electronics

MDPI

Article

Combustion Instability Monitoring through Deep-Learning-Based Classification of Sequential High-Speed Flame Images

Ouk Choi [1], Jongwun Choi [2], Namkeun Kim [2] and Min Chul Lee [3,4,*]

[1] Department of Electronics Engineering, Incheon National University, Incheon 22012, Korea; ouk.choi@inu.ac.kr
[2] Department of Mechanical Engineering, Incheon National University, Incheon 22012, Korea; whddnssnla@naver.com (J.C.); nkim@inu.ac.kr (N.K.)
[3] Department of Safety Engineering, Incheon National University, Incheon 22012, Korea
[4] Fire Disaster Prevention Research Center, Incheon National University, Incheon 22012, Korea
* Correspondence: LMC@inu.ac.kr; Tel.: +82-32-835-8295

Received: 27 April 2020; Accepted: 18 May 2020; Published: 20 May 2020

Abstract: In this study, novel deep learning models based on high-speed flame images are proposed to diagnose the combustion instability of a gas turbine. Two different network layers that can be combined with any existing backbone network are established—(1) An early-fusion layer that can learn to extract the power spectral density of subsequent image frames, which is time-invariant under certain conditions. (2) A late-fusion layer which combines the outputs of a backbone network at different time steps to predict the current combustion state. The performance of the proposed models is validated by the dataset of high speed flame images, which have been obtained in a gas turbine combustor during the transient process from stable condition to unstable condition and vice versa. Excellent performance is achieved for all test cases with high accuracy of 95.1%–98.6% and a short processing time of 5.2–12.2 ms. Interestingly, simply increasing the number of input images is as competitive as combining the proposed early-fusion layer to a backbone network. In addition, using handcrafted weights for the late-fusion layer is shown to be more effective than using learned weights. From the results, the best combination is selected as the ResNet-18 model combined with our proposed fusion layers over 16 time-steps. The proposed deep learning method is proven as a potential tool for combustion instability identification and expected to be a promising tool for combustion instability prediction as well.

Keywords: combustion instability; flame imaging; deep learning; residual network; power spectral density; temporal smoothing

1. Introduction

Combustion instability (CI) causes large oscillations of dynamic pressure and heat release, which can damage hot gas path components such as a combustor itself and turbine blades and vanes located at the downstream. Thus, CI has been one of the main issues for many combustion engines including gas turbines and internal engines. Although tremendous efforts have been exerted to notify and characterize the CI mechanisms [1–5] and to monitor and control the CI [6–11], the remorseless reinforcement of NOx emission regulation has made gas turbines to operate in ultra-lean premixed condition. This premixed combustion near the blowout limit makes safe operation of gas turbines uneasy by leading flame to oscillate, and, thus, amplifying CI [12,13]. However, the good news is that we are in the era of the 4th generation revolution, and a new paradigm of CI monitoring technique is being applied with the advent of advanced machine learning techniques.

Especially, with the advance of optical sensing and digital image processing methods, digital imaging devices have been employed in on-line combustion-state monitoring systems [14–23]. For example, Lu et al. extracted from flame images a variety of handcrafted features such as the ignition point and area, and the flickering of flame [14]. The variations of the features with furnace load were discovered in their work. Sun et al. defined a simple universal index that assesses the stability of flame in terms of color, geometry, and luminance [17]. The feasibility of oscillation frequency as an indicator of flame stability was also studied. Chen et al. proposed a unique processing algorithm based on multiway principal component analysis (MPCA) and a hidden Markov model (HMM) for monitoring of the combustion state in an industrial furnace [15,16]. From the MPCA-HMM analysis, spatial and temporal features of dynamic flame images were extracted and the feasibility of their model was verified by successfully detecting fault situations in a time-series control chart. Bai et al. segmented a flame image into premixed and diffused regions in which color and texture features were extracted [19]. They applied principal component analysis (PCA) to the features and then used a kernel support vector machine to identify the operation condition.

The advances in this digital image processing technique are accelerated by combining with machine learning methods, which can learn important features automatically. Thus, deep-learning-based methods of machine learning have recently been used for combustion state monitoring and identification of combustion instability onset [18,20–23]. Wang et al. proposed a convolutional neural network (CNN) to monitor furnace combustion states and predict the heat release rate [20]. The CNN showed higher accuracy than traditional methods based on handcrafted features and learned classifiers. Liu et al. proposed a deep belief network model (BNM) which can simply construct the complex nonlinear relationship between the flame images and the outlet oxygen content [21]. Their model was compared with a conventional linear feature extraction method of PCA and better performance of this model was verified through the demonstration of on-site tests in a real combustion system by proving not only the possibility of nonlinear feature extraction but also the higher accuracy of prediction with 5.543% relative error. They used a supervised learning framework to regress the oxygen content of combustion systems, which requires a large number of labeled images.

These labels are often given by manual labor. To reduce the laborious task, Akintayo et al. proposed a semi-supervised learning approach in which only stable flame images are assumed to be labeled [18]. A convolutional selective autoencoder was used to learn flame features, and the statistics of stable flame features were used to detect unstable flames. To avoid manual labeling, Qiu et al. used a convolutional autoencoder and PCA for feature extraction and dimension reduction [22]. An HMM was then used to model the transition probability between states and the conditional probability of features given a state. Han et al. proposed an unsupervised learning-based approach. In their approach, flame images are transformed into features by a stacked sparse autoencoder [23]. The training labels are then automatically determined by clustering the features and applying statistical analysis to handcrafted features [14]. These approaches are useful if the dataset consists only of images [22,23] or the labels of the images are partially known [18]; however, the approaches are prone to error if the unsupervised or semi-supervised features are not well separated across different combustion states.

In this paper, we propose a deep supervised learning-based method for combustion state prediction. An abundance of training data is obtained without manual labor by synchronized high-speed flame imaging and dynamic pressure measurement. A single model is trained to deal with multi-mode flame image sequences whose scale is unprecedented. Unlike previous network architectures [18,20–23] that take single images as input, our proposed method takes subsequent image frames and predicts the combustion state of the last frame. We propose two different network layers that can be combined with any backbone network to improve the performance. Our proposed early-fusion layer can learn to extract the power spectral density of subsequent image frames, which is time-invariant under certain conditions. Instead of modeling complicated temporal behavior of flames using a transition probability across neighboring time steps [15,22], this study proposes a late-fusion layer that can combine long-range outputs effectively to maximize the classification accuracy. The best

combination of our proposed layers with our adopted backbone network was determined through the validation test of which dataset is obtained from a gas turbine combustion experiment for six cases to ensure the robustness of prediction for various kinds of combustion instability.

2. ResNet-Based Models for Sequential Flame Images

Our proposed method takes subsequent flame image frames as input and produces the combustion state of the last frame. ResNet [24], which is the champion architecture that won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2015, serves as a backbone network. The proposed model architecture is composed of two additional sequential layers. The early-fusion layer provides the ResNet with input features extracted from the image frames so that the ResNet will extract deeper features and predict the combustion state better. The late-fusion layer combines the ResNet outputs at different time steps based on the assumption that previous outputs can help in improving the current output.

Figure 1 shows the proposed model architecture including the early-fusion layer, the ResNet, and the late-fusion layer. The input to the early-fusion layer is a sequence of T subsequent image frames and the output is a feature map with $4T$ channels. In this paper, we vary T from 1 to 16. Based on the feature map, the ResNet provides the probabilities of three different combustion states: stable, transient, and unstable states. The probabilities at the current time step τ as well as those at previous time steps are used as input to the late-fusion layer to obtain the final state probabilities. Figure 1 depicts a model with two previous times steps $\tau - 1$ and $\tau - 2$. The combustion state at τ is then determined as the one maximizing the probability from the late-fusion layer. The following subsections describe more details of the model.

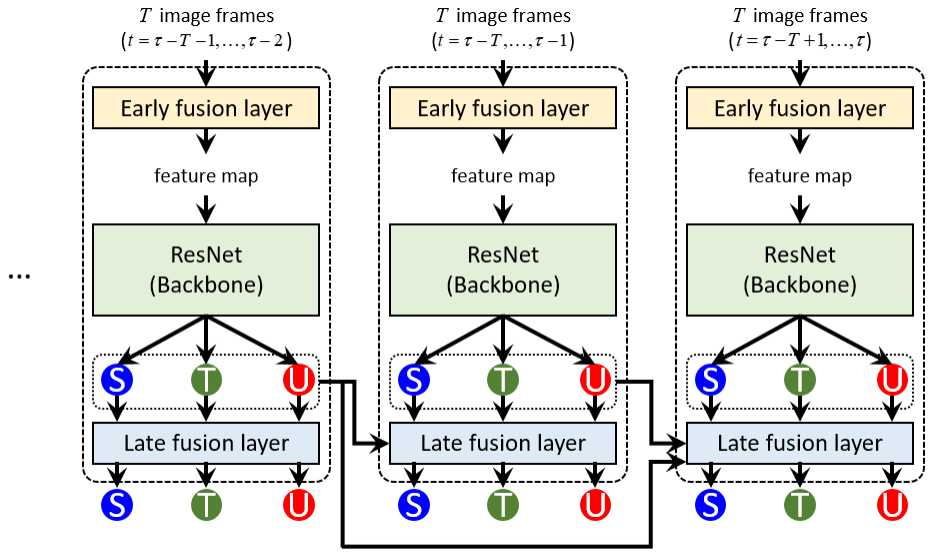

Figure 1. Proposed model architecture. The colored circles represent probabilities corresponding to three different combustion states: stable, transient, and unstable states colored in blue, green, and red, respectively. Please refer to the text for more detail. Best viewed in color.

2.1. Convolutional Neural Network

Before introducing the details of the proposed model, an understanding of the convolutional neural network and ResNet is necessary since they are used for the model configuration. Convolution has been widely used to extract image features such as edges [25], corners [26], and affine covariant regions [27]. Since an image is represented by a 2D grid of pixels, 2D convolution is performed on an

image by applying a linear filter to all pixels. The filter size typically ranges from 1×1 pixels to 7×7 pixels. For image I with C channels, the 2D convolution operation is defined as follows:

$$H_F(i,j) = \sum_{c=0}^{C-1} \sum_{u=-U}^{U} \sum_{v=-V}^{V} W(c,u,v) \cdot I(c, i+u, j+v) + b, \tag{1}$$

where (i,j) is the pixel coordinate vector and c is the channel index. $I(c, i+u, j+v)$ is the intensity of I at $(c, i+u, j+v)$. The filter size is $(2U+1) \times (2V+1)$ pixels and W is the filter weight. Finally, $H_F(i,j)$ is the filter response, or the feature, at pixel location (i,j). In the traditional feature extraction methods [25–27], handcrafted filter weights have been used along with a zero bias term $b = 0$. In deep learning, both W and b are learned from data.

Different filters can be applied to obtain a feature map with multiple channels. For example, we can apply K filters to an image to obtain feature map H_F with K channels. In this case, Equation (1) can be rewritten as follows:

$$H_F(k,i,j) = \sum_{c=0}^{C-1} \sum_{u=-U}^{U} \sum_{v=-V}^{V} W(k,c,u,v) \cdot I(c, i+u, j+v) + b(k), \tag{2}$$

where k is the filter index ranging from 0 to $K-1$. Considering a feature map X as an image, we can apply 2D convolution to X to obtain another feature map Y.

To reduce redundant features, the filters can be applied to pixels sampled with strides greater than 1. Or otherwise, the strongest features in small neighborhoods may be pooled, which will also reduce the feature map size. This operation is called max-pooling. With the reduced memory burden, the network can grow deeper.

Training a deep neural network changes the parameters of all layers. With a large learning rate, the distribution of each layer's input greatly changes, complicating the training. Thus, the learning rate needs to be small, leading to a long training time. With batch normalization [28], the input distribution is stabilized and the learning rate can be increased to reduce the training time. For a mini-batch with M training samples, batch normalization whitens each channel $H_F(k)$ using the mean $\mathrm{E}[H_F(k)]$ and variance $\mathrm{Var}[H_F(k)]$ over the mini-batch, as follows:

$$H_N(k,i,j) = \frac{H_F(k,i,j) - \mathrm{E}[H_F(k)]}{\sqrt{\epsilon + \mathrm{Var}[H_F(k)]}}, \tag{3}$$

where ϵ is a small positive number preventing the denominator from becoming zero. By the whitening, the mean and variance of $H_N(k)$ becomes 0 and 1, respectively, resulting in unwanted loss of representation power. To recover the representation power, the following linear transformation, with learnable parameters γ and β, is applied.

$$H_{BN}(k,i,j) = \gamma \cdot H_N(k,i,j) + \beta. \tag{4}$$

When using batch normalization, the bias term b or $b(k)$ is often omitted because β replaces the role of the bias term.

Applying linear filters repeatedly to an image is the same as applying their composite linear filter. Thus, such a pure deep convolutional network can hardly learn nonlinearity in data. To inject nonlinearity, the following rectified linear unit (ReLU) function is applied to the linear features.

$$H(k,i,j) = \max(0, H_{BN}(k,i,j)). \tag{5}$$

The ReLU function suffers less from the vanishing gradient problem [29], so it is helpful for increasing the depth of neural networks. In recent neural network architectures, convolution blocks

are typically defined by a series of convolution, batch normalization, and ReLU, as we reviewed in this subsection.

2.2. ResNet

The depth or the number of sequential layers of convolutional neural networks is of crucial importance in increasing the performance. However, adding more layers to a suitably deep model leads to higher training error. He et al. showed that this degradation problem is not caused by overfitting; they proposed convolution blocks with shortcut connections to reduce the degradation problem [24].

Let us denote a feature map by X and an underlying mapping to be fit by a few stacked layers by $\mathcal{H}(X)$. In residual learning, the residual mapping $\mathcal{F}(X)$ such that $\mathcal{F}(X) = \mathcal{H}(X) - X$ is learned, and $\mathcal{H}$ is recovered by

$$Y = \mathcal{H}(X) = \mathcal{F}(X) + X, \tag{6}$$

where Y is the output feature map of the mapping. If all the weights of $\mathcal{F}(X)$ are zero, then $\mathcal{H}(X)$ becomes an identity mapping such that

$$Y = X. \tag{7}$$

The identity mapping is implemented by using a shortcut connection, which directly connects the input to the output as shown in Figure 2. He et al. showed that it is easier to optimize a deep convolutional neural network based on building blocks with shortcut connections [24].

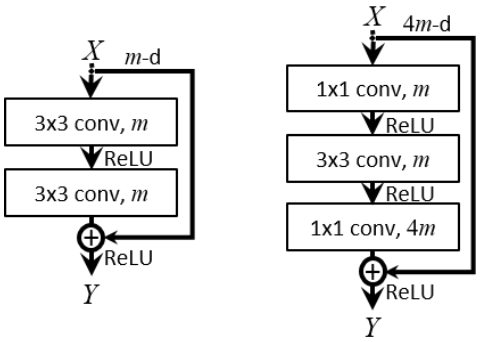

Figure 2. Examples of convolution blocks with shortcut connections [24]. (**Left**): basic building block for ResNet-18. (**Right**): bottleneck building block for ResNet-50. m and $4m$ are the number of filters. We note that every convolution layer is followed by batch normalization.

Based on the building blocks in Figure 2, He et al. proposed several network architectures ranging from ResNet-18 to ResNet-152. Among those we are interested in ResNet-18 and ResNet-50. ResNet-18 and ResNet-50 are based on basic blocks and bottleneck blocks, respectively. ResNet-50 showed higher accuracy than ResNets based on basic blocks (ResNet-18 and ResNet-34) by a considerable margin in the ILSVRC [24]. ResNet-50 also has an affordable training time for limited computational resources.

Figure 3 shows ResNet-18 and ResNet-50 architectures. For both architectures, the input image has a size of 224×224 pixels. 7×7 convolution with a stride of 2 is applied to the input image, resulting in a feature map with a size of 112×112 pixels. The feature map is then max-pooled to a size of 56×56 pixels. To the max-pooled feature map, a series of convolution blocks are applied. When the strided convolution is applied to a feature map, "/2" is used to note that. In each block, only the first 3×3 convolution has a stride of 2. In this case, the shortcut connection is replaced with 1×1 convolution with a stride of 2 to match the reduced size and increased dimension. The final feature map size is 7×7 pixels. The numbers of channels are 512 and 2048 for ResNet-18 and ResNet-50, respectively. The final feature map is average-pooled to obtain a 512- or 2048-dimensional vector, that is, each 7×7 channel is reduced to 1×1 by computing the average of the channel. For the ILSVRC, a 1000-dimensional fully connected layer is connected to the feature vector to output one of 1000 class

objects. In this paper, a 3-dimensional fully connected layer (fc 3 in Figure 3) is used to output one of the three combustion states.

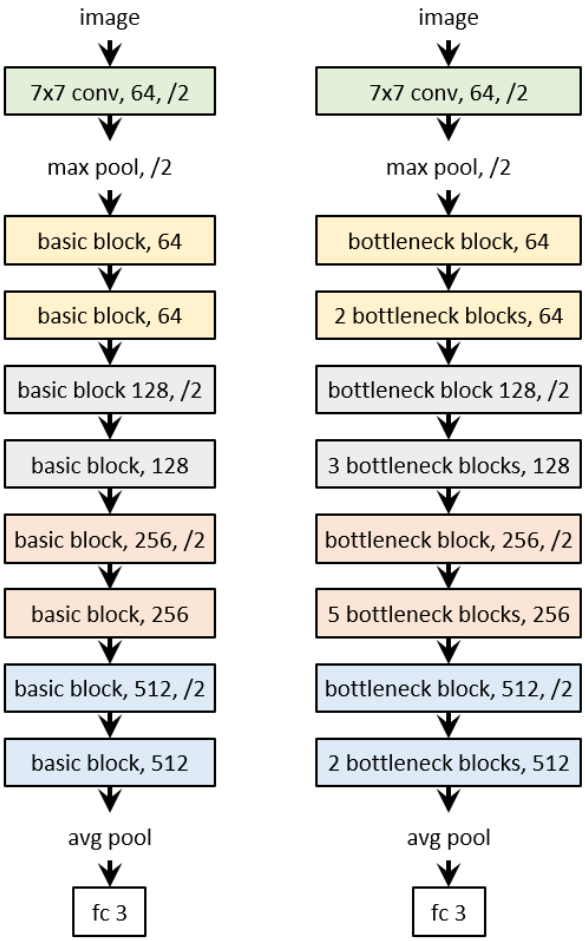

Figure 3. ResNet-18 and ResNet-50 architectures. (**Left**): ResNet-18. (**Right**): ResNet-50. The number to the right of each "block"is the number of filters m. "/2" represents that the feature map height and width are halved by convolution with a stride of 2 or by max-pooling. In every block with "/2", only the first 3×3 convolution has a stride of 2. Refer to the text for more detail.

To model the probability of each state, the softmax function is applied to the output of the fully connected layer. Denoting the set of states by $\mathcal{S} = \{\text{stable}, \text{transient}, \text{unstable}\}$, the softmax output is computed as

$$y(s) = P(s|I) = \text{softmax}(o(s)) = \frac{\exp(o(s))}{\sum_{r \in \mathcal{S}} \exp(o(r))}, \tag{8}$$

where $o(s)$ is the output of the fully connected layer for state s. The softmax output can be denoted as vector $\mathbf{y}$, whose components are $y(s)$ for $s \in \mathcal{S}$. Finally, the state is determined by finding s that maximizes $y(s)$.

2.3. Early Fusion

The simplest way to use subsequent image frames as input to a ResNet is to concatenate the image frames into a single multichannel image J. Denoting the image at time step t by $I^{(t)}$, we concatenate T images for $t = \tau - T + 1, \ldots, \tau$ as

$$J^{(\tau)}(c) = I^{(c+\tau-T+1)} \tag{9}$$

for $c = 0, \ldots, T-1$. Because our flame images are of a single channel, each time step corresponds to a channel in $J^{(\tau)}$.

In this paper, we vary T from 1 to 16. When $T = 1$, the output is the same as using a ResNet for a single image $I^{(\tau)}$. This will be our baseline method. When $T > 1$, our task becomes to infer the state at τ using the images from $I^{(\tau-T+1)}$ to $I^{(\tau)}$.

When combustion instability occurs, flame images rapidly change with time. It has been discovered that the change is periodic due to the periodical evolution and decaying of heat release, which is derived from the periodic oscillation velocity or equivalence of mixture and also affected by the periodic fluctuation of dynamic pressure inside the combustion chamber [5,10]. Based on the periodicity, one may want to transform the time-domain representation to a frequency-domain representation. For example, we can apply the discrete Fourier transform [30] at each pixel location as follows:

$$\begin{aligned} Re(k,i,j) &= \sum_{c=0}^{T-1} \cos(2\pi kc/T) \cdot J^{(\tau)}(c,i,j), \\ Im(k,i,j) &= -\sum_{c=0}^{T-1} \sin(2\pi kc/T) \cdot J^{(\tau)}(c,i,j), \end{aligned} \tag{10}$$

where Re and Im are the real and imaginary components of the frequency-domain signal.

In Equation (10), each k corresponds to frequency $2\pi kcf/T$, where f is the image sampling frequency, for example, in our work 8 kHz. We can compute the power spectral density (PS) for the kth frequency as

$$PS(k,i,j) = Re^2(k,i,j) + Im^2(k,i,j). \tag{11}$$

The power spectral density is time-invariant as long as the period of the image signal coincides with T [30]. Thus, training of a ResNet can be more stable with such an invariant input.

Because the period depends on various factors such as fuel composition or heat input, it may differ from sequence to sequence and from state to state. Besides, applying different T to different sequences will reduce the generality of the proposed method. In this paper we fix T for all sequences and learn the transform from data.

The computations in Equation (10) are both in the form of

$$F(k,i,j) = \sum_{c=0}^{T-1} W(k,c) \cdot J^{(\tau)}(c,i,j), \tag{12}$$

which is 1×1 convolution with T filters on $J^{(\tau)}$.

Let us assume that the exact period T_{exact} is less than T. If the learned weight $W(k,c)$ is $\cos(2\pi kc/T_{exact})$ for $c = 0, \ldots, T_{exact} - 1$ and otherwise 0, then Equation (12) is equivalent to the real part of the discrete Fourier transform with a period of T_{exact}. We can analogously show the equivalence to the imaginary part, concluding that 1×1 convolution is capable of learning the discrete Fourier transform with an exact period. Although T_{exact} can vary from sequence to sequence, 1×1 convolution has a chance of learning the best T_{exact} or the best weights that will result in the highest classification accuracy in an average sense.

To implement the block capable of extracting the power spectral density, we first apply two 1×1 convolutions with T filters to $J^{(\tau)}$. Denoting the two resulting feature maps by Re and Im, the power spectral density map PS is calculated as in Equation (11). J, Re, Im, and PS are then concatenated into a $4T$-channel feature map, which is used as input to a ResNet. This ResNet is referred to as PS-ResNet in the subsequent sections. Figure 4 depicts the PS-ResNet.

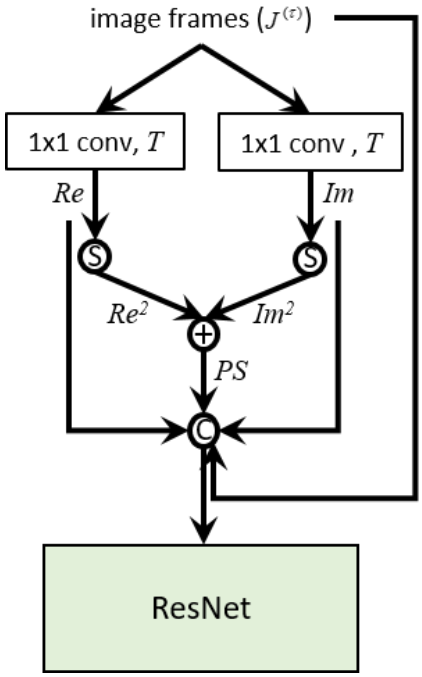

Figure 4. Our proposed PS-ResNet for early fusion. The two feature maps obtained by 1×1 convolutions are squared and added together to produce a feature map PS that can represent the per-pixel power spectral density. The input image sequence $J^{(\tau)}$ and the three feature maps Re, Im and PS are then concatenated to produce the input to a ResNet. In the figure, the "S" circle and the "C" circle represent the square operation and the concatenation operation, respectively. We note that neither batch normalization nor ReLU is applied after the convolutions. Please refer to the text for more detail.

On the other hand, the first layer of a ResNet is a 7×7 convolution layer. Because 7×7 convolution can represent 1×1 convolution, the 7×7 convolution layer is capable of learning to extract Re and Im. Indeed, a ResNet and any other CNNs are capable of learning to extract frequency-domain features. The main difference of a PS-ResNet from a ResNet is the explicit computation of the power spectral density. If the power spectral density is not a vital feature, the performance of the two architectures may be similar.

2.4. Late Fusion

Another way to use subsequent image frames is to provide each image frame as input to a ResNet and combine the outputs to determine the state of the last frame. For $t = \tau - T + 1, \ldots, \tau$, a ResNet can return T softmax outputs $\mathbf{y}^{(t)}$. The separate output vectors are then concatenated into a single $3T$-dimensional vector $\hat{\mathbf{y}}$. By employing a fully connected layer, we can combine the separate pieces of information as follows:

$$\tilde{y}^{(\tau)}(s) = \sum_{n=0}^{3T-1} w(s,n) \cdot \hat{y}(n), \tag{13}$$

where $w(s,n)$ denote the weights of the fully connected layer. By finding $\tilde{s}$ that maximizes $\tilde{y}^{(\tau)}(s)$, the state at τ is determined. Figure 5 illustrates our late fusion approach.

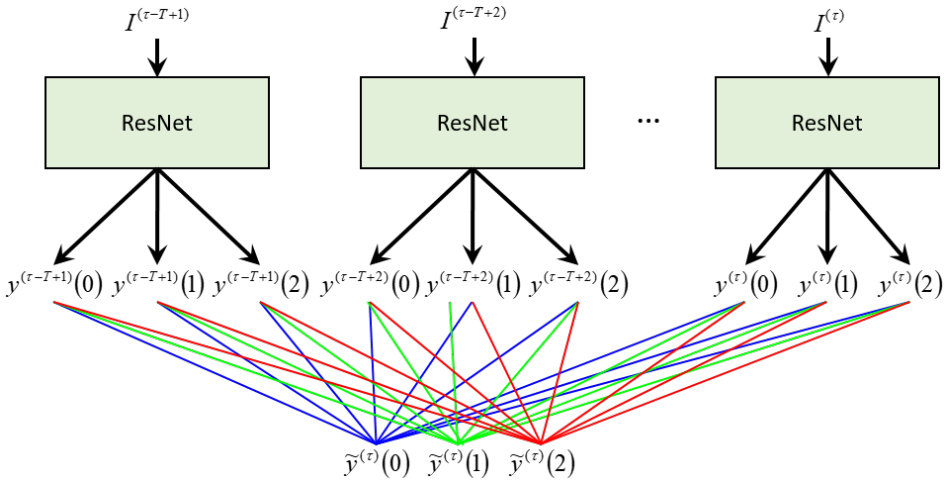

Figure 5. Our proposed layer for late fusion. The outputs of a ResNet for $t = \tau - T + 1, \ldots, \tau$ are concatenated into a $3T$-dimensional vector. To obtain the fused output $\tilde{\mathbf{y}}^{(\tau)}$, a fully connected layer is used. The colored lines represent the weights of the fully connected layer. Best viewed in color.

There are two different strategies to determine $w(s,n)$ and $d(s)$. The first is to learn the parameters from data. The second is to set the parameters manually. If we want the contributions from all outputs to be the same, then the weights can be determined as follows:

$$w(s,n) = \begin{cases} 1, & \text{if } n = 3q + s \text{ for } q = 0, \ldots, T-1, \\ 0, & \text{otherwise,} \end{cases} \tag{14}$$

considering that the stable, transient and unstable states correspond to labels 0, 1 and 2, respectively. For example, with $T = 2$, $w(s,n)$ can be represented in a matrix form as:

$$\begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}, \tag{15}$$

where the rows and columns correspond to s and n, respectively. In Section 3, we test both strategies.

Recurrent neural networks (RNNs) have been used to analyze a variety of time-series data [28]. Training of RNNs relies on backpropagation through time (BPTT) in which the required computing resources increase with T. In addition, the vanishing and exploding gradient problems prevent RNNs from using a long T. There exist RNNs [31,32] that suffer less from the gradient problems by gating the input and hidden states. However, the gating technique does not mitigate the required computing resources and training time.

Our approach is similar with the teacher-forcing RNN [28] in which not the previous hidden states but the previous outputs affect the current hidden state. By using the target outputs or the outputs of the current model, BPTT can be avoided. In our approach, $\tilde{\mathbf{y}}^{(\tau)}$ is connected to $\mathbf{y}^{(t)}$ instead of $\tilde{\mathbf{y}}^{(t)}$. This prevents a single false prediction of $\tilde{\mathbf{y}}^{(t)}$ from affecting the entire remaining predictions.

Our early fusion and late fusion approaches can be combined together to produce more accurate predictions. In Figure 5, we can replace the ResNet and $I^{(t)}$ with the PS-ResNet and $J^{(t)}$, respectively. In this manner, the prediction at time step τ depends on $2T-1$ images effectively. On the other hand, the two time periods T in the early and late fusion approaches do not have to be the same. To distinguish T in the late fusion from that in the early fusion, we write T as T_l or T_e using the subscripts for late or early from now on.

3. Model Validation Test

This section provides experimental results for the validation of the proposed models. We first introduce our gas turbine combustor and measurement system and then our multi-mode dataset consisting of image sequences acquired under different combustion conditions.

3.1. Model Gas Turbine Combustor and Measurements

A model gas turbine combustor, which forms partially-premixed flames to burn hydrogen and methane syngas without flashback risk, is used for this study. Figure 6 provides a schematic of the experimental setup for high-speed flame imaging and dynamic pressure measurement. Fuel gas is supplied from the central swirling nozzle and injected to the air within the inner side of swirl vanes at 2.7 mm upstream from the dump plane. The cylindrical combustor liner is composed of a 200 mm quartz part and a 1440 mm steel part. The flame was visualized through the quartz part liner of which diameter (D) is 130 mm and length (L) is 200 mm and the liner was cooled by compressed air to allow safe optical visualization of the flame. A water-cooled plug nozzle, which has approximately 90% of the blockage ratio of the combustor cross-sectional area is installed at the end of the combustor to form acoustically closed boundary conditions.

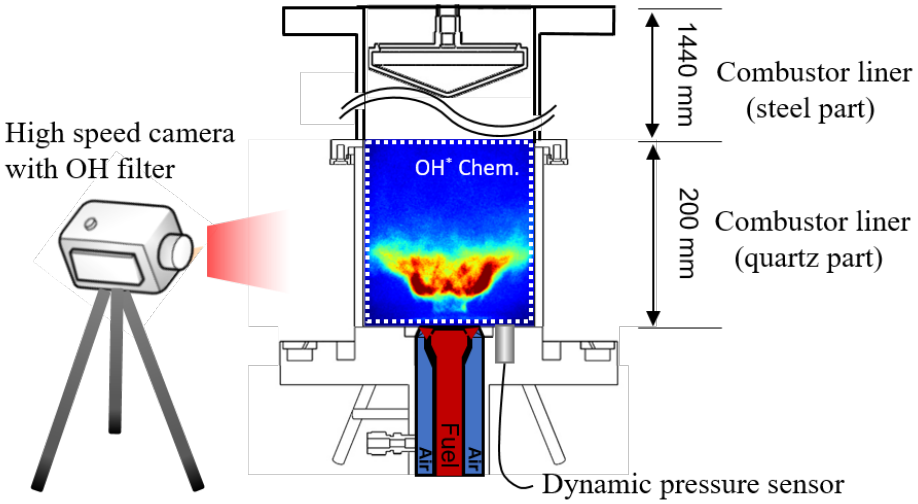

Figure 6. Schematic of the model gas turbine combustor, dynamic pressure sensor, and HI-CMOS camera position.

The dynamic pressure, which is used to monitor the level of combustion instability, is measured using piezoelectric dynamic pressure sensors (PCB-102A05, PCB Piezotronics, NewYork, U.S.A.) at the dump plane as depicted in Figure 6. High-speed OH chemiluminescence flame images are obtained through bandpass filters of WG-305 and UG-11 by a high-speed intensified CMOS camera (High-speed star8, Lavision, G ottinge, Germany). The recording time is synchronized with the dynamic pressure signal to observe flame structure according to the dynamic pressure oscillations. The images are taken during 1 second at 8 kHz with the resolution of 784 × 784 pixels and then resized to 224 × 224 pixels. More detailed descriptions of the experimental setup are provided in our previous articles [33,34].

3.2. High-Speed Flame Image Sequence Dataset

For the machine learning of this study, flame images and dynamic pressure data were acquired during transitions from an unstable to a stable state or vice versa. Table 1 shows our experimental conditions with 6 cases. Initial and final fuel supply conditions were predetermined from the results of our previous studies [33,34], which have reported instability characteristics at only steady-state conditions. In Case 1, stability changes from an unstable to a stable state at the constant heat input of 50 kW_{th} by changing fuel composition from 100% CH_4 to 12.5% H_2 and 87.5% CH_4. In other

cases, inverse conditions from stable to unstable were offered by changing either fuel composition or heat input.

Table 1. Experimental conditions for the combustion instability transition.

Case No.	Initial Fuel Supply Condition			Final Fuel Supply Condition			State Transition
	Heat Input (kWth)	H_2 (%)	CH_4 (%)	Heat Input (kWth)	H_2 (%)	CH_4 (%)	
1	50	-	100	50	12.5	87.5	unstable → stable
2	50	12.5	87.5	50	-	100	stable → unstable
3	50	100	-	50	75.0	25.0	stable → unstable
4	40	87.5	12.5	40	75.0	25.0	stable → unstable
5	40	87.5	12.5	50	87.5	12.5	stable → unstable
6	40	-	100	40	12.5	87.5	stable → unstable

One of the three phases of thermoacoustic stability (i.e., one of stable, transient, and unstable combustion states) was assigned to each flame image based on the dynamic pressure (DP) signal. Because the amplitude of dynamic pressure changes with operating conditions (e.g., fuel composition, equivalence ratio, flow rate), we used adaptive thresholds for different cases. To set the thresholds, we used stabilized DP signals computed as:

$$DP_{RMS} = \sqrt{\frac{\sum_{i=1}^{n} DP^2(i)}{n}}, \tag{16}$$

where $DP(i)$ is the dynamic pressure at time step i, and $n = 160$ is the duration corresponding to 10 ms.

For each case, we first manually selected stable sections based on our prior knowledge and the overall structure of the DP signal. The maximum DP_{RMS} was then found from the stable sections and used as the threshold separating stable states from transient states. The threshold separating transient states from unstable states was determined by introducing the concept of the cut-off frequency. For each case, the difference between the maximum DP_{RMS} and the minimum DP_{RMS} was computed. Denoting the minimum as $DP_{RMS,min}$ and the difference as ΔDP_{RMS}, the threshold was set to $DP_{RMS,min} + 1/\sqrt{2} \cdot \Delta DP_{RMS}$. This threshold is about 3 to 4% of the static pressure of the combustion chamber, which is almost the same as the standard used in gas turbines, 3% to 5%.

Figure 7 shows sample images from the 6 different cases. To apply early stopping [35] and to evaluate the generalization capability, each image sequence is divided into 80% of training, 10% of validation, and 10% of test sets. As depicted in the figures, the three sets are mutually exclusive.

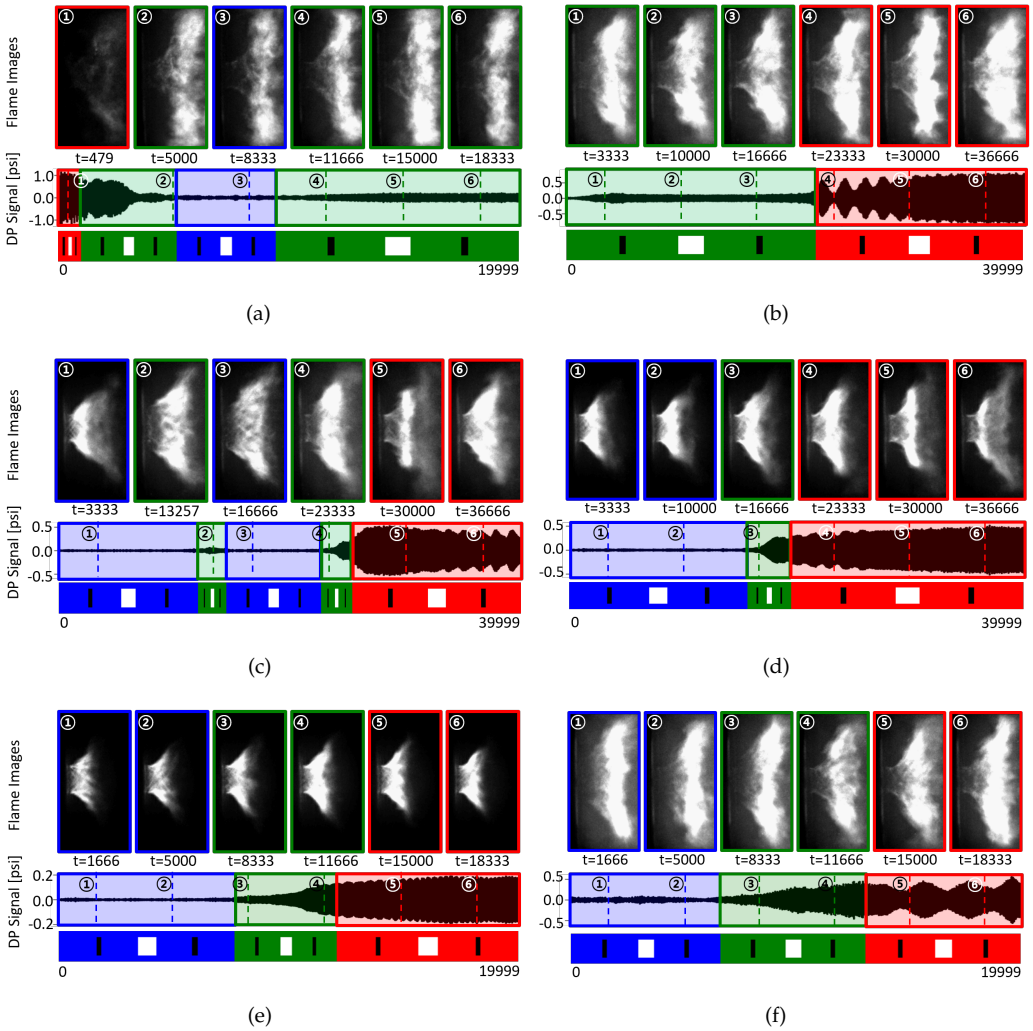

Figure 7. High-speed flame image sequence dataset. The images have been cropped and brightened for visibility. Below each row of images are shown dynamic pressure (DP), combustion states, and the configuration of training, validation, and test sets. The stable, transient, and unstable states are colored in blue, green, and red, respectively. The black and white boxes represent the validation and test sets, respectively. The remainders are the training sets. Best-viewed in color. (**a**) Case 1. (**b**) Case 2. (**c**) Case 3. (**d**) Case 4. (**e**) Case 5. (**f**) Case 6.

3.3. Training

For every network model, we minimize the cross entropy between the model's output $\mathbf{y}$ and its corresponding target output $\mathbf{z}$. For example, if the target state is stable, then $\mathbf{z} = (1, 0, 0)$. The cross entropy loss is defined as follows:

$$\mathcal{L}(\mathbf{y}, \mathbf{z}) = -\sum_{s \in \mathcal{S}} z(s) \log y(s). \tag{17}$$

The loss is minimized by the Adam optimizer [36] with a learning rate of 0.001, momentum coefficients of $(\beta_1, \beta_2) = (0.9, 0.999)$ and a mini-batch size of 32. The Adam optimizer is a variant of the stochastic gradient descent method with adaptive learning rates based on the first and second moment estimates. At each iteration, the loss in Equation (17) is averaged over the mini-batch and the average loss is minimized.

Each network model was trained over the entire training sets for 30 epochs while monitoring the validation accuracy. We repeated the training procedure three times with different random seeds and stored the best model parameters with the highest validation accuracy.

When training the late fusion layer, we used the pre-trained parameters of the ResNets and PS-ResNets. Only the parameters of the late fusion layer have been trained following the same aforementioned protocol. In this case, the cross entropy between the late-fused output $\tilde{\mathbf{y}}$ and the target output $\mathbf{z}$ was minimized. The initial weights of the late fusion layer were chosen as the manual setting described in Section 2.4. Thus, the training tries to find a local maximum of the validation accuracy near the initial weights.

3.4. Effect of Early Fusion

Tables 2 and 3 show the test set classification accuracy of the ResNets with and without the proposed early fusion. Although some outliers exist, the accuracy over the entire test set ("All" in Tables 2 and 3) tends to increase with T_e, showing the effectiveness of early fusion. The PS-ResNet gives higher overall accuracies for the majority of the combinations of the architectures and T_e. However, the difference is less than 1% for most pairs of accuracies. Thus, the power spectral density does not seem to be vital for our dataset.

Table 2. Classification accuracy (%, backbone: ResNet-18).

Case No.	T_e	1	2	4	8	16
1	ResNet	98.0	99.0	99.0	99.5	98.9
	PS-ResNet	98.7	98.0	98.9	97.4	99.0
2	ResNet	100	100	100	100	100
	PS-ResNet	100	100	100	100	100
3	ResNet	80.7	85.5	90.2	89.6	92.0
	PS-ResNet	86.9	87.7	87.5	84.5	93.0
4	ResNet	100	100	100	100	100
	PS-ResNet	100	100	100	100	100
5	ResNet	96.8	98.6	99.8	100	100
	PS-ResNet	97.4	98.0	99.8	100	100
6	ResNet	99.8	99.8	100	100	100
	PS-ResNet	99.7	100	100	100	100
All	ResNet	95.1	96.5	97.7	97.6	98.1
	PS-ResNet	96.6	96.8	97.1	96.2	98.3

Table 3. Classification accuracy (%, backbone: ResNet-50).

Case No.	T_e	1	2	4	8	16
1	ResNet	96.1	97.5	97.6	99.2	98.8
	PS-ResNet	97.0	96.6	98.2	99.2	99.6
2	ResNet	100	100	100	100	100
	PS-ResNet	100	100	100	100	100
3	ResNet	86.8	79.5	85.9	85.6	89.5
	PS-ResNet	88.1	83.2	86.4	87.3	88.9
4	ResNet	100	100	100	100	100
	PS-ResNet	100	100	100	100	100
5	ResNet	97.0	98.9	99.5	99.8	100
	PS-ResNet	97.7	98.2	99.2	100	100
6	ResNet	99.8	100	100	100	100
	PS-ResNet	100	100	100	100	100
All	ResNet	96.3	95.0	96.5	96.7	97.5
	PS-ResNet	96.8	95.7	96.7	97.1	97.5

To Cases 1, 2, 4, 5, and 6, all the models give high accuracy greater than 96.0%. When $T_e = 16$, the lowest accuracy is 98.8%. However, the models do not give as high accuracy to Case 3.

To analyze the relative degradation, we computed the approximate fundamental period of each image sequence by computing the average root mean square error (RMSE) between $I^{(t-\Delta t)}$ and $I^{(t)}$ as shown in Figure 8. Figure 9 shows subsequent image frames randomly sampled from each sequence, supporting the approximate fundamental periods. Case 3 clearly shows a period of $\Delta t = 5$ in Figure 8. Thus, the lack of periodicity does not seem to be the reason for the degradation. A greater setting of T_e may deliver better results at the expense of increased computational resources and time; however, $T_e = 16$ is already greater than all the approximate fundamental periods. According to our theory in Section 2.3, a greater setting of T_e than all the fundamental periods is sufficient.

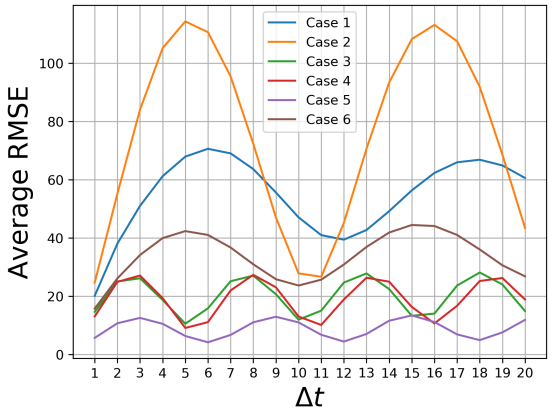

Figure 8. Approximate fundamental periods of image sequences. Case 1: $\Delta t = 12$, Case 2: $\Delta t = 11$, Case 3: $\Delta t = 5$, Case 4: $\Delta t = 5 \sim 6$, Case 5: $\Delta t = 6$, Case 6: $\Delta t = 10$.

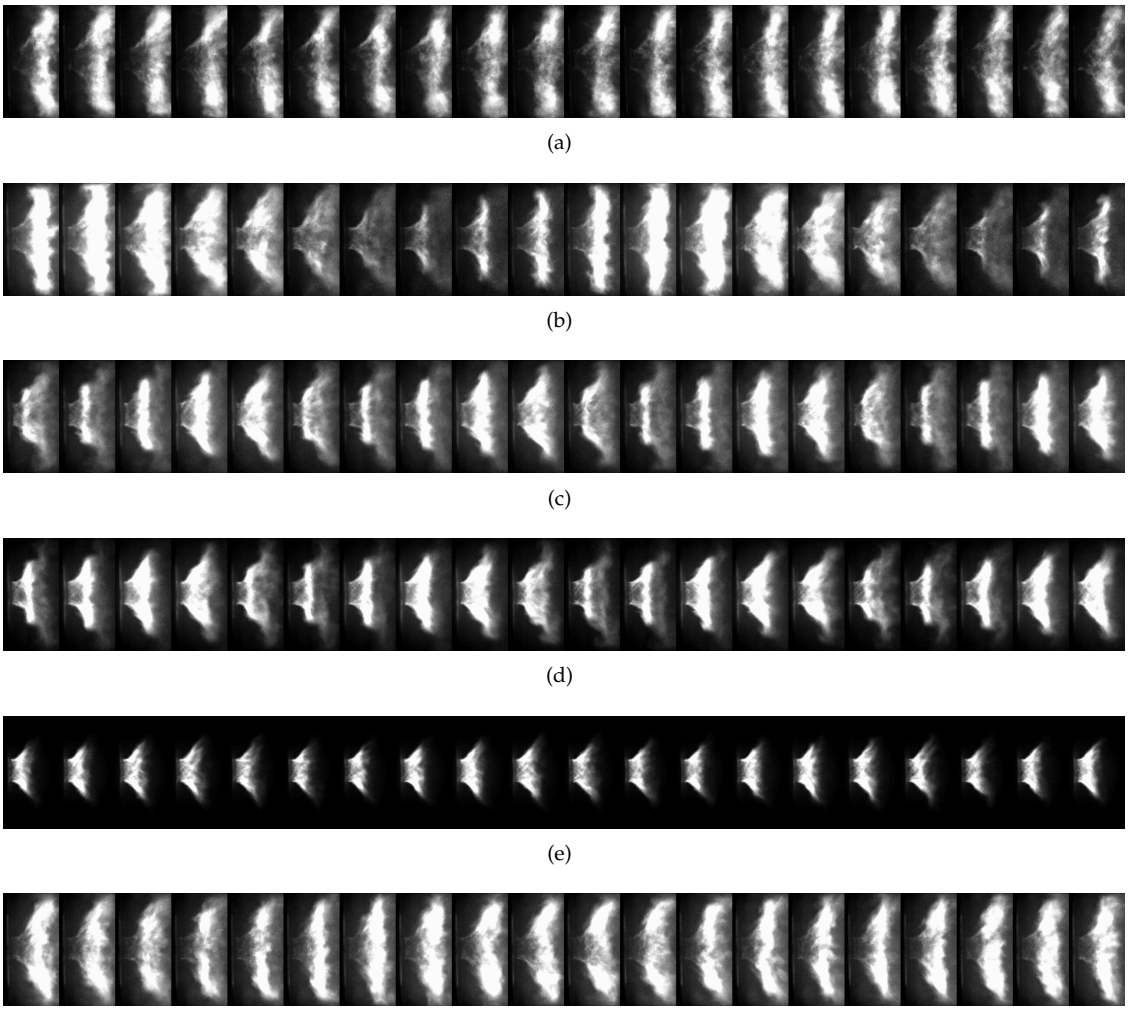

Figure 9. Subsequent image frames randomly sampled from image sequences. (**a**) Case 1, $t = 8800, \ldots, 8819$. (**b**) Case 2, t = 38,665, ..., 38,684. (**c**) Case 3, t = 29,204, ..., 29,223. (**d**) Case 4, t = 31,124, ..., 31,143. (**e**) Case 5, t = 10,921, ..., 10,940. (**f**) Case 6, t = 12,174, ..., 12,193.

For further analysis, we computed a confusion matrix of Case 3. Table 4 shows the confusion matrix corresponding to the worst accuracy attained by the ResNet-50 with $T_e = 2$. The confusion matrix shows that the main error source is the confusion between stable and transient states. Especially, more mistakes of transient states for stable states have been made. The test accuracy of the first transient section in Figure 7c is 9.2% while that of the second transient section is 100%. Not only the dynamic pressure of the first transient section is only slightly higher than that of the stable sections, but also the images of the first transient section are highly similar to those of the second stable section, as shown in Figure 10. Consequently, the classification across the two sections is more difficult than the remaining ones. Additionally, the stable and transient sections in Figure 10 do not show strong periodicity. This is the reason for lower performance of Case 3 with large T_e's in Tables 2 and 3.

Table 4. Confusion matrix (%, ResNet-50, $T_e = 2$, Case 3).

Target State \ Predicted State	Stable	Transient	Unstable
Stable	70.5	29.6	0
Transient	42.9	56.9	0.2
Unstable	0	0	100

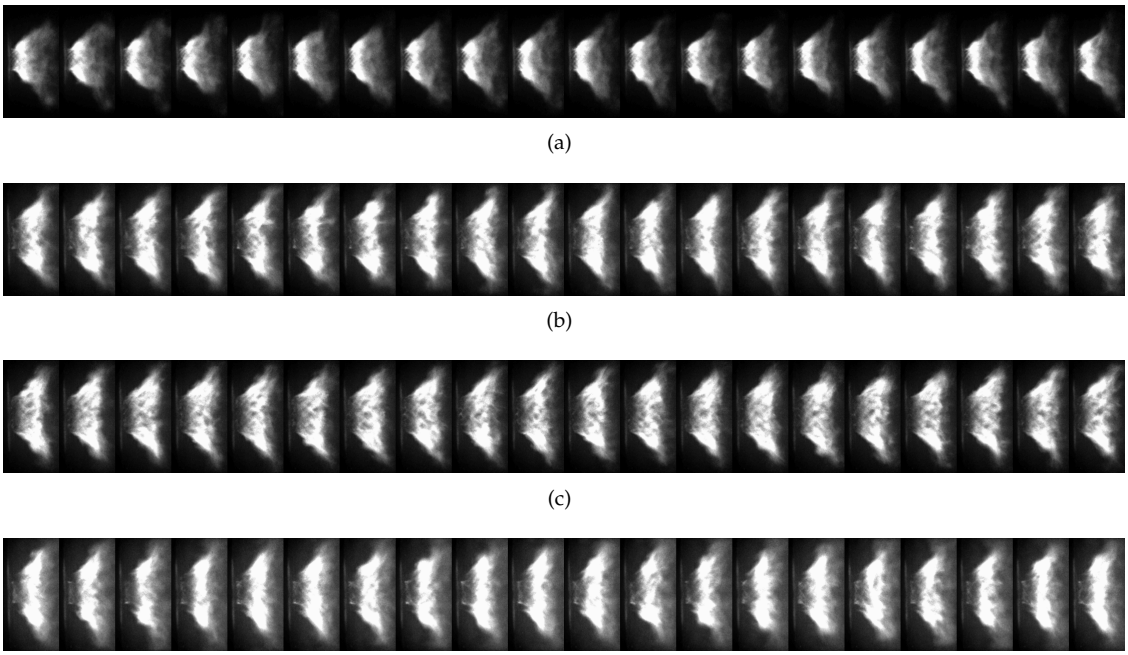

Figure 10. Subsequent image frames randomly sampled from stable and transient sections of Case 3. (**a**) Case 3, first stable section. (**b**) Case 3, first transient section. (**c**) Case 3, second stable section. (**d**) Case 3, second transient section.

3.5. Effect of Late Fusion

Our late fusion method can be applied to both ResNet and PS-ResNet outputs. In addition, the weights of the proposed late fusion layer can be either learned from data or handcrafted as we discussed in Section 2.4. Tables 5 and 6 show classification accuracies over all 6 cases according to different combinations of the proposed late fusion methods and backbone architectures. In Appendix A, Tables A1–A6 show detailed classification accuracies for each case. For all tests, we used a fixed setting of $T_l = 16$. Compared to the baseline method with single-image inputs (the ResNet-18 with $T_e = 1$), the proposed method (the PS-ResNet with $T_e = 16$ + handcrafted late fusion) gives approximately 3.5% improvement in classification accuracy.

Table 5. Overall classification accuracy (%, backbone: ResNet-18).

	T_e	1	2	4	8	16
w.o. late fusion	ResNet	95.1	96.5	97.7	97.6	98.1
	PS-ResNet	96.6	96.8	97.1	96.2	98.3
with handcrafted late fusion	ResNet	95.2	96.6	**98.4**	**98.0**	98.5
	PS-ResNet	**97.7**	**97.9**	97.5	96.5	**98.6**
with learned late fusion	ResNet	95.2	97.0	**98.4**	97.6	98.1
	PS-ResNet	97.1	97.0	97.7	96.1	98.4
with random initialization	ResNet	95.1	97.0	**98.4**	97.6	98.1
	PS-ResNet	97.0	97.1	97.7	96.1	98.5
with RMSprop	ResNet	95.0	97.4	**98.4**	97.4	97.8
	PS-ResNet	97.1	96.8	97.5	96.5	98.3

Table 6. Overall classification accuracy (%, backbone: ResNet-50).

	T_e	1	2	4	8	16
w.o. late fusion	ResNet	96.3	95.0	96.5	96.7	97.5
	PS-ResNet	96.8	95.7	96.7	97.1	97.5
with handcrafted late fusion	ResNet	97.1	95.2	**97.0**	96.7	97.9
	PS-ResNet	**98.0**	96.2	**97.0**	**97.5**	**98.0**
with learned late fusion	ResNet	96.9	95.2	96.7	96.7	97.7
	PS-ResNet	**98.0**	95.5	96.3	97.2	97.4
with random initialization	ResNet	96.9	95.2	96.9	96.8	97.8
	PS-ResNet	97.8	95.9	96.3	97.4	97.4
with RMSprop	ResNet	96.9	95.4	96.5	96.4	97.7
	PS-ResNet	97.6	**96.3**	96.6	97.3	97.4

It is interesting to notice that the late-fusion method with handcrafted weights is more effective for most combinations than that with learned weights. Because the initial weights of the learned late-fusion layer are set as the handcrafted ones, the learned weights may be worse if the initial weights are already locally optimal. In addition, improving the weights based on the validation accuracy can lead to overfitting to the validation data. We checked the validation accuracies and found that those of the learned late fusion is similar to those of the handcrafted late fusion. This means that the handcrafted weights tend to be very close to local optima already.

However, there exists a combination (the ResNet-18 with $T_e = 2$) for which the learned late-fusion most greatly improves the accuracy. Figure 11 shows the learned weights of the late fusion layer. In Figure 11, the absolute weights of the the last frame are the greatest and the absolute weights decrease with the temporal difference from the last frame. From Tables A1 and A2, we can see that the weights has learned to increase the accuracy of Case 3. We can conclude that learning the late-fusion weights can hardly lead to better generalization to the test data; however, it has the potential to produce a weight distribution that can better handle difficult cases.

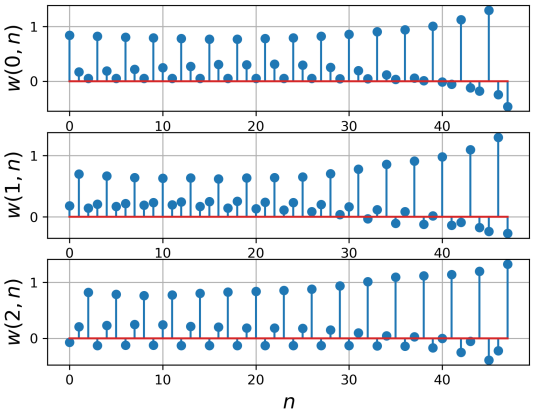

Figure 11. Learned weights of the proposed late fusion layer for the ResNet-18 with $T_e = 2$.

To verify our finding that the handcrafted weights give better results in an average sense, we varied learning of the late fusion layer in two ways: (1) We randomly initialized the weights according to the method in Reference [37]. (2) We applied a different optimization method named RMSprop [38]. In Tables 5 and 6, the two methods are denoted as "with random initialization" and "with RMSprop", respectively. Tables A3, A4, A7 and A8 show their detailed results. The random initialization typically gives the same result as the fixed initialization, although a slight improvement can be found for several cases. Changing the optimizer shows a similar effect. Both kinds of variations did not deliver better results than the handcrafted ones.

3.6. Processing Time

We measured the computation time of the proposed method using a computer running Ubuntu 18.04 with an Intel Xeon 4114 processor, an NVIDIA TITAN Xp graphics card, and 256 GB of RAM. The GPU was used for the network inference. Our code was written in Python and the PyTorch package [39] was used for the implementation. Table 7 shows the average processing time per time step. The best performing method takes 8.44 ms, which corresponds to 118 fps. Besides, application of the proposed fusion methods adds only 2 to 3 ms to the baseline. In our late fusion method, previous outputs are stored in memory so that they do not have to be recalculated to obtain the current output. Thus, the additional computational load is very low.

Table 7. Average processing time per time step.

Backbone	T_e	Early Fusion	Late Fusion	Time (ms)
ResNet-18	1	no	no	**5.21**
		yes	no	5.50
		yes	yes	7.88
	16	no	no	7.18
		yes	no	8.33
		yes	yes	**8.44**
ResNet-50	1	no	no	**10.06**
		yes	no	10.09
		yes	yes	11.88
	16	no	no	11.49
		yes	no	11.80
		yes	yes	**12.17**

4. Conclusions

We proposed two kinds of fusion layers for improving the performance of classification networks on sequential flame image inputs. The proposed early fusion layer can learn to extract the power spectral density of subsequent image frames. The proposed late fusion layer combines the outputs at different time steps to predict the current output. The proposed layers have been combined with ResNet architectures to classify the combustion state of hydrogen and methane syngas.

The proposed models for combustion instability classification based on the flame images were validated by achieving high accuracy of 95.1%∼98.6% and short processing time of 5.2∼12.2 ms. The accuracy approximately increased with T_e, the number of input flame images. With $T_e = 16$, PS-ResNets showed slightly higher performance than ResNets. Approximately 3.5% improvement in classification accuracy was obtained by the proposed fusion methods. The best performing method was the combination of our early fusion layer, ResNet-18, and our late fusion layer with handcrafted weights. Our in-depth validation tests showed interesting results such that simply increasing the number of input images is as competitive as using the proposed early-fusion layer. In addition, using handcrafted weights for the late-fusion layer was shown to be more effective than using learned weights. However, the learned weights were shown to have the potential to improve the performance on difficult cases in which the flame images are highly ambiguous.

In this work, we used regular time steps for the two fusion methods; however, for the late fusion method, there is no theoretical constraint not to use irregular time steps. In our future research, we are going to investigate learning the time steps for maximizing the classification accuracy.

Appendix A Tables of Classification Accuracies

Table A1. Classification accuracy (%, learned late fusion, backbone: ResNet-18).

Case No.	T_e	1	2	4	8	16
1	ResNet	100	100	100	99.9	99.2
	PS-ResNet	100	99.6	100	98.5	99.3
2	ResNet	100	100	100	100	100
	PS-ResNet	100	100	100	100	100
3	ResNet	78.5	86.7	92.7	89.4	91.8
	PS-ResNet	87.0	86.7	89.4	83.2	93.3
4	ResNet	100	100	100	100	100
	PS-ResNet	100	100	100	100	100
5	ResNet	99.6	99.9	100	100	100
	PS-ResNet	100	99.7	100	100	100
6	ResNet	100	100	100	100	100
	PS-ResNet	100	100	100	100	100
All	ResNet	95.2	97.0	98.4	97.6	98.1
	PS-ResNet	97.1	97.0	97.7	96.1	98.4

Table A2. Classification accuracy (%, handcrafted late fusion, backbone: ResNet-18).

Case No.	T_e	1	2	4	8	16
1	ResNet	100	100	100	100	99.5
	PS-ResNet	100	100	100	99.5	99.4
2	ResNet	100	100	100	100	100
	PS-ResNet	100	100	100	100	100
3	ResNet	78.4	84.7	92.9	91.2	93.3
	PS-ResNet	89.8	90.8	88.8	84.4	94.1
4	ResNet	100	100	100	100	100
	PS-ResNet	100	100	100	100	100
5	ResNet	99.8	100	100	100	100
	PS-ResNet	100	99.5	100	100	100
6	ResNet	100	100	100	100	100
	PS-ResNet	100	100	100	100	100
All	ResNet	95.2	96.6	98.4	98.0	98.5
	PS-ResNet	97.7	97.9	97.5	96.5	98.6

Table A3. Classification accuracy (%, learned late fusion (with random initialization), backbone: ResNet-18).

Case No.	T_e	1	2	4	8	16
1	ResNet	100	100	100	99.9	99.2
	PS-ResNet	100	99.8	100	98.8	99.3
2	ResNet	100	100	100	100	100
	PS-ResNet	100	100	100	100	100
3	ResNet	77.9	86.5	92.7	89.3	91.8
	PS-ResNet	86.5	87.3	89.8	83.2	93.7
4	ResNet	100	100	100	100	100
	PS-ResNet	100	100	100	100	100
5	ResNet	99.8	99.9	100	100	100
	PS-ResNet	100	99.7	100	100	100
6	ResNet	100	100	100	100	100
	PS-ResNet	100	100	100	100	100
All	ResNet	95.1	97.0	98.4	97.6	98.1
	PS-ResNet	97.0	97.1	97.7	96.1	98.5

Table A4. Classification accuracy (%, learned late fusion (with RMSprop), backbone: ResNet-18).

Case No.	T_e	1	2	4	8	16
1	ResNet	99.8	100	100	100	98.8
	PS-ResNet	100	99.5	100	98.8	98.9
2	ResNet	100	100	100	100	100
	PS-ResNet	100	100	100	100	100
3	ResNet	77.5	88.3	92.8	88.5	90.9
	PS-ResNet	86.9	86.0	89.0	84.8	92.9
4	ResNet	100	100	100	100	100
	PS-ResNet	100	100	100	100	100
5	ResNet	99.8	99.8	100	100	100
	PS-ResNet	100	99.8	100	100	100
6	ResNet	100	100	100	100	100
	PS-ResNet	100	100	100	100	100
All	ResNet	95.0	97.4	98.4	97.4	97.8
	PS-ResNet	97.1	96.8	97.5	96.5	98.3

Table A5. Classification accuracy (%, learned late fusion, backbone: ResNet-50).

Case No.	T_e	1	2	4	8	16
1	ResNet	99.7	99.8	99.8	99.8	99.1
	PS-ResNet	100	98.9	100	99.9	99.8
2	ResNet	100	100	100	100	100
	PS-ResNet	100	100	100	100	100
3	ResNet	86.3	78.6	85.3	85.3	90.3
	PS-ResNet	90.8	80.1	83.4	87.6	88.5
4	ResNet	100	100	100	100	100
	PS-ResNet	100	100	100	100	100
5	ResNet	100	100	100	100	100
	PS-ResNet	100	100	100	100	100
6	ResNet	100	100	100	100	100
	PS-ResNet	100	100	100	100	100
All	ResNet	96.9	95.2	96.7	96.7	97.7
	PS-ResNet	98.0	95.5	96.3	97.2	97.4

Table A6. Classification accuracy (%, handcrafted late fusion, backbone: ResNet-50).

Case No.	T_e	1	2	4	8	16
1	ResNet	99.7	100	100	100	99.2
	PS-ResNet	100	99.5	100	100	100
2	ResNet	100	100	100	100	100
	PS-ResNet	100	100	100	100	100
3	ResNet	87.2	78.2	86.4	85.0	90.9
	PS-ResNet	90.9	83.0	86.5	88.6	90.8
4	ResNet	100	100	100	100	100
	PS-ResNet	100	100	100	100	100
5	ResNet	100	100	100	100	100
	PS-ResNet	100	100	100	100	100
6	ResNet	100	100	100	100	100
	PS-ResNet	100	100	100	100	100
All	ResNet	97.1	95.2	97.0	96.7	97.9
	PS-ResNet	98.0	96.2	97.0	97.5	98.0

Table A7. Classification accuracy (%, learned late fusion (with random initialization), backbone: ResNet-50).

Case No.	T_e	1	2	4	8	16
1	ResNet	99.7	99.9	100	99.9	99.2
	PS-ResNet	100	99.3	100	100	99.8
2	ResNet	100	100	100	100	100
	PS-ResNet	100	100	100	100	100
3	ResNet	86.3	78.5	85.9	85.5	90.6
	PS-ResNet	89.9	82.0	83.4	88.4	88.5
4	ResNet	100	100	100	100	100
	PS-ResNet	100	100	100	100	100
5	ResNet	100	100	100	100	100
	PS-ResNet	100	100	100	100	100
6	ResNet	100	100	100	100	100
	PS-ResNet	100	100	100	100	100
All	ResNet	96.9	95.2	96.9	96.8	97.8
	PS-ResNet	97.8	95.9	96.3	97.4	97.4

Table A8. Classification accuracy (%, learned late fusion (with RMSprop), backbone: ResNet-50).

Case No.	T_e	1	2	4	8	16
1	ResNet	99.6	99.8	99.5	99.8	99.0
	PS-ResNet	100	99.7	100	99.9	99.8
2	ResNet	100	100	100	100	100
	PS-ResNet	100	100	100	100	100
3	ResNet	86.1	79.5	84.5	84.0	90.1
	PS-ResNet	89.1	83.6	84.9	87.8	88.3
4	ResNet	100	100	100	100	100
	PS-ResNet	100	100	100	100	100
5	ResNet	100	100	100	100	100
	PS-ResNet	100	99.8	100	100	100
6	ResNet	100	100	100	100	100
	PS-ResNet	100	100	100	100	100
All	ResNet	96.9	95.4	96.5	96.4	97.7
	PS-ResNet	97.6	96.3	96.6	97.3	97.4

Author Contributions: Conceptualization, O.C., N.K. and M.C.L.; methodology, O.C. and J.C.; software, O.C.; validation, O.C., J.C. and M.C.L.; formal analysis, O.C.; investigation, O.C., J.C. and M.C.L.; data curation, J.C., N.K. and M.C.L.; writing—original draft preparation, O.C., J.C. and M.C.L.; writing—review and editing, M.C.L.; visualization, O.C., J.C. and M.C.L.; supervision, M.C.L.; project administration, M.C.L.; funding acquisition, O.C., N.K. and M.C.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Incheon National University Institute of Convergence Science & Technology Research Grant in 2017(0006).

Acknowledgments: The authors are grateful to Youn Shik Byun for the fruitful discussion on signal processing topics.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Syred, N. A review of oscillation mechanisms and the role of the precessing vortex core (PVC) in swirl combustion systems. *Prog. Energy Combust. Sci.* **2006**, *32*, 93–161.
2. Palies, P.; Durox, D.; Schuller, T.; Candel, S. The combined dynamics of swirler and turbulent premixed swirling flames. *Combust. Flame* **2010**, *157*, 1698–1717.

3. Shahi, M.; Kok, J.B.; Casado, J.R.; Pozarlik, A.K. Assessment of thermoacoustic instabilities in a partially premixed model combustor using URANS approach. *Appl. Therm. Eng.* **2014**, *71*, 276–290.
4. Oh, S.; Kim, J.; Kim, Y. FDF-based combustion instability analysis for evolution of mode shapes and eigenfrequency in the multiple flame burner. *Appl. Therm. Eng.* **2017**, *124*, 695–706.
5. Zhang, Z.; Zhao, D.; Ni, S.; Sun, Y.; Wang, B.; Chen, Y.; Li, G.; Li, S. Experimental characterizing combustion emissions and thermodynamic properties of a thermoacoustic swirl combustor. *Appl. Energy* **2019**, *235*, 463–472.
6. Kelsall, G.; Troger, C. Prediction and control of combustion instabilities in industrial gas turbines. *Appl. Therm. Eng.* **2004**, *24*, 1571–1582.
7. Yi, T.; Gutmark, E.J. Combustion Instabilities and Control of a Multiswirl Atmospheric Combustor. *J. Eng. Gas Turbines Power* **2006**, *129*, 31–37.
8. Fichera, A.; Pagano, A. Monitoring combustion unstable dynamics by means of control charts. *Appl. Energy* **2009**, *86*, 1574–1581.
9. Song, W.J.; Cha, D.J. Temporal kurtosis of dynamic pressure signal as a quantitative measure of combustion instability. *Appl. Therm. Eng.* **2016**, *104*, 577–586.
10. Wu, G.; Lu, Z.; Pan, W.; Guan, Y.; Li, S.; Ji, C. Experimental demonstration of mitigating self-excited combustion oscillations using an electrical heater. *Appl. Energy* **2019**, *239*, 331–342.
11. Choi, J.; Choi, O.; Lee, M.C.; Kim, N. On the observation of the transient behavior of gas turbine combustion instability using the entropy analysis of dynamic pressure. *Exp. Therm. Fluid Sci.* **2020**, *115*, 110099.
12. Huang, Y.; Yang, V. Dynamics and stability of lean-premixed swirl-stabilized combustion. *Prog. Energy Combust. Sci.* **2009**, *35*, 293–364.
13. Bauerheim, M.; Nicoud, F.; Poinsot, T. Progress in analytical methods to predict and control azimuthal combustion instability modes in annular chambers. *Phys. Fluids* **2016**, *28*, 021303.
14. Gang Lu.; Yong Yan.; Colechin, M. A digital imaging based multifunctional flame monitoring system. *IEEE Trans. Instrum. Meas.* **2004**, *53*, 1152–1158.
15. Chen, J.; Hsu, T.Y.; Chen, C.C.; Cheng, Y.C. Monitoring combustion systems using HMM probabilistic reasoning in dynamic flame images. *Appl. Energy* **2010**, *87*, 2169–2179.
16. Chen, J.; Chan, L.L.T.; Cheng, Y.C. Gaussian process regression based optimal design of combustionsystems using flame images. *Appl. Energy* **2013**, *111*, 153–160.
17. Sun, D.; Lu, G.; Zhou, H.; Yan, Y.; Liu, S. Quantitative Assessment of Flame Stability Through Image Processing and Spectral Analysis. *IEEE Trans. Instrum. Meas.* **2015**, *64*, 3323–3333.
18. Akintayo, A.; Lore, K.G.; Sarkar, S.; Sarkar, S. Prognostics of Combustion Instabilities from Hi-speed Flame Video using A Deep Convolutional Selective Autoencoder. *Int. J. Progn. Health Manag.* **2016**, *7*, 14.
19. Bai, X.; Lu, G.; Hossain, M.M.; Yan, Y.; Liu, S. Multimode Monitoring of Oxy-Gas Combustion Through Flame Imaging, Principal Component Analysis, and Kernel Support Vector Machine. *Combust. Sci. Technol.* **2017**, *189*, 776–792.
20. Wang, Z.; Song, C.; Chen, T. Deep learning based monitoring of furnace combustion state and measurement of heat release rate. *Energy* **2017**, *131*, 106–112.
21. Liu, Y.; Fan, Y.; Chen, J. Flame images for oxygen content prediction of combustion systems using DBN. *Energy Fuels* **2017**, *31*, 8776–8783.
22. Qiu, T.; Liu, M.; Zhou, G.; Wang, L.; Gao, K. An Unsupervised Classification Method for Flame Image of Pulverized Coal Combustion Based on Convolutional Auto-Encoder and Hidden Markov Model. *Energies* **2019**, *12*, 1–17.
23. Han, Z.; Hossain, M.M.; Wang, Y.; Li, J.; Xu, C. Combustion stability monitoring through flame imaging and stacked sparse autoencoder based deep neural network. *Appl. Energy* **2020**, *259*, 1–16.
24. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 26 June–1 July 2016; pp. 770–778.
25. Canny, J. A Computational Approach to Edge Detection. *IEEE Trans. Pattern Anal. Mach. Intell.* **1986**, *PAMI-8*, 679–698.
26. Harris, C.; Stephens, M. A combined corner and edge detector. In Proceedings of the Fourth Alvey Vision Conference, Manchester, UK, 31 August–2 September 1988; pp. 147–151.

27. Mikolajczyk, K.; Tuytelaars, T.; Schmid, C.; Zisserman, A.; Matas, J.; Schaffalitzky, F.; Kadir, T.; Gool, L.V. A Comparison of Affine Region Detectors. *Int. J. Comput. Vis.* **2005**, *65*, 43–72.
28. Ioffe, S.; Szegedy, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In Proceedings of the International Conference on Machine Learning, Lille, France, 6–11 July 2015; pp. 448–456.
29. Kolen, J.F.; Kremer, S.C. Gradient Flow in Recurrent Nets: The Difficulty of Learning LongTerm Dependencies. In *A Field Guide to Dynamical Recurrent Networks*; IEEE: Piscataway, NJ, USA, 2001; pp. 237–243.
30. Heck, B.S.; Kamen, E.W. *Fundamentals of Signals and Systems: Using the Web and MATLAB*; Prentice Hall: Upper Saddle River, NJ, USA, 2007.
31. Hochreiter, S.; Schmidhuber, J. Long Short-term Memory. *Neural Comput.* **1997**, *9*, 1735–1780.
32. Cho, K.; van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; Bengio, Y. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In Proceedings of the Conference on Empirical Methods in Natural Language Processing, Doha, Qatar, 25–29 October 2014; pp. 1724–1734.
33. Lee, M.C.; Yoon, J.; Joo, S.; Kim, J.; Hwang, J.; Yoon, Y. Investigation into the cause of high multi-mode combustion instability of $H_2/CO/CH_4$ syngas in a partially premixed gas turbine model combustor. *Proc. Combust. Inst.* **2015**, *35*, 3263–3271.
34. Yoon, J.; Joo, S.; Kim, J.; Lee, M.C.; Lee, J.G.; Yoon, Y. Effects of convection time on the high harmonic combustion instability in a partially premixed combustor. *Proc. Combust. Inst.* **2017**, *36*, 3753–3761.
35. Goofellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016.
36. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. In Proceedings of the International Conference for Learning Representations, San Diego, CA, USA, 7–9 May 2015.
37. He, K.; Zhang, X.; Ren, S.; Sun, J. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015; pp. 1026—-1034.
38. Hinton, G. *Neural Networks for Machine Learning*, Video Lectures; Coursera: Mountain View, CA, USA, 2012.
39. Paszke, A.; Gross, S.; Chintala, S.; Chanan, G.; Yang, E.; DeVito, Z.; Lin, Z.; Desmaison, A.; Antiga, L.; Lerer, A. Automatic differentiation in PyTorch. In Proceedings of the NIPS-W, Long Beach, CA, USA, 28 October 2017.