Real-time Fire Detection and Localization within Video Imagery using Contemporary Advances in Deep Neural Networks

Student Name: Adam Read

Supervisor Name: Toby Breckon

Submitted as part of the degree of MSci in Computer Science and Mathematics to the

Board of Examiners in the Department of Computer Sciences, Durham University

Abstract —

Context/Background: The ineffectiveness of traditional smoke and heat based fire detection systems in well-ventilated, outdoor areas combined with the increasing omnipresence of security cameras highlights the possibility of a widespread computer-vision-based fire detection system. Furthermore, research into the field could introduce the use of autonomous fire fighting systems capable of saving time, money and lives.

Aims: The aim of this project is to explore the use of optical flow algorithms, specifically representational flow, with deep residual convolutional neural networks as a visual fire detection system. We will undertake an evaluation of the current state of the art in the field and produce a model able to detect the presence of fire within a video accurately.

Method: A fire video dataset is compiled and pre-processed using datasets produced by prominent papers in the field to train a deep network capable of binary fire classification containing at least one representational flow layer. To support this, we introduce several sampling strategies and examine the effect of varying network depth on performance.

Results: The most accurate computer-vision-based fire detection system produced correctly classified 78.06% of videos on an independent evaluation dataset.

Conclusions: Our results does not improve upon the state of the art, which achieves an accuracy of 97.92%. However, it does support the continued research into temporal fire detection systems with robust and rigorous investigation into the research question.

Keywords — fire detection, deep residual convolutional neural networks, computer vision, deep learning, machine learning, object recognition, optical flow, representational flow.

I INTRODUCTION

Worldwide, fire causes a substantial amount of destruction. In Califonia, wildfires resulted in an estimated $140+ billion of damage through 2018 alone (Wang et al. 2021) - a value predicted to grow in years to come. While it is not feasible to prevent fire from ever occurring, we can minimise the negative impact by producing systems that accurately identify its presence and provide alerts to take appropriate action.

Most fire detection systems, based either on smoke or heat, rely on specific environmental conditions to detect fire. Smoke detector systems fall into two categories: ionization alarms and photoelectric alarms. Ionization smoke alarms consist of a chamber containing oppositely charged electric plates and a small amount of radioactive material that ionizes the air causing current to flow. In comparison, photoelectric alarms contain a light-sensitive plate and a light source (not aimed at the plate). For both types, the presence of smoke causes the alarm to trigger; in ionization alarms, smoke reduces the flow of current, whereas, in photoelectric alarms, smoke particles will reflect light onto the sensing plate. Similarly, heat alarms depend on the heat produced by the fire to cause a temperature-sensitive device within the alarm to activate. After ignition, there are several hindrances to the activation of these detectors. Not only must the fire become large enough to generate conditions sufficient to activate the alarm, but the smoke and heat produced must reach the detector. These factors highlight numerous flaws. Most notably, if well-ventilated or outside, the area over which these alarms can effectively detect fire is significantly reduced. Additionally, there can be considerable delays between the ignition and detection of a fire.

Rather than smoke or heat, we will focus on a computer-vision-based fire alarm; that is, a system able to detect fire using an image or a video. Assuming the system is reliable and accurate, it would mitigate several critical deficiencies found in smoke and fire detectors. If positioned appropriately, a computer-vision-based detector could detect fire regardless of its surrounding environment. Contrary to conventional detectors, this would allow the system to operate effectively in large open, outdoor areas. Moreover, we would remove much of the lag between the ignition and detection of a fire. A computer-vision-based approach would require only the presence of a fire-like object - a condition fulfilled on ignition. Furthermore, we could obtain a much more significant amount of information than a simple binary classification of the presence of heat or smoke. With computer vision, it would be possible to obtain information about the type of fire and how the fire is evolving. This information could help inform the correct allocation of resources to extinguish the fire, minimising the monetary cost and damage caused.

With the ever-increasing popularity and omnipresence of CCTV, distributing a computer-vision-based fire detection system would require minimal cost and infrastructural change. For areas unmonitored by CCTV, or any other recording device, it is straightforward and inexpensive to place a camera able to monitor a large area. A computer-vision-based fire detection system could also integrate with autonomous vehicles, such as drones fitted with cameras, in use as a guidance system for machine-driven fire fighting able to provide faster support and reduce the risk for human personnel.

This paper aims to further research into the production of a computer-vision-based fire detection system to create an architecture viable for widespread implementation. We will evaluate the use of contemporary advances in deep neural networks and optical flow models, algorithms used to estimate the motion of objects between consecutive frames caused by the relative movement between the object and camera, to produce such a system.

A Background

Abstracting, we can consider that fire detection within a video or photo is just a simple object recognition task with two main approaches, temporal recognition or non-temporal recognition. Temporal information refers to the comparison of frames within a video to understand how the video is changing over time. Within fire detection, a conventional temporal approach would attempt to identify fire within a video. In contrast, non-temporal information considers only a particular slice of time, resulting in a system that would classify a video by considering each frame separately.

When producing a computer-vision-based fire-detection system, either temporal or non-temporal, we must be able to identify and select features within each frame of video. Due to their high performance, many object recognition architectures use deep neural networks (DNNs), specifically deep convolutional neural networks (DCNNs), for this purpose. An extension of artificial neural networks, DNNs are neural networks with three or more layers (Hinton et al. 2006). DCNNs are deep neural networks that contain convolutional layers. When applied to an image, convolutional layers can be understood intuitively as filters sliding over the image, applying some operation to produce an output map. DCNNs perform end-to-end classification, so there is no need to manually generate filters to select candidate regions - allowing for a wide range of applications. For these reasons, they are often a critical component of vision fire detection systems.

As will be discussed further in section III, this project focuses on temporal fire detection systems using the movement of objects in a video as a factor in the identification of fire. For this, we can implement optical flow, a technique which uses the relative change between consecutive frames to model the movement of objects in a video relative to the camera. This method, while computationally expensive, is highly accurate. In this project, we will utilise both these techniques as inputs to a neural network to explore multiple architectures capable of classifying whether a video contains fire.

B Project Objectives and Achievement

The research question asked is: *Can representational flow be successfully applied to the task of detecting fire within a video in order to advance the current state of the art?* When considering how to answer this question to produce a temporal computer-vision-based fire detection system, the following objectives were determined:

The minimum objectives of this project were to adapt and apply the representational flow model proposed by Piergiovanni and Ryoo (2019) for binary fire detection within a video. We aimed to train and evaluate the model with an established fire detection video dataset used within previous, similar work to achieve an accuracy of at least 65%. The purposes of the minimum objectives were to generate a foundational model from which we can adapt and develop a strong understanding of prior work.

The intermediate objectives were to design, implement and evaluate different network architectures, adapting the original model. We aimed to investigate the effect of training these models over several established datasets from reliable sources using varying sampling strategies. With these objectives, we aimed to extend the work beyond a basic adaptation of representational flow research to produce a novel architecture and advance research in the field.

The advanced objectives were to extend previous architectures by creating flow-of-flow models capable of learning complex patterns within temporal information. We also aimed to provide an in-depth comparison of all approaches explored to assess performance and usability. Finally, one of the models should achieve an accuracy that, at least, matches the state of the art computer-vision-based fire detection system - 97.92%. With this, we intended to improve the state of the art in temporal computer-vision-based fire detection and explore advanced techniques within optical flow to investigate its application in the field.

We successfully achieved all basic and intermediate objectives, adapting and testing several variants of the representational flow model proposed in (Piergiovanni & Ryoo 2019) using videos compiled from five datasets produced by prominent computer-vision-based fire detection papers and numerous sampling strategies. Furthermore, we produced a flow-of-flow model capable of binary fire classification and a detailed comparison of all architecures considered. Our best model was able to detect fire with a 78.06% accuracy; we were unable to produce a model capable of outperforming the current state of the art, which acheved an accuracy of 97.92%.

II RELATED WORK

As discussed previously, the necessity for early detection systems combined with the existing infrastructure for CCTV and surveillance systems highlights the possibility of fire detection systems powered by computer vision. Furthermore, the implementation of fire detection in autonomous vehicles (Bradshaw 1991), (Martínez-de Dios et al. 2006), (Yuan 2010) shows demand for accurate spatial fire information, with the latter extending this by taking advantage of data modeling the internal movement of the fire. Consequently, escalating need for research into a vision-based fire detection system is clear.

A Colour-Based Approaches

Initially introduced in (Healey et al. 1993), most early computer-vision-based fire detection systems relied on colour analysis. Healey et al. explored a non-temporal colour-threshold-based approach by partitioning images such that segments reflect equal areas when projected into the scene. Within each segment, we determine the presence of fire using colour-based characteristics derived from physical models. This idea was extended in (Phillips Iii et al. 2002) which introduced temporal information by observing the cardinal change in fire pixels as an indicator for basic flame motion. In (Chen et al. 2004), we see this expanded through the production of a formal set of chromatic colour rules for pixels in an image whilst continuing to consider basic temporal information. Researchers hypothesised that a pixel with relative RGB intensities R, G, and B could be identified as a fire pixel if the following conditions hold:

$$\begin{aligned} &\text{Condition 1: } R > R_T, \\ &\text{Condition 2: } R \geq G > B, \\ &\text{Condition 3: } ((255 - R) \cdot S_T / R_T, \end{aligned} \tag{1}$$

where R_T is some pre-determined threshold and S_T is the saturation when the value of the R channel is R_T. Alternative colour spaces were also considered with comparative chromatic rules researched for the YC_bC_r space (Celik & Demirel 2009) and the LAB space (Khalil et al. 2020).

More complex colour-based fire detection systems combine multiple approaches. In (Toreyin et al. 2005), fire detection is framed as a hidden Markov problem centred about the movement of fire pixels located using (1). In (Liu & Ahuja 2004), fire regions are extracted by creating interior colour probability density functions modelled as a mixture of Gaussian distributions in HSV space. The changing shape of these regions within the Fourier domain is used for classification.

Colour-based fire detection approaches have several advantages and can produce impressive results - the state of the art colour-based detector achieved an accuracy of 85.08% (Khalil et al. 2020). Moreover, as colour-based approaches identify fire within a frame on a pixel-by-pixel basis, we can localise the position of fire with relative ease and accuracy. This benefit highlights the suitability of colour-based methods for fire localisation with a dynamic camera, such as cameras mounted onto autonomous vehicles, which could be used to detect and extinguish fire remotely. These systems, however, have several fundamental limitations. Most importantly, it is not feasible to produce a set of rules which are suitable for every situation. Due to the considerable impact environmental conditions can have on the colour characteristics of a scene, many approaches have high false-positive rates. For example, in (Khalil et al. 2020), 14.38% of frames are falsely identified as containing fire.

B Machine learning

With the recent developments in artificial intelligence, more contemporary research has focused on exploring machine learning applications within fire detection. Firstly, (Ko et al. 2009) showed the use of wavelet functions (calculated by tracking the horizontal, vertical, and diagonal velocities of pixels) combined with support vector machines as an effectual fire detection system. This machine-learning-based research was followed by (Chenebert et al. 2011), which considered a non-temporal approach with the combined use of colour-texture feature descriptors as input to a shallow neural network or decision tree for classification.

As with many visual pattern recognition problems, deep convolutional neural network architectures have dominated contemporary machine-learning-based approaches to fire detection. (Frizzi et al. 2016) used a classical convolutional network, combining convolution and max pooling. The use of a support vector machine in place of the full connection layer of the convolutional neural network was explored in (Wang et al. 2017). Instead of adapting and exploring different approaches, recent work has focused on improving the architecture of the convolutional network used for feature extraction. In (Dunnings & Breckon 2018), researchers attempted to adapt popular existing DCNN architectures, reducing their complexity, and applying them to fire detection. As expected, the reduced complexity improved the performance of their network. The models produced were able to classify up to 17 frames per second accurately. A direct continuation of the work produced by Dunnings and Breckon (Samarth et al. 2019) sought to increase the accuracy of classification without reducing efficiency. Using similar low weight networks, they achieved an accuracy of 95.6% at 12 frames per second - the current state of the art in non-temporal computer-vision-based fire detection. Extending the use of convolutional neural networks with temporal information, in (Kim & Lee 2019) region-based CNNs are applied to extract suspected regions and fire and non-fire in video imagery. These identified regions are then summarised and classified by a LSTM model where a majority voting system is used for overall video classification. This approach is the state of the art in fire detection, achieving a 97.92% success rate.

C Optical Flow

Optical flow is the pattern of apparent motion of objects in a scene. In computer vision, this is represented by a vector field mapping all image changes. Early versions of optical flow used a brightness based technique, in which, by considering the small changes in pixel brightness between frames, movement is captured. This technique, however, is prone to aperture problems. If the overall brightness of the image was too large, the optical flow algorithm performed very poorly. The Lucas-Kanade method was developed to address this issue (Lucas & Kanade 1981). Small regions in the image are assumed to have movement and are grouped. Optical flow equations, derived in (Lucas & Kanade 1981), are solved iteratively to produce an optical flow vector for each region. This vector is said to be the optical flow of each pixel in the region. More modern methods, such as (Brox et al. 2004), and (Xie et al. 2017), use standard variational models to approximate the optical flow.

As computer-vision-based fire detection models began to explore the use of temporal information, optical flow models and their application within fire detection was investigated. Initial attempts revolved about identifying flame pixels using colour based methods and then applying optical flow to map their movement. In (Rinsurongkawong et al. 2012), if the movement of fire pixels identified using (1) appeared to follow the expected movement of turbulent fire plumes, rather than the laminar flow which would be expected in non-fire objects, a positive signal was sent. Later, the optical flow of experimentally identified smoke pixels was calculated and fed into a neural network trained to produce a binary classification (Mueller et al. 2013). Building upon this, (Yu et al. 2013) selects pixels based not on their current colour properties but on their change over time. Yu et al. took advantage of pixel transport energy, flow magnitude, sink/source matching and direction variance to pre-select pixels for which the optical flow is calculated and interpreted by a neural network. These methods produce relatively successful solutions. However, due to the high computational cost of optical flow models, they are not suitable for widespread implementation.

To improve the computational cost of optical flow, in (Piergiovanni & Ryoo 2019) representational flow is introduced. The representational flow algorithm learns motion representations, more specifically flow parameters, rather than computing optical flow. Traditional approaches utilising optical flow use the flow map as a separate input stream for final classification. In representational flow, however, the flow is included as a layer in the architecture. In action recognition, this produced results that are as accurate or more accurate than the state of the art solutions and up to ten times more efficient.

Overall, from the work of (Samarth et al. 2019) and (Kim & Lee 2019), we can see that the current state of the art applies complex convolutional neural networks. In both, the spatial information about scenes is applied with machine learning to produce overall frame and video classification. Samarth et al. use popular DCNN architectures adapted to binary fire classification. Whereas, in (Kim & Lee 2019), feature extraction and classification is performed using R-CNN and LSTM models. Building from this, we will be investigating contemporary advances in action recognition and their relevance within computer-vision-based fire detection. Adapting the representational flow model proposed in (Piergiovanni & Ryoo 2019), we aim to produce a model capable of accurately classifying videos by considering both the features identified and their temporal change.

III SOLUTION

When considering how to produce a computer-vision-based fire detection system, we must first decide whether to use temporal or non-temporal information. Although most modern computer-vision-based fire detection systems are non-temporal, we decided to produce a solution utilising temporal information. One of the critical issues faced by non-temporal models is the presence fire-like objects, which appear similar to - but do not behave like - fire. This issue can result in a significant false-positive rate. If we consider how an object changes over time, we can more easily distinguish between objects with a similar appearance to fire and actual fire.

We must now consider the workflow necessary to produce an effective temporal fire detection system. Firstly, when classifying a video, a per-frame deep convolutional neural network should be applied to extract features. After normalisation, we feed the extracted features into a representational flow layer capable of mapping any changes between frames. From here, we pass through several convolutional layers for further visual and temporal feature extraction before classification occurs. By monitoring the frame-by-frame change to model the movement of objects, our solution could be significantly affected by camera movement. Requiring video from a stationary camera does not affect the use of this architecture as a widespread fire detection system but does reduce its applicability within an autonomous agent.

A Implementation Tools

To implement our project, we chose to use the Python language due to the ease of development and the wide variety of packages available. As PyTorch (*PyTorch Open Source Machine Learning Library* n.d.) is the industry standard for artificial intelligence research, it is well documented and supported. Furthermore, the representational flow model produced in (Piergiovanni & Ryoo 2019) uses PyTorch. For these reasons, we used PyTorch to build both our deep neural networks and representational flow layer. To load frames of video for classification, we used Lintel (Duke n.d.). This library aims to provide a fast and straightforward Python interface to develop machine learning algorithms using video datasets. OpenCV (*OpenCV Open Source Computer Vision and Machine Learning Library* n.d.) was used to capture metadata within videos to help train our model more effectively. Finally, to create our dataset, we used Bandicut (*Bandicut Video Cutter, Joiner and Splitter Software* n.d.) to split videos into smaller, more appropriate segments and (*MoviePy* n.d.) to resize all videos to a standard format.

B Convolutional Neural Networks

Inspired by nature, neural networks are a popular machine learning algorithm often used in complex pattern recognition problems. Each network is comprised of several layers containing nodes, or perceptrons, similar to neurons found in brains. Traditionally, neural networks have an input layer that accepts data into the model and an output layer. If it is a deep neural network, the model will contain at least one hidden layer, in-between the input and output layers, that applies non-linear transformations to the inputs. After a node has received input, we can calculate its output as the result of an activation function on the weighted sum of all inputs (with a bias) where the weight of each input is just the weight of the edge used to pass the input to the node. Usually between 0 and 1, this value represents the amount that the node is activated. Most models use a feed-forward architecture where this output is passed to every node in the following layer

via weighted edges, and the process is repeated until the output layer is reached. In classification problems, the number of nodes in the output layer corresponds to the number of possible classes, the result of each output node corresponds to the likelihood that the input belongs to that class. In our network, the output layer contains two nodes representing fire and non-fire.

We are using a neural network as part of a supervised training problem; therefore, during training, we provide the model with both input and output data. For each input, the neural network will attempt to estimate the correct output. This estimate is compared with the correct output, and the loss between them is calculated. Loss is a measure of the model's accuracy; the more similar the model's output and the correct answer are, the smaller the loss between them. Neural networks use this loss with back-propagation to update weights and biases between nodes, allowing them to predict the expected output with higher accuracy. Thus, through training, the model learns how to correctly classify each input by adjusting weights to identify patterns.

When applying standard neural networks to computer-vision-based tasks, they can become inefficient due to the vast number of parameters, or node weights, required to consider each pixel in an image. To overcome this, we introduce convolutional layers into our network. As shown in Figure 1, convolutional layers use a sliding filter to generate a feature map. Each value in the feature map is the dot product of the filter with the corresponding pixel values from the input map. These filters are initialised randomly and updated during training; they are weights in the network. By having a filter, convolutional layers effectively reuse the same weights - massively reducing the parameters required by a model. Convolutional filters are also spatially invariant; they can identify objects with comparable accuracy no matter their location in an image, a key benefit in object recognition tasks such as fire detection. As the model learns, its filters are able to extract more relevant features from each image. Intuitively, convolutional layers earlier in the architecture extract low-level features such as edges and colour, while layers later in the architecture aim to extract high-level features like objects and shapes.

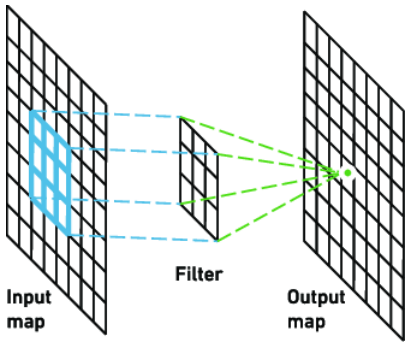

Figure 1: Visualisation of a convolutional layer (Yakura et al. 2018).

After extracting features using convolutional layers, we can add pooling layers to a network. Similarly to convolutional layers, pooling applies a sliding filter over a map, however, the filter is chosen, not learnt. Most commonly, we use a max-pooling operation that returns the maximum value selected by the filter, downscaling the feature map and the parameters required by the model. Moreover, such layers remove noise from convolutions and help prevent overfitting. After convolutional and pooling layers, CNNs use fully connected layers to produce a final classification. Feature maps are flattened and passed into a traditional neural network capable of classifying the image using the identified features.

C Deep Residual CNNs

As discussed, deep neural networks are networks containing at least three layers (Hinton et al. 2006). As networks become deeper, they contain more hidden layers. Therefore, deeper networks can learn more complex, non-linear functions. Furthermore, a greater number of convolutional layers will result in the extraction of more high-level features. These benefits imply that deeper networks can discriminate between different classes more accurately, given sufficient training data. Despite the advantages, the correlation between the number of layers and model accuracy does not always hold (He et al. 2016). For each model, there is an optimal depth of network at which the loss is minimised. This reduction in performance at greater depths can occur because of overfitting, or the degradation problem. It is beneficial to increase the depth at which this optimisation occurs, so we attempt to mitigate these issues.

With the increased ability to identify patterns in a training dataset, deeper networks often overfit and struggle to adapt to unseen data. We tackle overfitting using two techniques: batch normalisation and data dropout. Batch normalisation standardises outputs between layers such that they follow some standard normal distribution across the batch. Applying batch normalisation can not only help reduce overfitting in deep networks but also stabilises the learning process, dramatically reducing the number of training epochs required. Dropout is also a popular and effective technique to reduce overfitting in neural networks. Using dropout, we randomly discard a predetermined number of nodes and their relevant connections from neural networks during training. This prevents nodes from co-adapting too much. This technique can also greatly reduces the amount of computation required, making it an effective choice for complex neural networks.

When using neural networks in pattern recognition problems, as we pass through layers, we are effectively propagating information about the input down the neural network. As we deepen our model, it can become more difficult to propagate this information through the network; the quality of the information begins to degrade as it passes through layers. This is called the degradation problem. Residual neural networks (ResNets) (He et al. 2016) are an extension of traditional deep learning models based on concepts observed in pyramidal cells found in the cerebral cortex. These networks can help to reduce the degradation problem by using skip connections to jump over layers. These shortcuts directly connect shallow neurons to deeper neurons helping to pass information through the network with less degradation. As fire detection is a complex problem, increasing the depth of our network could significantly improve our results - the use of batch normalisation, dropout and residual neural networks allows us to do so.

D Representational Flow

In computer vision, optical flow is a method by which we can model the change between two images often applied to capture the movement of objects in a video. Given images I_1 and I_2, an optical flow algorithm aims to find the updated position of each point of I_1 within I_2. As these images should be consecutive within a video, there is only a small time-step between them. From this, we can reasonably make two key assumptions: between I_1 and I_2 the brightness of each point should remain constant, and any movement which does occur should be small. The brightness assumption allows us to state that for each point (x, y) in I_1 moved by $[\Delta x, \Delta y]$ between images, $I_1(x, y) = I_2(x + \Delta x, y + \Delta y)$. As the movement is small, this can be approximated using a Taylor series:

$$I_2 = I_1 + \frac{\delta I}{\delta x}\Delta x + \frac{\delta I}{\delta y}\Delta y. \tag{2}$$

If we solve this (2) we can easily calculate the flow between I_1 and I_2, $\mathbf{u} = [\Delta x, \Delta y]$. However, as (2) contains two unknown variables, it can only be approximated. Research in optical flow aims to produce algorithms approximating the solutions to (2) as accurately and efficiently as possible.

Variational models, applied in (Brox et al. 2004) (Xie et al. 2017), are often used for optical flow. In such methods, we estimate **u** using an iterative optimisation technique. When optimising, models use manually set hyperparameters such as the smoothness of output and time-step with experimentally calculated values to improve results. Variational approaches have mixed success. They obtain excellent flow estimates but are very computationally expensive, taking a significant amount of time to produce accurate results.

Representational flow (Piergiovanni & Ryoo 2019) extends variational models by utilising a fully-differentiable, learnable convolutional representational flow layer. The flow layer takes two CNN feature maps as input and applies convolutional filters with the iterative optimisation algorithm proposed in (Xie et al. 2017) to compute the flow between them. By doing so, hyperparameters normally set manually are weights in a convolutional layer. Rather than capturing brightness, when using CNN feature maps we assume that feature values extracted by the CNN are consistent between inputs. As, by design, convolutional layers spatially invariant, this is a valid assumption. The use of both convolutional layers and lower resolution CNN feature maps as input make representational flow up to ten more efficient than traditional optical flow models (Piergiovanni & Ryoo 2019). Furthermore, there is no need to set the hyperparameters for optimisation manually, as with all weights in the network, they are learned through training. By learning hyperparameters, the representational flow layer can be optimised for any optical flow task. In Figure 2, we can see the flow layer calculated on two frames from a video in the MIVIA dataset. In this project, we aim to use a representational flow layer in a deep network to reliably determine whether a video contains fire.

Figure 2: Flow layer calculated for two frames using representational flow.

As discussed, optical flow algorithms are typically used to calculate a map of the flow between two images. We can, however, use them to calculate the changes between two flow images. By computing the flow of two flow images, we essentially track points showing similar motion. In practice, this typically leads to worse performance as optical flow results do not follow the brightness consistency assumption. However, Piergiovanni and Ryoo found that adding an intermediate convolutional layer to smooth the flow and convert from movement vectors to feature

values suppressed these inconsistent results and improved the performance for representational 'flow-of-flow' models. In (Piergiovanni & Ryoo 2019), adding an extra representational flow layer improved results. This project will explore the use of a second representational layer for a computer-vision-based fire detection system. Follwing previous logic, we could introduce a third representational flow layer. However, in (Piergiovanni & Ryoo 2019) this decreased the accuracy of action recognition, the increased depth of the network begins to degrade the quality of flow information. Therefore, we will not include the use of a third representational flow layer in our project.

E Network Architecture

First proposed in (Piergiovanni & Ryoo 2019), the basic architecture for our computer-vision based fire detection system is shown in Figure 3. We begin by passing each input into a 3D convolutional layer with kernel size 7 and stride 2 for low-level feature extraction. These low-level features help identify high-level features later in the model, so dropout and max-pooling are applied to reduce overfitting. Output feature values are then fed into a residual neural network for high-level feature extraction. The representational flow layer is included here to map the movement of high-level objects identified by the first residual neural network. Through experimentation, Piergiovanni and Ryoo found this to be the optimal position for the flow layer. These flow maps are fed into another residual neural network used to learn patterns within the movement of high-level objects. Finally, we feed into a convolutional layer, with kernel size and stride 1, capable of producing a classification for each video. This convolution outputs a 2x1 vector containing the estimated likelihoods that a video either containing fire or not. Classification of a video by our model occurs by selecting the option with the greater estimated likelihood.

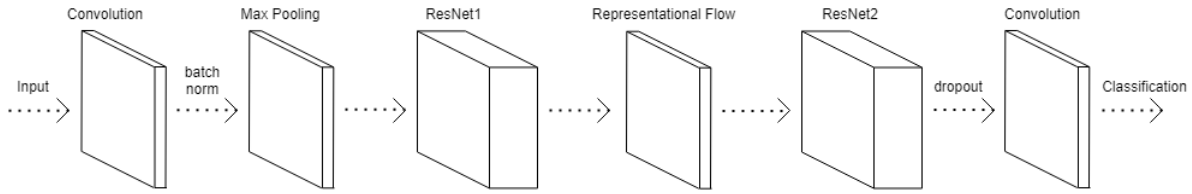

Figure 3: Our network architecture

Each residual neural network is made up of residual neural network blocks, or ResNet blocks. The architecture of each is shown below in Figure 4. Every layer in a block is a convolution - the filter properties of each convolution vary with both the position of the layer within the block and the block within the ResNet. Between each layer, we apply batch normalisation. We can modify the depth of our residual neural networks by changing the number of blocks it comprises. Our project investigates architectures with residual neural networks containing 36, 100 and 202 blocks for accurate fire detection systems.

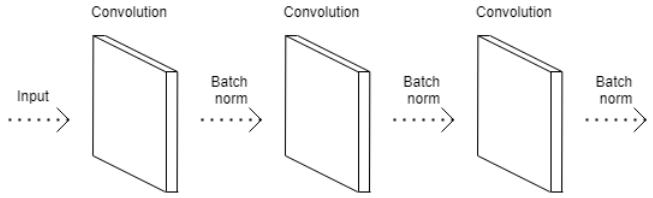

Figure 4: ResNet Block

As detailed in subsection D, we explore the use of flow-of-flow architectures. In such an architecture, the block shown in Figure 5 replaces the representational flow layer from our original architecture, Figure 3. The block contains two representational flow layers with a 3D convolutional layer to convert from movement vectors to feature values for improved performance. The convolutional layer has kernel size (2, 1, 1), stride 1 and padding (1, 0, 0). Batch normalisation is applied to the output of the convolutional neural network as CNN features extracted are quite small, < 0.5 on average (Piergiovanni & Ryoo 2019). The flow optimisation algorithm used in representational flow is designed for standard image values ranging from [0, 255]. This normalisation step stabilises the input for the second flow layer.

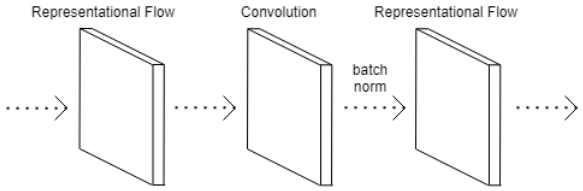

Figure 5: Flow-of-flow block

F Training and Evaluation Datasets

We searched the relevant literature for suitable datasets with proven effectiveness when generating a video dataset for training and evaluation of our model. We found five relevant datasets compiled by prominent papers in the field. Firstly, we decided to use the dataset produced in (Chenebert et al. 2011), a standard fire detection dataset used in many previous papers. Next, we included the dataset generated in (Steffens et al. 2016), which was produced for the sole purpose of evaluating models in this field. We also used the MIVIA dataset produced in (Foggia et al. 2015) and (Di Lascio et al. 2014), another dataset used in many previous models. Finally, we used the (Dunnings & Breckon 2018) dataset and (Kim & Lee 2019). These datasets were especially valuable as they were used in (Samarth et al. 2019) and (Kim & Lee 2019) respectively, allowing us to make direct comparisons between our solution and the state of the art. By incorporating several datasets from different sources, we increased the variance of data used to train and test, ensuring we produce a reliable and robust model.

Many of these datasets, however, were used for non-temporal computer-vision-based fire detection systems. Consequently, we noticed that they were ill-suited to training our model. Videos often featured camera cuts, causing the model to mistakenly identify camera change as object movement, making it difficult to accurately learn the movement of objects, specifically fire, in scenes. Furthermore, the length of each video was unsuitable for training on our model; in many

videos labelled as containing fire, significant portions contained no fire. Our model may sample such sections from positively labelled videos when training, causing incorrect information to be backpropagated through the network. To mitigate this issue, we decided to split videos into smaller sections. A 10s splitting interval was chosen as a suitable length as the model produced in (Piergiovanni & Ryoo 2019) is trained and evaluated on the Kinetics dataset produced by DeepMind containing videos approximately 10s long. Moreover, state of the art temporal model produced in (Kim & Lee 2019) trains using 10s videos. After splitting videos using Bandicut (*Bandicut Video Cutter, Joiner and Splitter Software* n.d.), we combed through videos to remove unsuitable segments. Finally, we labelled each video as either containing fire by assigning a 1 or not containing fire by assigning a 0.

When training and testing deep-learning models, it is essential to standardise all inputs to ensure reliable and representative identification of features. When using 3D convolutional layers, we do this by ensuring all videos are the same size. In (Piergiovanni & Ryoo 2019), standardising is done by randomly cropping videos to a standard size. By cropping randomly, overfitting to training data is reduced as the number of unique videos shown to the model increases. Despite this, we found cropping to be an ineffective standardisation technique for fire detection tasks. When cropping on videos containing a fire, there is a chance that we remove the fire. Such occurrences have a significant adverse effect on the performance of a model as incorrect information is backpropagated. Instead, we decided to resize our dataset. Resizing videos, instead of cropping, ensures that we retain all features within each video; there is no possibility of removing the fire. Following the standard 2:3 aspect ratio, we chose to resize videos to 60x90px. The final dataset produced contained 5513 resized videos, 2714 labelled as positive fire videos, and 2799 labelled as negative fire videos.

G Sampling Strategy

When training, we must consider how to load videos such that the model is provided with the most appropriate information to help achieve optimal classification performance. In (Piergiovanni & Ryoo 2019), a random selection of 16 frames is used as a representative subsample of the entire video - their model then classifies the action occurring in a video using the architecture shown in Figure 3. This strategy, however, is ineffective for binary fire detection. Human actions are repetitive, periodic and simple: a random selection of frames can reliably represent this information. The flow between the frames selected for action movement can reasonably show the movement occurring throughout the video. This assumption does not hold when monitoring the movement of fire. Flames and smoke plumes characterising fire follow turbulent motion; movement defined by irregular fluctuations and random mixing. The modelling and understanding of such motion is notoriously complex; in fact, it is the basis of an unsolved Millenium Prize Problem. Hence, we decided to produce and test several different video sampling strategies: *Start*, *Any*, and *Window* (each implemented using the (Duke n.d.) package for Python).

(i) *Start*: We sample and train using the first 16 frames from each video in the training dataset.

(ii) *Any*: We sample and train using a randomly selected group of 16 consecutive frames from each video in the training dataset.

(iii) *Window*: We sample and train using a sliding window that iteratively selects all groups of 16 consecutive frames (without group overlap). A video or window with less than 16 frames is not used when sampling.

These sampling strategies use a contiguous block of frames to help the model better learn to distinguish between the movement of the fire and non-fire objects. With each method, we train the model by interpolating the given classification for a video as the correct classification for the sampled selection. When using *Window*, each group is loaded and trained independently, increasing the size of the training set dramatically. To produce classifiactions using the *Window* sampling strategy, we allow the model to make predictions on each group in a video and use the modal result as the overall prediction. If there is no mode, we use the most recent prediction by the model.

IV RESULTS

When experimenting, we used a standard 75:25 training set to evaluation set split. Furthermore, to ensure the validity of our results, we chose the training and evaluation sets to be completely independent. As with many deep neural networks, the computational demands of training our model exceeded any standard computer's hardware capabilities. Due to their ability to perform parallel computing, we often use GPUs for this purpose. We trained our model using the Durham University NVIDIA CUDA Centre (NCC) GPU system.

To compare with previous papers and assess the quality of our model we will calculate the accuracy, true-positive rate (TPR), false-positive rate (FPR) and precision (PPV) statistics for each experiment. We will also use the critical success index (CSI), F1 score, and Matthews Correlation Coefficient (MCC) to provide a comprehensive analysis of our results.

$$\text{CSI} = \frac{\text{TP}}{\text{TP + FN + FP}} \quad (3) \qquad \text{F}_1 = 2 \times \frac{\text{PPV} \times \text{TPR}}{\text{PPV} + \text{TPR}} \quad (4)$$

$$\text{MCC} = \frac{\text{TP} \times \text{TN - FP} \times \text{FN}}{\sqrt{(\text{TP + FP})(\text{TP + FN})(\text{TN + FP})(\text{TN + FN})}} \quad (5)$$

A Binary Video Classification

When training deep models, it is crucial to determine the number of training epochs required to optimise performance. If we overtrain, deep learning models will overfit to the training data, reducing the performance on the evaluation set. Comparatively, if we undertrain, the model will not have learned any patterns in the data required for accurate classification, again reducing performance. Consequently, to determine the optimal training time and sampling strategy for each residual neural network depth, we trained each model depth and sample strategy pair over 24 hours and evaluated their performance at five regular intervals: 0 hours, 6 hours, 12 hours, 18 hours and 24 hours. Training this way not only allowed us to investigate the optimal training time, but also the effect of our varying sample strategies and residual neural network depth on performance. As can be seen in Figure 6, all three depth-36 models optimised quickly, with results plateauing at around 12 hours. However, below this depth *Window-100* was the only model to produce satisfactory results.

Following these results, we retrained each model twice using the data produced by our initial test to inform training times. To produce the results showin in Table 1, took the average of both trainining attempts. This was done to reduce any random error which may occur. Again, we can see that regardless of sampling strategy, depth-36 models outperform deeper residual neural networks in binary fire detection. Our *Any-36* model was the most successful, achieving the

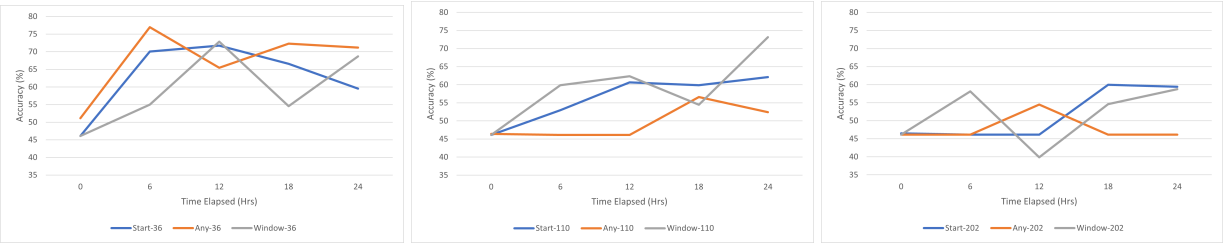

Figure 6: Evaluation accuracy over time for ResNet depth (a) 36, (b) 100 and (c) 202 blocks.

highest accuracy, CSI, F1 score and MCC in the batch of base models. Disregarding *Window-100*, the Matthews Correlation Coefficient scores achieved by depth-100 and depth-202 models are close to 0, indicating a performance comparative to random selection. We suspect that the degradation problem has significantly affected the performance of such models. By training using the *Window* sampling strategy, our *Window-100* model produced the best performance of any model deeper than depth-30.This is because, the *Window* sampling strategy provides the greatest variety in data shown to the model during training, reducing the degradation problem.

Model	Accuracy	TPR	FPR	Precision	CSI	F1	MCC
Start-36	71.75	83.43	41.90	69.94	61.41	76.10	43.20
Any-36	**76.98**	79.51	26.04	**78.12**	**65.07**	**78.83**	**53.62**
Window-36	72.82	80.57	36.23	72.22	61.51	76.17	45.15
Start-100	62.12	95.14	76.46	59.25	57.51	73.03	27.28
Any-100	56.58	35.43	**18.70**	68.89	30.52	46.79	18.63
Window-100	73.13	79.29	34.06	73.12	61.39	76.08	45.75
Start-202	59.97	93.14	78.80	58.01	55.63	71.49	20.94
Any-202	54.50	**98.14**	96.49	54.31	53.76	69.92	5.15
Window-202	58.12	83.14	71.12	57.74	51.69	68.15	14.38

Table 1: Average results achieved over two trains for each model depth and sampling strategy pair.

Despite their accuracy, neither the highest true-positive rate nor the lowest false-positive rate was achieved by a depth-36 model. We can see that *Any-202* achieved the highest TPR. However, it also produced the highest FPR, and an the lowest accuracy of all models trained. It is clear that, instead of producing an accurate classification system, *Any-202* classified almost all videos as containing fire. Similarly, *Any-100* achieved the lowest false-positive rate by classifying almost all videos negatively. For future experimentation, we decided to only include models that achieved an accuracy of 70% or over: *Any-36*, *Start-36*, *Window-36* and *Window-100*.

B Hybrid Sampling and Data Augmentation

To further experimentation with sampling, we decided to implement hybrid combinations of our strategies - *Start*, *Any* and *Window*. We created models combining different training and evaluation sampling strategies. As shown in Table 2, we used *Window* sampling for evaluation on pre-trained *Any-36* and *Start-36* models. This tested whether models capable of producing a high classification accuracy using a single window per video could increase the accuracy of

the modal result on multiple windows. However, our results show that varying the training and evaluation methods did not have any significant effect. In fact, the models using hybrid sampling performed slightly worse than their non-hybrid counterparts in almost every evaluation category.

Model	Accuracy	TPR	FPR	Precision	CSI	F1	MCC
Start-36	71.75	**83.43**	41.90	69.94	61.41	76.10	43.20
Any-36	**76.98**	79.51	**26.04**	**78.12**	**65.07**	**78.83**	**53.62**
Start-Window-36	71.59	82.76	41.29	69.82	60.95	75.73	42.96
Any-Window-36	76.37	79.43	27.21	77.33	64.42	78.36	52.36

Table 2: Results of hybrid sampling strategies over pre-trained models.

When training, we noticed that our models showed a much higher accuracy during training than in evaluation. Therefore, we decided to test the effect of using overfitting mitigation by applying data augmentation. In augmented training, there was a 50% chance of each loaded video being either flipped along the vertical axis or rotated by a multiple of 90 degrees. This increases the variance of training data shown to each model. The results, shown in Table 3, highlight data augmentation as a viable method to improve the results for the deeper models; the accuracy of the *Window-100* model increased by 4.93%. This increase occurred as, due to the depth of the model, without data augmentation significant overfitting had occurred. By using augmentation, the *Window-100* model was able to learn more complex patterns representative of the entire dataset. Achieving the highest accuracy, precision, CSI, F1 score and MCC, this was the best model produced.

Model	Accuracy	TPR	FPR	Precision	CSI	F1	MCC
Start-36-Aug	64.28	45.27	**13.52**	79.65	40.59	57.74	34.35
Any-36-Aug	67.21	**90.00**	59.43	63.90	59.99	74.73	35.63
Window-36-Aug	75.29	77.14	26.88	77.03	62.72	77.09	50.27
Window-100-Aug	**78.06**	78.51	22.54	**80.29**	**65.87**	**79.42**	**55.95**

Table 3: Results of data augmentation on training.

C Flow of Flow

Replacing the single representational flow layer, we expermiented with the use of flow-of-flow architecture in our models. As shown in (Piergiovanni & Ryoo 2019), flow-of-flow can produce systems capable of identifying complex patterns present that can be applied generally across the entire distribution of data. However, the results in Table 4 do not support this. Our flow-of-flow models failed to identify any pattern in the data suitable for binary fire classification in videos. Both flow-of-flow models achieved high true-positive rates. However, we can see that they identified almost all videos positively by comparing this to their false-positive rates. Moreover, both models achieved a negative Matthews Correlation Coefficient implying that the models performed worse than random classification.

Model	Accuracy	TPR	FPR	Precision	CSI	F1	MCC
Window-36	**53.89**	**99.29**	99.66	**54.02**	**53.81**	**69.97**	-2.56
Window-100	53.43	96.71	**96.71**	53.77	52.80	69.12	**-1.29**

Table 4: Results of flow-of-flow models.

D State of the Art Comparison

Finally, we compared the results of our experimentation to the current state of the art in computer-vision-based fire detection, allowing us to understand the performance of our model relative to the literature. As shown in Table 5, our best model, *Window-100-Aug*, significantly underperformed compared with the current state of the art. Our model achieved a lower accuracy, true-positive rate and false-positive rate. These results show the limitation of our method for binary fire classification in videos.

Model	Accuracy	TPR	FPR
Kim & Lee (2019)	**97.92**	**97.53**	**1.38**
Bhowmik and Breckon (2019)	95.60	96	5
Proposed Model	78.06	78.51	22.53

Table 5: Comparison between our model and the state of the art.

V EVALUATION

In this section, we will evaluate the strengths and limitations of both our approach and the final solution within the context of our research question: *Can representational flow be successfully applied to the task of detecting fire within a video in order to advance the current state of the art?*

A Solution Strengths

Whilst we have not produced a solution extending the state of the art in binary fire classification, we have produced a model capable of correctly classifying a significant proportion of videos, achieving a 78.06% accuracy. Furthermore, our model does not show any bias towards either positive or negative classification; the true-positive rate (78.51) and the true-negative rate (77.47), calculated as 100-FPR, are almost identical. To that extent, our findings support the addition of movement information into complex convolutional neural networks to produce accurate classification. Extensions to this research could investigate the use of alternative, more accurate feature extraction and pattern recognition models. As in (Samarth et al. 2019), we could adapt popular deep learning architectures, such as Inception-V4, capable of highly accurate object recognition to perform feature extraction. Moreover, extensions could explore the application of LSTM models for pattern recognition within flow maps, as is shown to be effective in the state of the art model produced by (Kim & Lee 2019).

We also consider the substantial experimentation undertaken and evaluated in our report as a meaningful strength. We explored the use of multiple sampling techniques, residual neural

network depths and extensions to the base architecture - such as flow-of-flow models. By introducing the *Window* sampling strategy and applying data augmentation, we discovered that deeper residual neural networks can integrated with the architecture proposed in (Piergiovanni & Ryoo 2019) for accurate identification of patterns and subsequent binary classifications of videos for fire detection. While our shallow neural networks can produce accurate results, we observed that increasing the variance of data shown to deeper networks generates higher performance. Moreover, validating the results of this experimentation through the use of binary classification evaluation techniques such as the critical score index, F1 score, and the Matthews Correlation Coefficient allows for robust comparison both within our work and surrounding literature.

B Solution Limitations

A clear, fundamental limitation of our solution was the inability to improve upon the state of the art research in computer-vision-based fire detection. Our best model produced an accuracy 19.86% lower the state of the art (Kim & Lee 2019). To this point, our model performed poorly compared to (Kim & Lee 2019) in all basic evaluation metrics; it also achieved a lower true-positive rate and false-positive rate. Furthermore, we were unsuccessful in our introduction of flow-of-flow architecture to our solution. Achieving negative Matthews Correlation Coefficient scores, the models performed worse than random classification. In future, we could extend our research into flow-of-flow by exploring the use of varying dataset input sizes. The use of a relatively small videos resized to 60x90px may have reduced the ability of our models to extract information. As these extracted features propagated through the network, information may have degraded at the flow-of-flow block reducing overall performance. Our solution also appeared to overfit significantly to the training data. The most significant boost in performance came when applying data augmentation to the *Window-100* model. Therefore, further techniques such as dropout or parameter reduction could be applied in future research.

C Approach

When completing our project, we applied an agile workflow. By using iterative optimisation of a base solution, despite unsuccessful flow-of-flow models we were still able to generate a viable computer-vision-based fire detection system. Developing our solution using an agile workflow, we were also obtain a significant amount of results. As our deep models took up to 24 hours to train and the Durham University NCC limits the number of simultaneous jobs to 4, this was very important. We were still able to obtain results and address issues in our model whilst aspects of the solution were still in development.

The amount of time spent compiling and pre-processing our dataset was a significant limitation to our approach. Difficulties using the Lintel package to load videos meant that we could not automate the splitting of videos using software such as FFmpeg. Consequently, we had to split all videos manually using Bandicut (*Bandicut Video Cutter, Joiner and Splitter Software* n.d.). Moreover, after splitting, we had to clean our datasets by manually classifying video sections to ensure our model was not training on incorrectly labelled video. Repeating this process over thousands of videos collected from multiple datasets was very time-consuming. This time could have been used more effectively by investigating further experimentation and exploring the use of contemporary technologies and their applications to our solution. Despite the effect these shortcomings may have had on the overall performance of our project, we still believe our project

to be successful. We followed scientific methods to produce robust results from meaningful experimentation.

VI CONCLUSIONS

In this project, we produced a model capable of identifying the presence of fire within a video with a 78.06% accuracy. To achieve this, we implemented contemporary advances in action recognition and assessed their usefulness within computer-vision-based fire detection. Building from the architecture proposed by (Piergiovanni & Ryoo 2019), we experimented with several differing residual neural network depths, sampling strategies and the use of data augmentation. The best model produced was created using residual neural networks built using 100 ResNet blocks (shown in Figure 4), with a *Window* sampling technique and data augmentation. Furthermore, we extended the basic architecture by generating flow-of-flow models aimed at producing complex temporal information. However, the flow-of-flow models produced were ineffective, achieving results comparable to random classification.

Our solution was unable to advance state of the art in this field, produced by (Kim & Lee 2019), which has an accuracy of 97.92%. Despite this, our results show key improvements compared to many implementations - most notably to many non-temporal approaches. The inclusion of movement information in our architecture reduced any bias between positive and negative data classification, a problem often faced by non-temporal approaches. For this reason, the need for continued research into the inclusion of temporal information for accurate fire detection is clear. We believe that the clear progression of our project would be the introduction of more accurate object recognition algorithms and pattern recognition models as explored in the state of the art models produced by (Kim & Lee 2019) and (Samarth et al. 2019).

REFERENCES

Bandicut Video Cutter, Joiner and Splitter Software (n.d.). Online; accessed 12th April 2021.
URL: *https://www.bandicam.com/bandicut-video-cutter/*

Bradshaw, A. (1991), The uk security and fire fighting advanced robot project, *in* 'Proc. Int. Conf. Advanced Robotic Initiatives in the UK', IET, pp. 1–1.

Brox, T., Bruhn, A., Papenberg, N. & Weickert, J. (2004), High accuracy optical flow estimation based on a theory for warping, *in* 'Proc. European Conf. Computer Vision', Springer, pp. 25–36.

Celik, T. & Demirel, H. (2009), 'Fire detection in video sequences using a generic color model', *Fire safety journal* **44**(2), 147–158.

Chen, T., Wu, P. & Chiou, Y. (2004), An early fire-detection method based on image processing, *in* '2004 Int. Conf. Image Processing', Vol. 3, IEEE, pp. 1707–1710.

Chenebert, A., Breckon, T. & Gaszczak, A. (2011), A non-temporal texture driven approach to real-time fire detection, *in* 'Proc. Int. Conf. image processing', IEEE, pp. 1741–1744.

Di Lascio, R., Greco, A., Saggese, A. & Vento, M. (2014), Improving fire detection reliability by a combination of videoanalytics, *in* 'Proc. Int. Conf. Image Analysis and Recognition'.

Duke, B. (n.d.), 'Lintel: Python video decoding'. Online; accessed 12th April 2021.
URL: *https://github.com/dukebw/lintel*

Dunnings, A. & Breckon, T. (2018), Experimentally defined convolutional neural network architecture variants for non-temporal real-time fire detection, *in* 'Proc. Int. Conf. Image Processing', IEEE, pp. 1558–1562.

Foggia, P., Saggese, A. & Vento, M. (2015), 'Real-time fire detection for video surveillance applications using a combination of experts based on color, shape and motion', *Proc. Int. Conf. Circuits and Systems for Video Technology* .

Frizzi, S., Kaabi, R., Bouchouicha, M., Ginoux, J., Moreau, E. & Fnaiech, F. (2016), Convolutional neural network for video fire and smoke detection, *in* 'Proc. Conf. Industrial Electronics Society', IEEE, pp. 877–882.

He, K., Zhang, X., Ren, S. & Sun, J. (2016), Deep residual learning for image recognition, *in* 'Proc, Int. Conf. Computer Vision and Pattern Recognition', pp. 770–778.

Healey, G., Slater, D., Lin, T., Drda, B. & Goedeke, A. (1993), A system for real-time fire detection, *in* 'Proc. Conf. Computer Vision and Pattern Recognition', IEEE, pp. 605–606.

Hinton, G., Osindero, S. & Teh, Y. (2006), 'A fast learning algorithm for deep belief nets', *Neural computation* **18**(7), 1527–1554.

Khalil, A., Rahman, S., Alam, F.and Ahmad, I. & Khalil, I. (2020), 'Fire detection using multi color space and background modeling', *Fire Technology* pp. 1–19.

Kim, B. & Lee, J. (2019), 'A video-based fire detection using deep learning models', *Applied Sciences* **9**(14), 2862.

Ko, B., Cheong, K. & Nam, J. (2009), 'Fire detection based on vision sensor and support vector machines', *Fire Safety Journal* **44**(3), 322–329.

Liu, C. & Ahuja, N. (2004), Vision based fire detection, *in* 'Proc. Int. Conf. Pattern Recognition', Vol. 4, IEEE, pp. 134–137.

Lucas, B. & Kanade, T. (1981), An iterative image registration technique with an application to stereo vision, Vancouver, British Columbia.

Martínez-de Dios, J., Merino, L., Caballero, F., Ollero, A. & Viegas, D. (2006), 'Experimental results of automatic fire detection and monitoring with uavs', *Forest Ecology and Management* **234**(1), S232.

MoviePy (n.d.). Online; accessed 12th April 2021.
URL: *https://github.com/Zulko/moviepy*

Mueller, M., Karasev, P., Kolesov, I. & Tannenbaum, A. (2013), 'Optical flow estimation for flame detection in videos', *Transactions on image processing* **22**(7), 2786–2797.

OpenCV Open Source Computer Vision and Machine Learning Library (n.d.). Online; accessed 12th April 2021.
URL: *https://opencv.org/*

Phillips Iii, W., Shah, M. & Vitoria, L. (2002), 'Flame recognition in video', *Pattern recognition letters* **23**(1-3), 319–327.

Piergiovanni, A. & Ryoo, M. (2019), Representation flow for action recognition, *in* 'Proc. Int. Conf. Computer Vision and Pattern Recognition', pp. 9945–9953.

PyTorch Open Source Machine Learning Library (n.d.). Online; accessed 12th April 2021.
URL: *https://pytorch.org/*

Rinsurongkawong, S., Ekpanyapong, M. & Dailey, M. (2012), Fire detection for early fire alarm based on optical flow video processing, *in* 'Proc. Int. Conf. Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology', IEEE, pp. 1–4.

Samarth, G., Bhowmik, N. & Breckon, T. (2019), Experimental exploration of compact convolutional neural network architectures for non-temporal real-time fire detection, *in* 'Proc. Int. Conf. Machine Learning And Applications', IEEE, pp. 653–658.

Steffens, C., Rodrigues, R. & Costa Botelho, S. (2016), Non-stationary vfd evaluation kit: Dataset and metrics to fuel video-based fire detection development, *in* 'Robotics', Springer, pp. 135–151.

Toreyin, B., Dedeoglu, Y. & Cetin, A. (2005), Flame detection in video using hidden markov models, *in* 'Proc. Int. Conf. Image Processing', Vol. 2, IEEE, pp. II–1230.

Wang, D., Guan, D., Zhu, S., Mac Kinnon, M., Geng, G., Zhang, Q., Zheng, H., Lei, T., Shao, S. & Gong, P. (2021), 'Economic footprint of california wildfires in 2018', *Nature Sustainability* **4**(3), 252–260.

Wang, Z., Wang, Z., Zhang, H. & Guo, X. (2017), A novel fire detection approach based on cnn-svm using tensorflow, *in* 'Proc. Int. Conf. Intelligent Computing', Springer, pp. 682–693.

Xie, S., Sun, C., Huang, J., Tu, Z. & Murphy, K. (2017), 'Rethinking spatiotemporal feature learning for video understanding', *arXiv preprint arXiv:1712.04851* **1**(2), 5.

Yakura, H., Shinozaki, S., Nishimura, R., Oyama, Y. & Sakuma, J. (2018), Malware analysis of imaged binary samples by convolutional neural network with attention mechanism, *in* 'Proc. ACM Conf. Data and Application Security and Privacy', pp. 127–134.

Yuan, F. (2010), 'An integrated fire detection and suppression system based on widely available video surveillance', *Machine Vision and Applications* **21**(6), 941–948.