Deep Neural Network with Walsh-Hadamard Transform Layer For Ember Detection during a Wildfire

Hongyi Pan*, Diaa Badawi*, Chang Chen*, Adam Watts†
Erdem Koyuncu*, Ahmet Enis Cetin*,

*Department of Electrical and Computer Engineering, Pacific Northwest Research Station
*University of Illinois at Chicago, †U.S. Forest Service
*{hpan21, dbadaw2, cchen269, ekoyuncu, aecyy}@uic.edu, †Adam.Watts@usda.gov

Abstract

In this article, we describe an ember detection method in infrared (IR) video. Embers, also called firebrands, can act as wildfire super-spreaders. We develop a novel neural network with a Walsh-Hadamard Transform (WHT) layer to process the IR video. The WHT layer is used to process the temporal dimension of the video data to model the high-frequency activity due to ember movements. We insert the WHT layer to ResNet-18 and obtained higher accuracy compared to the standard single slice ResNet-18 and the ResNet-18 processing the entire video block. We also repeat the experiments on ResNet-34, but we found that ResNet-18 is sufficient for this task. Therefore, we choose the ResNet-18 with the WHT layer as the proposed model.

1. Introduction

Global climate change has led to an increased wildfire season length, frequency, and burned area, with devastating consequences, especially in recent years [1]. For example, the year 2018 marked the deadliest and the most destructive wildfire season in the State of California in recorded history. Over several months, thousands of wildfires spanning almost 2 million acres claimed the lives of 98 civilians and 6 firefighters. In particular, the Camp Fire, which occurred over a period of two weeks in mid-November 2018, destroyed most of the town of Paradise, resulting in the deaths of 85 people. Close to 20,000 buildings were destroyed, with estimated economic damage of $20B [2]. On the global scale, the "Black Summer" fires in Australia in the 2019-2020 season killed at least 35 people [3]. Also, as many as one billion animals perished, and thousands of homes were destroyed. The unprecedented amount of CO2 emissions resulting from the fires also created a huge environmental impact. Once a wildfire begins, it can very rapidly expand to a large area and may become uncontrollable due to several factors such as high winds or low humidity as in the Camp Fire.

Computer-vision-based detection of wildfires have received much attention [4–20]. However, automated UAV-based IR real-time monitoring of wildfires using deep learning has not been studied in the literature. The goal of a complete IR computer vision system is to monitor the firefront, embers, and the firemen as well as other people on the ground. The firefront monitoring system will provide general-purpose situational awareness for firemen during a wildfire. A characteristic feature of wildfires is ember attacks where small-sized burning or charred pieces of wood or vegetation are carried from the firefront to remote locations via winds. Embers may then start new fires at where they land. Embers are burning pieces of airborne wood and vegetation produced by wildfires, and they can travel miles in the wind [21]. Depending on the type of fuel and the wind speed, the spotting distance (i.e., the distance between the origin of the ember and its destination) may reach more than a mile. Embers can cause wildfires to spread not only quickly but also unpredictably and they can ignite the fire behind the firemen [22]. Fig. 1 shows flying embers.

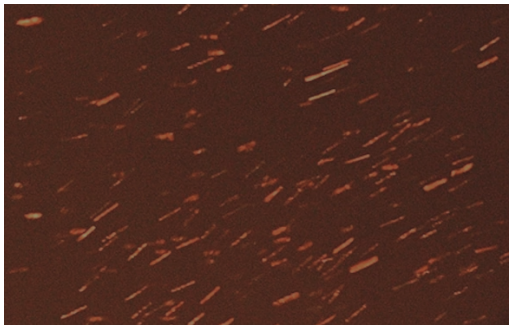


Figure 1. Flying embers during a wildfire. The image is downloaded from [21].

Convolutional neural networks (CNNs) have been successfully employed in many detection fields, and one of the applications is wildfire detection [13–20]. Since embers are clusters of burning objects we should use video to recognize them. In this article, we propose an ember detection method in infrared videos which are obtained from a UAV monitoring the wildfire front. Our algorithm will automatically detect ember attacks near the firefront using a novel deep learning network. In this paper, we design a novel convolutional neural network that will accept a sequence of image frames and process the image frames using a Walsh-Hadamard Transform (WHT) layer in the temporal domain. In a regular CNN, the first layer filters should be very deep to process the sequence of images. Training such filters is difficult compared to calculating their temporal Walsh-Hadamard transforms which model the movements of ember clusters based on the "frequency" components of temporal behavior. In this article, we use the WHT instead of a standard wavelet or the complex Fourier Transform to avoid complex arithmetic. If there is an ember cluster in a block of image data there should be a high amount of high-frequency activity due to fast ember movements compared to other segments of the video.

Other transform-based layers include [23–30], but neither of them are used to process the data in video processing to model the temporal behavior of objects.

In Section 2, we will introduce the methodology. It describes the methods we propose the WHT layer and we implement it into ResNet-18 and ResNet-34. In Section 3, we will introduce our video dataset and how do we make the video clips for training and testing, then we will present our experimental results to show that the proposed model, the ResNet-18 with the WHT layer, is most suitable for this task.

2. Methodology

In this section, we will first review the Walsh-Hadamard Transform in Section 2.1. Then, we will describe the WHT layer with smooth-thresholding activation function in section 2.2. Next, we will introduce how to insert the WHT layer into the ResNet architecture in Sections 2.3 and 2.4.

2.1. Walsh-Hadamard Transform (WHT)

The Walsh-Hadamard Transform (WHT) is an orthogonal binary transform and it is equivalent to block-Haar wavelet transform [31]. The 1-D WHT is equivalent to applying N band-pass filters to an input vector of size N. Therefore, it can be used in the time-scale analysis of data. The low-index (high-index) transform domain coefficients approximately represent the low-frequency (high-frequency) content of the input vector.

Let $\{x_n\}$ denote the input sequence, where $n = 0, 1, ..., N-1, N = 2^k, k \in \mathbb{N}$ (0 may be padded to guarantee $N = 2^k$), and let $\{X_n\}$ denote the sequence in the transform domain, then the Walsh-Hadamard Transform and its inverse can be computed as:

$$\mathbf{Y} = \sqrt{\frac{1}{N}}\mathbf{W}_k\mathbf{X}, \tag{1}$$

$$\mathbf{X} = \sqrt{\frac{1}{N}}\mathbf{W}_k\mathbf{Y}, \tag{2}$$

where, $\mathbf{Y} = [Y_0, ..., Y_{N-1}]^T$, $\mathbf{X} = [x_0, ..., x_{N-1}]^T$. $\mathbf{W}_k$ is called the Walsh matrix. Y_0 is the so-called the DC value and it is the sum of the input vector elements as in discrete cosine and Discrete Fourier Transforms (DCT and DFT), and $Y_{N-1} = (-1)^n x_n$, which is the $(\frac{N}{2}+1)$-th coefficient (the highest-frequency component) of the DFT for a real input vector. The WHT essentially applies N different filters to the input vector. The Walsh Transform and its inverse are actually identical if the forward transform is normalized by $\sqrt{\frac{1}{N}}$ as in Eq. (1).

The Walsh matrix $\mathbf{W}_k$ can be generated via the following steps: [32]:

1. Construct the Hadamard matrix $\mathbf{H}_k$:

$$\mathbf{H}_k = \begin{cases} 1, & k = 0, \\ \begin{bmatrix} \mathbf{H}_{k-1} & \mathbf{H}_{k-1} \\ \mathbf{H}_{k-1} & -\mathbf{H}_{k-1} \end{bmatrix}, & k > 0, \end{cases} \tag{3}$$

   Alternatively, for $k > 1$, $\mathbf{H}_k$ can also be computed using Kronecker product $\otimes$:

$$\mathbf{H}_k = \mathbf{H}_1 \otimes \mathbf{H}_{k-1}. \tag{4}$$

2. Shuffle the rows of $\mathbf{H}_k$ to obtain $\mathbf{W}_k$ by applying the bit-reversal permutation and the Gray-code permutation on row index.

Therefore, the Walsh matrix and the Hadamard matrix are both symmetric scaled-unitary matrices:

$$\mathbf{W}_k = \mathbf{W}_k^T, \tag{5}$$

$$\mathbf{H}_k = \mathbf{H}_k^T, \tag{6}$$

$$\mathbf{W}_k\mathbf{W}_k = \mathbf{H}_k\mathbf{H}_k = N\mathbf{I}_N, \tag{7}$$

where, $\mathbf{I}_N$ is an N by N identical matrix.

Similar to the FFT, the WHT can be computed using butterfly operations described in Eq. (1) in [33]. In this way, the complexity of the WHT is also $O(N \log_2 N)$. Moreover, WHT is more efficient than FFT because the operator only contains $+1$ and -1 instead of any complex exponential terms. Thus, there is no need to use complex arithmetic to implement the WHT.

2.2. WHT Layer for Video Clip Analysis

We process the video data in short-time windows. Suppose that we have an N image frames in a short-time window: $\mathbf{X} = \{\mathbf{X}_n\}$ where $n = 0, 1, ..., N-1$, with the image frame size H by W. Therefore, $\mathbf{X}_n \in \mathbb{R}^{H\times W}$ denotes the n-th frame. We first apply one-dimensional (1D) WHT along the temporal axis (along n) on $\mathbf{X}$. In this way, we convert the tensor into the transform domain along the time axis. Then, we feed the tensor into a scaling layer. The scaling layer is achieved by element-wisely multiplying the tensor with N trainable parameters. The scaling layer contains no bias term and its activation is the smooth-thresholding operation. The smooth-thresholding function is a modified version of the soft-thresholding operator used in wavelet domain denoising. The threshold parameters are learned during training. The smooth threshold removes the slow amplitude transform domain components. Finally, we apply another WHT along the temporal axis to convert the tensor back into the time domain.

The smooth-thresholding function is first defined in [26]:

$$y = \mathcal{ST}(x) = \tanh(x)(|x| - T)_+, \tag{8}$$

where, T is a trainable threshold, $()_+$ denotes the ReLU function [34] which is defined as:

$$y = (x)_+ = \max(x, 0). \tag{9}$$

Compared to soft-thresholding, which is commonly used in wavelet domain denoising algorithms [35, 36] and defined as

$$y = \text{sign}(x)(|x| - T)_+, \tag{10}$$

the smooth-thresholding can make the network converge to a higher accuracy with the same amount of parameters because the derivative is not just 1, 0, and -1 [26]:

$$\frac{\partial \mathcal{ST}(x)}{\partial T} = \begin{cases} -\tanh(x), & |x| > T \\ 0, & |x| \le T \end{cases} \tag{11}$$

We do not use the ReLU function because both positive and negative values in the transform domain are equally important.

In general, the WHT layer is summarized as follows:

$$\mathbf{Z} = \mathcal{W}(\mathcal{ST}(\mathbf{K} \cdot \mathcal{W}(\mathbf{X}))), \tag{12}$$

where, $\mathcal{W}()$ denotes the WHT along the temporal axis, and $\mathcal{ST}()$ denotes the smooth-thresholding function which makes the layer non-linear, $(\mathbf{K} \cdot \mathcal{W}(\mathbf{X}))$ is the scaling layer that takes the tensordot operation between the transform domain weights $\mathbf{K}$ and $\mathcal{W}(\mathbf{X})$. The WH transform does not have any adjustable weights. The scaling layer applies weights to transform domain coefficients.

Fig. 2 shows the block diagram of the WHT layer. Algorithm 1 describes the implementation of the WHT layer.

Algorithm 1 The WHT layer for video clip analysis

Input: Input tensor $\mathbf{X} \in \mathbb{R}^{B\times H\times W\times N}$, where B is the batch size, H and W are the frame size, and N is the length of the video clip.

Output: Output tensor $\mathbf{Z} \in \mathbb{R}^{B\times H\times W\times N}$

Find minimum $k \in \mathbb{N}$, s.t. $2^k \ge N$

$\hat{\mathbf{X}} = \text{pad}(\mathbf{X}, 2^k - N) \in \mathbb{R}^{B\times H\times W\times 2^k}$

$\mathbf{Y} = \text{WHT}(\hat{\mathbf{X}}) \in \mathbb{R}^{B\times H\times W\times 2^k}$

$\hat{\mathbf{Y}} = \text{ST}(\mathbf{K} \cdot \mathbf{Y}) \in \mathbb{R}^{B\times H\times W\times 2^k}, \mathbf{K} \in \mathbb{R}^{2^k}$

$\hat{\mathbf{Z}} = \text{WHT}(\hat{\mathbf{Y}}) \in \mathbb{R}^{B\times H\times W\times 2^k}$

$\mathbf{Z} = \hat{\mathbf{Z}}[:, :, : N]$

return $\mathbf{Z}$.

Comments: Function pad($\mathbf{A}, b$) pads b zeros on the last axis of tensor $\mathbf{A}$. WHT($\cdot$) is the normalized Walsh-Hadamard transform on the last axis. We do not pad any zeros in this paper because $N = 32$. ST($\cdot$) performs smooth-thresholding. Index follows Python's rule.

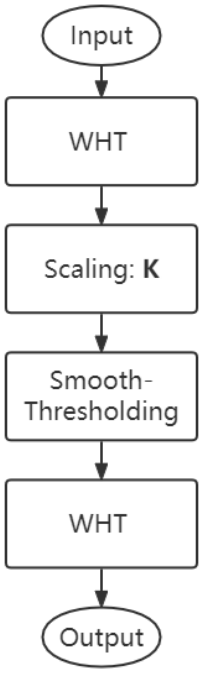


Figure 2. The WHT layer.

2.3. WHT in Inner Layers of ResNet

We use WHT not only in the input layer but also in inner layers of ResNet to process the tensor data. The original architecture of ResNet-18 is shown in Table 1. We introduce 1-D Walsh-Hadamard (WH) transforms before the "Conv1" layer as shown in Table 2. In addition, we introduce WHT layers described in Section 2.2 inside the resid-

ual blocks of ResNet as shown in Fig. 3. We apply the WH transform along the channel (last) axis to organize the activity among the channel axis according to their "frequency" components.

The original ResNet residual convolutional block is shown in Fig. 3a. Before the 3×3 2D convolutions, the WHT layer rearranges the channel data as shown in Fig. 3b. In the transform domain we use soft-thresholding to denoise the data. Finally, we batch normalize [37] the data after each WHT layer before feeding them to 3×3 Conv2D filters.

Table 1. Structure of ResNet-18.

Layer	Output Shape	Implementation Details
Conv1	$64 \times 64 \times 32$	Conv2D $3 \times 3, 32$
Conv2_x	$64 \times 64 \times 32$	$\begin{bmatrix} \text{Conv2D } 3 \times 3, 32 \\ \text{Conv2D } 3 \times 3, 32 \end{bmatrix} \times 2$
Conv3_x	$32 \times 32 \times 64$	$\begin{bmatrix} \text{Conv2D } 3 \times 3, 64 \\ \text{Conv2D } 3 \times 3, 64 \end{bmatrix} \times 2$
Conv4_x	$16 \times 16 \times 128$	$\begin{bmatrix} \text{Conv2D } 3 \times 3, 128 \\ \text{Conv2D } 3 \times 3, 128 \end{bmatrix} \times 2$
Conv5_x	$8 \times 8 \times 256$	$\begin{bmatrix} \text{Conv2D } 3 \times 3, 256 \\ \text{Conv2D } 3 \times 3, 256 \end{bmatrix} \times 2$
GAP	256	Global Average Pooling
Output	2	Dense(unit = 2, softmax)

2.4. ResNet-18 and ResNet-34 Architectures Used in Ember Cluster Detection

In this section, we describe the ResNet-18 and ResNet-34 with WHT layers used in ember cluster detection in IR video. The baseline regular ResNet-18 model is summarized in Table 1, and the regular ResNet-34 model is described in Table 3, respectively. As described in [38], if the output shape is different from the input shape, a 1×1 convolutional layer matches the sizes and performs the shortcut connection. The convolutional blocks include Conv3_1, Conv4_1, and Conv5_1 layers. Other convolutional blocks use the identity function ($a(x) = x$) to achieve the shortcut connection whenever the output shape is the same as the input shape. We apply a global average pooling (GAP) layer after the last convolutional residual block, and it is followed by a dense layer which is the output of the model. The dense layer contains two outputs and computes the softmax function. Weights in the convolutional layers and the dense layer are initialized using He normal initializer [39]. Batch normalization is applied after each 3×3 convolutional layer.

The ResNet-18 and ResNet-34 with the WHT layer are shown in Tables 2 and 4. As we mention in Section 2.3, they start with a WHT layer. The ResNet-18 with the WHT

Table 2. Structure of ResNet-18 with the WHT layer. We implement it by inserting one WHT layer before Conv1 and at the beginning of Conv2_x, Conv3_x, Conv4_x, Conv5_x. WHTL is the WHT layer.

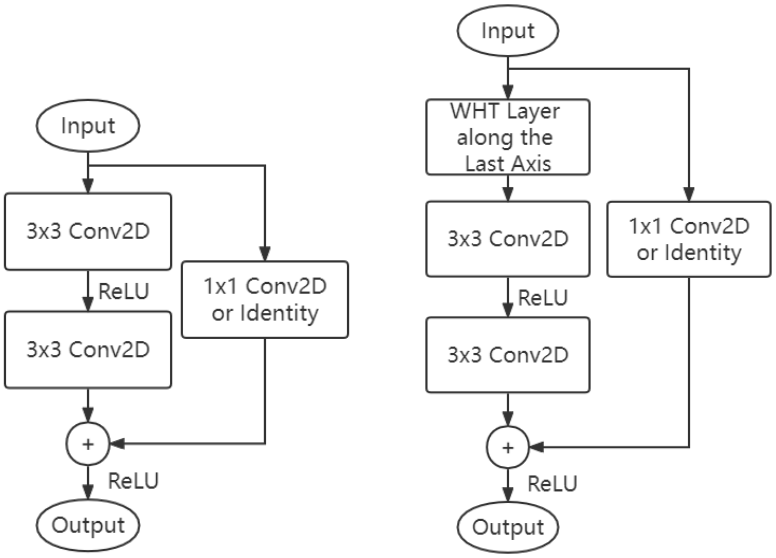
Layer	Output Shape	Implementation Details
WHT1	$64 \times 64 \times 32$	WHTL along last axis
Conv1	$64 \times 64 \times 32$	Conv2D $3 \times 3, 32$
Conv2_x	$64 \times 64 \times 32$	$\begin{bmatrix} \text{WHTL along last axis} \\ \text{Conv2D } 3 \times 3, 32 \\ \text{Conv2D } 3 \times 3, 32 \end{bmatrix} \times 2$
Conv3_x	$32 \times 32 \times 64$	$\begin{bmatrix} \text{WHTL along last axis} \\ \text{Conv2D } 3 \times 3, 64 \\ \text{Conv2D } 3 \times 3, 64 \end{bmatrix} \times 2$
Conv4_x	$16 \times 16 \times 128$	$\begin{bmatrix} \text{WHTL along last axis} \\ \text{Conv2D } 3 \times 3, 128 \\ \text{Conv2D } 3 \times 3, 128 \end{bmatrix} \times 2$
Conv5_x	$8 \times 8 \times 256$	$\begin{bmatrix} \text{WHTL along last axis} \\ \text{Conv2D } 3 \times 3, 256 \\ \text{Conv2D } 3 \times 3, 256 \end{bmatrix} \times 2$
GAP	256	Global Average Pooling
Output	2	Dense(unit = 2, softmax)

layer contains 8 convolutional residual blocks after the first convolutional layer, and the ResNet-34 with the WHT layer contains 16 convolutional residual blocks after the first convolutional layer. Each convolutional block contains two 3×3 convolutional layers and 1 WHT layer. Batch normalization is applied after each 3×3 convolutional layer and each WHT layer.

Table 3. Structure of ResNet-34.

Layer	Output Shape	Implementation Details
Conv1	$64 \times 64 \times 32$	Conv2D $3 \times 3, 32$
Conv2_x	$64 \times 64 \times 32$	$\begin{bmatrix} \text{Conv2D } 3 \times 3, 32 \\ \text{Conv2D } 3 \times 3, 32 \end{bmatrix} \times 3$
Conv3_x	$32 \times 32 \times 64$	$\begin{bmatrix} \text{Conv2D } 3 \times 3, 64 \\ \text{Conv2D } 3 \times 3, 64 \end{bmatrix} \times 4$
Conv4_x	$16 \times 16 \times 128$	$\begin{bmatrix} \text{Conv2D } 3 \times 3, 128 \\ \text{Conv2D } 3 \times 3, 128 \end{bmatrix} \times 6$
Conv5_x	$8 \times 8 \times 256$	$\begin{bmatrix} \text{Conv2D } 3 \times 3, 256 \\ \text{Conv2D } 3 \times 3, 256 \end{bmatrix} \times 3$
GAP	256	Global Average Pooling
Output	2	Dense(unit = 2, softmax)

Since we are feeding the network with gray-scale video



(a) ResNet residual block [38].

(b) ResNet residual block including a WHT layer that has a soft-thresholding nonlinearity as in Figure 2.

Figure 3. ResNet residual block by inserting a WHT layer at the beginning. Batch normalization is applied after each 3×3 convolutional layer and the WHT layer.

Table 4. Structure of ResNet-34 with the WHT layer. We implement it by inserting one WHT layer before Conv1 and at the beginning of Conv2_x, Conv3_x, Conv4_x, Conv5_x. WHTL is the WHT layer.

Layer	Output Shape	Implementation Details	
WHT1	$64 \times 64 \times 32$	WHTL along last axis	
Conv1	$64 \times 64 \times 32$	Conv2D $3 \times 3, 32$	
Conv2_x	$64 \times 64 \times 32$	WHTL along last axis Conv2D $3 \times 3, 32$ Conv2D $3 \times 3, 32$	$\times 3$
Conv3_x	$32 \times 32 \times 64$	WHTL along last axis Conv2D $3 \times 3, 64$ Conv2D $3 \times 3, 64$	$\times 4$
Conv4_x	$16 \times 16 \times 128$	WHTL along last axis Conv2D $3 \times 3, 128$ Conv2D $3 \times 3, 128$	$\times 6$
Conv5_x	$8 \times 8 \times 256$	WHTL along last axis Conv2D $3 \times 3, 256$ Conv2D $3 \times 3, 256$	$\times 3$
GAP	256	Global Average Pooling	
Output	2	Dense(unit = 2, softmax)	

clips, the input of the network is $64 \times 64 \times 32$. TensorFlow API "model.summary()" shows that the ResNet-18 with WHT layer totally contains 2,813,922 parameters and the ResNet-34 with WHT layer totally contains 5,353,954 parameters. Therefore, the 9 WHT layers with the batch normalization in ResNet-18 and the 17 WHT layers with the batch normalization in ResNet-34 only bring 0.16% (4,608) and 0.19% (10,176) additional parameters, respectively.

3. Experimental Results

We have four 720×480 infrared wildfire videos with a 30 frames-per-second capture rate. The imagery that provided input for this study was extracted from videos recorded using a FLIR Tau2 thermal imager (Wilsonville, OR USA). The imager was integrated into a remotely-operated gimbal system and flown aboard a DJI Matrice 600P unmanned aircraft system (SZ DJI Technology Co., Ltd., Shenzen, Guangdong, China). We conducted several flights over prescribed fires in frequently-burned, pine-dominated forest plots in Florida, USA, during October 2019 and February 2021. Altitudes ranged from 50m to 120m. A combination of moderate fire weather and low levels of fuel resulted in low-intensity fires, with short flame lengths and relatively low ember production compared with fires in areas with higher fuel loading or in different forest types. Ignition methods for these burns were primarily via hand (drip torch) and ATV-mounted torch, as is common throughout

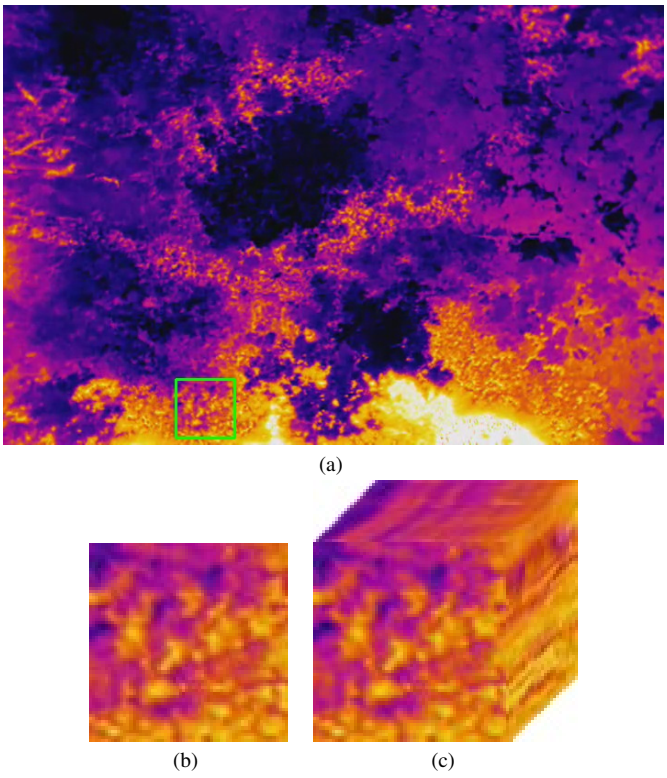


Figure 4. (a) An infrared video frame. Pseudo-color is applied for display purposes. We feed the neural network with the gray-scale video clips. Region with a green square bounding box appears to be an ember cluster. (b) The ember cluster region (with the green square shaped bounding box in (a)). (c) The video clip of the ember cluster consisting of 32 frames (about a second long). The size of the video clip is $64 \times 64 \times 32$.

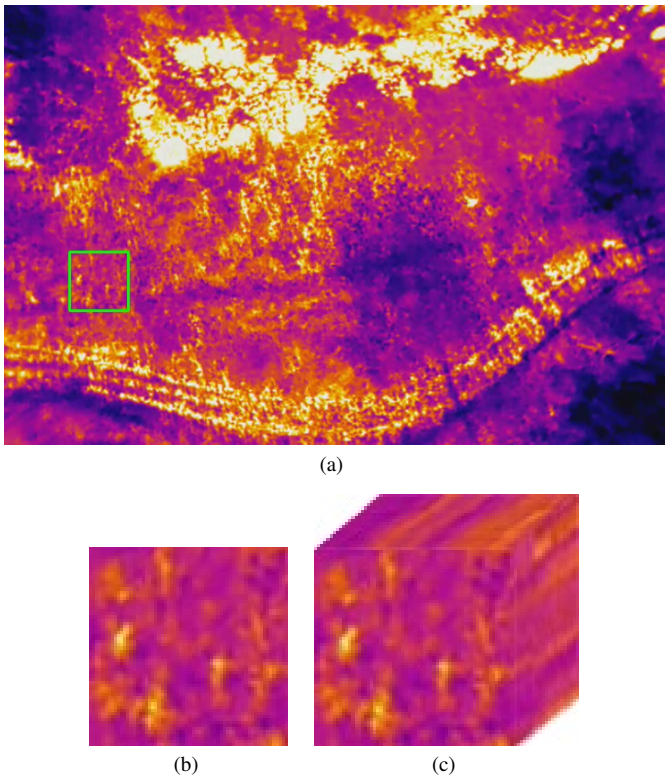


Figure 5. (a) An infrared video frame. Region with a green square bounding box contains no ember. (b) Details of the green square region in (a). (c) Temporal frames show that hot spots are not moving as in Fig. 4c. The size of the video clip is $64 \times 64 \times 32$.

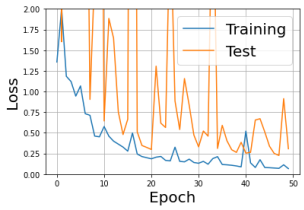
the region. Fig. 4a and Fig. 5a show two image frames from our collection of video frames. Pseudo-color is applied to the videos for better visualization.

We feed gray-scale data to the neural network, so the input data only has one color channel instead of three. We manually annotated the video clips containing moving ember clusters. We crop 64×64 windows around ember clusters as shown in Fig. 4b. We extract 32 temporal frames and make a small video clip of about a one-second-long video clip. We use 32 because this duration is long enough for our eyes to observe the movement of the ember, and we do not need to pad zeros in the WHT layer because 32 is an integer power of 2. The data size of the video clip is $64 \times 64 \times 32$ as shown in Fig. 4c. We feed $64 \times 64 \times 32$ to the neural network. Similarly, we randomly crop $64 \times 64 \times 32$ short-time windows that do not contain ember clusters. An example is shown in Fig 5c. We have generated 296 video clips with ember clusters and 676 video clips without ember clusters in our training dataset. We have also generated 248 video clips with ember clusters, and 438 video clips without ember clusters in our test dataset. We make the data unbalanced because we want the networks more likely to report the video clip containing no ember to reduce the false-alarm rate. After all, the high false-alarm rate may disturb the people in the forest fire department in a long surveillance task.

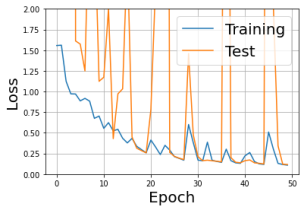
3.1. Training ResNets with the WHT Layer

We implemented ResNet-18 and ResNet-34 on an HP-Z820 workstation with 2 Intel Xeon E5-2695 v2 CPUs, 2 NVIDIA RTX A4000 GPUs, with 128GB RAM. The code is written in TensorFlow-Keras in Python 3.

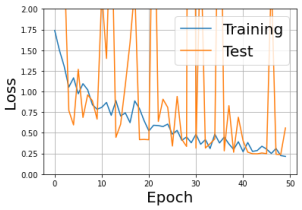
We train ResNet-18 with the WHT layer using the training dataset. The input size of this model is $64 \times 64 \times 32$. To compare our design, we also train two regular ResNet-18s. One is trained using only the first frame in each video clip, so its input size is $64 \times 64 \times 1$. The other one is trained using the video clip dataset, so its input size is $64 \times 64 \times 32$. The architectures of these two ResNet-18s are the same except in the first layer. We also studied the effect of scaling layer as a part of the WHT layer. Without the scaling layer, it is



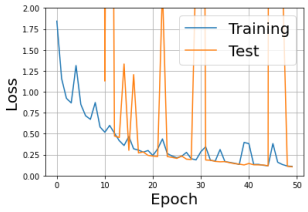
(a) Cross-entropy vs. epoch number of ResNet-18 that do not use the temporal information. The input is the first frame of the video



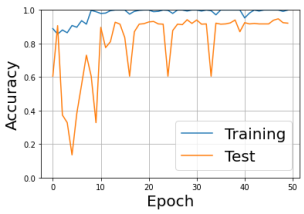
(b) Cross-entropy vs. epoch number of the regular ResNet-18. The input is video clips.



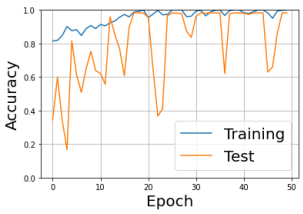
(c) Cross-entropy vs. epoch number of ResNet-18 with the WHT layer. Scaling is not applied in the WHT layer. The input is video clips.



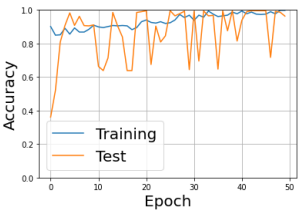
(d) Cross-entropy vs. epoch number of ResNet-18 that do not use the temporal information. Scaling is applied in the WHT layer. The input is the first frame of the video



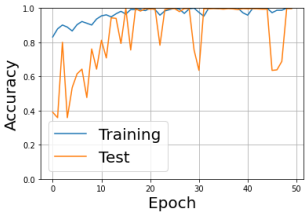
(e) Accuracy vs. epoch number of ResNet-18 that do not use the temporal information. The input is the first frame of the video



(f) Accuracy vs. epoch number of the regular ResNet-18. The input is video clips.



(g) Accuracy vs. epoch number of ResNet-18 with the WHT layer. Scaling is not applied in the WHT layer. The input is video clips.



(h) Accuracy vs. epoch number of ResNet-18 that do not use the temporal information. Scaling is applied in the WHT layer. The input is the first frame of the video

Figure 6. Cross-entropy and accuracy vs. epoch number of ResNet-18 models. The highest accuracy of each model is reported in Table 5.

Table 5. Accuracy, true positive rate (TPR) and false negative rate (FNR) values, and the number of parameters of the networks

Base Model	**WHT**	**Scaling in WHT**	**Input**	**Parameters**	**TPR**	**FNR**	**Accuracy**
ResNet-18	No	-	First Frame ($64 \times 64 \times 1$)	2,800,386	94.35%	5.02%	94.75%
ResNet-18	No	-	Video Clip ($64 \times 64 \times 32$)	2,809,314	95.16%	0.00%	98.25%
ResNet-18	Yes	No	Video Clip ($64 \times 64 \times 32$)	2,813,154	98.79%	0.00%	99.56%
ResNet-18	**Yes**	**Yes**	**Video clip ($64 \times 64 \times 32$)**	**2,813,922**	**99.19%**	**0.00%**	**99.71%**
ResNet-34	No	-	First Frame ($64 \times 64 \times 1$)	5,334,850	94.76%	3.88%	95.63%
ResNet-34	No	-	Video Clip ($64 \times 64 \times 32$)	5,343,778	95.56%	0.00%	98.40%
ResNet-34	Yes	No	Video Clip ($64 \times 64 \times 32$)	5,352,258	99.19%	0.00%	99.71%
ResNet-34	Yes	Yes	Video Clip ($64 \times 64 \times 32$)	5,353,954	99.19%	0.00%	99.71%

the same as the WHT layer proposed in [20]:

$$\mathbf{Z} = \mathcal{W}(\mathcal{ST}(\mathcal{W}(\mathbf{X}))), \tag{13}$$

though the WHT layer in [20] is employed only to replace the 1×1 convolutions layers to reduce the number of parameters in image classification tasks. This approach reduces the number of parameters but it leads to a slight loss in TPR rate in ResNet-18 as shown in Table 5. We also repeat the above experiment using ResNet-34. In ResNet-34 there is no need to have a scaling layer but the number of parameters of ResNet-34 is already twice that of ResNet-18.

We train all networks with the same parameter setting: RMSprop optimizer [40] with learning rate = 0.001 on 50 epochs with the batch size = 32. We use cross-entropy as the loss function. In Table 5, accuracy on the test dataset is used for model evaluation. TPR (True positive rate, true detected rate) and FNR (false negative rate, false alarm rate) of each model are provided in Table 5:

$$TPR = \frac{\text{Number of true positive cases}}{\text{Total number of positive cases}}, \tag{14}$$

and

$$FNR = \frac{\text{Number of false negative cases}}{\text{Total number of negative cases}}, \tag{15}$$

Table 6. Confusion matrix of ResNet-18 with the WHT layer. Scaling layer is part of the WHT layer. We totally have 248 video clips of size of size $64 \times 64 \times 32$ containing ember clusters and 438 regular video clips that do not contain ember clusters in the test set.

Actual \ Predicted	Ember	No Ember
Ember	246	2
No Ember	0	438

Table 7. Confusion matrix of ResNet-18 with the WHT layer. Scaling layer is not applied in the WHT layer. The input is video clips of size $64 \times 64 \times 32$.

Actual \ Predicted	Ember	No Ember
Ember	245	3
No Ember	0	438

Table 8. Confusion matrix of ResNet-18 without using the temporal information. The input is the first frame of size 64×64 of the video clips of the test dataset.

Actual \ Predicted	Ember	No Ember
Ember	234	14
No Ember	22	416

Table 9. Confusion matrix of ResNet-18. The input is video clips of size $64 \times 64 \times 32$.

Actual \ Predicted	Ember	No Ember
Ember	236	12
No Ember	0	438

Plots of cross-entropy and accuracy versus epoch number are shown in Figure 6. Confusion matrices of all ResNet-18 models are shown in Tables 6, 7, 8 and 9. Video-based models report no false-alarm case, but the single-frame-based model reports 22 false-alarm cases in our test set. The proposed ResNet-18 with WHT layer achieves the best TPR and accuracy as shown in Table 5. The ResNet-18 models whose inputs are video clips reach a higher accuracy than the model whose input is a single image frame. This is because ember flickering and motion can be observed only in the video clips. It cannot be observed in a single frame. That is why the FNR of the single-frame model is 5.02 %. The model with the WHT layer has a 4.03% higher TPR than the model without the WHT layer in ResNet-18 model. This is because our WHT layer enhances the neural network's ability to extract the information from the temporal domain. The smooth thresholding in the transform domain also eliminates small variations and noise in the video. Scaling layer as a part of the WHT layer brings more parameters and this improves the accuracy 0.15% with a slight increase in the number of parameters. The WHT transform does not have any adjustable weights the scaling layer applies weights to the transform coefficients.

When we make the network deeper to ResNet-34, we notice that the accuracy of the model with WHT does not increase. We think the accuracy is bounded by the amount and quality of the dataset. However, ResNet-34 contains twice that of the parameters of ResNet-18. Since the additional layers will increase the computational cost, we choose the ResNet-18 with the WHT layer as the proposed model fro ember cluster recognition in infrared video. Moreover, in this case, scaling in the WHT layer does not contribute the accuracy because the accuracy is already very high.

4. Conclusion

In this paper, we proposed an ember cluster detection paper using a deep neural network with a novel layer based on the Walsh-Hadamard Transform (WHT). We incorporated the WHT layer into ResNet-18 for the ember cluster detection task in IR video. In this task, the WHT layer models the high-frequency activity caused by ember movements. Compared to the regular ResNet-18 models, the proposed model has 4% higher TPR rate and a 1.46% higher accuracy with a slight increase in the number of parameters. Making the model deeper did not improve the accuracy in our dataset. It is necessary to use video data to recognize ember movements. The WHT layer successfully models the temporal dimension of the video data.

References

[1] USGCRP. Impacts, risks, and adaptation in the united states: Fourth national climate assessment. *US Global Change Research Program*, 2, 2018. 1

[2] NIST. *New Timeline of Deadliest California Wildfire Could Guide Lifesaving Research and Action*, February 2021. https://www.nist.gov/news-events/news/2021/02/new-timeline-deadliest-california-wildfire-could-guide-lifesaving-research. 1

[3] The conversation. *Australia's massive fires could become routine, climate scientists warn*, January 2020. https://theconversation.com/australias-black-

summer-of-fire-was-not-normal-and-we-can-prove-it-172506. 1

[4] B Uğur Töreyin, Yiğithan Dedeoğlu, and A Enis Cetin. Wavelet based real-time smoke detection in video. In *2005 13th European signal processing conference*, pages 1–4. IEEE, 2005. 1

[5] B Uğur Töreyin, Yiğithan Dedeoğlu, Uğur Güdükbay, and A Enis Cetin. Computer vision based method for real-time fire and flame detection. *Pattern recognition letters*, 27(1):49–58, 2006. 1

[6] Y Hakan Habiboglu, Osman Gunay, and A Enis Cetin. Real-time wildfire detection using correlation descriptors. In *2011 19th European Signal Processing Conference*, pages 894–898. IEEE, 2011. 1

[7] Yusuf Hakan Habiboğlu, Osman Günay, and A Enis Çetin. Covariance matrix-based fire and flame detection method in video. *Machine Vision and Applications*, 23(6):1103–1113, 2012. 1

[8] Paulo Vinicius Koerich Borges and Ebroul Izquierdo. A probabilistic approach for vision-based fire detection in videos. *IEEE transactions on circuits and systems for video technology*, 20(5):721–731, 2010. 1

[9] Turgay Çelik, Hüseyin Özkaramanlı, and Hasan Demirel. Fire and smoke detection without sensors: Image processing based approach. In *2007 15th European Signal Processing Conference*, pages 1794–1798. IEEE, 2007. 1

[10] Turgay Celik and Hasan Demirel. Fire detection in video sequences using a generic color model. *Fire safety journal*, 44(2):147–158, 2009. 1

[11] Feiniu Yuan. A fast accumulative motion orientation model based on integral image for video smoke detection. *Pattern Recognition Letters*, 29(7):925–932, 2008. 1

[12] A Enis Cetin, Bart Merci, Osman Günay, Behçet Ugur Töreyin, and Steven Verstockt. *Methods and techniques for fire detection: signal, image and video processing perspectives*. Academic Press, 2016. 1

[13] Renjie Xu, Haifeng Lin, Kangjie Lu, Lin Cao, and Yunfei Liu. A forest fire detection system based on ensemble learning. *Forests*, 12(2):217, 2021. 1, 2

[14] Yakhyokhuja Valikhujaev, Akmalbek Abdusalomov, and Young Im Cho. Automatic fire and smoke detection method for surveillance systems based on dilated cnns. *Atmosphere*, 11(11):1241, 2020. 1, 2

[15] Behçet Uğur Töreyin. Smoke detection in compressed video. In *Applications of Digital Image Processing XLI*, volume 10752, page 1075232. International Society for Optics and Photonics, 2018. 1, 2

[16] Süleyman Aslan, Uğur Güdükbay, B Uğur Töreyin, and A Enis Çetin. Early wildfire smoke detection based on motion-based geometric image transformation and deep convolutional generative adversarial networks. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8315–8319. IEEE, 2019. 1, 2

[17] Minsoo Park, Dai Quoc Tran, Daekyo Jung, Seunghee Park, et al. Wildfire-detection method using densenet and cyclegan data augmentation-based remote camera imagery. *Remote Sensing*, 12(22):3715, 2020. 1, 2

[18] Hongyi Pan, Diaa Badawi, Xi Zhang, and Ahmet Enis Cetin. Additive neural network for forest fire detection. *Signal, Image and Video Processing*, pages 1–8, 2019. 1, 2

[19] Hongyi Pan, Diaa Badawi, and Ahmet Enis Cetin. Computationally efficient wildfire detection method using a deep convolutional network pruned via fourier analysis. *Sensors*, 20(10):2891, 2020. 1, 2

[20] Hongyi Pan, Diaa Badawi, and Ahmet Enis Cetin. Fourier domain pruning of mobilenet-v2 with application to video based wildfire detection. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 1015–1022. IEEE, 2021. 1, 2, 7

[21] National Geographic. *Meet the wildfire superspreaders*, October 2020. https://www.nationalgeographic.com/science/article/meet-the-wildfire-superspreaders-embers. 1

[22] WSRB. *Embers, Wind and Fire: A Dangerous Mix*, August 2021. https://www1.wsrb.com/blog/embers-wind-and-fire-a-dangerous-mix. 1

[23] James Lee-Thorp, Joshua Ainslie, Ilya Eckstein, and Santiago Ontanon. Fnet: Mixing tokens with fourier transforms. *arXiv preprint arXiv:2105.03824*, 2021. 2

[24] T Ceren Deveci, Serdar Cakir, and A Enis Cetin. Energy efficient hadamard neural networks. *arXiv preprint arXiv:1805.05421*, 2018. 2

[25] Ritchie Zhao, Yuwei Hu, Jordan Dotzel, Christopher De Sa, and Zhiru Zhang. Building efficient deep neural networks with unitary group convolutions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11303–11312, 2019. 2

[26] Hongyi Pan, Diaa Badawi, and Ahmet Enis Cetin. Fast walsh-hadamard transform and smooth-thresholding based binary layers in deep neural networks. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 4645–4654. IEEE, 2021. 2, 3

[27] Hongyi Pan, Diaa Badawi, and Ahmet Enis Cetin. Block walsh-hadamard transform based binary layers in deep neural networks. *ACM Transactions on Embedded Computing Systems (TECS)*, 2022. 2

[28] John Guibas, Morteza Mardani, Zongyi Li, Andrew Tao, Anima Anandkumar, and Bryan Catanzaro. Adaptive fourier neural operators: Efficient token mixers for transformers. *arXiv preprint arXiv:2111.13587*, 2021. 2

[29] Matej Ulicny, Vladimir A Krylov, and Rozenn Dahyot. Harmonic networks with limited training samples. In *2019 27th European Signal Processing Conference (EUSIPCO)*, pages 1–5. IEEE, 2019. 2

[30] Matej Ulicny, Vladimir A Krylov, and Rozenn Dahyot. Harmonic convolutional networks based on discrete cosine transform. *arXiv preprint arXiv:2001.06570*, 2020. 2

[31] A Enis Cetin, Omer N Gerek, and Sennur Ulukus. Block wavelet transforms for image coding. *IEEE Transactions on Circuits and Systems for Video Technology*, 3(6):433–435, 1993. 2

[32] Joseph L Walsh. A closed set of normal orthogonal functions. *American Journal of Mathematics*, 45(1):5–24, 1923. 2

[33] Bernard J. Fino and V. Ralph Algazi. Unified matrix treatment of the fast walsh-hadamard transform. *IEEE Transactions on Computers*, 25(11):1142–1146, 1976. 2

[34] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Icml*, 2010. 3

[35] PM Agante and JP Marques De Sá. Ecg noise filtering using wavelets with soft-thresholding methods. In *Computers in Cardiology 1999. Vol. 26 (Cat. No. 99CH37004)*, pages 535–538. IEEE, 1999. 3

[36] David L Donoho. De-noising by soft-thresholding. *IEEE transactions on information theory*, 41(3):613–627, 1995. 3

[37] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015. 4

[38] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 4, 5

[39] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015. 4

[40] Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. *Cited on*, 14(8):2, 2012. 7