Optimizing Hyperparameters of a Multi-Scale Convolutional Neural Model Tailored to Describe Amorphous Materials Behavior

Marek Pakosta
Dept. of Process Control
Faculty of Electrical Engineering and Informatics,
University of Pardubice, Czech Republic
pakostamarek@gmail.com

***Abstract*— This paper presents an optimization procedure of parameters in a multi-scale neural model, which is crucial for accurately characterizing the behavior of amorphous materials; more specifically glass transition kinetics, which is considered one of the most important yet not fully understood phenomena of solid-state physics and chemistry, with wide-ranging applications. Through systematic exploration of a hyperparameter grid space and rigorous evaluation of resultant models, a highly effective configuration was identified – Conv1D kernel size=8-24-48, Conv1D #filters=16 & Dense #neurons=16 – which offers optimal performance with remarkably short mean epoch times. Our findings suggest a promising strategy of increasing kernel size while decreasing the number of filters and neurons in the dense layer, supported by the superior performance of key competitors sharing this trend. However, further examination reveals comparable performance levels among subsequent competitors, indicating the need for additional samples to draw definitive conclusions. Our experiment highlights the intricate relationship between model accuracy and computational resources, emphasizing the necessity for further results to gain a comprehensive understanding of hyperparameter impact. Three promising combinations for future experimentation emerge, with Conv1D kernel size=8-24-48, Conv1D #filters=16 & Dense #neurons=16 standing out as the clear winner for practical application.**

I. Introduction

Glass transitions represent a pivotal phenomenon within the realm of amorphous materials, holding substantial promise for diverse applications spanning electronics, medicine, and various industrial sectors [1], [2]. The glass transition temperature stands out as a crucial characteristic of amorphous materials, marking the point where the material transitions from a glassy solid to a supercooled liquid. This temperature significantly impacts the mechanical, thermal, and transport properties of amorphous materials. Therefore, accurate determination and understanding of glass transition temperatures and their related parameters are indispensable for effectively leveraging and comprehending these materials. The Tool-Narayanaswamy-Moynihan (TNM) model [3], [4] is a widely used empirical model, that explains the enthalpy relaxation patterns of amorphous materials in proximity to their glass transition temperature.

The work has been supported by the SGS grant at the Faculty of Electrical Engineering and Informatics, University of Pardubice, Czech Republic. This support is very gratefully acknowledged.

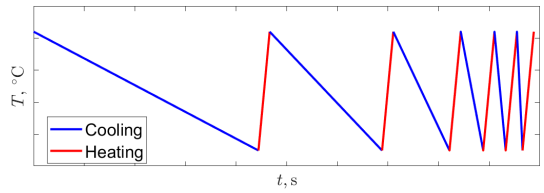
While the TNM model is extensively employed, it faces several limitations. A primary challenge lies in establishing suitable values for its parameters. This task is frequently intricate due to the interdependence of parameters, coupled with potential variations based on the material and experimental conditions. Consequently, achieving a precise determination of TNM model parameters becomes crucial for making accurate predictions concerning enthalpy relaxation behavior and glass transition temperatures. The application of deep learning shows a promising strategy for addressing the main challenge of TNM model, that is determination of its parameters.

In the realm of deep learning, multi-scale convolutional Neural Networks (MCNNs) have emerged as a prominent paradigm for signal analysis. These networks are engineered to capture information across multiple spatial scales, enabling them to process and comprehend signals with varying levels of detail. Notably, MCNNs offer a significant advantage by adeptly extracting both fine-grained and high-level features from input signals, rendering them resilient to changes in signal frequency, modulation, and current state [5], [6]. Moreover, it is noteworthy that MCNNs demonstrate exceptional resilience to noise and parasitic signals, further highlighting their robustness in practical applications [7].

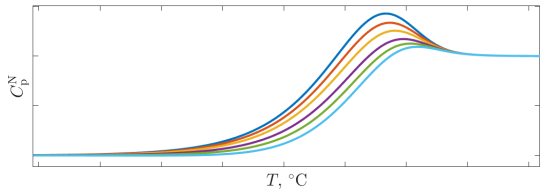
To address the main challenge of determining TNM model parameters, we have, in our previous study [8], implemented one such multi-scale convolutional neural network. Our results has shown, that the proposed neural model can effectively extract the desired values from the input data with high accuracy. Thus, in pursuit of enhancing its suitability for real-time applications and facilitating further development, a commitment has been made to optimize its hyperparameters. The core goal of this research is to achieve maximal accuracy with minimal time investment, emphasizing the primary motivations behind hyperparameter optimization. By fine-tuning the hyperparameters of the MCNN, the aim is to achieve improved efficiency in both training and inference phases. Optimization holds the key to unlocking the network's latent capabilities, enhancing its ability to discern intricate patterns and extract parameters with higher accuracy. [9] The optimization process entails constructing a grid of diverse hyperparameter combinations, training each model for a fixed number of epochs, and

subsequently conducting a comprehensive evaluation of the most promising combination.

In summary, the optimization of hyperparameters for the MCNN is a deliberate effort aimed at enhancing efficiency in parameter determination while steadfastly adhering to commitment to accuracy and reproducibility. The subsequent sections delve into the specific strategies employed for hyperparameter optimization and their corresponding impact on the MCNN's performance in the context of parameter extraction tasks.



(a)



(b)

Fig. 1: Example of input temperature program – constant heating rate cycles (a). Example of output relaxation signals used in the dataset depending on both the input temperature program (b) & TNM model's A, Δh^*, x and β values. Note that 6 output relaxation signals corresponding to the cooling part of the temperature subprogram are not shown because these are not of interest.

II. Materials and Methods

A. TNM model and its parameters

The TNM model can be expressed by the following equations.

$$\Phi(t) = \exp\left[-\left(\int_0^t \frac{\mathrm{d}t}{\tau(T,T_f)}\right)^{\beta}\right], \tag{1}$$

$$\tau(T,T_f) = A \cdot \exp\left[\chi\frac{\Delta h^*}{RT} + (1-\chi)\frac{\Delta h^*}{RT_f}\right], \tag{2}$$

where: $\Phi(t)$ is the relaxation function of the given property, t is time, τ is the relaxation time, β is the nonexponentiality parameter, A is the pre-exponential factor, χ is the nonlinearity parameter, Δh^* is the apparent activation energy of the structural relaxation, R is the universal gas constant, T is temperature, and T_f is the fictive temperature, which is defined as the temperature of the undercooled liquid with the same structure as that of the relaxing glass. [10]

Each phenomenon described by the TNM model is determined by four parameters: A, Δh^*, x and β. The other variables of the model are either determined by the ambient conditions or are universal natural constants. See [10], [11] for the detailed information about this model and its features.

In our previous study [8], we have employed a multi-scale convolutional neural model, which processes the DSC curves (Fig. 1b), and, consequently, it provides the four parameters of interest: A, Δh^*, x and β.

B. Dataset

To gather the necessary data for training the neural model, a collection of input-output pairs must be compiled. Here, the input comprises six DCS curves, while the output consists of four TNM model parameters (A, Δh^*, x and β).

To compute relaxation signals, a carefully designed temperature input program must be used, more specifically a standard constant heating rate (CHR) cycles. In the CHR cycles temperature program, the cooling rates vary with each cycle, while the heating rate remains constant throughout, see Fig. 1a. Although the primary focus is on how the system responds during the heating phase, it is essential to perform simulation for complete temperature program, that is for both the cooling and heating step. This is because the relaxation behavior of the material is affected by its complete thermal history, as indicated by the integral in Eq. (1).

Keep in mind, that both the input temperature program & TNM model parameters (A, Δh^*, x and β) uniquely determine relaxation signals; example of relaxation signals can seen on Fig. 1b. Discrete combinations of A, Δh^*, x and β then determine the explored state space, effectively forming a database of input-output pairs. The ranges of each parameter are defined on the basis of naturally occurring values in real amorphous materials, and are listed in Tab. I.

TABLE I: TNM model parameters, their ranges & units

Property	Range	Unit
A	$-18, -630$	–
Δh^*	$2 \cdot 10^5, 12 \cdot 10^5$	$\mathrm{J \cdot mol^{-1}}$
x	$0.2, 1$	–
β	$0.2, 1$	–

There is an additional limit imposed on $\ln(A)$ values, which is based on current value of Δh^*, this limit is to ensure that values are physically meaningful, these two limit inequalities are not further described for brewity, but can be seen on Fig. 3c. In these limits a 10^5 samples is obtained randomly with a uniform probability distribution, see scatter of Δh^*, $\ln(A)$ values on Fig. 3c, each point on the scatter has its own value of x and β. The dataset is then shuffled and separation is done in $70-15-15$ % ratio to obtain training, validation, and testing set, respectively.

C. Multi-scale convolutional neural model

After conducting an extensive review of existing literature, we have – in our previous study [8] – developed an initial architecture of a multi-scale neural model aimed at extracting TNM model parameters (A, Δh^*, x and β) from DSC

TABLE II: Optimisation results sorted by final val_mae metrics

#	kernel_size	nr_filters	nr_neurons	final loss	final mae	final val_loss	final val_mae	mean epoch time, s
1	2-6-12	16	16	1.716e-02	9.881e-02	1.498e-02	8.830e-02	6.637e+02
3	2-6-12	16	64	1.647e-02	9.562e-02	1.481e-02	8.655e-02	6.705e+02
2	2-6-12	16	32	1.621e-02	9.471e-02	1.466e-02	8.509e-02	6.687e+02
22	8-24-48	32	16	1.064e-02	7.719e-02	1.223e-02	8.487e-02	3.513e+03
5	2-6-12	32	32	1.462e-02	8.934e-02	1.342e-02	8.383e-02	8.950e+02
8	2-6-12	64	32	1.535e-02	9.169e-02	1.381e-02	8.347e-02	1.939e+03
4	2-6-12	32	16	1.582e-02	9.409e-02	1.352e-02	8.258e-02	8.936e+02
10	4-12-24	16	16	1.588e-02	9.359e-02	1.355e-02	8.192e-02	7.127e+02
12	4-12-24	16	64	1.503e-02	9.045e-02	1.330e-02	8.158e-02	6.962e+02
6	2-6-12	32	64	1.453e-02	8.893e-02	1.300e-02	8.009e-02	9.035e+02
7	2-6-12	64	16	1.432e-02	8.903e-02	1.206e-02	7.832e-02	1.969e+03
15	4-12-24	32	64	1.413e-02	8.779e-02	1.217e-02	7.749e-02	1.815e+03
11	4-12-24	16	32	1.405e-02	8.738e-02	1.190e-02	7.676e-02	6.868e+02
20	8-24-48	16	32	1.417e-02	8.733e-02	1.203e-02	7.608e-02	6.833e+02
14	4-12-24	32	32	1.345e-02	8.483e-02	1.162e-02	7.570e-02	1.800e+03
17	4-12-24	64	32	1.181e-02	7.950e-02	1.274e-02	7.555e-02	3.512e+03
13	4-12-24	32	16	1.414e-02	8.803e-02	1.201e-02	7.555e-02	1.773e+03
21	8-24-48	16	64	1.374e-02	8.536e-02	1.189e-02	7.481e-02	6.883e+02
18	4-12-24	64	64	1.212e-02	8.072e-02	1.065e-02	7.324e-02	3.448e+03
9	2-6-12	64	64	1.313e-02	8.345e-02	1.116e-02	7.242e-02	1.968e+03
16	4-12-24	64	16	1.291e-02	8.396e-02	1.045e-02	7.158e-02	3.442e+03
24	8-24-48	32	64	1.260e-02	8.167e-02	1.068e-02	7.149e-02	3.590e+03
23	8-24-48	32	32	1.291e-02	8.408e-02	1.061e-02	7.103e-02	3.389e+03
27	8-24-48	64	64	1.132e-02	7.763e-02	1.013e-02	7.018e-02	6.711e+03
26	8-24-48	64	32	1.142e-02	7.859e-02	9.508e-03	6.752e-02	6.595e+03
25	8-24-48	64	16	9.581e-03	7.289e-02	7.343e-03	6.284e-02	6.650e+03
19	8-24-48	16	16	1.032e-02	7.585e-02	7.644e-03	6.255e-02	6.835e+02

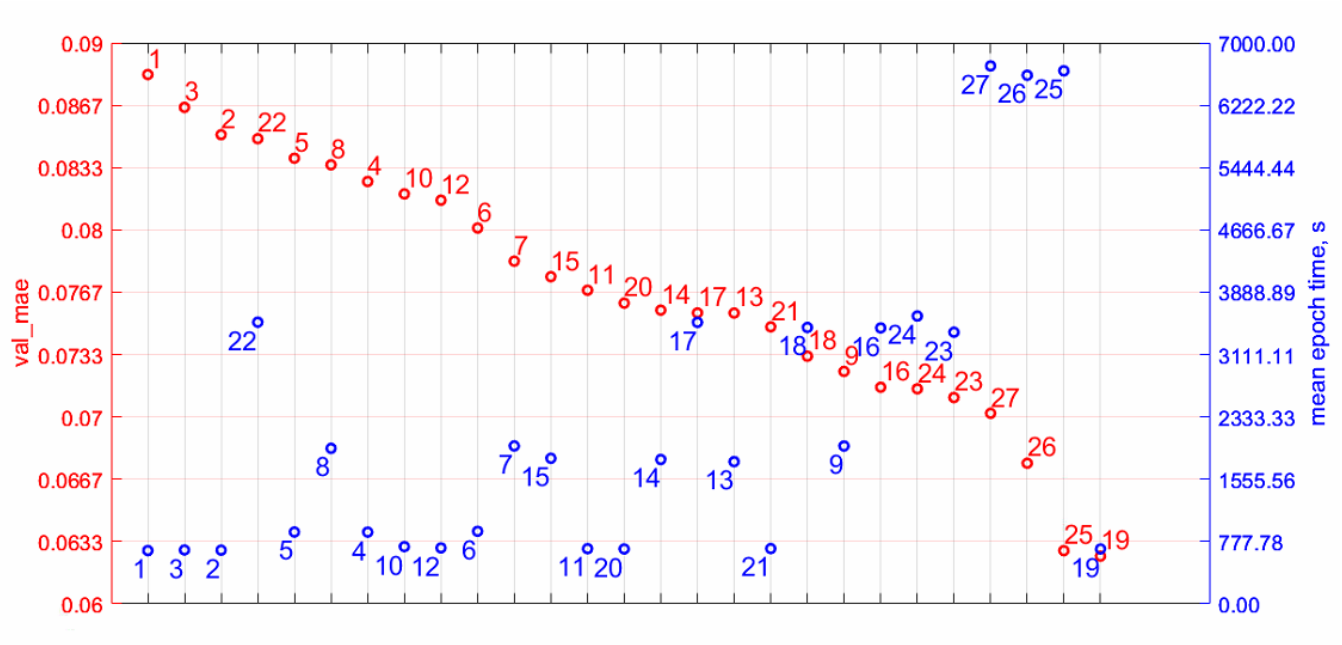


Fig. 2: Optimisation results: final val_mae versus mean epoch time

curves. The model's multi-scalability is achieved through its opening section, comprising three parallel and independent branches. Each branch incorporates an extraction module designed to capture features of varying natures from the input data, operating at different frequency scales. Additionally, the extraction module divides the six DSC curves into individual branches, processing each curve separately. The outputs from all branches are then combined, and the resulting signal undergoes further processing through a series of dense layers – here a `n_neurons` parameter has to be chosen. The output

of the multi-scale neural model consists of four neurons with linear activation functions, each representing one of the TNM model parameters. The overall architecture, as implemented in the experiments, is illustrated in Figure 3a. Furthermore, the architecture of the extraction module is detailed in Figure 3b. The kernel size of the convolutional layer is different for each of the three modules. Additionally, the lower module comprises the original set of DCS curves, while the middle and upper modules receive downsized inputs, reduced to one half and one third of the original set of DCS curves, respectively. Hence the different value of kernel size.

D. Training details

As mentioned previously, the MCNN model has the following hyperparameters, that are primal candidates for optimisation:

- Conv1D #filters – In the context of the Multi-channel Convolutional Neural Network (MCNN), the `nr_filters` parameter determines the number of output filters, representing the dimensionality of the output space.
- Conv1D kernel size – The `kernel_size` parameter specifies the length of the 1D convolution window, controlling the receptive field of the filters over the input sequence.
- Dense #neurons – The `nr_neurons` parameter (referred as `units` in documentation) in the Dense layer is a positive integer that signifies the dimensionality of the output space.

Hence, the objective is to methodically devise various configurations of these parameters, creating a grid of settings. These settings will serve as the foundation for generating a corresponding set of models. Subsequently, these models will be subjected to a training process for a fixed number of 30 epochs, as time constraints has not allowed for more repetitions. After 30 epochs, models they undergo qualitative evaluation. To achieve this, three values were chosen for each hyperparameter, culminating in a total of 27 unique combinations (See Fig. 3e).

In addition to varying hyperparameters, all models undergo training under identical conditions. Specifically, the Stochastic Gradient Descent (SGD) algorithm serves as the optimizer, chosen for its widely acknowledged merits: efficiency, strong generalization capabilities, scalability to large datasets, and straightforward implementation [12]. Due to a stochastic nature of training, each model is trained on fixed number of epochs. The training process is evaluated using the Mean Square Error loss function, while validation employs the Mean Absolute Error function. A summary of the training parameters is presented in Table III.

TABLE III: Parameters of training

Input shape	18501×6
Output shape	4×1
Training algorithm	SGD algorithm
Loss function	Mean square error
Validation metric	Mean absolute error
Number of training experiments	1
Maximum epochs	30
Stopping criterion	Maximum epochs reached
Learning rate α	0.01
Batch size	128

III. Results and discussion

Several trends can be observed based on results summarized in Tab. II & visualised on Fig. 2; these results are discussed in this section. The analysis primarily identified a highly effective configuration, denoted as point #19, which not only yields the best value but also operates within the shortest time range.

The three leading competitors (2nd, 3rd and 4th setting) characterized by their performance measured in final val_mae, exhibit the longest training times among all models. They share a commonality: they possess the highest kernel size and number of filters. Conversely, the favored model features the highest kernel size yet the lowest number of filters and dense layer neurons. This observation suggests that increasing the kernel size while decreasing the number of filters and neurons in the dense layer is a viable approach. Furthermore, an analysis of the impact of either the number of filters or neurons reveals that reducing the number of neurons leads to an improvement in quality. Notably, all three competitors share the same number of filters.

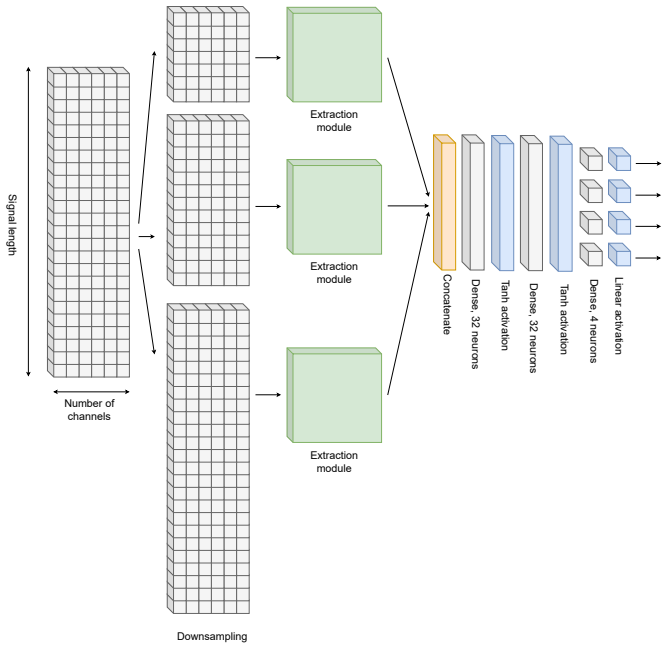
The subsequent competitors in the sequence, represented by settings #16, #23, and #24, demonstrate low final val_mae scores and moderate training times. While their hyperparameters vary, making it challenging to determine the influence of specific settings, these points exhibit similar performance levels in both metrics. Therefore, drawing further conclusions based solely on these samples would be premature.

Nevertheless, in the absence of the optimal setting #19, settings #9 and #21 emerge as promising alternatives. Despite their final val_mae scores being slightly lower than those of points #16, #23, and #24, settings #9 and #21 boast significantly shorter training times. This favorable balance of quality and efficiency positions them as attractive choices for further consideration. Together with the optimal setting #19, these points could provide valuable insights for both future experiments and practical applications.

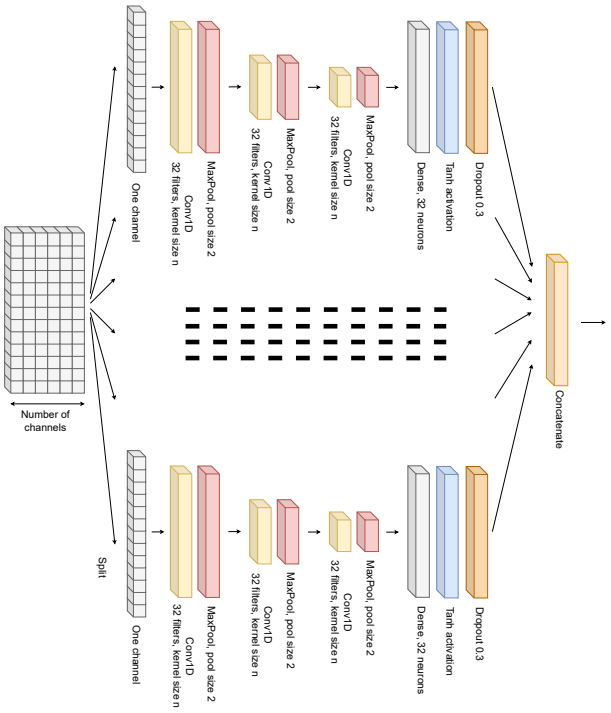
It can also be observed, that mean epoch time metrics generally improve as final val_mae worsens. Lower values of kernel size lead to the worst final val_mae values, but at the same time, this parameter influences training time the most. The relationship between time metrics and final val_mae scores reveals a notable trend: as final val_mae scores worsens, time metrics tend to improve. Notably, lower values of kernel size correspond to poorer final val_mae scores, yet simultaneously exert the most significant influence on training time.

IV. CONCLUSIONS

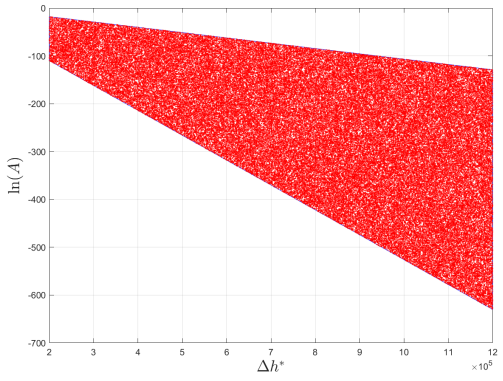
This paper successfully achieves the objective of optimizing parameters for the multi-scale neural model, a crucial and



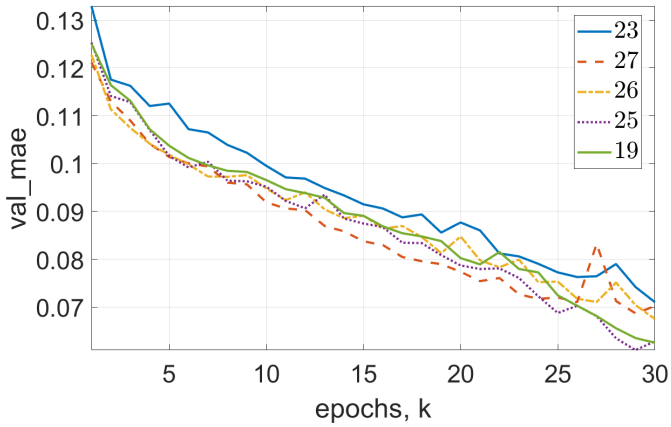
(a) Overall architecture of implemented a multi-scale convolutional neural model. [8]



(b) Extraction module. Input multi-channel signal is split into six separate branches, each of them processed individually. [8]



(c) Dataset, scatter of (Δh^*, $\ln(A)$) values



(d) Evolution of top 5 val_mae metrics

(e) Used hyperparameter combinations

kernel_size	nr_filters	nr_neurons	kernel_size	nr_filters	nr_neurons	kernel_size	nr_filters	nr_neurons
2-6-12	16	16	4-12-24	16	16	8-24-48	16	16
2-6-12	16	32	4-12-24	16	32	8-24-48	16	32
2-6-12	16	64	4-12-24	16	64	8-24-48	16	64
2-6-12	32	16	4-12-24	32	16	8-24-48	32	16
2-6-12	32	32	4-12-24	32	32	8-24-48	32	32
2-6-12	32	64	4-12-24	32	64	8-24-48	32	64
2-6-12	64	16	4-12-24	64	16	8-24-48	64	16
2-6-12	64	32	4-12-24	64	32	8-24-48	64	32
2-6-12	64	64	4-12-24	64	64	8-24-48	64	64

Fig. 3: Neural architecture, visualistaion of dataset, training metrics and grid combinations.

innovative step towards accurately describing the behavior of amorphous materials.

Central to the methodology was the design of an extensive dataset and the systematic exploration of a grid state space of hyperparameters. Through rigorous training and comprehensive evaluation of resultant models, a valuable insights were gleaned.

Our analysis has led us to identify a highly effective configuration – Conv1D kernel size=8-24-48, Conv1D #filters=16, Dense #neurons=16 – which not only delivers optimal performance but also boasts remarkably short mean epoch times compared to its counterparts. Notably, the findings suggest a promising strategy of increasing kernel size while decreasing the number of filters and neurons in the dense layer, as evidenced by the superior performance of three key competitors sharing this trend.

However, further examination of subsequent competitors reveals comparable performance levels, making it challenging to discern the specific influence of individual settings. This underscores the need for additional samples to draw more definitive conclusions.

Moreover, the experiment has unveiled a notable pattern: a consistent decrease in final validation mean absolute error alongside increasing mean epoch time, highlighting the intricate relationship between model accuracy and computational resources.

However, to gain a comprehensive understanding of the impact of hyperparameters on this function, further results are necessary. Ultimately, the strength of observations and analyses would benefit from additional data. In conclusion, three emerge as the most promising starting points for future experimentation, more specifically the following combinations of hyperparameters: Conv1D kernel size=2-6-12, Conv1D #filters=64, Dense #neurons=64, Conv1D kernel size=8-24-48, Conv1D #filters=16, Dense #neurons=64 & Conv1D kernel size=8-24-48, Conv1D #filters=16, Dense #neurons=16 Among these setting the latter stands out as the clear winner in terms of practical application. Moreover, the prediction time is well within application demands, that is obtaining results promptly by skilled personnel; this applies to all configurations. Specifically, the prediction for a single input takes approximately 15 milliseconds for the best configuration.

V. Data and software availability

This dataset is a product of a precisely crafted and optimized real-time glass relaxation software suite that creatively incorporates and evolves on the outlined principles. Moreover, this software harnesses parallelization and offers a versatile range of settings – approximately 30 program options for simulations alone – allowing customization to diverse simulation requirements. For more details about the licensed software and how to use it, refer to https://numcraft.eu/.

References

[1] Y. V. Kuznetsova and I. D. Popov, "Design and technological aspects of novel cds quantum dots doped glass–ceramics," *Ceramics International*, vol. 48, no. 13, p. 18972 – 18982, 2022.

[2] M. Wuttig and N. Yamada, "Phase-change materials for rewriteable data storage," *Nature Materials*, vol. 6, no. 11, p. 824 – 832, 2007.

[3] C. T. Moynihan, A. J. Easteal, M. a. De Bolt, and J. Tucker, "Dependence of the fictive temperature of glass on cooling rate," *Journal of the American Ceramic Society*, vol. 59, no. 1-2, p. 12 – 16, 1976.

[4] O. Narayanaswamy, "A model of structural relaxation in glass," *Journal of the American Ceramic Society*, vol. 54, no. 10, p. 491 – 498, 1971.

[5] S. W. Chen, S. L. Wang, X. Z. Qi, S. M. Samuri, and C. Yang, "Review of ecg detection and classification based on deep learning: Coherent taxonomy, motivation, open challenges and recommendations," *Biomedical Signal Processing and Control*, vol. 74, 2022.

[6] Q. Zhang, D. Zhou, and X. Zeng, "Heartid: A multiresolution convolutional neural network for ecg-based biometric human identification in smart health applications," *IEEE Access*, vol. 5, p. 11805 – 11816, 2017.

[7] N. Liu, Y. Long, C. Zou, Q. Niu, L. Pan, and H. Wu, "Adcrowdnet: An attention-injective deformable convolutional network for crowd understanding," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2019-June, p. 3220 – 3229, 2019.

[8] M. Pakosta, P. Dolezel, R. Svoboda, and B. B. Zanón, "Multi-scale neural model for tool-narayanaswamy-moynihan model parameter extraction," in *18th International Conference on Soft Computing Models in Industrial and Environmental Applications (SOCO 2023)*. Cham: Springer Nature Switzerland, 2023, pp. 24–33.

[9] I. Sarker, "Deep learning: A comprehensive overview on techniques, taxonomy, applications and research directions," *SN Computer Science*, vol. 2, 08 2021.

[10] R. Svoboda and J. Málek, "Description of enthalpy relaxation dynamics in terms of tnm model," *Journal of Non-Crystalline Solids*, vol. 378, p. 186 – 195, 2013.

[11] I. M. Hodge and A. R. Berens, "Effects of annealing and prior history on enthalpy relaxation in glassy polymers. 2. mathematical modeling," *Macromolecules*, vol. 15, no. 3, p. 762 – 770, 1982.

[12] G. Habib and S. Qureshi, "Optimization and acceleration of convolutional neural networks: A survey," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 7, p. 4244 – 4268, 2022.