Fire Technology
© 2020 Springer Science + Business Media, LLC, part of Springer Nature
Manufactured in The United States
https://doi.org/10.1007/s10694-020-01037-2

Deep-Learning Emulators of Transient Compartment Fire Simulations for Inverse Problems and Room-Scale Calorimetry

Tyler Buffington and Jan-Michael Cabrera, Department of Mechanical Engineering, The University of Texas at Austin, Austin, TX 78712, USA*
Andrew Kurzawski, Sandia National Laboratories, Albuquerque, NM 87185, USA
Ofodike A. Ezekoye, Department of Mechanical Engineering, The University of Texas at Austin, Austin, TX 78712, USA

Received: 5 June 2020/**Accepted:** 20 August 2020

Abstract. This work describes a deep learning methodology for "emulating" temperature outputs produced by the Fire Dynamics Simulator (FDS), a CFD software. An array of artificial neural networks (ANNs) is trained to predict transient temperatures at specified locations for a transient heat release rate (HRR) input. These locations correspond to the locations of thermocouples used in an experimental burn structure. In order to build the training set, A Gaussian process (GP) framework is used to develop a generative model that produces random viable HRR ramps. Although this procedure may require thousands of FDS runs to build a sufficient training set, the application of transfer learning can reduce the required number of runs by nearly an order of magnitude. This refers to the process of initially training an ANN to predict the output of the Consolidated Model of Fire and Smoke Transport (CFAST) and then transferring its knowledge to an ANN that learns to predict FDS outputs. CFAST is a much faster model than FDS, so a large training set can be generated quickly. The final state of the ANN trained to emulate CFAST is used as the initial state of an ANN that learns to emulate FDS. The result is a model that produces FDS temperature predictions with a mean absolute error (MAE) of less than 2°C and runs over five orders of magnitude faster than FDS. The emulators are also capable of learning inverse mappings; i.e. for a given temperature output, they can predict the HRR ramp that would cause FDS to produce the temperature response. This ability to invert for the HRR profile is exercised on data collected from eight fire experiments with peak HRRs up to 200 kW, including four propane burner fires, two methanol pool fires, and two n-Hexane pool fires. The model inverts for the experimental HRR with a MAE of 5.8 kW-15.4 kW (11.3%–16.7%) for the burner tests and 5.0 kW–25.5 kW (12.1%–28.6%) for the pool fire tests, with a tendency to underestimate the HRR of the pool fires. Finally, the computational speed of the emulators allows for the incorporation of CFD physics in Bayesian parameter inversion. As an example, this is demonstrated to infer the radiative fraction from experi-

* Correspondence should be addressed to: Tyler Buffington, E-mail: tyler.c.buffington@utexas.edu

Published online: 10 September 2020

mental and synthetic data in conjunction with reported uncertainties from the FDS Validation Guide.

1. Introduction

Compartment fire models are useful for many applications. For one, accurate predictions of potential fire scenarios help quantify the risk of adverse consequences such as property damage and loss of life. They also can provide insight in forensics problems by identifying credible fire scenarios in light of observed evidence. The value of accurate models is also augmented by the fact that compartment fire experiments are difficult and expensive to perform.

The computational expense of fire simulation tools varies significantly. For instance, "zone" models typically run at super-real-time speeds because of their relatively simple physics. These models are useful for probabilistic assessments [1, 2] and other applications that require results from many simulations. Zone models typically divide a compartment into two homogeneous zones- an upper gas layer of hot combustion products, and a cooler lower layer. Examples of commonly used zone models include the Consolidated Model of Fire and Smoke Transport (CFAST) [3] and BRANZFIRE [4]. Conversely, computational fluid dynamics (CFD) models are significantly more computationally expensive, but are generally more accurate. These models are also capable of making predictions for specific locations in the compartment, rather than for large spatially averaged zones.

There have been various efforts to combine the advantages of CFD and zone models. Hostikka et al. [5] proposed a two-model Monte Carlo approach that uses a relatively small number of FDS runs to correct the results from a larger number of CFAST runs to produce a probabilistic output. Outside of compartment fire modeling, there are many studies that have explored the coupling of zone models with CFD models for various applications [6–11].

The recent advances in machine learning have allowed for novel methods of producing models that combine the accuracy of CFD models with the computational speed of zone models. Hodges et al. [12] and Lattimer et al. [13] have proposed using transpose convolutional neural networks (TCNNs) for producing spatially resolved temperature and velocity fields for compartment fires. Although the present work is similar to these studies in the sense that deep learning is leveraged to develop "emulators" of FDS, there are several key differences. First, the emulators in the previously mentioned works are primarily designed to capture spatial variations of temperatures and velocities rather than temporal variations. Conversely, the emulators described in the present work are designed for the purpose of inferring transient quantities using a specific experimental setup.

Another key difference is the role of zone models in the methodology. Rather than using zone model results as inputs to the deep learning models, they are used for *transfer learning* [14]. This is a concept heavily used in Natural Language Processing (NLP) [15] and computer vision [16] applications. The idea is that an agent can learn a new task more efficiently if it has already been trained to perform a similar task. In the present study, an artificial neural network (ANN) is first trained to produce outputs from CFAST, with which a large training set can be obtained easily. The parameters of this ANN are then used as the initial state of an ANN that learns to produce outputs from FDS. In a similar fashion, the knowledge of the first ANN that learns to emulate FDS is transferred to all other ANNs that learn to emulate FDS. The result is that the ANNs can learn to produce accurate FDS results with much fewer FDS runs if they are initialized in this manner rather than from a random state. An advantage of this approach is that when the model makes predictions for a new case, the zone model does not need to be run again, which would be a speed-limiting step.

This paper also explores the utility of using FDS emulators for inverse problems. These problems involve using observations to infer the causal factors that produced them. This essentially involves running FDS in reverse; from a set of simulation outputs, the goal is to determine the simulation inputs that produced them. This cannot be done directly in FDS, so iterative techniques requiring many runs for a single scenario are often required, which is especially computationally expensive. Deep learning can aid this process not only by reducing the time it takes to produce FDS results, but also by obviating the need for iteration altogether. This is because ANNs can be trained to map FDS outputs to FDS inputs directly, which makes the inversion process even faster.

The specific inversion problem explored in this paper is to determine a fire's transient heat release rate (HRR) from a series of transient thermocouple measurements. The HRR has been described as the most important variable in fire hazard assessment [17]. It describes the amount of energy that is released from the combustion reactions of a fire, and it is a predictor of adverse fire consequences such as flashover [18] and secondary ignition [19]. Many methods exist to measure HRR at various scales. Perhaps the most common calorimetry apparatus at the bench scale is the cone calorimeter [20] developed by Babrauskas at NIST. This approach is based on measuring the oxygen consumed during the fire. Because the heat of combustion per unit of oxygen consumed is approximately constant across most fuels encountered in fires [21], the oxygen consumption measurements can be easily converted to estimates of the heat release rate. Another apparatus is the OSU calorimeter [22]. Unlike oxygen consumption calorimeters, this apparatus measures temperatures at various locations that allow for the HRR to be estimated by a simple energy balance.

Full scale calorimetry measurements are more difficult to obtain. Open burning calorimeters have been developed to measure the HRR of furniture items [23]. However, a limitation of these approaches is that the burning conditions do not necessarily resemble those of an actual compartment fire. To account for room effects, oxygen depletion calorimetry has been applied to room-scale fire tests [24]. These measurements are often expensive and difficult to set up, which has led to an effort to develop more cost effective full scale HRR inversion frameworks. Most of these approaches rely on using correlations or two-zone models to invert for the HRR. Richards et al. [25] used temperature data from ceiling sensors to estimate the HRR using LAVENT, a two-zone model. Overholt and Ezekoye [26] developed an inversion framework that uses measurements of the upper gas layer temperature with the zone-model CFAST. Kurzawski and Ezekoye also developed a methodology using heat flux measurements from Directional Flame Thermometers (DFTs) with FDS as the forward model. The present work is a natural extension of these studies in that it utilizes temperature measurements with the physics of FDS for HRR inversion.

2. Experimental Setup

An experimental compartment measuring 5.7 m by 4.6 m by 2.17 m previously utilized for various room scale fire experiments including positive pressure ventilation [27], wildfire fuel bed characterization [28], and HRR inversion [29] was used to conduct further testing for HRR inversion experiments. All interior walls are lined with 1.6 cm gypsum wallboard and the floor is 2.54 cm thick and made of concrete. For the burner experiments, one sand burner 0.3 m by 0.3 m square with a height of 0.4 m constructed in accordance with the standard *NFPA 286* was electronically controlled using an Alicat Scientific MCR-series 250 SLPM PID mass flow controllers to follow specified HRRs [30]. The sand burner was placed in the center of the compartment and its peak HRR is typically set to be 200 kW. A National Instruments data acquisition system was used to send setpoint signals to the Alicat mass flow controllers to control the flow of propane to the burner within the compartment. The data rate for all experiments was set to 1 Hz. Four burner experiments were conducted, each with a different prespecified HRR ramp. These included two triangle fires- one with a 300 second time to peak and one with a 100 second time to peak, a symmetric t-squared fire, and an arbitrary ramp that was chosen to resemble the potential behavior of an actual burning item whose HRR may grow and diminish unpredictably throughout an experiment. The four HRR ramps are shown in Figure 1.

In addition to the propane burner experiments, four pool fire experiments were conducted. These include two n-Hexane fires conducted in an aluminum pan with a 12-inch diameter and two methanol fires conducted in an aluminum pan with a

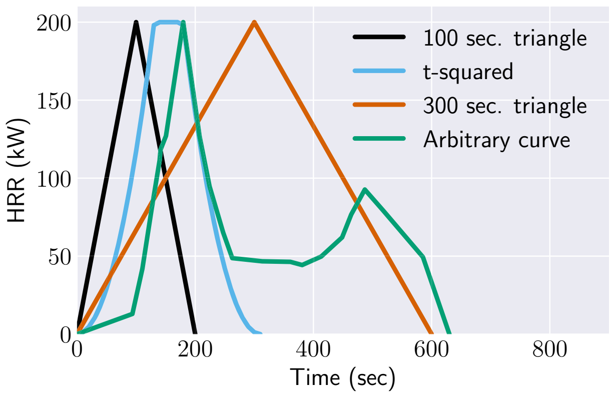


Figure 1. The four specified HRR ramps for the burner experiments.

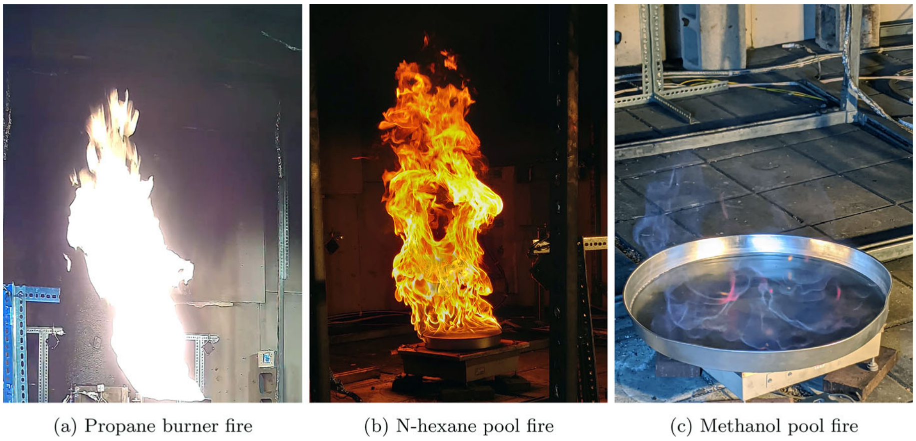


Figure 2. Photos from each of the three types of fire experiments conducted for this work.

22-inch diameter. The pans were placed on a load cell, which measured the mass of the pan and the fuel during the tests with a sampling rate of 1 Hz. The HRR was determined by smoothing the mass loss data with a kernel smoother using a bandwidth of 60 seconds before computing a first-order numerical difference. This difference was then multiplied by the effective heat of combustion, which was assumed to be 44.7 kJ/g for n-Hexane and 20 kJ/g for methanol (taken from Table 26.21 in [31]). Example photos of the tests used in this work are shown in Fig. 2.

Temperatures were collected from 17 thermocouples arranged according to Fig. 3 in the experimental chamber. There are 8 stands with 2 thermocouples each located at heights of 1.6 m and 2.11 m. The stands are arranged in a 4 by 2 grid with 1 m between stands in the x-direction and 2.2 m between stands in the y-direction. The final 2 thermocouples (16 and 17) are located along the west and east walls at a height of 1.97 m and 0.95 m from the north wall. A single thermocouple (marked with an X in Fig. 3) failed during testing and was omitted from the present work.

3. Computational Models

Two computational models of the experimental setup were used in this study- one in CFAST and one in FDS. The CFAST model serves only for pretraining the ANNs that eventually learn to emulate FDS.

3.1. Consolidated Model of Fire and Smoke Transport (CFAST)

Because the CFAST model is only used for transfer learning, it is a relatively simple description of the experimental setup. It consists of a single 6 m by 5 m by 2.4 m compartment with all surfaces specified as gypsum. The door is modeled as

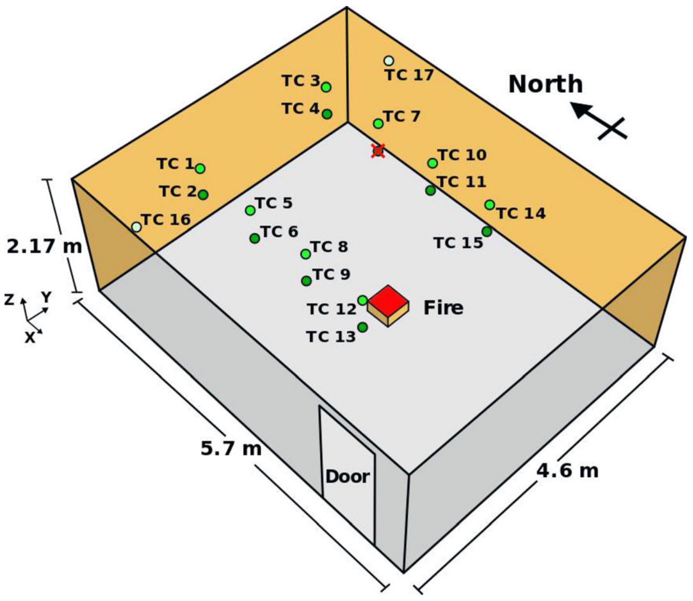


Figure 3. A diagram of the experimental burn structure and the thermocouple locations.

a 1 m by 2 m vent. The burner is specified as a fire with an area of 0.1 m and a height of 0.4 m. When making the training set of CFAST runs, the HRR is specified according to the same procedure as that used for making the training set of FDS runs.

3.2. Fire Dynamics Simulator (FDS)

The FDS model consisted of a 6 m by 5 m by 2.6 m structure with 0.2 m walls on all sides. This makes it so that the fluid domain inside the structure is 5.6 m by 4.6 m by 2.2 m which is consistent with the dimensions of 3 within the length of one grid cell (0.2 m). The door was modeled as a hole in the west wall, and the wall thickness of 0.2 m was specified so that the door would be one cell thick. It is important to note that FDS allows for the specification of different thicknesses for calculating heat transfer between the walls and surrounding air. The four exterior walls and the ceiling were modeled as gypsum boards with thermal thicknesses of 0.016 m. The thermal properties were specified according to Bruns and Prasad [32] and Kukuck [33]. The floor was modeled as concrete with a thermal thickness of 0.04 m. The thermocouples were specified according to the properties described by Kurzawski [34]. The computational domain extended outside the structure 1 m in the negative-Y direction to allow FDS to resolve the flow outside the door. A mesh size of 0.2 m was used because of the large number of runs required for the present work. Although this is a fairly coarse mesh, most of the temperature predictions did not exhibit significant grid size sensitivity. This was evaluated by running FDS simulations of the t-squared fire ramp shown in Fig. 1

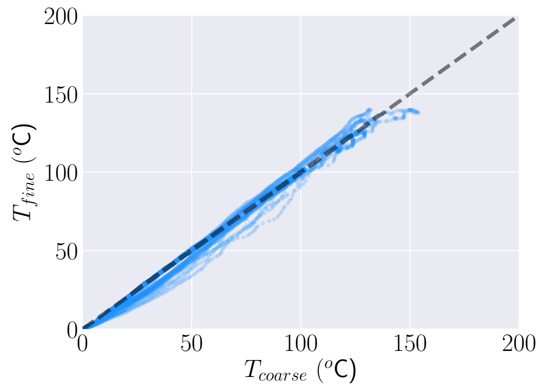


Figure 4. Comparison of FDS results using a 20 cm grid and a 10 cm grid for the t-squared fire ramp shown in Fig. 1. The results are pooled across all 17 thermocouples at all times in the simulation.

using both a 20 cm grid and a 10 cm grid. The results are shown in Fig. 4 The predicted temperatures from the 10 cm simulations (T_{fine}) are plotted against the predicted temperatures from the 20 cm simulations at each timestep (T_{coarse}) for all 17 thermocouples. The fact that the the data generally lie along the diagonal denoting equality shows that the results are not very sensitive to the grid size.

It is important to note that the grid sensitivity depends heavily on the geometry of the experimental structure being modeled. If grid sensitivity is an issue, it is expected that the transfer learning procedure described in this paper could offset the computational expense of running many fine-mesh FDS runs. The emulators could first train on coarse-mesh FDS runs, with which it is easier to build a large training set. Then, if the parameters of these trained ANNs are used to initialize ANNs that learn to emulate the fine-mesh cases, it is expected that the required size of the fine-mesh case training set would be relatively small.

4. Emulators of FDS with ANNs

4.1. Building the Training Set

A key step developing the emulators is developing a viable strategy for building the training set. One approach might be to specify the functional form of the HRR curve (e.g., t-squared or triangle ramps) and randomly draw the corresponding parameters. This approach is limiting because the ANNs then would likely only be able to produce reliable results for these *parametric* HRR curves. It would be preferable to create randomly generated HRR curves with properties that encompass the true variability in actual burn tests. To do this involves creating a *non-parametric* description of the HRR curves. For this study, non-parametric means that the HRR curve is drawn without a-priori specifying a functional form. There are also potential issues with this approach; if one draws independent random values at different times to construct the HRR curve, then that could produce unphyisically "noisy" HRR curves.

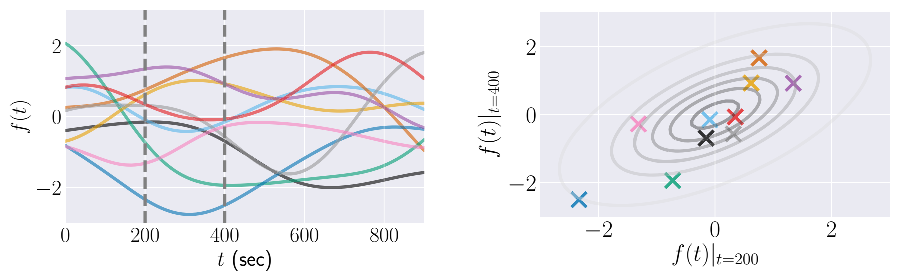
The Gaussian process formalism allows for the development of a generative model that circumvents these issues. A Gaussian process is a stochastic process in which any finite collection of random variables has a joint multivariate normal distribution [35]. This concept can be a bit abstruse for those unfamiliar with this type of statistical modeling, so a more in-depth explanation is provided here. The overall goal is to describe a probability distribution of reasonably smooth functions. If this distribution exists, then one could draw a random sample of N functions, i.e. $f_n(t), n \in \{1, 2, .., N\}$, where each $f_n(t)$ is a randomly drawn function defined over t. At any specified time, t', one could evaluate every drawn function at t'. This would result in a sample of size N of scalar function evaluations, $f_n(t')$. According to the definition of a Gaussian process, $f_n(t')$ is normally distributed for any t'. Now, one could imagine evaluating each of the functions at *two* points, t_1 and t_2, resulting in two scalar function evaluations for each $f_n(t)$. Again, if the distribution of functions is a Gaussian process, then the stochastic vectors $[f_n(t_1), f_n(t_2)]^T$ will have a bivariate normal distribution. Similarly, for any size k vector of times, $\boldsymbol{t} \in \mathbb{R}^k$, $f(\boldsymbol{t})$ is k-variate normally distributed. The general equation for a multivariate normal distribution is shown in equation 1.

$$p\bigg(f(\boldsymbol{t})\bigg) = (2\pi)^{-\frac{k}{2}} \det\big[\Sigma\big]^{-\frac{1}{2}} \exp\bigg(-\frac{1}{2}\big[\mathrm{f}(\boldsymbol{t}) - \boldsymbol{\mu}\big]^{\mathrm{T}} \Sigma^{-1} \big[\mathrm{f}(\boldsymbol{t}) - \boldsymbol{\mu}\big]\bigg) \tag{1}$$

Just as a univariate normal distribution is uniquely characterized by a scalar mean and a scalar variance, a multivariate normal distribution is uniquely characterized by a mean *vector*, $\boldsymbol{\mu}$ and a covariance *matrix*, Σ, which must be symmetric and positive semi-definite. The entries of Σ are described in equation 2.

$$\Sigma_{i,j} = E\bigg([x_i - \mu_i][x_j - \mu_j]\bigg) \tag{2}$$

where E is the expectation operator, $E(x) = \int_{-\infty}^{\infty} xp(x)dx$. The diagonal terms of Σ describe the variance of the corresponding function evaluations. The off-diagonal terms describe the correlation between two function evaluations. To make this more clear, the bi-variate example is revisited. If $\boldsymbol{t} = [t_1, t_2]^T$, and $t_1 = t_2$, then obviously $f_n(t_1) = f_n(t_2)$ for any randomly drawn continuous function, $f_n(t)$. Said differently, $f_n(t_1)$ and $f_n(t_2)$ are perfectly correlated. If instead, $t_2 = t_1 + \Delta t$, then there is no guarantee that $f_n(t_1) = f_n(t_2)$, but it is reasonable to expect that $f_n(t_1)$ and $f_n(t_2)$ are correlated if Δt is small. If Δt is large, then one would expect that $f_n(t_1)$ and $f_n(t_2)$ are uncorrelated. Figure 5 provides an illustration of the idea of covariance between function evaluations. Figure 5a shows 10 randomly drawn functions from an arbitrary Gaussian process. Figure 5b is a scatterplot of the function evaluations at 400 seconds versus those at 200 seconds. Because the functions are somewhat smooth, the scatterplot shows a positive correlation, which would increase as the difference between the two times is reduced. The defining



(a) Random draws from an arbitrary GP

(b) A scatterplot of function evaluations with bivariate PDF

Figure 5. 5(a) shows 10 randomly drawn functions from an arbitrary Gaussian process. Each randomly drawn function is evaluated at $t = 200$ seconds and at $t = 400$ seconds. A scatterplot of these function evaluations is shown in 5(b). Note that the function evaluations at the two dierent times are correlated. The contours in 5(b) indicate the bivariate normal probability density function of the points.

characteristic of a Gaussian process is that these quantities must be normally distributed, so the contours of the bivariate normal probability density function are also shown. If this process were repeated for many more randomly drawn functions, the contours would indicate the density of the points.

Just as a multivariate normal distribution is uniquely characterized by a mean vector and a covariance matrix, a Gaussian process is uniquely characterized by a mean *function*, $m(t)$, and a covariance *function*, $K(t_i, t_j)$. Then for any vector of times, $\boldsymbol{t}$, the resulting multivariate normal distribution can be characterized computing the mean vector $m(\boldsymbol{t})$ and the covariance matrix $K(\boldsymbol{t}, \boldsymbol{t})$. Based on the intuition that the degree of correlation between $f_n(t_i)$ and $f_n(t_j)$ should depend on the magnitude of $\Delta t = t_i - t_j$, the squared exponential kernel is a common choice for a covariance function. Its form is shown in equation 3,

$$K(t_i, t_j) = \tau^2 exp\left(-\frac{(t_i - t_j)^2}{2b^2} \right) \tag{3}$$

where τ^2 is a hyperparameter that describes the magnitude of the function draws, and b, known as the bandwidth, relates to the time scale over which function evaluations are correlated. Intuitively, b defines how smooth the function draws will be, with a larger bandwidth giving smoother functions. This covariance function has a ‘‘bell’’ shape; it reaches a maximum as $t_i - t_j$ approaches zero and asymptotically approaches zero as $t_i - t_j$ goes to positive infinity or negative infinity.

For generating the training set of the FDS emulators, an initial Gaussian process is specified for the transient HRR, i.e. $\dot{Q}(t) \sim \mathcal{GP}\left(m_0(t), K_0(t_i, t_j) \right)$, with $m_0(t) = 150$ kW, meaning at at any time, the function evaluations will be centered around 150 kW. The covariance function, $K_0(t_i, t_j)$ is specified according to equation 3. The hyperparameter, τ^2 is chosen to be 8,100 kW^2, meaning that the standard deviation of any individual function evaluation is $\sqrt{8100} = 90$ kW, producing a reasonable variation of HRR values centered around 150 kW. This is somewhat arbitrary, but it produces HRR curves that are generally consistent with the sand burner peak HRR of 200 kW. Two bandwidth values, b, were used. In about half of the draws, the bandwidth is set to 30 seconds and in the other half, the bandwidth is set to 60 seconds. This was done so that the training set would contain both smooth and more rapidly changing HRR curves.

In order to draw HRR curves for the ANN training set, a time vector $\boldsymbol{t}^{\mathrm{NN}} = [0, 10, \ldots, 900]^T$ is specified. This vector contains 91 evenly spaced points between 0 and 900 seconds (inclusive). 900 seconds (15 minutes) was chosen because it is the maximum simulation duration for the cases of interest in this paper. For other applications, this vector could be specified differently to allow longer simulations. Evaluations of randomly generated functions can then be drawn by specifying a multivariate normal distribution with mean $m_0(\boldsymbol{t}^{\mathrm{NN}})$ and covariance $K_0(\boldsymbol{t}^{\mathrm{NN}}, \boldsymbol{t}^{\mathrm{NN}})$. The Python package Scipy [36] was used for performing the random multivariate normal draws.

A few additional restrictions are imposed in the generative model to make the HRR draws more physically viable. The HRR should be zero at the point of ignition, i.e. $\dot{Q}(0) = 0$. Also, at the end of the fire, the HRR should return to zero, i.e. $\dot{Q}(t_{end}) = 0$, where t_{end} is the duration of the fire. It is desired that the emulators can produce FDS outputs of simulations with varying t_{end}. As a result, t_{end} is uniformly drawn between 150 and 900 seconds for each case in the training set. Fortunately these constraints can be easily imposed into the mean vector and covariance matrix. Let $\boldsymbol{t}^{\mathrm{zero}} = [0 \quad t_{end}]^T$ be a vector of times at which $Q(t)$ is zero, i.e. $Q(\boldsymbol{t}^{\mathrm{zero}}) = \boldsymbol{0}$. If one truncates the original multivariate distribution to include only draws that satisfy the constraints, the result is another multivariate normal distribution whose form is shown in equation 4.

$$\dot{Q}(\boldsymbol{t}^{\mathrm{NN}}) \sim \mathcal{N}\Big(\boldsymbol{m}, \boldsymbol{C}\Big) \tag{4}$$

where

$$\boldsymbol{m} = m_0(\boldsymbol{t}^{\mathrm{NN}}) + K_0(\boldsymbol{t}^{\mathrm{NN}}, \boldsymbol{t}^{\mathrm{zero}})\Big[K(\boldsymbol{t}^{\mathrm{zero}}, \boldsymbol{t}^{\mathrm{zero}})\Big]^{-1} m(\boldsymbol{t}^{\mathrm{zero}})$$

and

$$\boldsymbol{C} = K_0(\boldsymbol{t}^{\mathrm{NN}}, \boldsymbol{t}^{\mathrm{NN}}) - K_0(\boldsymbol{t}^{\mathrm{NN}}, \boldsymbol{t}^{\mathrm{zero}})\Big[K_0(\boldsymbol{t}^{\mathrm{zero}}, \boldsymbol{t}^{\mathrm{zero}})\Big]^{-1} K_0(\boldsymbol{t}^{\mathrm{zero}}, \boldsymbol{t}^{\mathrm{NN}})$$

The derivation of equation 4 is not given here, but it is based on the procedure of calculating posterior distributions in Gaussian process regression [37]. Two post-processing steps are performed on each draw from the distribution described in equation 4. First, the absolute value of $\dot{Q}(\boldsymbol{t}^{\mathrm{NN}})$ is taken to ensure that the HRR curve does not contain negative values. One could use a cutoff other than zero given that it is unlikely that a fire would become fully extinguished and then reignite during an experiment. Nevertheless the use of this cutoff appears to result in a sufficiently trained model. Second, the HRR curve is forced to zero for times after t_{end}. This is done so that the HRR does not ramp back up after the specified end of the fire. Examples of the randomly generated HRR curves used to build the training set are shown in Fig. 6. Figure 6a shows that the drawn functions are smoother when using a bandwidth of 60 seconds compared to those shown in Fig. 6b when using a bandwidth of 30 seconds. The emulators train on functions drawn from both distributions. Note that the HRR curves also have a varying duration so that the emulators can learn to produce the results from FDS simulations with arbitrary duration up to 900 seconds.

A Python script was used to draw random HRR ramps, write them into FDS or CFAST input files, and then run the respective simulations.

4.2. Forward FDS Emulators

In this section, the design of the *forward* emulators of FDS is described. For each thermocouple, an ANN is trained to map the time series vector $\dot{Q}(\boldsymbol{t}NN)$ to the corresponding temperature time series vector from a FDS simulation, $T_i(\boldsymbol{t}NN)$, where the index i denotes the *ith* thermocouple. Recall that $\boldsymbol{t}NN = [0, 10, ..., 900]^T$. Note that FDS does not necessarily output a prediction at the times in $\boldsymbol{t}NN$, so linear interpolation is used. For each ANN, the input layer and output layer both contain 91 nodes, which is equal to the length of $\boldsymbol{t}NN$. There are four hidden layers, each with 128 *perceptrons*. These are nodes that compute a weighted sum of their inputs and a bias, and then pass this sum through an activation function. The rectified linear activation function (ReLU) is used as the activation function. This function simply returns its input if the input is positive and returns zero otherwise.The neural networks are fully connected, meaning that each output of a given layer is an input to every perceptron in the next layer. The architecture of

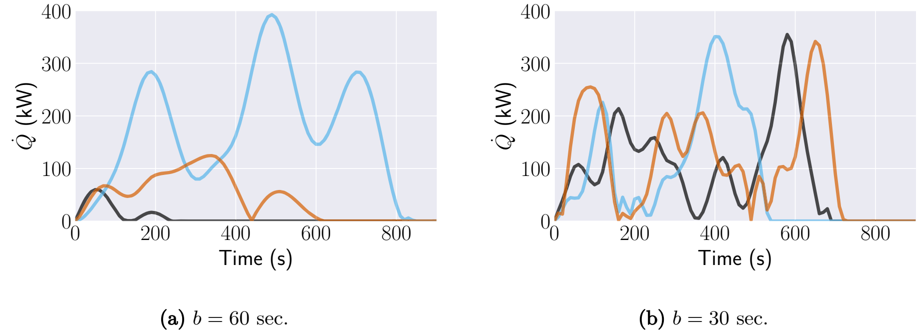
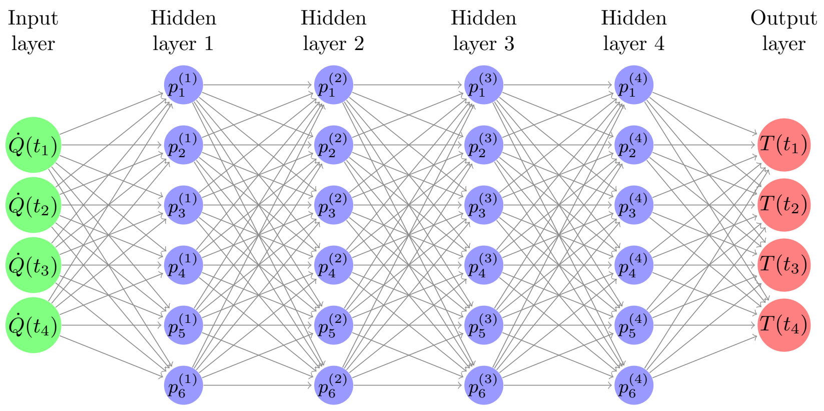
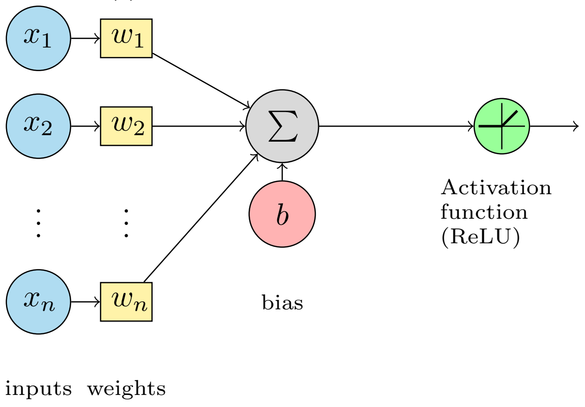


Figure 6. Example draws from the Gaussian process generative model used to build the ANN emulator training set.



(a) A simplified diagram of an ANN



(b) A diagram of an individual perceptron

Figure 7. 7a shows a simplified diagram of the neural network architecture used for each thermocouple response. In reality, the input layer and output layers each have 91 nodes as opposed to four, and each of the hidden layers has 128 perceptrons as opposed to six. 7b is a diagram showing the operation of an individual perceptron.

the ANNs is shown in Fig. 7a and the diagram of an individual perceptron is shown in Fig. 7b

Each perceptron in the first hidden layer has 92 parameters to learn (91 weights plus one bias) and all other perceptrons have 129 parameters to learn (128 weights plus one bias). This totals to 73,051 trainable parameters. The ANNs are constructed using the sequential method in the Python package Keras [38], which is built on top of TensorFlow. The parameters are trained using the Adam method [39], which is a stochastic optimization method. The batch size is set to 31, and the validation split is 20% of the data. Also, the authors found that the performance improved by dividing $\dot{Q}(t)$ by 200 kW and $T(t)$ by 200 $^\circ C$, which roughly

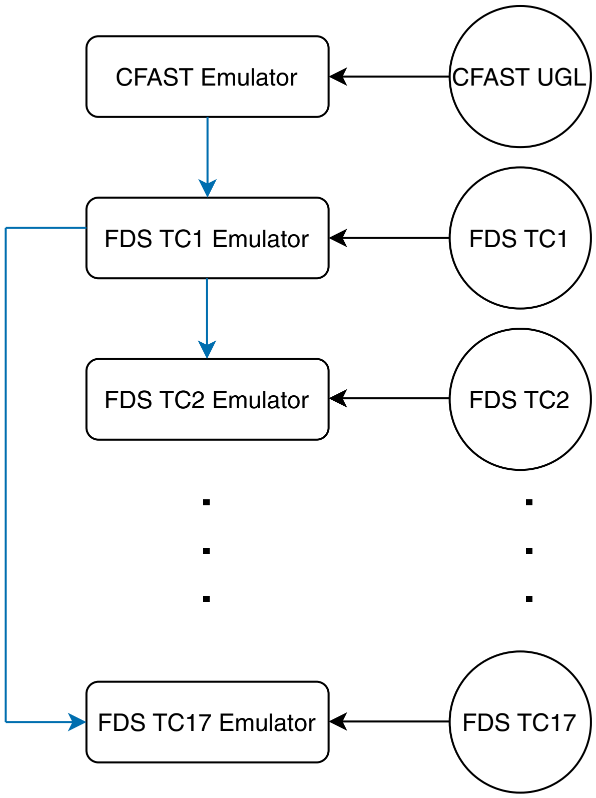


Figure 8. A flow chart summarizing the transfer learning approach used during training. The circles represent outputs from compartment fire simulations. The rectangles represent emulator ANNs. The black arrows represent direct training and the blue arrows represent knowledge transfers. For example, the first emulator trains on examples of the transient CFAST UGL whose weights and biases are randomly initialized. The learned weights and biases from CFAST are passed to the first FDS emulator, which trains on the respective thermocouple in FDS. These new, updated weights and biases trained using FDS are then used as initial states for the remaining FDS thermocouples.

corresponds to the peak temperature associated with a peak HRR of 200 kW (see Fig. 10b).

This normalization makes it so that the inputs and outputs will generally vary on the order of one, which can lead to improved training [40].

The training process also leverages transfer learning, which is based on the idea that an agent can learn a task more efficiently if it has already been trained to perform a similar task. If transfer learning is not used, the weights and biases of the emulator ANNs are initialized with random values at the start of training.

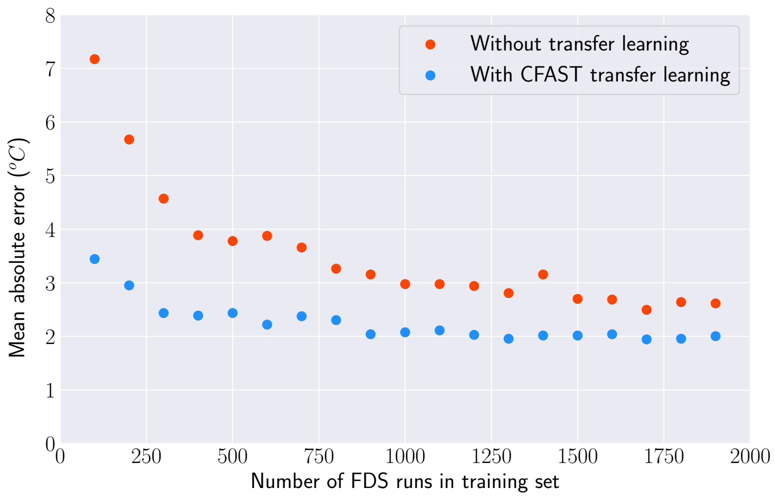


Figure 9. A plot demonstrating the effect of pretraining an FDS emulator with the CFAST training set. The red points denote the MAE vs. training set size for a ANN initialized from a random state and the blue points denote the MAE vs. training set size for an ANN initialized from the state of an ANN that has been trained to emulate CFAST.

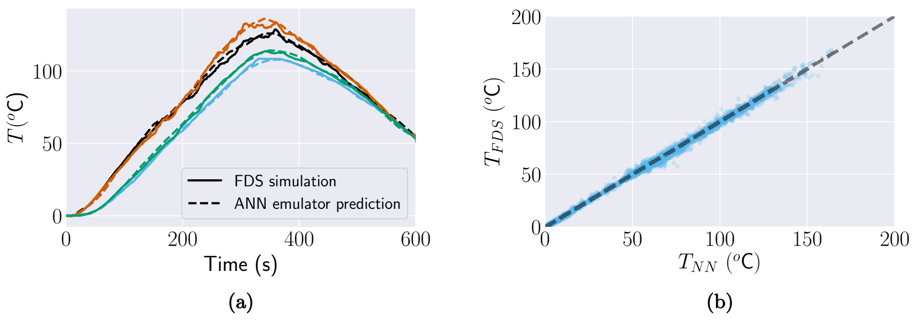


Figure 10. A comparison of the predictions of the forward ANN emulators to the FDS simulations. Figure 10(a) is an example of the emulator predictions alongside the corresponding FDS outputs. Figure 10(b) is a scatterplot of the FDS simulation results against the emulator results for all times, across all four HRR ramps shown in Fig. 1 and all 17 thermocouples.

Instead, an ANN was first trained to predict the transient upper gas layer (UGL) temperature in CFAST. A training set of 20,000 randomly generated CFAST cases was used for this initial training step. The weights and biases from this trained ANN are then used as the initial state for the ANN that is trained to predict the first FDS thermocouple. This process essentially transfers the knowledge of the simplified zone model physics of CFAST to the ANNs that learn the more complicated physics of FDS. Once the first FDS emulator is trained, its weights and biases are used to initialize each of the 16 remaining emulators. The transfer learning approach is depicted in the flow chart, Fig. 8.

In order to evaluate the effect of the this transfer learning procedure on training efficiency, 100 randomly generated FDS runs were set aside as a "test set." The HRR curves for these runs are not present in either the FDS training sets or the CFAST training set. An ANN emulator (for a single thermocouple) was then trained using a variable number of FDS cases ranging from 100 to 1900 cases. Once trained, the emulator made predictions on the 100 cases in the test set and the mean absolute error was calculated. This process was done twice- once with a "cold start" ANN (with randomly initialized parameters) and again with an ANN that was initialized with the parameters from the trained CFAST ANN. Figure 9 shows the effect of this initialization. It appears that the use of this transfer learning procedure dramatically reduces the required size of the FDS training set. If the FDS emulator is initialized with the weights and biases from a well trained CFAST UGL emulator, it can achieve lower error with a training set of 300 FDS runs than a "cold start" emulator can with a training set of 1900 FDS runs. It is important to note that the impact of transfer learning may depend on the geometry under consideration. In the present study, the compartment geometry is relatively simple and so the CFAST and FDS outputs may be more similar than would be the case for complex geometries.

For the results described in the rest of the paper, a training set of 1000 FDS cases is used. Once trained, the emulators made predictions on the four HRR ramps that were used in the burner experiments shown in Fig. 1. Note that none of these ramps are in the randomly generated training set.

Figure 10a shows a typical output from the ANN emulators compared to an actual FDS simulation. The emulator predictions are typically smoother than the FDS outputs. This is likely due to the fact that the fluctuations in the FDS outputs are due to random turbulent effects, which are very difficult for the ANNs to learn. Figure 10b shows a scatterplot of the FDS simulation results vs. the emulator predictions for all four HRR ramps shown in Fig. 1 and all 17 thermocouples. Overall the mean absolute error (MAE) of the emulator predictions compared to FDS is 1.6° C or 2.5% of the average temperature rise, indicating that they can accurately produce FDS outputs for new HRR curves. The emulators typically issue a full set of predictions (all 17 temperature time series) at a speed of about 0.01 seconds. On the same computer, it takes about 25 minutes to achieve the respective results from an FDS simulation, which equates to a speedup of over five orders of magnitude.

4.3. Inverse FDS Emulators

Another advantage of using ANN emulators to produce FDS outputs is that they can learn inverse mappings directly. The same ANN archeticure shown in Fig. 7a is used to produce 17 inverse emulators, except the quantities of the input and

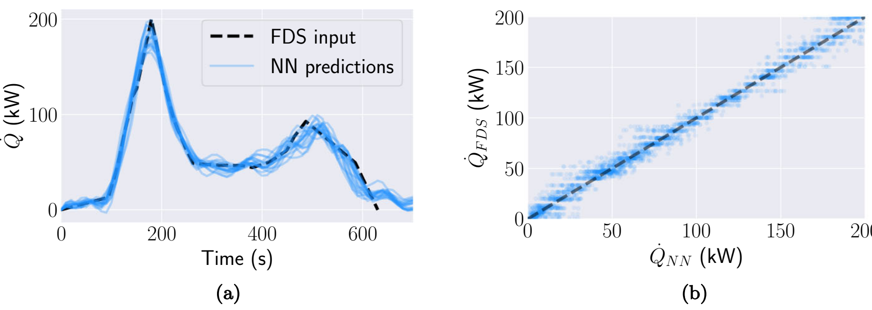


Figure 11. Predictions of the inverse FDS emulators compared to the actual HRR inputs. 11(a) is an example of the 17 predicted ramps (blue lines) generated by the emulators relative to the HRR input to FDS (black line). 11(b) is a scatterplot of the true FDS HRR input vs. the predicted input for all times, all 17 thermocouples, and all four ramps shown in Fig. 1.

output layers are switched. Each inverse emulator takes a temperature time series as an input and predicts the HRR ramp that would make FDS produce such an output for its respective thermocouple. This model can be especially useful because FDS does not natively have this functionality and iterative techniques are usually required. As before, the ANNs are first trained using the CFAST predicted upper gas layer. Using the FDS outputs for the four HRR curves shown in Fig. 1, the inverse emulators each predicted the HRR ramp input that would cause FDS to produce the outputted transient temperature results. The results are shown in Fig. 11. In Fig. 11a, the dashed black line is the HRR input ramp for an FDS simulation. For each of the 17 thermocouple locations, an inverse ANN emulator uses the corresponding simulated temperature time series to predict the HRR ramp that would cause FDS to produce such an output. The 17 blue lines show the predicted HRR curves for the 17 different ANNs. Figure 11b shows the actual HRR input compared to the predicted input for all 17 thermocouple locations, all times, and all four HRR ramps shown in Fig. 1.

The MAE for the inverse predictions on the four ramps is 5.2 kW, or 6% of the average HRR. Not surprisingly, the error as a percentage is greater for the inverse ANN emulators than for the forward ANN emulators. This may in part be due to the fact that the problem is ill-posed, in the sense that there is no guarantee that a temperature output implies a unique HRR input. Nevertheless, the emulators generally reconstruct the HRR inputs accurately. It appears that they produce the most error when the slope of the HRR curve rapidly changes signs. This may be an artifact of the bandwidth selections in the Gaussian process generative model. Also, the emulators struggle to tell when the HRR goes to zero at the end of the ramp. This is likely due to the fact that the temperatures remain elevated for quite some time, even after the fire is extinguished. In experimental applications, this issue may be mitigated by the fact that it is often visually clear when a fire is extinguished from video footage.

5. Experimental Applications

This section explores the application of the ANN emulators of FDS for two experimental efforts- estimating the transient heat release rate of a burning item and inferring the radiative fraction of a burning fuel. Both of these efforts depend heavily on the accuracy of the FDS model.

5.1. Heat Release Rate Inversion

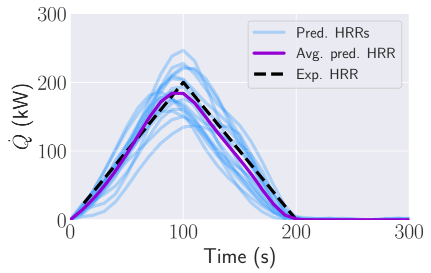
The simplest approach for using the emulators to estimate a transient HRR is to pass the transient thermocouple measurements at the times in $\boldsymbol{t}NN$ to the respective inverse ANN emulators. The FDS Validation Guide [41] indicates that FDS tends to overpredict both hot gas layer and ceiling jet temperature rises by a bias factor, δ, of 1.03. All measured temperature rises are multiplied by this factor before being passed to the emulators. This results in 17 different predicted HRR curves (one for each thermocouple). In the absence of additional information, it is not clear whether any of these predictions is more credible than any of the others. As a result, an average of all 17 predictions is taken for each timestep.

The results of this procedure using the thermocouple data from the burner experiments is shown in Fig. 12. The procedure was repeated for the pool fire experiments, shown in Fig. 13. For each experiment, the measurements from each thermocouple are passed into the respective ANN emulator. The emulator predicts the HRR ramp that would cause FDS to output the same temperature time series. Each emulator performs this task, resulting in the 17 blue curves. The purple curve is the average of all the individual emulator predictions.

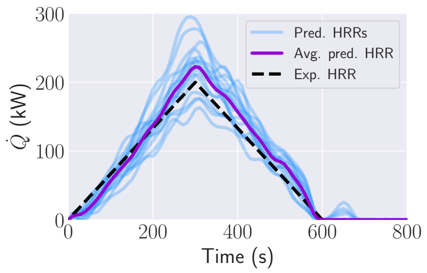
The results in Figs. 12 and 13 indicate that the inversion framework is generally more accurate for the burner experiments than the pool fire experiments. In general, the model predicts that the HRR from the pool fires is lower than that measured from the mass loss data. A summary of the results shown in Figs. 12 and 13 is shown in Table 1. Also shown is the standard deviation of the emulator predictions at the peak HRR for each of the eight experiments.

5.2. Bayesian Inference of the Radiative Fraction

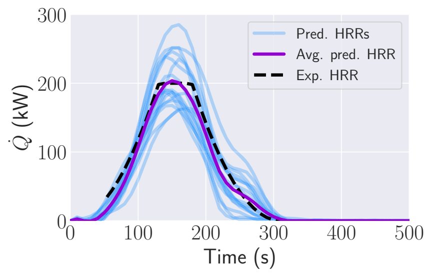
Another potential application of the FDS emulators is Bayesian inference. This is a statistical framework for updating prior beliefs in light of observed data and models. It is commonly used for inferring kinetic parameters [42] and testing fire scenario hypotheses [43]. The general idea is that one is interested in learning about some arbitrary parameter, θ. Before observing any data, prior beliefs about the possible values of θ are quantified using a probability distribution. Then, if data are observed, and one has a model for the data that takes θ as an input, the prior distribution can be updated to produce a posterior distribution that generally is more confident. Intuitively, this is because some values of θ that were originally believed to be credible produce model outputs that do not align with observed data. This approach is useful because it outputs probabilistic estimates that convey uncertainty rather than point estimates, but it can be computationally prohibitive because it relies on Markov chain Monte Carlo (MCMC) samplers,



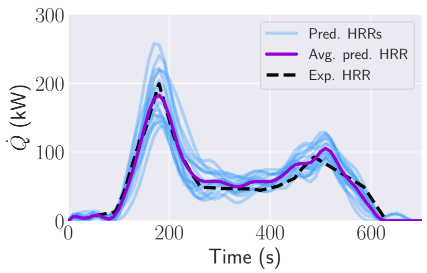
(a) 100 sec. triangle
MAE: 11.4 kW (11.3% of average exp. HRR)



(b) 300 sec. triangle
MAE: 14.9 kW (14.9% of average exp. HRR)



(c) t-squared
MAE: 15.4 kW (16.7% of average exp. HRR)



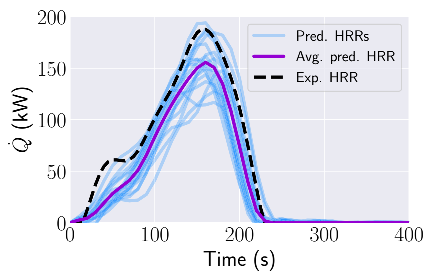
(d) Arbitrary curve
MAE: 5.8 kW (13.6% of average exp. HRR)

Figure 12. HRR inversion results for all four burner experiments. The blue lines show the HRR curves generated by the individual emulators, the purple line shows the average of these curves, and the dashed black line shows the experimental HRR ramp.

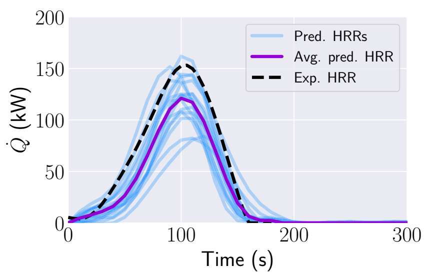
which can require tens of thousands of model evaluations performed in series. As a result, MCMC techniques have mostly been limited to simplified models that can be run quickly. However, the ability to produce CFD results quickly with deep learning makes it more practical to incorporate high fidelity physics into Bayesian inference techniques. Even if training the ANN emulators requires the same number of model evaluations as an MCMC simulation, the ability to parallelize the runs can still greatly reduce the time required to achieve MCMC results.

As an illustrative example, the Bayesian inference procedure is demonstrated to infer the radiatve fraction, χ_R of a burning fuel using a combination of synthetic and experimental data. This quantity refers to the fraction of energy that is released as radiation for a burning item. In principle, the measurement of the radiative heat flux q_0'' at a specified distance R_0 from the burning item can be used to infer the radiative HRR, Q_R. As presented in [44], assuming isotropic radiation,

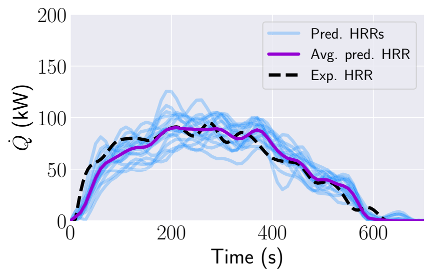
$$Q_R = 4\pi R_0^2 q_0'' \tag{5}$$



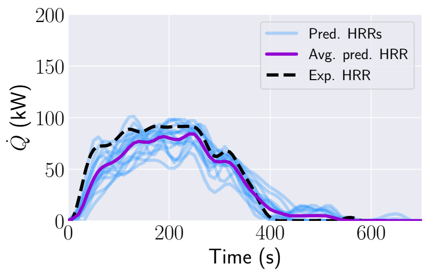
(a) N-hexane test 1
MAE: 25.5 kW (24.6% of average exp. HRR)



(b) N-hexane test 2
MAE: 21.1 kW (28.6% of average exp. HRR)



(c) Methanol test 1
MAE: 5.0 kW (12.1% of average exp. HRR)

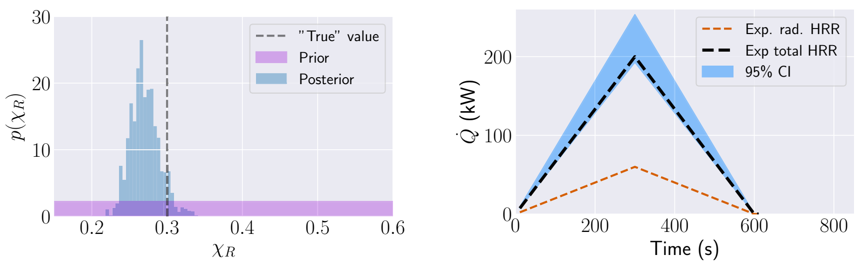


(d) Methanol test 2
MAE: 10.4 kW (18.8% of average exp. HRR)

Figure 13. HRR inversion results for all four pool fire experiments. The blue lines show the HRR curves generated by the individual emulators, the purple line shows the average of these curves, and the dashed black line shows the experimental HRR ramp determined from mass loss data.

then, the emulators described in this work give an estimate of Q, which allows for the estimation of χ_R by the relation $\chi_R = \frac{Q_R}{Q}$. As this is an illustrative example, it is assumed that $\dot{Q}_R$ is known using this approach; this is the only use of synthetic data in this section. In reality, there would be uncertainty due to the measurement error of q''_0 as well as the modeling error of equation 5. The 300 second triangle ramp (Fig. 1) is used for the demonstration because it is the burner test that exhibited the most error in the previous section. The radiative fraction, χ_R is specified to be 0.30 for propane [45], which is used to calculate $\dot{Q}_R(\boldsymbol{t}^{NN})$ (the radiative fraction at the times in $\boldsymbol{t}NN$). For the illustrative example, it is assumed that $\dot{Q}_R(\boldsymbol{t}NN)$ is the only known quantity, which could have been inferred from heat flux measurements. The Bayesian inference model attempts to learn χ_R (and by extension, $\dot{Q}$) from the observed experimental data and the FDS emulators.

Formally, the objective is to produce the posterior distribution, $p\big(\chi_R|D\big)$, which is the probability distribution for χ_R *given* observed data, D. Bayes theorem then gives equation 6:



(a) The prior and posterior distributions for the 300 second triangle fire.

(b) The 95% credible interval for the total HRR curve implied by the posterior of χ_R.

Figure 14. An illustration of the Bayesian inference process for χ_R using the reported uncertainties in the FDS validation guide. In 14a, the purple uniform distribution is the prior. After performing the MCMC sampling, the blue distribution is the posterior. 14b shows the HRR curve credible interval that is implied by the posterior of χ_R.

$$p(\chi_R|D) = \frac{p(D|\chi_R)p(\chi_R)}{p(D)} \tag{6}$$

where the *likelihood*, $p(D|\chi_R)$, quantifies how likely the observed data are given an assumed value of χ_R. This is where the FDS model is used, but it will be explained in more detail below. The *marginal likelihood*, $p(D)$ generally does not need to be addressed explicitly because it is determined from the constraint that $p(\chi_R|D)$ must integrate to one. Finally the prior, $p(\chi_R)$ quantifies prior beliefs about the value of χ_R. Fleury [46] notes that χ_R varies from approximately 0.15 to 0.60. Therefore, a uniform prior is used, $\chi_R \sim Uniform(0.15, 0.60)$.

Next, the construction of the likelihood, $p(D|\chi_R)$ is described. Specifically, the observed data D is an array of 17 measured temperature rises $\boldsymbol{T}(\boldsymbol{t}NN)$ at each of the times in $\boldsymbol{t}NN$. For a given χ_R, predicted temperature rises, $\hat{\boldsymbol{T}}(\boldsymbol{t}NN, \dot{Q}(\boldsymbol{t}NN))$, can be produced by passing the ANN emulators the HRR ramp, $\dot{Q}(\boldsymbol{t}NN) = \frac{\dot{Q}_R(\boldsymbol{t}NN)}{\chi_R}$. Because $\dot{Q}_R(\boldsymbol{t}NN)$ is given, the predictions of the forward FDS emulators will hereafter be written as a function of χ_R for simplicity, i.e. $\hat{\boldsymbol{T}}(\boldsymbol{t}NN, \dot{Q}(\boldsymbol{t}NN)) = \hat{\boldsymbol{T}}(\boldsymbol{t}NN, \chi_R)$. Because a given value of χ_R produces a specific emulator output, the likelihood can be viewed as the probability of the observed temperatures given a set of emulator predictions, i.e.

$$p(D|\chi_R) = p\left(\boldsymbol{T}(\boldsymbol{t}NN)\middle|\hat{\boldsymbol{T}}(\boldsymbol{t}NN, \chi_R)\right) \tag{7}$$

It is assumed that the difference between the forward emulator outputs and the actual FDS simulation outputs is negligible. Therefore, equation 7 is essentially the uncertainty associated with the FDS predictions. In other words, if FDS predicts a set of temperatures for a given scenario, what is the probability distribu-

Table 1
Summary of Experimental HRR Inversion Results

Experiment	MAE (% of mean HRR)	SD at peak HRR
100 sec. triangle	11.4 kW (11.3%)	31.3 kW (15.6%)
300 sec. triangle	14.9 kW (14.9%)	34.3 kW (17.1 %)
t-squared	15.4 kW (16.7%)	35.7 kW (17.8 %)
Arbitrary curve	5.8 kW (13.6%)	36.4 kW (18.2 %)
N-hexane test 1	25.5 kW (24.6%)	22.6 kW (12.0 %)
N-hexane test 2	21.1 kW (28.6%)	22.5 kW (14.7 %)
Methanol test 1	5.0 kW (12.1%)	11.4 kW (12.0 %)
Methanol test 2	10.4 kW (18.8%)	8.3 kW (9.1 %)

tion for the corresponding temperatures in the real fire scenario? Answering questions such as these is a central aim of the FDS Validation Guide [41], which serves as the primary resource for constructing this likelihood distribution. However, several steps are required before equation 7 can be written in terms of the uncertainties reported in the validation guide. First, the validation guide typically condenses time series measurements into scalar quantities. For temperatures, this is done by evaluating the peak temperature rise for both the experimental data and the FDS output. As such, the maximum temperature rise is calculated for each thermocouple across all of $\boldsymbol{t}NN$ and similarly for each emulator output. This results in two vectors each of length 17, $\boldsymbol{T}_{max}$, which contains the maximum temperature rise measured by each thermocouple and $\hat{\boldsymbol{T}}_{max}(\chi_R)$, which contains the maximum predicted temperature rise for each ANN emulator for a given χ_R. This further reduces the likelihood to

$$p(D|\chi_R) = p\left(\boldsymbol{T}_{max}\middle|\hat{\boldsymbol{T}}_{max}(\chi_R)\right) \tag{8}$$

Based on equation 4.23 of the FDS Validation Guide, the PDF of the peak temperature rise measured by thermocouple i is normally distributed, shown in equation 9[1]

$$T_{i,max}\middle|\hat{T}_{i,max}(\chi_R) \sim N\left(\frac{\hat{T}_{i,max}(\chi_R)}{\delta}, \sigma_i^2\right) \tag{9}$$

where

$$\sigma_i^2 = \left(\frac{\hat{T}_{i,max}(\chi_R)}{\delta}\right)^2 (\tilde{\sigma}_m^2 + \tilde{\sigma}_e^2)$$

[1] The experimental uncertainty is added here because $T_{i,max}$ is a measured quantity. In equation 4.23 of the Validation Guide, θ denotes the "true" quantity.

where δ is a bias term. As previously noted, it is reported in the Validation Guide as 1.03 for both the naturally ventilated hot gas layer (HGL) and ceiling jet simulations, which were determined to be the most similar quantities to those used in this paper. The relative modeling uncertainty, $\tilde{\sigma}_m$ is reported as 0.11 for naturally ventilated HGL predictions and 0.12 for ceiling jet predictions, and the value of 0.12 is used in the present study. The experimental uncertainty $\tilde{\sigma}_e$ is reported as 0.07 for both of the relevant quantities in the Validation Guide. The magnitude of the temperature rise is similar across all 17 thermocouples, so a simplifying assumption is that the σ_i^2 is the same across all thermocouples. Also, $\hat{T}_{i,max}(\chi_R)$ should be of similar magnitude to $T_{i,max}$. As a result, the following approximation is used:

$$\sigma_i^2 \approx \left(\frac{T_{global,max}}{\delta}\right)^2 (\tilde{\sigma}_m^2 + \tilde{\sigma}_e^2) = \sigma^2 \tag{10}$$

where $T_{global,max}$ is the maximum peak temperature of the 17 thermocouples. This approximation makes it so that the variance is the same across all thermocouples, and does not need to be recalculated for each iteration of the MCMC sampler. Now that the likelihood for a single thermocouple has been specified, the next step is to compute the full likelihood for the full array of thermocouples.

If the measured values of $T_{i,max}$ are independently distributed around $\frac{\hat{T}_{i,max}(\chi_R)}{\delta}$, then the resulting likelihood is equation 11.

$$p\left(\boldsymbol{T}_{max}\middle|\hat{\boldsymbol{T}}_{max}(\chi_R)\right) = \prod_{i=1}^{17} p\left(T_{i,max}\middle|\hat{T}_{i,max}(\chi_R)\right) \tag{11}$$

Although this is a common approach, there is reason to expect that the measurements are not distributed independently around the predicted values from FDS. Instead, spatial correlation is likely. For example, if FDS underpredicts the temperature measured at a location in the compartment, it probably underpredicts nearby temperatures as well. Treating correlated observations as independent can lead to biased parameter estimates and overconfident inferences due to an underestimation of the variance [47]. To circumvent these issues, the correlations of errors between FDS predictions is modeled explicitly. First, recognize that equation 11 in conjunction with equation 9 is actually a special case of the *multivariate* normal distribution,

$$\boldsymbol{T}_{max}\middle|\hat{\boldsymbol{T}}_{max}(\chi_R) \sim N\left(\frac{\hat{\boldsymbol{T}}_{max}(\chi_R)}{\delta}, \Sigma\right) \tag{12}$$

where Σ is a covariance matrix. Equation 11 is equivalent to setting $\Sigma = I\sigma^2$, where I denotes the identity matrix. In other words, the FDS Validation guide implies that the diagonal terms of Σ are each τ^2, but it is unclear what the off-diagonal terms, which describe correlations, should be. It is expected that the correlation between the errors in model outputs between two thermocouples should be a function of the distance between them. If they are very close to each other, the error associated with the FDS model will likely be similar for both of them. As the distance between them increases, the modeling errors are likely to become more uncorrelated. The intuition here is very similar to that described in the development of the Gaussian process generative model, in which function evaluations that are close in time are assumed to be correlated. In fact, the squared exponential covariance function (equation 3) is used again here to model the covariance matrix. This model is often used to capture correlation effects in the geostatistical modeling [48]. This results in the following model for the covariance matrix:

$$\Sigma_{i,j} = \sigma^2 exp\left(- \frac{d_{i,j}^2}{2b^2} \right) \tag{13}$$

where $d_{i,j}$ is the euclidean distance between the *ith* and *jth* thermocouples. It is not obvious what the value for the bandwidths, b, is most appropriate. As a result a uniform prior is also placed over it ranging from 0 to 10 m. Because of the introduction of this extra parameter, the MCMC sampler actually samples the joint distribution, $p(\chi_R, b \mid \boldsymbol{T}_{max})$. The final distribution of interest is obtained by marginalizing out b, which is described by equation 14.

$$p(\chi_R \mid \boldsymbol{T}_{max}) = \int_{-\infty}^{\infty} p(\chi_R, b \mid \boldsymbol{T}_{max}) db \tag{14}$$

The Python package PyMC [49] was used for the MCMC sampler. The results are shown in Fig. 14. It takes about 60 seconds to run a chain consisting of 10,000 iterations. It was found that modeling the spatial correlation between measurements resulted in a 95% credible interval that was about 19% wider than that obtained assuming independence.

6. Conclusions and Future Work

This work presents a methodology for training ANNs to produce FDS outputs for a specified experimental setup. To build a training set, a Gaussian process generative model provides randomly generated viable HRR curves that do not have a specified form, i.e. t-squared or triangle. Although the approach described in this paper appears to be effective in the sense that the model is able to emulate FDS results for new HRR curves in the range of interest, the approach is not optimized. Future work may focus on exploring the impact of different covariance functions on training efficiency. The present work also indicates that transfer learning is an effective way to reduce the number of required runs in the training set. Specifically, it was found that when an ANN is initialized with the weights and biases of an ANN that has learned to emulate CFAST outputs, the emulator exhibits lower error with a training set of 300 FDS runs than it does with a training set of 1900 FDS runs when it starts from a "cold start" random state. A potentially interesting topic for future work is the effect of the compartment geometry on the impact of transfer learning. Given that the present work uses a fairly simple single-compartment geometry, the outputs of CFAST and FDS may be more similar than they would be for more complex geometries. As the outputs from the two models becomes more dissimilar, the impact of transfer learning would likely diminish. However, there may be other computationally fast models besides CFAST that could be used in the transfer learning procedure for these applications.

When running the emulators on the four ramps shown in Fig. 1, the temperature predictions from the emulators align with those from FDS with a mean absolute error of 1.6° C or 2.5% of the average temperature, which is typically a negligible error compared to experimental uncertainties. It is interesting that this error is smaller than that observed when running the emulators on HRR curves from the generative model that were not in the training set (Fig. 9). This may be due to the fact that the experimental ramps are generally smoother than those produced by the generative model, which may make their respective results easier to emulate. The model produces its results in about 0.01 seconds which is a speedup of over five orders of magnitude relative to running FDS. This speedup is expected to be much greater if a finer mesh is used, but this also makes it more computationally expensive to build the training set. Another potential avenue for future work is to explore the idea of transfer learning for mesh refinement. For example, if ANNs first train on a large training set of coarse FDS runs, how many fine mesh FDS runs are required before the emulators can reliably produce the fine-mesh results? Transfer learning could also mitigate a another major limitation of this work, which is that it assumes a specific geometric configuration. For example, if others wanted to produce FDS results for a new compartment that has different dimensions or ventilation configurations, perhaps initializing the ANNs with the parameters of the ANNs used in this work would allow for faster training.

The experimental application of the emulators demonstrates their utility for heat release rate inversion using relatively inexpensive measurements. However, the HRR inversion framework appears to be much more accurate for burner fires than pool fires (ranging from 5.8 kW-15.4 kW (11.3%-16.7 %) for the burner tests and 5.0 kW-25.5 kW (12.1%-28.6%) for the pool fire tests). This is likely due to the fact that the FDS model is based on a burner fire. Future work should explore an FDS model of a pool fire and examine how different the predictions are. One potential difference between the difference in the results is that the burner imparts upward momentum to the fire, whereas the pool fires are entirely buoyancy driven. It is also possible that the pool fire error is due to uncertainty surrounding the effective heat of combustion of the fuels used. The fact that the error is more significant for the n-Hexane pool fires than for the methanol pool fires may support this hypothesis.

Finally, the ability to package FDS physics into fast deep learning models allows for the incorporation of higher fidelity physics into Bayesian inference techniques. As an example, this is demonstrated for inferring the radiative fraction of a fuel. This methodology assumes that the radiative HRR is given from measurements that are not reported in this work. Additional future work may involve using heat flux measurements to estimate the radiative component of the HRR and then using the Bayesian inference procedure to infer the total HRR. The Bayesian inference methodology used in this work is based on uncertainties reported in the FDS Validation guide. It also models spatial correlations between thermocouple measurements, which results in a 95% credible interval that is about 19% wider than that obtained assuming independence between measurements.

All of the efforts described in this work will benefit from the continued development of compartment fire models and artificial intelligence. The utility of transfer learning also motivates the continued development of computationally inexpensive "reduced order" models that can serve as stepping stones for artificial intelligence models before they learn more sophisticated, computationally expensive models.

Acknowledgements

This work was funded by the Federal Emergency Management Agency's Assistance to Fire Fighters Grant Program, grant number AFG-EMW-2017-FP-00117 and the National Science Foundation under grant number 1707090. The authors would like to thank Cory Yan of the University of Texas at Austin for his contribution to this work.

References

1. Anderson A, Ezekoye OA (2018) Quantifying generalized residential fire risk using ensemble fire models with survey and physical data. Fire Technol 54(3):715–747
2. Baker G, Wade C, Spearpoint M, Fleischmann C (2013) Developing probabilistic design fires for performance-based fire safety engineering. Procedia Eng 62:639–647
3. Peacock RD, Forney GP, Reneke PA, Jones WW, Portier RW (1993) CFAST: the consolidated model of fire growth and smoke transport. Technical report
4. Wade CA (2000) *BRANZFIRE technical reference guide*. BRANZ
5. Simo H, Timo K, Olavi K-R (2005) Two-model Monte Carlo simulation of fire scenarios. Fire Saf Sci 8:1241–1252
6. DrEng YI, DrEng SK (2008) Method for coupling three-dimensional transient pollutant transport into one-dimensional transport simulation based on concentration response factor. ASHRAE Trans 114:259
7. Gang T, Glicksman LR (2005) Application of integrating multi-zone model with CFD simulation to natural ventilation prediction. Energy Build 37(10):1049–1057
8. Wang L, Chen Q (2007) Validation of a coupled multizone-CFD program for building airflow and contaminant transport simulations. HVAC&R Res 13(2):267–281
9. Colella F, Rein G, Borchiellini R, Carvel R, Torero JL, Verda V (2009) Calculation and design of tunnel ventilation systems using a two-scale modelling approach. Build Environ 44(12):2357–2367
10. Floyd J (2011) Coupling a network HVAC model to a computational fluid dynamics model using large eddy simulation. Fire Saf Sci 10:459–470

11. Colella F, Rein G, Verda V, Borchiellini R (2011) Multiscale modeling of transient flows from fire and ventilation in long tunnels. Comput Fluids 51(1):16–29
12. Hodges JL, Lattimer BY, Luxbacher KD (2019) Compartment fire predictions using transpose convolutional neural networks. Fire Saf J 108:102854
13. Lattimer BY, Hodges JL, Lattimer AM (2020) Using machine learning in physics-based simulation of fire. Fire Saf J 114:102991
14. Pan SJ, Yang Q (2009) A survey on transfer learning. IEEE Trans Knowl Data Eng 22(10):1345–1359
15. Ruder S, Peters ME, Swayamdipta S, Wolf T (2019) Transfer learning in natural language processing. In: Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: tutorials, pp 15–18
16. Gopalakrishnan K, Khaitan SK, Choudhary A, Agrawal A (2017) Deep convolutional neural networks with transfer learning for computer vision-based data-driven pavement distress detection. Constr Build Mater 157:322–330
17. Babrauskas V, Peacock RD (1992) Heat release rate: the single most important variable in fire hazard. Fire Saf J 18(3):255–272
18. McCaffrey BJ, Quintiere JG, Harkleroad MF (1981) Estimating room temperatures and the likelihood of flashover using fire test data correlations. Fire Technol 17(2):98–119
19. Jahn W, Rein G, Torero JL (2008) The effect of model parameters on the simulation of fire dynamics. Fire Saf Sci 9:1341–1352. https://doi.org/10.3801/IAFSS.FSS.9-1341
20. Babrauskas V (1984) Development of the cone calorimeter—a bench-scale heat release rate apparatus based on oxygen consumption. Fire Mater 8(2):81–95
21. Huggett C (1980) Estimation of rate of heat release by means of oxygen consumption measurements. Fire Mater 4(2):61–65
22. Smith EE (1996) Heat release rate calorimetry. Fire Technol 32(4):333–347
23. Babrauskas V, Lawson JR, Walton WD, Twilley WH (1982) Upholstered furniture heat release rates measured with a furniture calorimeter. Technical report
24. Abecassis-Empis C, Reszka P, Steinhaus T, Cowlard A, Biteau H, Welch S, Rein G, Torero JL (2008) Characterisation of Dalmarnock fire test one. Exp Therm Fluid Sci 32(7):1334–1343
25. Richards RF, Munk BN, Plumb OA (1997) Fire detection, location and heat release rate through inverse problem solution. Part I: theory. Fire Saf J 28(4):323–350
26. Overholt KJ, Ezekoye OA (2012) Characterizing heat release rates using an inverse fire modeling technique. Fire Technol 48(4):893–909
27. Ezekoye OA, Weinschenk C, Beal CM (2011) Modeling fan-driven flows for firefighting tactics using simple analytical models and cfd. J Fire Prot Eng 21:85–114
28. Overholt KJ, Cabrera J, Kurzawski A, Koopersmith M, Ezekoye OA (2014) Characterization of fuel properties and fire spread rates for little bluestem grass. Fire Technol 50(1):9–38
29. Kurzawski AJ (2017) Inverse modeling and characterization of an experimental testbed to advance fire scene reconstruction. Ph.D. thesis, The University of Texas at Austin, Austin, Texas
30. Standard methods of fire tests for evaluating contribution of wall and ceiling interior finish to room fire growth. Standard, National Fire Protection Agency (2019)
31. Babrauskas V (2016) Heat release rates. In: SFPE handbook of fire protection engineering. Springer, pp 799–904
32. Bruns M, Prasad K (2014) Parametric analysis of heat transfer in gypsum wallboard partitions. Fire Saf Sci 11:598–611
33. Kukuck SR (2009) Heat and mass transfer through gypsum partitions subjected to fire exposures. Technical report

34. Kurzawski A, Ezekoye OA (2017) Inversion for fire heat release rate using transient heat flux data. In: ASME 2017 heat transfer summer conference. American Society of Mechanical Engineers Digital Collection
35. Lee J, Bahri Y, Novak R, Schoenholz SS, Pennington J, Sohl-Dickstein J (2017) Deep neural networks as Gaussian processes. arXiv:1711.00165
36. Jones E, Oliphant T, Peterson P (2001) SciPy: Open source scientific tools for Python. http://www.scipy.org
37. Rasmussen CE (2003) Gaussian processes in machine learning. In: Summer school on machine learning. Springer, pp 63–71
38. Chollet F (2015) keras. GitHub repository. https://github.com/fchollet/keras
39. Kingma DP, Jimmy B (2014) Adam: a method for stochastic optimization. arXiv:1412.6980
40. Sola J, Sevilla J (1997) Importance of input data normalization for the application of neural networks to complex industrial problems. IEEE Trans Nucl Sci 44(3):1464–1468
41. McGrattan K, Hostikka S, Floyd J, McDermott R, Vanella M (2020) Fire dynamics simulator technical reference guide volume 3: validation. NIST Spec Publ 1018(3):1001
42. Bruns MC (2016) Inferring and propagating kinetic parameter uncertainty for condensed phase burning models. Fire Technol 52(1):93–120
43. Overholt KJ, Ezekoye OA (2015) Quantitative testing of fire scenario hypotheses: a Bayesian inference approach. Fire Technol 51(2):335–367
44. Koseki H (1989) Combustion properties of large liquid pool fires. Fire Technol 25(3):241–255
45. Beyler CL (2016) Fire hazard calculations for large, open hydrocarbon fires. In: SFPE handbook of fire protection engineering. Springer, pp 2591–2663
46. Fleury R (2010) Evaluation of thermal radiation models for fire spread between objects. Master Thesis, University of Canterbury, Australia
47. Dormann CF (2007) Effects of incorporating spatial autocorrelation into the analysis of species distribution data. Glob Ecol Biogeogr 16(2):129–138
48. Dormann CF, McPherson JM, Araújo MB, Bivand R, Bolliger J, Carl G, Davies RG, Hirzel A, Jetz W, Kissling WD et al (2007) Methods to account for spatial autocorrelation in the analysis of species distributional data: a review. Ecography 30(5):609–628
49. Patil A, Huard D, Fonnesbeck CJ (2010) PyMC: Bayesian stochastic modelling in Python. J Stat Softw 35(4):1