A dynamic weighted loss function for enhancing the performance of neural networks

Chetra Mang, Axel Tahmasebimoradi, David Danan, Mouadh Yagoubi

16th World Congress in Computational Mechanics (WCCM)
21-26 July 2024, Vancouver, Canada
A. Korobenko, M. Laforest, S. Prudhomme, R. Vaziri (Eds)

A DYNAMIC WEIGHTED LOSS FUNCTION FOR ENHANCING THE PERFORMANCE OF NEURAL NETWORKS

Chetra Mang[1], Axel TahmasebiMoradi[1], David Danan[1] and Mouadh Yagoubi[1]

[1] Institut de Recherche Technologique SystemX
2 Bd Thomas Gobert, 91120 Palaiseau, France
chetra.mang@irt-systemx.fr,
a.tahmasebimoradi@irt-systemx.fr,
david.danan@irt-systemx.fr,
mouadh.yagoubi@irt-systemx.fr

Key words: Dynamic weighted loss function, Hyper parameters optimization, Goldstein equation, Radiative transfer equation, Neural network.

Abstract. In machine learning process, hyper parameters are chosen in a way to decrease the prediction error and improve the convergence. However, the optimized hyper parameters have a limit in terms of enhancing the performance of the neural networks. In this work, the datasets used for the numerical experiments arise from the resolution of partial differential equations (PDE) defined on a spatial domain. We propose a DYNAmic WEIghted Loss (DYNAWEIL) function-based approach for neural networks that are used to learn these PDE's solutions. This a two-step process: first we train for a few numbers of epochs in a classical way then the dynamic weighted loss function replaces the classical loss function by leveraging the information from past training error histories. To validate this method, we carry out numerical experiments with different neural networks on datasets arising on two different physics: Goldstein equation [1] and radiative transfer equation [2]. Thus, in order to demonstrate the relevance of this approach, we provide a comparison among a neural network model using a classical loss function, with and without hyper parameters optimization, and a dynamic weighted loss function for both versions.

1 INTRODUCTION

The goal of artificial intelligence (AI) is to imitate human intelligence in computers through a set of theories and methods. As a branch of artificial intelligence, machine learning (ML) studies the creation and analysis of statistical algorithms that enable computers to learn from data and then generalize their "knowledge" to predict new data. A sub-domain of ML techniques is Deep Learning (DL) that employs Artificial neural Networks (ANNs) to solve complex tasks. In DL, multiple layers with nonlinear activation functions are used within the network. Then, a loss function is defined and by minimizing (or maximizing) it, the unknown parameters of the network are found. A significant factor in the learning process is the type of loss function.

Therefore, one way to improve the learning process of neural networks is to define a suitable loss function. Such a choice arises from the data and problem type at hand.

Data imbalance, usually observed in large-scale data, causes a learning bias towards the majority of data (dominant samples) and neglecting the minority. This is crucial since rare events carry a high degree of uncertainty. Thus, training unbiased models from imbalanced data remains an open problem among researchers.

The multitude of initialization, optimizer, and hyper-parameter combinations makes it impossible to identify the optimal choice, even for a given set of tasks. Given how challenging it can be to choose the most suitable loss function initially, one may wonder if the accuracy of the network can be improved by changing iteratively the loss function during training process.

To tackle the problem of imbalanced data, a dynamic weight loss is introduced in the literature. Setting data weights inversely proportional to the data frequency is a popular and straightforward heuristic technique for balancing loss in the presence of data imbalance [3, 4]. In contrast to the previous research, [5] suggested a novel loss function known as Dynamically Weighted Balance (DWB) Loss, which could manage the unbalanced data and result in better calibration performance. The DWB was particularly assessed on hard to train samples.

In [6], the authors presented a dynamic loss function that took into account a time-dependent weight for every class (of images). In order to emphasize one class after another during training, the weight assigned to each class fluctuates and shifts within each cycle. They showed that we demonstrate that this approach leads to better training and test accuracy when the neural network is unable to optimize the standard (static) loss function in the underparametrized regime. Surprisingly, in the overparametrized regime, it also improved the accuracy of prediction.

Many efforts have been made in the last few years to identify a suitable network substitute for the physical solvers used to solve problems in solid mechanics, heat transfer, and computational fluid dynamics (CFD). These networks are ML-substitute of physical solvers based on a set of partial differential equations (PDEs) with boundary/initial conditions. In most cases, the geometry is divided into several elements; these elements are distinguished by their associated nodes. For each node, field/source variables are defined. The output of the network is usually mapped on the field points. Unsurprisingly, the prediction accuracy for each node may not be the same compared to other nodes.

To the best of our knowledge, there is no research on dynamic weighted loss function for networks of a geometric domain. In this work, we propose a neural network method based on the DYNAmic WEIghted Loss (DYNAWEIL) function for learning the solutions to PDEs defined over a geometric domain. This is a two-step process wherein we first train classically for a few hundred epochs, and then we replace the classical loss function with a dynamic weighted one by utilizing the historical training error histories. We conduct numerical experiments using various neural networks on datasets arising from two different physics: radiative transfer equation and Goldstein equation. Moreover, we present a comparison between a neural network model using a classical loss function, with and without hyper parameters optimization, and a dynamic weighted loss function for both versions in order to illustrate the applicability of this approach.

2 DYNAWEIL ALGORITHM

Training a neural network is principally based on optimization of a loss function defined on training data label. Its performance can be enhanced by many factors; one of them is the definition of loss function . In this section, we develop an algorithm based on dynamic a weighted loss function aggregated on a spatial and temporal domain of a partial differential equation.

Let's consider a partial differential equation (PDE) as follows:

$$F(p, y(x,t)) = 0 \tag{1}$$

where
- $p \in \mathcal{X} \subset \mathbf{R}^N$ and $\mathcal{X}$ is a set of input data p and N is the input data dimension;
- $y(x,t)$ is the solution to the equation (1);
- $(x,t) \in \Omega \times \mathcal{T}$, Ω is a spatial domain and $\mathcal{T}$ is a temporal domain.

The equation (1) can be solved using discretized methods such as finite difference or element method. To simplify the notation, we define the discretized solution as $y \in \mathcal{Y} \subset \mathbf{R}^M$ where M is the number of the spatial discretization and $\mathcal{Y}$ is the set of discretized solution corresponding to the input data set $\mathcal{X}$.

The proposed algorithm is compatible with any neural network that is defined on spatial dataset and can be extended to temporal dataset. In the scope of the work, the algorithm is only limited to the spatial dataset.

For simplicity, we consider a Multi-Layer Perceptron Neural Network (MLPNN) model which is a function $f_w(p) : \mathbf{R}^N \rightarrow \mathbf{R}^M$ where w is its weights. Hence, the predicted solution of the network is $\hat{y}(p) = f_w(p)$.

Let's consider a loss function vector $\bar{L}_w$ of the reference and predicted solution y and $\hat{y}$ respectively, with a metric ρ. The loss function vector can be written as :

$$\bar{L}_w(p, y) = (\rho(y_1(p) - \hat{y}_1(p)), ..., \rho(y_M(p) - \hat{y}_M(p))), \forall p \in \mathcal{X}, \forall y \in \mathcal{Y} \tag{2}$$

where $y_i(p) = y(p)(x_i)$ and $\hat{y}_i = f_w(p)(x_i), i \in [1...M]$ are the component of the reference and predicted solution respectively at each discretized spatial coordinate.

Without loss of generality, we consider the following metric: $\rho(y_i - \hat{y}_i) = (y_i - \hat{y}_i)^2$, also called Squared Error (SE). The associated weighted vector ξ is then defined as a collection of weights associated to each component of the loss function vector. If $\xi = (1, ..., 1)$, then the classical Mean Squared Error (MSE) loss function can be written as:

$$L_w(p, y) = \frac{1}{M} \xi^T \bar{L}_w(p, y), \forall p \in \mathcal{X}, \forall y \in \mathcal{Y} \tag{3}$$

where ξ^T is the transpose of ξ.

Now, we can describe the proposed algorithm DYNAWEIL. Let $\hat{\xi}$ be the maximum of the absolute error vector of the reference and predicted solution in all training dataset, defined by

$$\hat{\xi} = (max_{p \in \mathcal{X}}(|y_1(p) - \hat{y}_1(p)|), ..., max_{p \in \mathcal{X}}(|y_M(p) - \hat{y}_M(p)|)) \tag{4}$$

To simplify the notation, we denote ξ_i as the component of vector $\hat{\xi}$.

We compute this weighted vector at every certain training's epoch, called updated epoch. Then, the weighted loss used to train the neural network associated to MSE loss is defined as follows:

$$\hat{L}_w^k(p,y) = \frac{1}{M}\hat{\xi}^T \bar{L}_w^k(p,y), \forall p \in \mathcal{X}, \forall y \in \mathcal{Y} \tag{5}$$

with k being the epoch index. The loss function is changed (weighted) at each optimization operation because of the local absolute error changes at each epoch. In the proposed method, we only update the weight vector $\hat{\xi}$ at a given frequency. Therefore, between these epoch, the weighted loss stays the same and is to be minimized.

Before going further, let's look at how the weighted loss can have an impact on optimization. At each spatial discretized coordinate, the solution's local error is used as the weight coefficient at each loss function vector's component. The larger the local error is, the more important the weight will be at this coordinate point. Hence, the loss function vector component where the weight is important tends to reach a better precision, whereas, for the classical MSE loss, the local error is equally weighted.

The difference for each MLPNN's weight component $w_{i,p}$ between epoch $k+1$ and k can be written as:

$$|w_{i,k+1} - w_{i,k}| = \gamma \frac{1}{M} |\xi^T \nabla \bar{L}_{w_i}^k| \leq \gamma max_{\xi_i \in \xi}(\xi_i) |\nabla L_{w_i}^k| \tag{6}$$

where γ is the initial learning rate of a given optimizer. This means that, at each update, comparing to MSE, the descend amplitude is, at most, of factor $max_{\xi_i \in \xi}(\xi_i)$.

For solving the PDE, we seek a solution in the domain under the constraints of either boundary or initial conditions. As for training its solutions, we only need the predicted solutions to be precise on the unknown parts (i.e. excluding boundary/initial points) of the reference solution. Taking advantage of this, we can further amplify the weighted vector with the weighted amplitude a on the coordinate's unknown solution, hence the weighted vector can be expressed as follows:

$$\tilde{\xi} = (\xi_{i_1}, ..., \xi_{i_K}, a\xi_{i_{K+1}}, ..., a\xi_{i_M}) \tag{7}$$

where $\xi_{i_1}, ..., \xi_{i_K}$ are corresponding to the known components of the reference solutions (boundary/initial conditions) y and the set $\{i_1, ..., i_M\}$ is bijective to the set $\{1, ..., M\}$. The weighted loss function is as the same as in equation (5) with using equation (7).

At each update epoch, comparing to MSE, the weighted loss's descend amplitude is, at most, of factor $a\ max_{\xi_i \in \xi}(\xi_i)$. If either weighted amplitude or maximum weighted vector is too large, the amplified weighted loss may diverge. To avoid this, we normalize the initial learning rate by the average of the weighted vector and weighted amplitude. For evaluation's purpose, we can replace the predicted solution's boundary/initial point by their reference value.

We can, further, improve the algorithm by updating the amplitude according to the behaviour of the average weight vector corresponding to the unknown components of the reference solu-

tion, which is defined as:

$$\zeta = \frac{1}{M-K}\Sigma_{i\in[i_{K+1},...,i_M]}(\xi_i) \tag{8}$$

We seek to decrease ζ at each update epoch for a given amplitude a. If ζ decreases, we increase the the amplitude for that the weighted vector of the loss function vector is amplified at the local error, else we just decrease the amplitude.

The pseudo code of DYNAWEIL is described in algorithm (1).

Algorithm 1 DYNAWEIL

$f_w \leftarrow$ Initialize model's hyperparameters
$k \leftarrow 0$
$\zeta_0 \leftarrow 0$
while $k \leq$ epoch number **do**
  $\hat{\xi}_k \leftarrow (\xi_1, ..., \xi_M)$ compute weighted vector
  $\zeta_k \leftarrow$ compute average weighted vector
  if $k <$ update epoch **then**
    $\tilde{\xi} \leftarrow \xi = (1, ..., 1)$
  end if
  $\gamma_k = \gamma$
  if $k \equiv 0 \mod$ update epoch **then**
    $\gamma_k = \frac{\gamma}{a\zeta_k}$
    if $a \leq 0$ **then**
      $a = 1$
    end if
    if $\zeta_{k-1} < \zeta_k$ **then**
      $a \leftarrow a - \varsigma da$
    else
      $a \leftarrow a + da$
    end if
    if $\zeta_k \leq \varepsilon$ **then**
      break
    end if
    $\tilde{\xi}_k \leftarrow (\xi_{i_1}, ..., \xi_{i_K}, a\xi_{i_{K+1}}, ..., a\xi_{i_M})$ amplify weighted vector
  end if
  $\bar{L}_k \leftarrow$ compute loss function vector
  $\tilde{L}_k \leftarrow \frac{1}{M}\tilde{\xi}_k^T \bar{L}_k$
  training f_w with weighted loss $\tilde{L}_k$
  $k \leftarrow k + 1$
end while

3 NUMERICAL EXPERIMENTS

3.1 Problem statements

3.1.1 Goldstein equation

Any fields dealt with sound such as musics and noises, acoustic problem plays a major role. For a joyful sound like concerts or opera, one would love to amplify the sound magnitude as much as one can, however, for noises such as car or plane noises, one is eager to reduce as much as possible. Concerning plane noises, a famous Goldstein equation [1] is used to model the sound propagation. A simplification of this model is to solve a 1D harmonic transport problem with a high variation source function.

The harmonic transport equation is described by the following partial differential equation:

$$F(p, y(x)) = m\frac{\partial y}{\partial x} - iky(x) - g(x) = 0 \tag{9}$$

where
- $i = \sqrt{-1}$;
- m is mach number which describes the velocity of the propagate wave function;
- k is wave number;
- g is the source function which describes the external force pushing on the wave medium;
- $x \in [0, 1]$ is spatial domain;
- $y(x = 0) = 0$ is the boundary condition.

The discretized solution of the equation (9) is $y \in \mathbf{C}^M$. It can be written as $y_R = (Re(y), Im(y)) \in \mathbf{R}^{2M}$.

We work on the high dimensional input data where $m = m(x) \in \mathbf{R}^M$ and $g = g(x) \in \mathbf{C}^M$ or $g_R = (Re(g), Im(g)) \in \mathbf{R}^{2M}$ are represented by a parametric Chebeshev polynomial that vary at each discretized spatial position and the parameter k is fixed. So, the input and output data are $p = (m, g_R) \in \mathbf{R}^{3M}$ and $y_R \in \mathbf{R}^{2M}$ respectively.

3.1.2 Radiative transfer equation

The steady-state radiative transfer equation for a spectral participative, non-scattering gas is [2, 7]

$$\frac{\partial I_\nu(\vec{r}, \vec{s})}{\partial s} = \kappa_\nu I_{b\nu}\left(T\left(\vec{r}\right)\right) - \kappa_\nu I_\nu\left(\vec{r}, \vec{s}\right) \tag{10}$$

where sub-script ν refers to the spectral dependency of the corresponding variables, $\vec{s}$ is the direction along the ray, $\vec{r}$ is the spatial coordinate, I_ν is the radiative intensity, κ_ν is the absorption coefficient, T is the temperature, and $I_{b\nu}$ is the Planck's function.

On a diffusive wall (a surface that reflects diffusively) with a known temperature T_0, the boundary condition at a given point $\vec{r}_0$ is given by

$$I_\nu^0\left(\vec{r}_0,\vec{s}_0\right)=\varepsilon_\nu I_{b\nu}\left(T_0\right)+\frac{1-\varepsilon_\nu}{\pi}\int_{\vec{n}.\vec{s'}<0} I_\nu\left(s'\right)|\vec{n}.\vec{s'}|d\Omega' \tag{11}$$

where index 0 refers to a location on a boundary where the ray is drawn, $\vec{n}$ is the normal of the boundary, ε_ν is the spectral emissivity of the boundary, $\vec{s'}$ is the direction along the incident/incoming ray and Ω' is the solid angle corresponding to $\vec{s'}$.

By integrating over spectral bands, the hemispherical irradiation on a boundary surface can be written as:

$$H=\int_0^\infty H_\nu d\nu=\int_{\vec{n}.\vec{s'}}\int_0^\infty I_\nu\left(s'\right)d\nu|\vec{n}.\vec{s'}|d\Omega' \tag{12}$$

The statistical narrow band (SNB) model with the Curtis-Godson (CG) modification can be used to calculate the spectral absorption coefficients of participative gases for a given spectral band. It is outside the scope of this work to discuss the SNB-CG model; interested readers should consult [8, 9, 10] for further information. Based on SNB-CG model for gases, the absorption coefficients are functions of temperature (T), wavenumber (ν), total pressure (p), and molar fractions of participative gases (x_g).

In order to create a reference dataset, the Discrete Transfer Radiation Method (DTRM), proposed by Lockwood and Shah [11], is used to solve the radiative transfer equation (10) with its boundary condition (11). For the sake simplicity, a 2D rectangular shape furnace with CO2, CO, H2O gases is considered. The boundary and the domain are meshed in a Cartesian way. For this problem, the input parameters are emissivities (ε) and temperature (T_0) of the boundary, and the temperature (T) of the domain. On the other hand, the output is the hemispherical irradiation (H) for each boundary point. For n_x and n_y discretization points on the boundaries, we have $2(n_x+n_y)$ boundary points and $(n_x n_y)$ domain points.

3.2 DYNAWEIL's parameter impacts

In this section, we describe the influence of the DYNAWEIL's parameters such as update epoch frequency, amplitude a, amplitude step da, and learning rate γ on the performance of MLPNN applied on the dataset generated by transport harmonic problem. We adopt the neural network's hyper-parameters as shown in Tab. 3's first column. In this case, we train the neural network for 100 epochs. Table 1 illustrates the impact of DYNAWEIL's parameters on Mean Absolute Error (MAE) and Mean Relative Error (MRE).

For a fixed update epoch of 20 with amplitude step $da=0$ and initial learning rate $\gamma=0.1$, the weighted amplitude a is only changed. It is shown that for the weighted amplitude of 0.1, the MAE and MRE are smaller than those of others. In general, the smaller or larger weighted amplitude cannot guarantee a better performance of neural network.

For a fixed update epoch of 20 with amplitude $a=3$ and initial learning rate $\gamma=0.1$, the amplitude $da=2$ provides a better neural network's performance. As in the above observation, we need to find a suitable amplitude step da for a better neural network's performance.

For a fixed amplitude $a=3$, $da=2$ and initial learning rate $\gamma=0.1$, the larger the update epoch is, the better the neural network's performance is.

For a fixed update epoch of 20 with weighted amplitude $a = 3$ and amplitude step $da = 2$, we observe that a smaller or larger learning rate cannot guarantee the better neural network's performance. For learning rate of 0.2, it provides a better performance. We can observe that the change of weighted loss for each update epoch provides a better neural network's performance.

From this parametric study, it shows that the better neutral network's performance can be obtained by choosing the suitable learning rate, weighted amplitude, amplitude step, and a large enough update epoch.

Table 1: DYNAWEIL's parametric study

Fixed parameters	update epoch = 20 da = 0, γ=0.1			update epoch = 20 a = 3, γ=0.1		
Studied parameters	amplitude a			amplitude step da		
Values	0.05	0.1	3	0.1	2	5
MAE	0.014	0.013	0.017	0.017	0.016	0.0167
MRE	11.90	11.23	22.06	21.74	18.80	19.55
Fixed parameters	a=3, da = 2, γ=0.1			update epoch=20 a = 3, da=2		
Studied parameters	update epoch			Learning rate γ		
Values	3	20	101	0.05	0.2	0.5
MAE	0.017	0.016	0.017	0.022	0.015	0.024
MRE	22.30	20.02	19.15	28.17	16.29	35.14

3.3 Comparison of classical and DYNAWEIL neural network on different physics

This section describes the performance of a neural network model using a classical and dynamic weighted loss function. For the comparison, we considered also the model with and without optimized hyper-parameters. A random DYNAWEIL's parameters are chosen to demonstrate the capability of this algorithm as shown in Tab. 2.

Table 2: DYANWEIL's parameters

amplitude a	amplitude step da	step factor ς	update epoch
3	2	0.1	20

First, fully connected neural network's hyper-parameters are randomly chosen for training Goldstein and radiative transfer dataset as shown in Tab. 3. Then, we employ hyper-parameter

searches using optuna [12] to optimize the number of layers, number of nodes for each layer, training batch-size, and as well as learning rate for the two datasets.

Second, the hyper-parameters and the optimized one are used to train the fully connected neural network classically and with/without DYNAWEIL. The optimized hyper-parameters are displayed in the Tab. 3. The neural network performance evaluation are performed on MAE and MRE, and are shown in Tab. 4.

Table 3: Fully connected neural networks hyper-parameters

Physical problems	Goldstein		radiative transfer	
Hyper-parameters	Initial	Optimized	Initial	Optimized
Input dimension	3072	3072	2096	2096
Output dimension	2048	2048	280	280
Epoch	1000	1000	1000	1000
Training samples	1000	1000	1000	1000
Activation function	Tanh	Tanh	Tanh	Tanh
Optimizer	SGD	SGD	SGD	SGD
#Layer	1	1	1	2
#Layer's Node	4096	3307	4096	(4046,3538)
Batch sizes	100	50	100	200
Learning rate	0.1	0.15	0.1	0.375

Table 4: Fully connected neural networks performance

Goldstein				
Neural network	Classic		DYNAWEIL	
Hyper-parameters	Initial	Optimized	Initial	Optimized
MAE	0.0794	0.0301	0.0121	0.0117
MRE	102.364	50.042	7.878	7.171
radiative transfer				
Neural networ	Classic		DYNAWEIL	
Hyper-parameters	Initial	Optimized	Initial	Optimized
MAE	0.807	0.730	0.732	0.728
MRE	0.662	0.575	0.572	0.570

For Goldstein problem, the classic neural network performance with optimized hyper-parameters

are improved 2.6 and 2 times better for MAE and MRE, respectively, comparing to another classical networks with the non-optimized hyper-parameters. Without optimized hyper-parameters, the performances of the DYNAWEIL neural networks are improved 6.56 and 13 times for MAE and MRE, respectively, comparing to the performances of the classical networks. Furthermore, the DYNAWEIL neural network performance, without optimized hyper-parameters, are improved 2.48 and 6.35 times for MAE and MRE respectively comparing to the classical one with optimized hyper-parameters. As for the DYNAWEIL neural network performance with optimized hyper-parameters are not further improved comparing to that without optimized hyper-parameters. We conclude, for this problem, that using DYNAWEIL, alone, without optimized hyper-parameters, can improve the classical neural network performance, even with optimized hyper-parameters.

For radiative transfer problem, the classic neural network performance with optimized hyper-parameters are improved 1.1 and 1.16 times for MAE and MRE, respectively, comparing to the network non-optimized hyper-parameters. As for DYNAWEIL neural networks with and without optimized hyper-parameters, its performance is as good as the classical neural network performance with optimized hyper-parameters.

4 CONCLUSION

We proposed a dynamic weighted loss function and its algorithm (DYNAWEIL) to improve the neural network performance. Two physical problems, Goldstein and radiative transfer equations were used for generating the datasets. First, DYNAWEIL's parametric study was performed. We observe that the suitable DYNAWEIL's parameters can improve neural network performance. Second, the optimized hyper-parameter searches using optuna for the above datasets were carried out. Then, we compared different neural network models to see the impact of the dynamic weighted loss function. We observed that, for a random DYNAWEIL's parameters, DYNAWEIL neural network performance without optimized hyper-parameters is, at least, as good as the classical one with optimized hyper-parameters.

5 ACKNOWLEDGEMENT

The authors also thank the industrial partners for their supports in the framework of the HSA project [13]. This project is as well supported by the French government's aid in the framework of PIA (Programme d'Investissement d'Avenir) for Institut de Recherche Technologique SystemX.

REFERENCES

[1] Antoine Bensalah, Patrick Joly, and Jean-Francois Mercier. Mathematical analysis of goldstein's model for time-harmonic acoustics in flows. *ESAIM: Mathematical Modelling and Numerical Analysis*, 56(2):451–483, 2022.

[2] Michael F Modest and Sandip Mazumder. *Radiative heat transfer*. Academic press, 2021.

[3] Yuri Sousa Aurelio, Gustavo Matheus De Almeida, Cristiano Leite de Castro, and Antonio Padua Braga. Learning from imbalanced data sets with weighted cross-entropy function. *Neural processing letters*, 50:1937–1949, 2019.

[4] Zhenyu Wu, Yang Guo, Wenfang Lin, Shuyang Yu, and Yang Ji. A weighted deep representation learning model for imbalanced fault diagnosis in cyber-physical systems. *Sensors*, 18(4):1096, 2018.

[5] K Ruwani M Fernando and Chris P Tsokos. Dynamically weighted balanced loss: class imbalanced learning and confidence calibration of deep neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 33(7):2940–2951, 2021.

[6] M Ruiz-Garcia, G Zhang, SS Schoenholz, and AJ Liu. Tilting the playing field: Dynamical loss functions for machine learning in international conference on machine learning.(pmlr). 2021.

[7] Weeratunge Malalasekera, Hendrik K Versteeg, Jonathan C Henson, and JC Jones. Calculation of radiative heat transfer in combustion systems. *Clean Air*, 3(1):113–143, 2002.

[8] AR Curtis. A statistical model for watervapour absorption. *QJ Roy. Met. Soc.*, 78:639–640, 1952.

[9] WL Godson. The evaluation of infra-red radiative fluxes due to atmospheric water vapour. *Quarterly Journal of the Royal Meteorological Society*, 79(341):367–379, 1953.

[10] John R Howell, M Pinar Mengüç, Kyle Daun, and Robert Siegel. *Thermal radiation heat transfer*. CRC press, 2020.

[11] FC Lockwood and NG Shah. A new radiation solution method for incorporation in general combustion prediction procedures. In *Symposium (international) on combustion*, volume 18, pages 1405–1414. Elsevier, 1981.

[12] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2623–2631, 2019.

[13] https://www.irt systemx.fr/projets/hsa/.