Fire Detection with CNNs and Optical Flow

Adithya Palle
Northeastern University
Khoury College of Computer Sciences
Boston, MA
palle.a@northeastern.edu

***Abstract*—Detecting fires rapidly in both indoor and outdoor environments is essential for minimizing damage and ensuring timely response. We propose two vision-based models that predict the presence of fire in each frame of a real-time video feed. The first is a lightweight convolutional neural network (CNN) inspired by AlexNet, which achieves over 99% recall with a 10% false positive rate on a small test set of videos, and generalizes well across varied scenarios. The second is an optical flow-based model which calculates the percentage of flow in flame-colored regions. This model performs effectively on large fires but struggles to generalize to smaller or more complex scenes. The CNN was trained on a custom dataset of approximately 20,000 images and supports real-time inference at up to 124 frames per second on a CPU, making it suitable for deployment on everyday devices. While the CNN shows strong potential as a general-purpose fire detector for both indoor and outdoor use, further refinement is needed to reduce false positives. The optical flow approach, although limited as a standalone solution, may serve as a useful complementary input when integrated with image data to improve overall classifier robustness.**

I. Introduction

Fires are one of the most destructive hazards, responsible for significant property loss, injury, and death across the globe. According to the International Association of Fire and Rescue Services (CTIF), there were 23,535 structure fire incidents reported in eighteen selected cities around the world in 2017 alone, with 6,581 fire-related casualties recorded across forty-four cities over a four-year period (2013–2017) [1]. The consequences of undetected or delayed fire response are severe, especially in high-risk indoor environments or remote outdoor areas.

Traditionally, fire detection systems have relied on physical sensors such as smoke detectors or gas-based fire alarms. However, these systems are often limited in scope and performance. For instance, particle-based sensors perform poorly outdoors due to diffusion and air flow, making them unreliable for open spaces. The rapid advancement in camera technology, coupled with breakthroughs in computer vision and machine learning, has enabled a new generation of visual fire detection systems. Cameras offer the ability to capture fire and smoke signatures in real time, while computer vision models allow for the fast and automated interpretation of visual data. These developments open the door to robust fire detection systems that are accurate, fast, and scalable.

This paper proposes a real-time fire detection system that processes a live video feed to determine whether fire is present in each frame. The proposed system is designed to operate in diverse environments and provide immediate alerts. To achieve this, we compare two different approaches: (1) a classical computer vision method based on optical flow analysis, and (2) a deep learning-based image classification model.

II. Related Work

Early approaches to fire detection relied on physical sensors, such as smoke and gas detectors. These methods are widely used in indoor environments due to their low cost and reliability in enclosed spaces. However, their effectiveness diminishes in outdoor environments where environmental factors such as wind can prevent smoke particles from reaching the sensors. Furthermore, there is often a significant delay from when the fire starts till its detection due the time taken for particles to reach the sensor. That time, even if only a few seconds, is crucial in dangerous scenarios where victims may need to respond quickly. As a result, researchers have explored vision-based alternatives that leverage camera data to detect fire and smoke.

Vision-based fire detection methods can be broadly classified into three categories: patch-level, blob-level, and pixel-level approaches [1]. Patch-level methods divide the image into smaller regions and analyze features such as color and texture within each patch to determine the presence of fire or smoke. Blob-level methods focus on identifying connected regions of pixels that exhibit fire-like characteristics and then analyzing the shape, size, or motion of these blobs. Pixel-level methods make fine-grained decisions at the individual pixel level, often using thresholds on color or motion features to classify whether a pixel belongs to a flame or smoke region. In contrast, our approach focuses on higher-level binary classification at the frame level, whereas the methods described below operate primarily at the patch and blob levels.

An early vision-based technique was proposed by Liu and Ahuja (2004), who used shape and temporal features of flame regions to classify fire using a Support Vector Machine with a Radial Basis Function kernel [2]. Similarly, Chen et al. (2004) combined color-based segmentation with dynamic features such as disorder and region growth to detect early-stage fires in video streams [3]. These classical computer vision methods take advantage of visual cues like flame color (orange-red) and motion dynamics to classify regions as fire or smoke. Although these methods are computationally lightweight and do not require large datasets, they often struggle with generalization

across different environments. To overcome this, recent work has turned to deep learning.

Deep learning-based fire detection systems eliminate the need for handcrafted features by learning representations directly from data. A notable example is the work by Muhammad et al. (2018), who used a modified GoogleNet architecture with transfer learning for fire detection in surveillance videos [4]. Their model's weights are initialized from a pretrained GoogleNet and only the final classification layer is retrained on the dataset. Another notable paper by Frizzi et al. introduces a CNN architecture loosely inspired by AlexNet, trained on a dataset of over 28,000 images spanning fire, smoke, and negative samples [5]. To enable coarse localization, they apply a sliding window over the final feature map, passing 12×12 regions through the network to detect the presence of fire in specific areas of the frame.

A hybrid approach based on motion rather than appearance was proposed by Mueller et al. (2013), who introduced optical flow techniques specifically tailored for fire detection [6]. Mueller et al. apply optical flow to extract motion features, which are then fed into a neural network classifier. Since fire and smoke exhibit unique motion patterns, optical flow is a natural tool for capturing their behavior over time. Their paper proposed two methods: Optimal Mass Transport (OMT) and Non-Smooth Data (NSD). OMT models fire as a dynamic texture and tracks the "mass" of intensity across frames, while NSD focuses on edge motion in saturated fire regions where texture is weak.

Among classical optical flow algorithms, Farnebäck optical flow is particularly relevant due to its speed and robustness [7]. It approximates each small neighborhood of the image with a polynomial function, and estimates how this surface shifts between frames to derive motion vectors at each pixel. Unlike traditional methods like Lucas-Kanade [8], which operate on raw intensity values, Farnebäck's method leverages a smoother quadratic approximation, making it more robust to noise and better suited for chaotic motion like fire.

Our proposed system differs from these prior approaches by implementing and comparing both classical and deep learning models in a unified real-time pipeline. We adopt Farnebäck optical flow as a lightweight motion-based method and contrast it with a CNN trained on fire and non-fire images. This dual approach allows us to evaluate tradeoffs in speed, accuracy, and generalization, with the goal of building a robust fire detection system suitable for both indoor and outdoor environments. Instead of detailed segmentation or region-based analysis, our system operates at a coarser level by assigning a binary label to each frame, indicating the presence or absence of fire. This design choice reflects practical disaster response needs, where a rapid and reliable alert often matters more than precise localization.

III. Methods

We implement and evaluate two complementary approaches for real-time fire detection: (1) a deep learning-based image classifier using a lightweight convolutional neural network (CNN), and (2) a classical computer vision method based on Farnebäck optical flow. Both approaches process video frames and output a probability that fire is present, which is thresholded to yield a binary decision.

A. CNN-Based Frame Classifier

1) Dataset and Preprocessing: We constructed a balanced dataset of 17,878 labeled images, consisting of 8,939 fire images from a Roboflow dataset [9] and 8,939 non-fire images sourced from the Places365 dataset [10]. The non-fire images include a variety of scenes (e.g., urban, indoor, nature), selected to challenge the model's ability to distinguish fire from visually similar distractors such as sunsets or lighting artifacts. The dataset was split into 80% training (14,302 images), 10% validation (1,787 images), and 10% test (1,789 images). All images were resized to 128 × 128 pixels and normalized using ImageNet means and standard deviations [11]. The validation set was used to tune hyperparameters and apply early stopping, while the test set was held out entirely for final performance evaluation.

2) Architecture: Our convolutional neural network is a lightweight feedforward model inspired by AlexNet [12], specifically designed for binary classification of fire presence in 128×128 RGB images. The architecture consists of two main components: a convolutional feature extractor and a fully connected classification head.

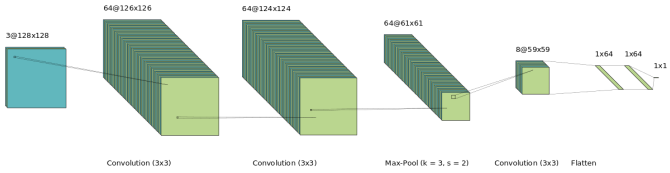

Fig. 1: Architecture of Convolutional Neural Network used for Fire Detection

The feature extractor begins with a two-dimensional convolutional layer with 64 filters, each of size 3×3, applied with a stride of 1 and no padding. This is followed by a LeakyReLU activation function and a dropout layer with a dropout probability of 0.5. The second convolutional layer also uses 64 filters of size 3×3, again with stride 1 and no padding. Following this, a 3×3 max pooling operation is applied with a stride of 2 to reduce spatial dimensions. This is again followed by a LeakyReLU activation and a dropout layer with the same dropout probability. The third convolutional layer reduces the number of feature maps to 8, using the same kernel size and stride as before. A LeakyReLU activation and dropout complete the convolutional block.

The output from the final convolutional layer is flattened into a one-dimensional vector. This vector is then passed to the fully connected classifier. The first fully connected layer maps the flattened feature vector to 64 hidden units, followed by a LeakyReLU activation and dropout. The second fully

connected layer also contains 64 units and is followed by another LeakyReLU and dropout. The final output layer consists of a single neuron that produces a scalar logit, representing the unnormalized probability of fire presence.

We do not apply a sigmoid activation to the output during training. Instead, we use the BCEWithLogitsLoss function, which combines the sigmoid activation and binary cross-entropy loss into a single numerically stable operation. During inference, a sigmoid function is applied to the final logit to convert it into a probability between 0 and 1. The total number of trainable parameters in the network is approximately 1,829,897, making it significantly more compact than standard architectures while retaining sufficient expressivity for fire classification.

The model was constructed and trained using PyTorch [13].

3) Training Procedure: We trained the model for 15 epochs using the Adam optimizer [14], with a batch size of 128 and an initial learning rate of 0.001. A learning rate scheduler reduced the rate by a factor of 0.5 upon plateauing validation loss. Early stopping was triggered if validation loss did not improve by at least 0.0001.

The loss function used was BCEWithLogitsLoss() [13], a numerically stable formulation that combines sigmoid activation with binary cross-entropy loss. This design avoids gradient instability caused by saturating sigmoid outputs during training.

Training was conducted on an NVIDIA V100 SXM2 GPU (32 GB), and inference benchmarks were collected on both this GPU and a CPU (Intel Xeon Gold 6132 @ 2.60 GHz, 8 cores, 50 GB RAM).

4) Limitations: The Roboflow dataset contains some mislabeled images, including scenes with bright orange or red lighting (e.g., digital signage, sunsets) and digitally generated images labeled as fires. These outliers may cause the model to learn spurious correlations between fire and certain visual features common to these scenes, such as color intensity or hue. Given the relatively small size of the dataset, such mislabeled samples could impact the model's generalization. However, as these cases represent a small fraction of the data, their overall influence is likely limited. Future work should focus on curating a more robust and accurately labeled dataset to further improve model performance and reliability.

B. Optical Flow-Based Fire Detection

Our second approach uses Farnebäck optical flow [7], a technique that computes per-pixel motion vectors between consecutive frames. It models the neighborhood of each pixel with a polynomial approximation and infers motion by comparing these approximations across frames. After computing optical flow between two frames, we apply a fire-color mask using RGB thresholds. We define fire-colored pixels using lower bounds of [150, 50, 0] and upper bounds of [255, 255, 150]. These thresholds are chosen to match the hue and brightness characteristics of flame regions. We then calculate the total magnitude of motion vectors (optical flow) inside and outside the fire-colored region. The fire probability for each frame is defined as:

$$P_{\text{fire}} = \frac{\sum \|\vec{v}\|_{\text{fire-colored}}}{\sum \|\vec{v}\|_{\text{all pixels}}} \tag{1}$$

This quantity reflects the proportion of motion in flame-colored regions, with values close to 1 indicating strong, large flame-like movement, and values near 0 suggesting either no motion or motion unrelated to fire. This method was implemented using OpenCV and runs entirely on the CPU. Since it relies on motion across frames, at least two consecutive frames are required for inference.

C. Experiments

To evaluate both models, we compiled a test set of videos of fires and non-fires from the FIRESENSE dataset [15] and ran the models on each frame of each video to evaluate its classification accuracy. The test video set is constructed so that each video is labeled as entirely fire or non-fire due to the lack of per-frame annotations. This design simplifies evaluation but limits granularity.

1) Threshold Selection: Both methods produce continuous probabilities, which are binarized using a decision threshold to create a fire or no-fire prediction. To select this threshold, we set aside a small portion (33%) of our video data as a validation set. For each model, we plot a recall vs. false positive rate (FPR) curve on this set and select a threshold that balances high recall with low FPR. The final test set results are then computed using this fixed threshold.

This ensures the test set remains a fair benchmark for generalization, as no data from it was used in threshold tuning. The validation video set consists of 5 non-fire videos and 4 fire videos, while the test video set consists of 11 non-fire videos and 7 fire-videos.

2) Frame-Level Classification: For image-based CNN evaluation, we use the held-out test images from the original training split. This allows comparison with prior deep learning-based methods that operate on single images.

3) Inference Speed: We measure the average inference time per frame for both models on the video test set, including all necessary pre- and post-processing. These benchmarks help assess the practicality of each method for real-time fire detection in embedded or constrained environments.

IV. Results

We evaluated the CNN model and the optical flow model on the same testing video sets and found that the CNN model performed the best overall, with high recall, accuracy, and inference speed.

While the optical flow model appears to have a lower false positive rate on the test set, we found that it is unlikely to generalize well in practice. When both models were evaluated on three videos from Youtube [16], the CNN continued to perform as expected, but the optical flow model exhibited significantly more false positives and false negatives. The link

TABLE I: Comparison of CNN and Optical Flow Fire Detection Models. Thresholds for binary classification are chosen from Recall-FPR curves of validation sets seen in Figure 3 and Figure 7

Model	Accuracy	Recall	FPR	Avg Time (ms)	FPS	t
CNN	0.9354	0.9997	0.1054	8.043	124.3	0.9
Optical Flow	0.9781	0.9478	0.0027	61.43	16.2	0.15

to the trials on these videos can be found in the references at [16].

We attribute this discrepancy to the nature of the test and validation video sets, which mostly feature large fires that occupy a substantial portion of the frame. In contrast, the additional evaluation videos contain smaller, more localized fires. Since the optical flow method depends on the proportion of motion in fire-colored regions, the same threshold determined from the validation set may be ineffective for frames with small or subtle fires. This highlights a core limitation of the method: it lacks generalizability across different fire scales, making it difficult to find a single decision threshold that works reliably for both large and small fire events.

Now, we take a deeper look at the results for each model separately.

A. CNN Results

1) Training Loss:

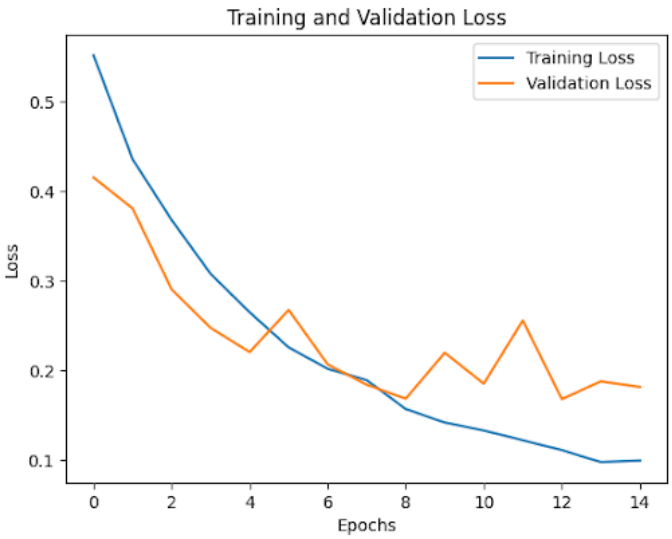

Fig. 2: Loss Curve during training of CNN

The loss curves for the CNN model reveal that it has low bias and low variance, as evidenced by the similarity between training and validation loss. It is worth noting that the validation loss begins lower than the training loss because it is recorded after each epoch, once the weights have already been updated, while the training loss reflects the average batch loss across the entire epoch — including the early batches before weights have stabilized. Nevertheless, the curves clearly illustrate the model's generalization capability, with only a few epochs required for convergence.

2) Threshold Selection:

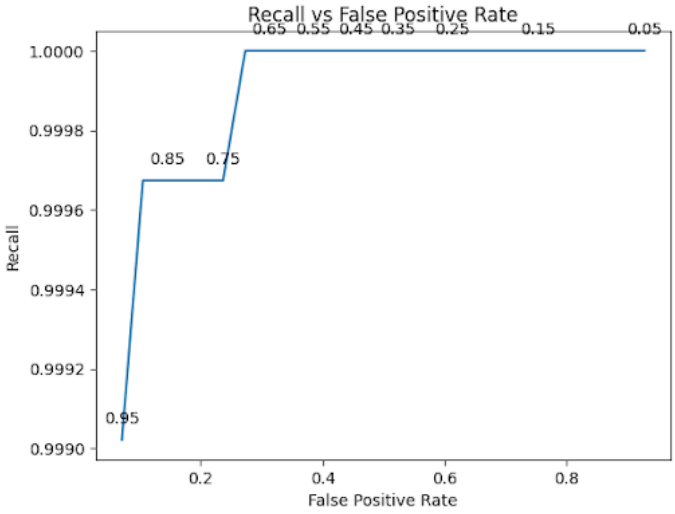

Fig. 3: Recall-FPR Curve for CNN on validation set of videos

The recall vs. FPR plot above illustrates the trade-off between recall and false positives for different decision thresholds on the validation video set. This curve was used to select the threshold of 0.9, which achieves high recall with an acceptable false positive rate and serves as a reasonable operating point for real-world scenarios.

Furthermore, we also evaluated the CNN model on a separate image-based test set, leveraging the fact that it can process individual frames independently. This evaluation provides additional insight into model behavior on still images, as opposed to video.

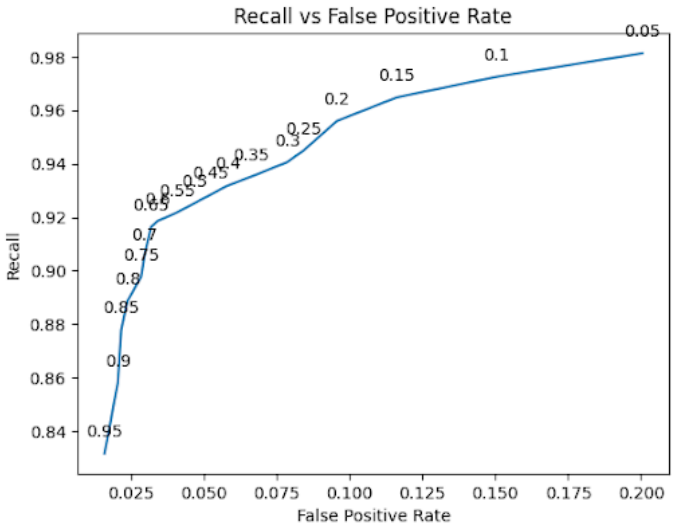

Fig. 4: Recall-FPR Curve for CNN on validation set of videos

The recall curve for the image dataset suggests that a threshold of 0.9 still offers a favorable balance of recall

and precision. The false positive rate on the image dataset ($< 2.5\%$) is noticeably lower than that of the video dataset (10.5%), likely due to the higher quality and more curated nature of the still images, whereas video data—often captured from real-world camera feeds—tends to be lower in quality, introducing more noise.

3) Comparison with Related Models: Using the threshold of 0.9 determined from Recall-FPR curve on the video validation set, we obtained the following performance on the image test set:

TABLE II: Table Comparing our CNN's performance with those from table 2 in Gaur et. al's literature review on Fire Detection algorithms [1]

Model (Reference)	Accuracy (%)	FPR (%)
Our CNN	90.44	2.28
CNN (GoogleNet)	94.43	0.054
CNN (Alexnet variant)	98.10	0.21
CNN (AlexNet)	94.39	9.07

Compared to other models, our architecture has slightly lower accuracy and a moderately higher false positive rate. However, this is expected given our lightweight design, limited dataset, and our goal of enabling real-time deployment on resource-constrained devices. Future work may explore increasing model capacity or increasing input resolution to improve performance while maintaining efficiency.

4) GPU speed: We also evaluated inference speed on a GPU to assess the model's suitability for real-time processing on modern hardware. Using the same GPU from training, we observed an average inference time of 1.178 ms per frame, corresponding to a maximum throughput of approximately 850 frames per second. This suggests the model can be deployed alongside other workloads on GPU-equipped systems without significantly affecting performance. On the CPU, the model achieves around 124 fps, which is more than sufficient for real-time video analysis in practical settings.

5) Activation Analysis:

Fig. 5: Original Image used in activation analysis

Fig. 6: Feature Map after applying 8 randomly selected filters from the first layer of the CNN to the image in Figure 5

We visualized the filters learned in the first convolutional layer by applying them to a random fire image. The sampled filters reveal diverse feature sensitivities — some emphasize background or non-fire regions, one acts as a near-skip connection allowing fire to pass through, and others focus sharply on fire edges and textures. This diversity indicates that the model is learning a broad set of visual patterns, enhancing its ability to generalize across varied fire appearances.

B. Optical Flow Results

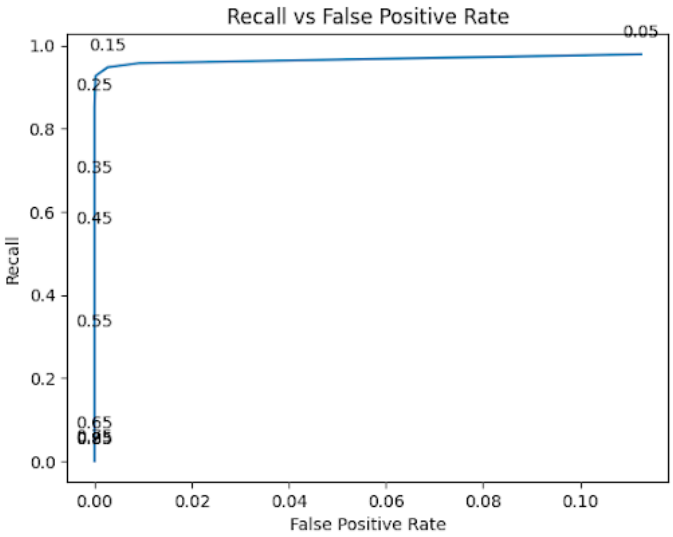

Fig. 7: Recall-FPR Curve for Optical Flow model on validation set of videos

The recall vs. FPR curve for the optical flow model differs notably from that of the CNN. It achieves a low false positive rate on the test video set, particularly at a threshold of 0.15, but exhibits a sharp decline in recall as the threshold increases. This behavior implies that the model only captures large fires well and will struggle to detect small-scale fires that occupy less than 15% of the frame without capturing many more false positives — a limitation confirmed by its poor performance on the external evaluation videos.

Despite its limitations, the optical flow method performs very well on large fire events and could potentially serve as a complementary signal in future systems. Specifically, it could be used to generate motion-based feature maps (of shape 2×128×128) that are fused with raw images in a CNN classifier. This could enable the network to leverage both motion and appearance cues, enhancing robustness.

V. Conclusion

We developed two frame-level models for real-time fire detection in live video streams. Both models are capable of reliably identifying fires shortly after ignition due to their high recall, making them strong candidates for early warning systems. Among the two, the CNN-based model shows the most promise due to its high inference speed and strong generalizability across diverse fire patterns. It achieves a recall of over 0.99 on test videos, indicating it can detect nearly all fire events. However, it suffers from a relatively high false positive rate. Despite this, the model remains practical in scenarios where false alarms are tolerable and early detection is critical. Given its impressive speed, future iterations could explore increasing the model size to reduce false positives while maintaining real-time performance. In contrast, the optical flow-based model struggles with generalizability, especially when detecting small fires or performing on demo videos [16]. Nevertheless, it excels at detecting large fires with a very low false positive rate. This suggests potential as a complementary feature input to reduce false alarms in future systems. Future work could explore temporal modeling approaches, such as recurrent CNNs [17] or 3D CNNs that incorporate sequences of frames, to better capture the temporal dynamics of fire emergence. Additionally, combining optical flow feature maps with raw image data may further enhance model robustness and accuracy.

VI. Acknowledgment

I would like to thank Bruce Maxwell for his instruction of CS 5330 at Northeastern University which birthed this project. I'd also like to thank Northeastern University's Discovery Cluster team for letting me use their computing these resources for model training.

References

[1] A. Gaur, A. Singh, A. Kumar, A. Kumar, and K. Kapoor, "Video flame and smoke based fire detection algorithms: A literature review," *Fire Technology*, vol. 56, no. 5, p. 1943–1980, Apr. 2020. [Online]. Available: http://dx.doi.org/10.1007/s10694-020-00986-y

[2] C. Liu and N. Ahuja, "Vision based fire detection," *Proceedings - International Conference on Pattern Recognition*, vol. 4, pp. 134–137, 2004, proceedings of the 17th International Conference on Pattern Recognition, ICPR 2004 ; Conference date: 23-08-2004 Through 26-08-2004.

[3] T.-H. Chen, P.-H. Wu, and Y.-C. Chiou, "An early fire-detection method based on image processing," in *2004 International Conference on Image Processing, 2004. ICIP '04.*, vol. 3, 2004, pp. 1707–1710 Vol. 3.

[4] K. Muhammad, J. Ahmad, I. Mehmood, S. Rho, and S. W. Baik, "Convolutional neural networks based fire detection in surveillance videos," *IEEE Access*, vol. 6, p. 18174–18183, 2018. [Online]. Available: http://dx.doi.org/10.1109/ACCESS.2018.2812835

[5] S. Frizzi, R. Kaabi, M. Bouchouicha, J.-M. Ginoux, E. Moreau, and F. Fnaiech, "Convolutional neural network for video fire and smoke detection," in *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society*, 2016, pp. 877–882.

[6] M. Mueller, P. Karasev, I. Kolesov, and A. Tannenbaum, "Optical flow estimation for flame detection in videos," *IEEE Transactions on Image Processing*, vol. 22, no. 7, p. 2786–2797, Jul. 2013. [Online]. Available: http://dx.doi.org/10.1109/TIP.2013.2258353

[7] G. Farnebäck, *Two-Frame Motion Estimation Based on Polynomial Expansion*. Springer Berlin Heidelberg, 2003, p. 363–370. [Online]. Available: http://dx.doi.org/10.1007/3-540-45103-X_50

[8] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI)*, 1981, pp. 674–679.

[9] F. Dataset, "Fire detection dataset," https://universe.roboflow.com/fire-dataset-tp9jt/fire-detection-sejra, aug 2022, accessed: 2025-04-06. [Online]. Available: https://universe.roboflow.com/fire-dataset-tp9jt/fire-detection-sejra

[10] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba, "Places: A 10 million image database for scene recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[11] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105. [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[13] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," 2019. [Online]. Available: https://arxiv.org/abs/1912.01703

[14] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014. [Online]. Available: https://arxiv.org/abs/1412.6980

[15] C. Filo, "Firesense dataset," https://www.kaggle.com/datasets/chrisfilo/firesense, 2022, accessed: 2025-04-06.

[16] YouTube, "Custom youtube dataset for fire detection," https://drive.google.com/drive/folders/1GLSxCgdm0lU-UBnkRTfptQrsf-lBjziE, 2025, accessed: 2025-04-06.

[17] M. Liang and X. Hu, "Recurrent convolutional neural network for object recognition," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 3367–3375.