdictToTensor class. As for the other two transformations, their rationale will be further clarified in the following paragraph titled "Crop Issue."

It's important to note that all these operations are applied to every frames of the videos in the dataset.

preprocessing

The transformation used is Resize, which resizes the frame dimensions according to the specified height and width parameters, conforming to the network's standard input size of 224x224. This operation is applied to each frame in the video.

augmentation

The augmentation operation is performed with a 10% probability on all frames of the video, and it applies one of the following transformations:

- **ColorJitter**: it applies a random alteration to the color channels of the image, such as brightness, contrast, saturation, or hue. The probability of applying this transformation is 0.5.
- **GaussianBlur**: it applies Gaussian blur to the image, resulting in a smoothed appearance. The probability of applying this transformation is 0.5
- **HorizontalFlip**: it horizontally flips the image, effectively mirroring it. The probability of applying this transformation is 1.
- **RandomBrightnessContrast**: it applies a random alteration to the brightness and contrast of the image. The probability of applying this transformation is 0.5.
- **HueSaturationValue**: it applies a random alteration to the hue, saturation, and value of the image. The probability of applying this transformation is 0.5.

These augmentation transformations help increase the diversity of the training data, improving the network's ability to generalize and handle variations in lighting, colors, and orientation.

Crop Problem

The CenterCropVideo transformation was removed following some initial tests on the network, as described in the "Crop Tries" section. It was observed that this transformation placed a significant emphasis on positive samples, resulting in the network consistently classifying every input as a video containing fire (class 1). This raised suspicions about the CenterCropVideo transformation since it performed a central crop, which could potentially remove fire or smoke completely from the processed video. As a result, the network could learn incorrectly from such videos labeled as class 1, leading to false learning.

Based on this analysis, the decision was made to remove the CenterCropVideo transformation and instead perform a Resize operation on the image. This way, all the original information of the video is retained, albeit acknowledging that the network was not

trained in this specific manner. Following this adjustment, the initial training attempts for our binary classification problem were initiated.

Training Steps

After identifying and addressing various issues in the network and preparing the data to be suitable for the network in terms of format and normalization, the final layer of the original network was replaced.

```
(6): ResNetBasicHead(
  (dropout): Dropout(p=0.5, inplace=False)
  (proj): Linear(in_features=2304, out_features=400, bias=True)
  (output_pool): AdaptiveAvgPool3d(output_size=1)
)
```

With a custom layer specifically designed for the binary classification problem:

```
(6): ResNetBasicHead(
  (dropout): Dropout(p=0.5, inplace=False)
  (proj): Linear(in_features=2304, out_features=1, bias=True)
  (output_pool): Sigmoid()
)
```

A custom layer was introduced specifically for the binary classification problem. The original linear layer, which had 400 output classes, was replaced with an untrained linear layer with a single output. Additionally, the AdaptiveAvgPool3d layer was replaced with a sigmoid activation function for the classification of videos with and without fire.

Initially, the training of the network focused solely on training this last layer, preventing the other layers from being trained. Afterward, parameter tuning was performed by modifying learning rate, weight decay, momentum, and batch size, as well as the number of epochs for each training step. The Stochastic Gradient Descent optimizer, as taught in class, was used for the training phase.

To handle the training process, utility functions were employed to divide the video frames into various folds, enabling K-fold cross-validation. The frames were appropriately labeled as 0 or 1 to indicate the absence or presence of fire, respectively.

Crop Tries

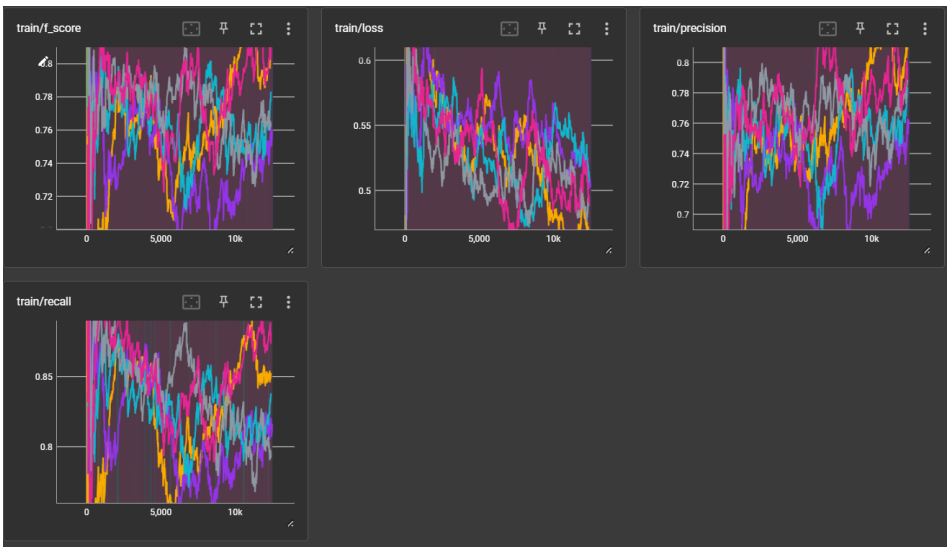
In the initial stages, it was decided to provide the network with input data in the same format as it was trained on. Therefore, the data was normalized according to the network's standard and cropped at the center after resizing the shorter side of the video frames.

This approach aimed to maintain consistency with the network's training data and leverage the transformations that were found effective during its training. By resizing the frames to a standard size and cropping them at the center, the original information of the videos was retained while ensuring compatibility with the network's input requirements.

Training parameters chosen

Initially, different values for learning rate, weight decay, and momentum were tested to observe how the network would respond.

- One initial choice was a high learning rate of 0.1, a weight decay of 0.00001, and a momentum of 0.9. However, the results obtained showed a high number of false positives. The network tended to converge to a state of classifying all inputs as positive (fire) epoch after epoch.
- Similar results were obtained even when decreasing the learning rate and attempting various fine-tuning strategies. Despite gradually reducing the time to convergence, the network still tended to classify all inputs as positive.
- These observations suggest that the chosen hyperparameters and training configurations did not effectively train the network to discriminate between positive and negative samples. Further adjustments and experimentation were necessary to achieve the desired classification performance.



Despite the heavy smoothing of the graphs, it is evident that the trend is not progressive, and the results are highly fluctuating. From the analysis conducted, it is believed that this phenomenon is primarily due to the small batch size used, which was set to 2 in this case, driven by the limited available RAM on the device used for training. With a WSL there were constraints on the amount of video frames that could be loaded by the data loaders, hence the decision to use a batch size of 2.

This small batch size, along with the learning rate, contributed to the oscillatory behavior observed in the training process. While reducing the learning rate mitigated the oscillations to some extent, they were still present. Additionally, the obtained results align with the figures, where a limited decrease in the training set loss does not correspond to a similar decrease in the validation set loss, which remains stagnant or even slightly worsens.

Considering these results and subsequent incomplete tests due to the evident degradation the network was experiencing, it was evident that the training time for the SlowFast network was quite demanding (approximately 190 seconds per epoch). As a result, further investigation was pursued, which involved removing the center crop transformation in favor of a more conservative resizing approach.

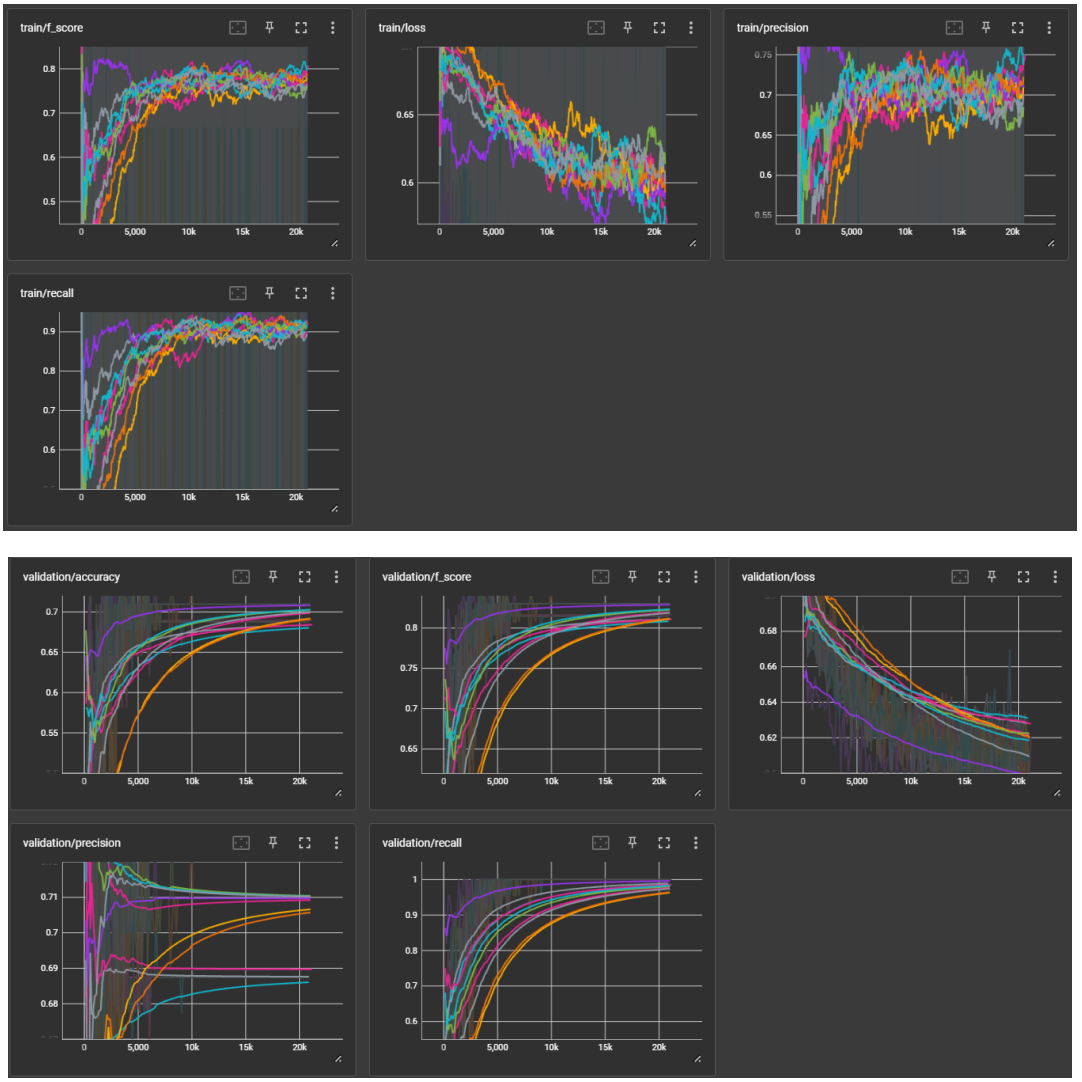
Tries with resize and no crop

Following the attempts made with the center crop transformation, the decision was made to remove it and introduce frame resizing using the Resize transformation. The frames of the video were resized to the desired network format of 224x224 during the data preprocessing phase, eliminating the need for the ShortSideScale transformation as well.

These changes resulted in a noticeable improvement in the data fetching speed by the data loaders, reducing the training time to approximately 130 seconds per epoch.

Training parameters chosen

In this training phase, a fine-tuning approach was adopted using the same parameters as before to verify whether the crop transformation was indeed the cause of the analyzed issue.



Following several training iterations and prolonged training sessions with the parameters described earlier, a similar oscillatory trend to that observed with the crop transformation was obtained. The loss values remained unsatisfactory, accompanied by precision and recall values of approximately 0.7 and 1, respectively.

Despite various training attempts, parameter adjustments, and even modifying the number of trainable layers, the network consistently exhibited this type of behavior. Through analysis conducted during the training days, it was concluded that the issue might be stemming from

dataset imbalance. The dataset had a significantly larger number of positive examples compared to negative examples, with a ratio of 70/30. This imbalance could have destabilized the network during the training phase.

Positive classification problem

The initially provided dataset had an imbalance in the ratio of videos with fire or smoke to videos without fire or smoke, with a ratio of 70 to 30. This imbalance caused the network, under the chosen parameters, to bias towards classifying every frame as fire. As a result, the obtained results were unsatisfactory, with false positives for every negative sample.

By rebalancing the dataset and providing the network with a more equal representation of both classes, it is expected to improve the network's ability to distinguish between fire-related and non-fire-related videos, resulting in more accurate and reliable predictions.

Dataset Balancing and training on the MIVIA account

To address this issue, the dataset was balanced by adding videos without fire, adjusting the ratio from 70/30 to 60/40. However, since the dataset was still somewhat imbalanced, further measures were taken to address it.

One approach was to modify the network architecture by introducing a weight parameter in the CrossEntropyLoss. This modification involved replacing the final layers of the network with two new layers. The weight parameter helps to assign higher weights to the minority class (videos without fire) during the training process, giving it more importance and reducing the bias towards the majority class (videos with fire).

By adjusting the architecture in this way and incorporating the weight parameter, it aims to mitigate the effects of the dataset imbalance and improve the network's ability to learn and make accurate predictions for both fire and non-fire videos.

```
(6): ResNetBasicHead(
  (dropout): Dropout(p=0.5, inplace=False)
  (proj): Sequential(
    (0): Linear(in_features=2304, out_features=128, bias=True)
    (1): ReLU()
    (2): Linear(in_features=128, out_features=2, bias=True)
  )
  (output_pool): Softmax(dim=None)
)
```

Following the dataset balancing and the provision of the A100 GPU by the MIVIA laboratory, the initial training tests were conducted on the laboratory's account. In this phase, the Adam optimizer was chosen instead of SGD.

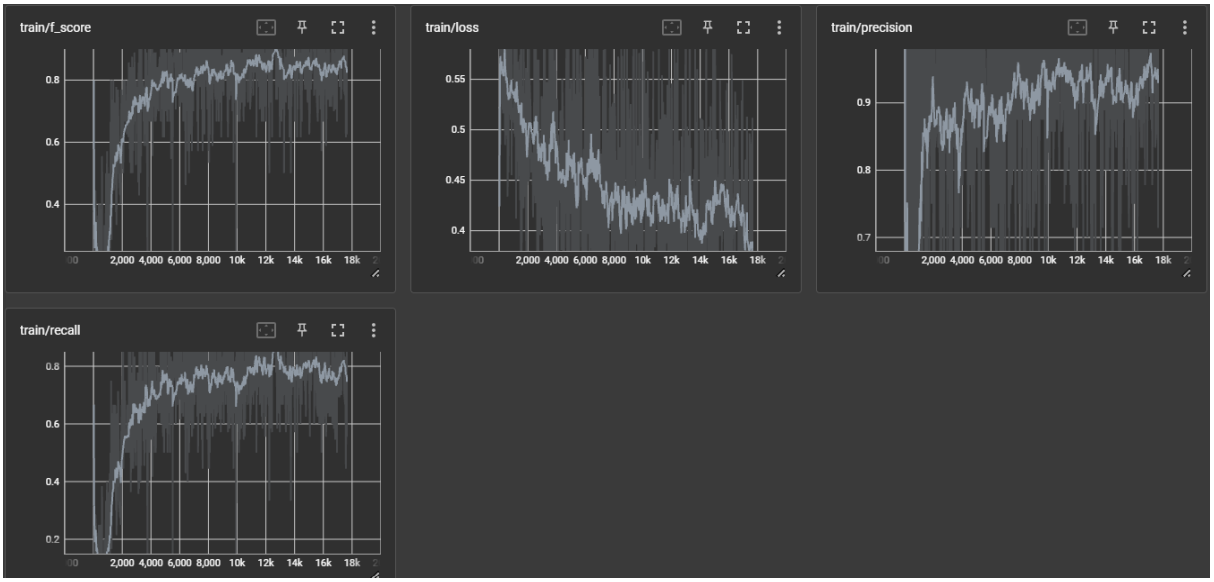
Adam is known for its adaptive learning rate capabilities, as it adapts the learning rate individually for each parameter in the model. This makes the training process more stable and helps alleviate the oscillations observed in previous analyses. By using Adam, the aim is to achieve smoother and more consistent training, improving the overall stability and convergence of the network.

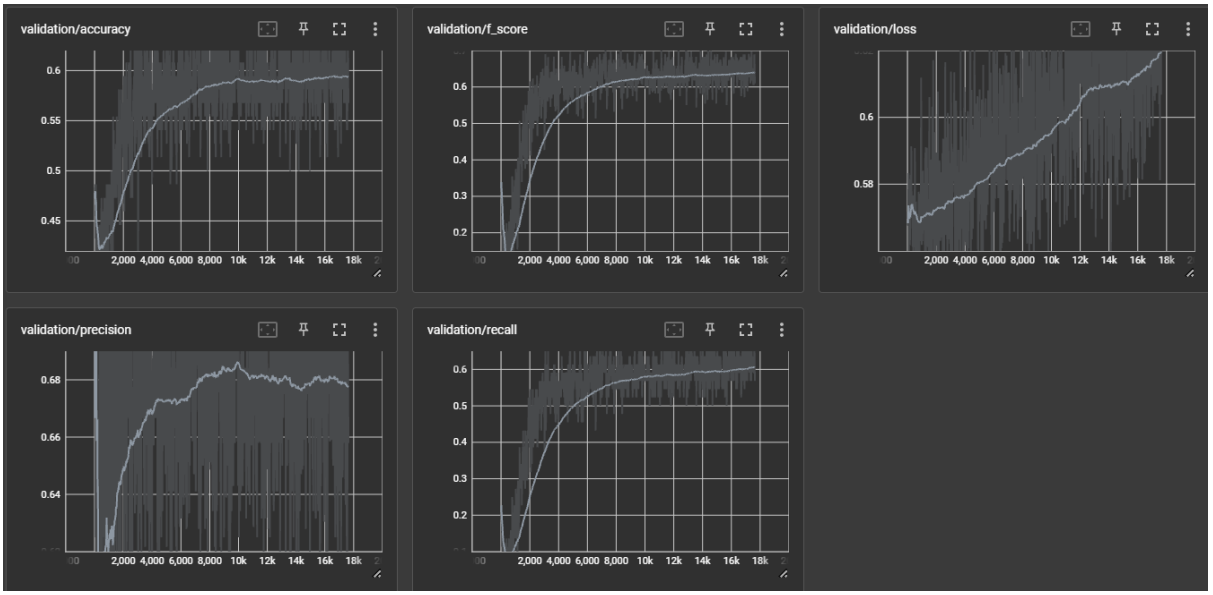
Training parameters chosen

After resolving the dataset imbalance issue, several prolonged training sessions were conducted to investigate whether the problem had been addressed successfully. These training sessions followed a similar approach to the previous fine-tuning of the network.

Considering the limited results obtained thus far, a decision was made not to perform a full training but instead focus on a single fold of the k-fold cross-validation. This allowed for an evaluation of how the network responds to variations in the fine-tuning process.

In this case, a learning rate of 0.0001 was chosen, along with weight decay and momentum set to 0. The aim was to assess the network's behavior under these specific fine-tuning parameters and observe any improvements or changes in its performance.





Despite the reduced oscillatory behavior observed in the training with an increased batch size of 16, the results are still unsatisfactory. It is worth noting that the number of epochs conducted remains high, on the order of 100 epochs with 32 frames per epoch, and the desired results have not been achieved.

Although the loss appears to decrease relative to the number of epochs, the validation set's loss diverges upward, indicating potential overfitting and strong reliance on the training set. While the issue of having a recall of 1 at the expense of poor precision has been addressed through dataset balancing, the obtained results remain highly negative.

These observations suggest that the network may be overfitting to the training data, failing to generalize well to unseen data in the validation set.

Data analysis, dataset augmentation and test on CNNs

Following the limited results obtained by the network and the limited time available, a decision was made to intervene on the dataset, in parallel with conducting experimental training using CNN architectures, particularly ResNet and GoogleNet.

Dataset augmentation

Due to the overfitting observed in the graphs, a deeper analysis of the provided dataset was conducted, focusing on removing a significant number of videos with the same scenario. It

was believed that having these videos simultaneously present in both the training and validation sets could destabilize the network. Additionally, most of these videos had a considerably long duration, allowing the network to already capture a reasonable number of frames from these scenarios.

After removing these videos, a substantial augmentation of the dataset was performed by adding videos with and without fire through manual labeling. A dataset obtained from Kaggle [1] was utilized, along with numerous videos sourced from YouTube [2], as well as others obtained from platforms like Pixabay [3] and StoryBlocks [4]. Additionally, several videos were created specifically by the project authors.

Considering that the issues encountered so far seem to be primarily caused by the quantity of similar videos in the dataset, the decision was made to reintroduce the Crop transformation. This choice was motivated by the fact that the original network was designed with the use of this transformation.

Training GoogleNet/ResNet

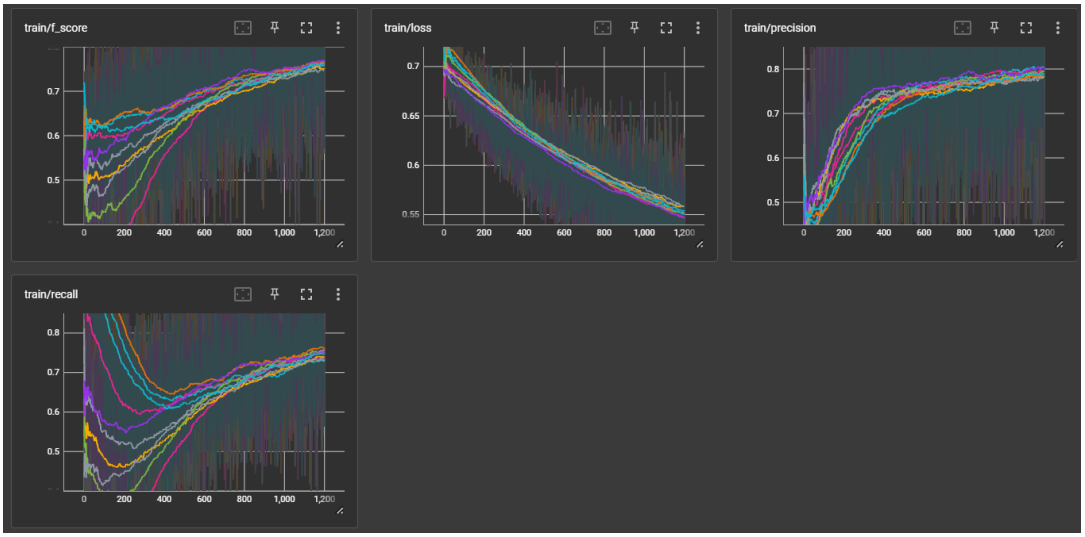
Simultaneously with the dataset augmentation, various tests were conducted on CNNs, particularly on GoogleNet:

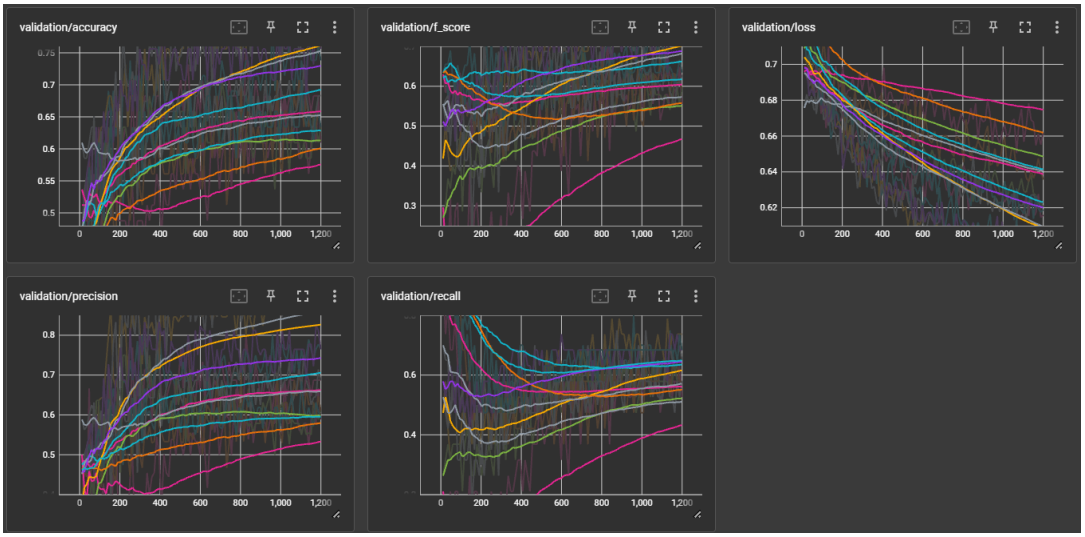
```
  (avgpool): AdaptiveAvgPool2d(output_size=(1, 1))
  (dropout): Dropout(p=0.2, inplace=False)
  (fc): Sequential(
    (0): Linear(in_features=1024, out_features=128, bias=True)
    (1): ReLU()
    (2): Linear(in_features=128, out_features=2, bias=True)
    (3): Softmax(dim=None)
  )
)
```

and ResNet (ResNet50).

The networks, including GoogleNet, were trained with larger batch sizes compared to SlowFast_R50 (ranging from 16 to 64), and different learning rates ranging from aggressive to less aggressive values (in the range [0.1:10-7]). The momentum was consistently set to 0, and the weight decay was varied within the range [0.001:10-7].

Here are the results obtained from training the GoogleNet CNN with a learning rate of 0.0005, a weight decay of 0.0001, and a momentum of 0:

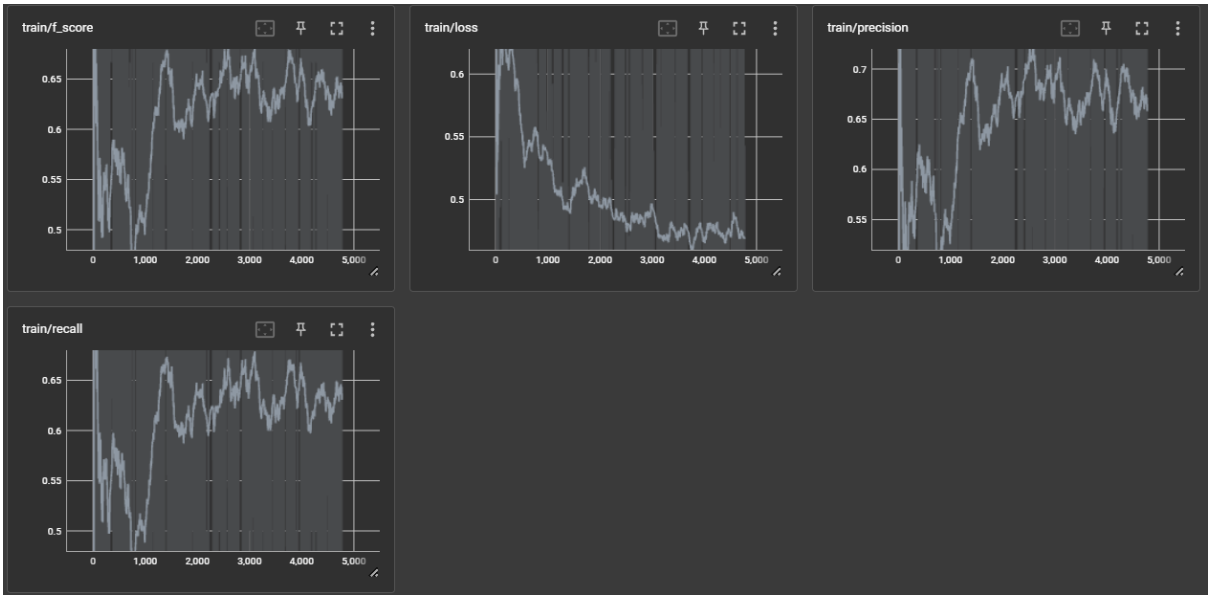


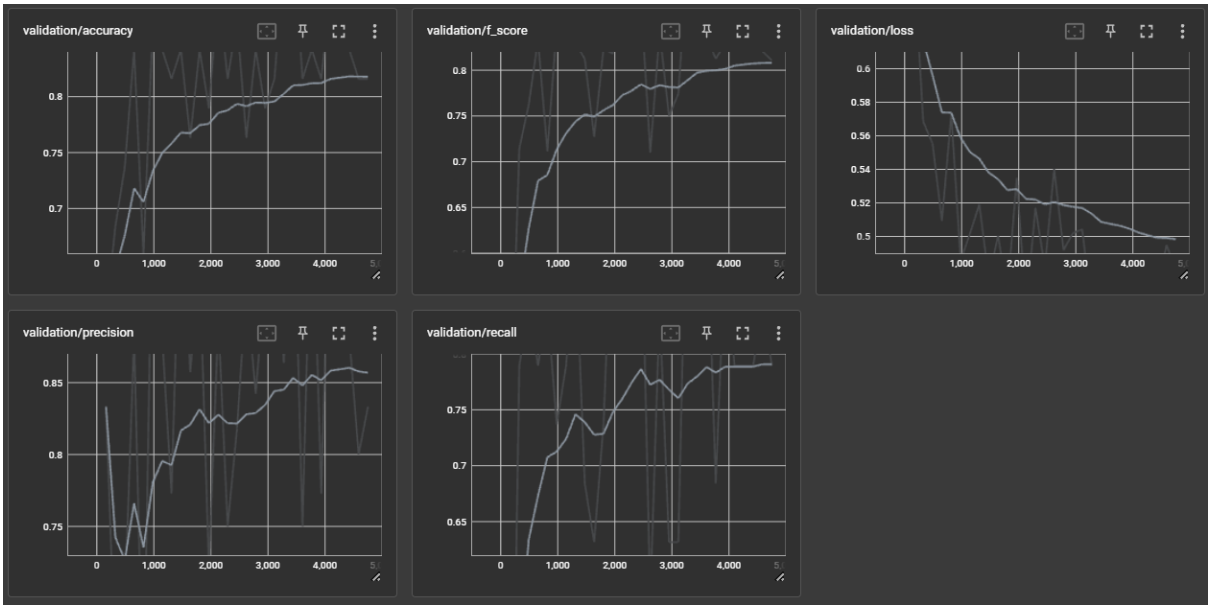


SlowFast further tests

Following the training conducted on the CNNs and their less satisfactory outcomes, a decision was made to revert to the SlowFast_R50 network architecture and examine the network's responses to changes in the dataset by conducting various tests during the dataset augmentation process.

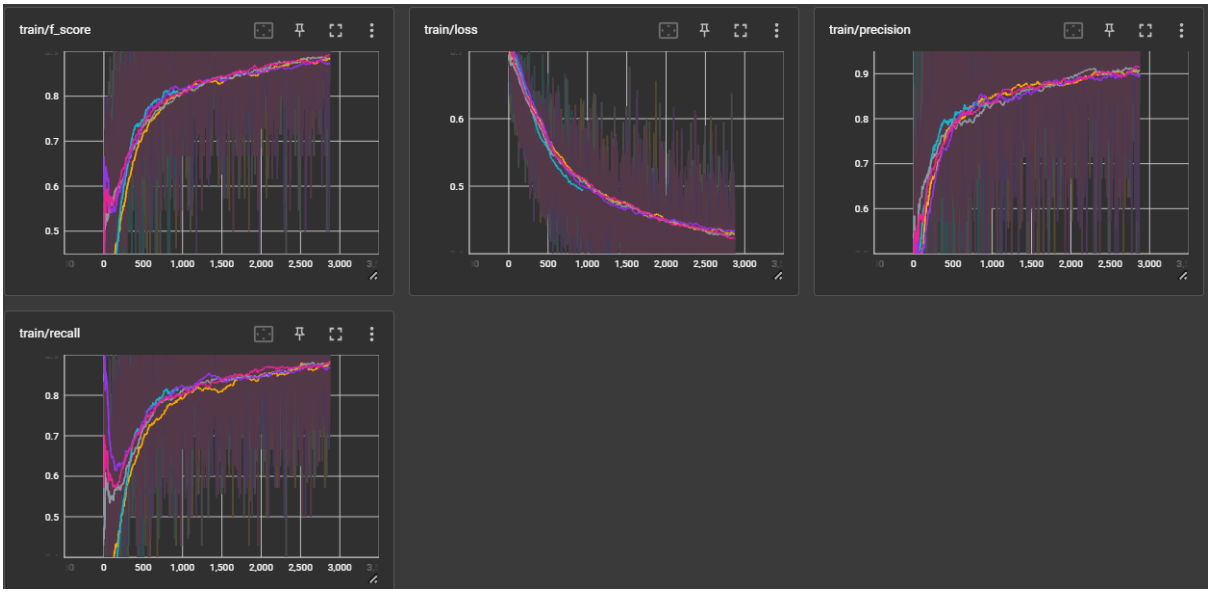
In one of the initial attempts, the network was trained during the dataset augmentation itself to observe its reaction. The chosen parameters were a learning rate of 0.001, a batch size of 2, 125 epochs, and a momentum and weight decay of 0.

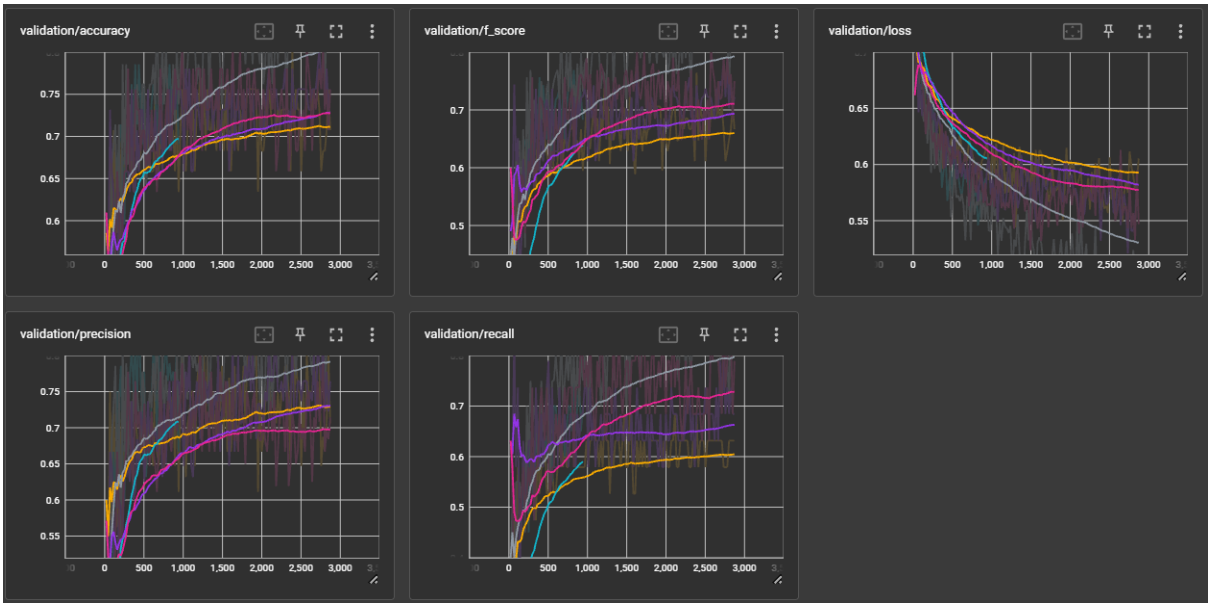




By achieving significantly better results than those obtained so far, with even better values in the validation set compared to the training set, possibly due to the presence of Dropout layers during training, where a percentage of features are set to 0. We chose to conduct one final K-fold validation with all the data available to us in the "TRAINING_SET" folder on the drive due to the limited time remaining for model training.

It was decided to unlock more training layers of the network immediately, specifically the last six convolutional layers, and perform a 5-fold cross-validation with a learning rate of 0.000015 and an early stopping criterion set to 5. The results obtained with a batch size of 16 are as follows:





The results obtained are worse than the test conducted before the complete augmentation of the dataset, but they definitely represent a more reliable outcome as we have performed cross-validation. For this reason, we have chosen to deliver one of these latter models for problem resolution. Although the results are not very satisfactory, especially in terms of the loss value, we performed inference on a test set derived from data that has not been provided to the network that given good results.

Test Phase

After training the SlowFast_R50 network, a policy had to be chosen to define how to handle videos obtained from the test set. It was decided to opt for a fairly conservative approach given the network's not-too-high precision, declaring the presence of fire only if the network

responds with fire for 3 consecutive inputs. Considering that the network takes 32 frames at a time as input, the minimum time to detect fire would be 96 frames of fire. Taking into account an average frame rate of 24 FPS in modern cameras, this choice allows the system to detect the presence of fire after 4 seconds, which is still a reasonable time frame. Additionally, this helps reduce the number of false positives that the network might identify, avoiding unnecessary alarms.

Several precautions were also added for fire detection: in the case of videos with very low frame rates (less than 32x3 frames) or videos that end with the presence of flames for a very short time, the system will still identify the flame with 1 or 2 occurrences, respectively. These choices are made to address the limitation of needing at least 96 frames to identify a flame or smoke, which may not always be possible at the end of a video or for very short videos.

An alternative approach that considered overlapping frames between two processing steps was also considered to reduce the number of frames necessary for accurate fire or smoke detection. However, after careful analysis, this solution was discarded because it would require more resources and consequently worsen the system's performance. Furthermore, sharing frames between the two processing steps could propagate analysis errors from one step to another, reducing the overall accuracy of the system.

During the testing phase, the decision to use SlowFast_R50 was further reinforced compared to CNNs like GoogleNet or ResNet50. The latter models, processing a single frame at a time, showed a drop in accuracy when dealing with adjacent frames processed incorrectly by the system. To overcome this problem, one would have to avoid using adjacent frames and space them apart, but that would increase the number of frames needed to detect fire or smoke, as well as waste many intermediaries.

Additionally, it was noticed that the GoogleNet network requires about 0.04 seconds to perform a single evaluation, while SlowFast_R50 requires approximately 0.9 seconds. However, it should be emphasized that SlowFast_R50 uses 32 frames at a time, which would require GoogleNet about 1.28 seconds to process.

Quotes

[1] APA: Aremu, T., Zhiyuan, L., Alameeri, R., Khan, M., & Saddik, A. E. (2023). SSIVD-Net: A Novel Salient Super Image Classification & Detection Technique for Weaponized Violence.

[2] Youtube Videos. https://www.youtube.com/

[3] Pixabay Videos, https://pixabay.com/it/

[4] Storyblocks Videos, https://www.storyblocks.com/