Article

Approximating the Steady-state Temperature of 3D Electronic Systems with Convolutional Neural Networks

Monika Stipsitz [1] and Hèlios Sanchis-Alepuz [2,*]

[1] Silicon Austria Labs GmbH, Inffeldgasse 33, 8010 Graz, Austria; monika.stipsitz@silicon-austria.com
[2] Silicon Austria Labs GmbH, Inffeldgasse 33, 8010 Graz, Austria; helios.sanchis-alepuz@silicon-austria.com
* Correspondence: helios.sanchis-alepuz@silicon-austria.com

Abstract: Thermal simulations are an important part in the design of electronic systems, especially as systems with high power density become common. In simulation-based design approaches, a considerable amount of time is spent by repeated simulations. In this work, we present a proof-of-concept study of the application of convolutional neural networks to accelerate those thermal simulations. The goal is not to replace standard simulation tools but to provide a method to quickly select promising samples for more detailed investigations. Based on a training set of randomly generated circuits with corresponding Finite Element solutions, the full 3D steady-state temperature field is estimated using a fully convolutional neural network. A custom network architecture is proposed which captures the long-range correlations present in heat conduction problems. We test the network on a separate dataset and find that the mean relative error is around 2 % and the typical evaluation time is 35 ms per sample (2 ms for evaluation, 33 ms for data transfer). The benefit of this neural-network-based approach is that, once training is completed, the network can be applied to any system within the design space spanned by the randomised training dataset (which includes different components, material properties, different positioning of components on a PCB, etc.).

Keywords: Physics simulations; Neural Networks; Electronic design; Heat equation

1. Introduction

Physics simulations are becoming an essential aspect in the design of electronic systems. For an optimally designed electronic system, the interplay of many different physical domains has to be taken into account. For instance, for the design of an efficient power converter co-simulations including the generation of heat via electronic losses as well as the conduction and heat rejection via forced convection have to be studied [1]. Such coupled simulations lead, however, to large computational requirements. Often, the large simulation times render thorough automatic optimizations of designs impossible.

Machine learning (ML) techniques have been identified as possible candidates to increase the computational speed. The application of machine-learning techniques to the design of electronic systems focuses mainly on two aspects [2]. The first aims at reducing the number of required design iterations, see e.g. [3,4]. As an example of such an approach, an optimal (from a thermal point of view) placement of a chip on a system required, applying genetic algorithms, around 1000-10000 FEM simulations [5], which, however, still takes a considerable amount of time. A second use of ML techniques consists of using model for components based on neural networks (NNs) that can capture the nonlinear behaviour in circuit simulations. Among others, this method has been successfully applied to model an inductor [6], a MOSFET [7], a power diode [8], and a supercapacitor [9]. For the design optimization, sweeps over the design space to evaluate the Pareto front [6] or a combination with conventional design optimization methods [10] are feasible due to the fast evaluation time of these NN based models.

In addition to thermal simulations, the application of different NN architectures to perform physics simulations has been the focus of much research. For instance, the use of NNs has been explored for fluid dynamics problems [11–13], for electroconvection [14], transport problems [15,16], magnetic field estimation [17], classical mechanics [18] and in replicating multi-particle evolutions by learning the pair-wise interaction function [19].

Citation: Stipsitz, M.; Sanchis-Alepuz, H. Approximating the Steady-state Temperature of 3D Electronic Systems with Convolutional Neural Networks. *Preprints* **2021**, *1*, 0. https://doi.org/

Received:
Accepted:
Published:

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 16 December 2021 doi:10.20944/preprints202112.0275.v1

Although these physical phenomena clearly have a three-dimensional nature, the machine learning community has mostly focused on approximations of physical simulations in lower dimensions only. Especially, the representation of three-dimensional full-field solutions for complex geometries remains a challenge. One major difficulty when going from 2D to 3D representations of a system is the limited memory of the GPUs [20] used to train and evaluate NNs. One way to reduce memory requirements would be to use a smaller representation of the geometry instead of images [21]. Although some alternatives, like point clouds based on CAD geometries [22–24] or octrees were proposed [25], most available NN architectures are still developed for images.

For this type of applications, so-called physics-informed NNs have been developed that aim at embedding prior physics knowledge into the design of the NNs [26]. For this purpose, the autogradient feature of the NN is used to construct a loss term based on the underlying differential equation. These techniques require the network to be defined in terms of the absolute coordinate positions, i.e. given the input coordinates, the NN is trained to give the solution at the given position. Thus, these techniques lead to promising results in simple geometries e.g. for flow problems [27]. However, it is not clear how complex geometries could be included. Additionally, advanced ML features like convolutional kernels (which automatically lead to the preservation of translational invariance [28]) cannot be used.

In this paper we explore the possibility of generating full-field 3D approximations of the temperature distribution in an electronic system using NNs. We consider simplified representations of electronic components and systems with no electronic functionality, keeping only the essential properties for the present work to serve as a proof of principle. Here, we focus on the steady-state, equilibrium thermal configuration of the systems since training data can be easily obtained from FEM simulations.

2. Materials and Methods

2.1. Data set generation

In this work a fully-convolutional neural network (FCN) was trained to approximate the 3D FEM solutions of electronic systems using supervised learning. This procedure requires a large dataset of random systems with corresponding FEM solutions. The dataset needs to be representative of the design space. This entails that all properties that one wants to change during the design process (component type / number / placement, material properties, heat sink specifications, external temperature), are randomised in the dataset. In total, 464 unique systems were generated using an automatised workflow in python (see Fig. 1 for a graphical summary):

1. **System generation:** For each system the number and type of components is randomly chosen and they are placed at random locations on a PCB. Material parameters are assigned to the different parts of each component.
2. **Generation of FEM solutions:** Initial and boundary values are randomly chosen, heat sources (i.e. losses) are assigned to some of the components, and the mesher and FEM solver are called.
3. **Voxelization:** Postprocessing of the systems and the FEM solutions to generate a set of 3D images per system as input for the NN.

2.1.1. System generation

The main challenge in generating systems was to make them replicate the complexity of the 3D geometries and variation of material properties while ensuring that the resulting temperature fields stay within a physically plausible range. To this end, all systems were based on six different components. The components were designed manually using `FreeCAD` (one large and one small IC, one large and one small capacitor, and copper layers of different shapes and sizes, see Fig. 1a). Most components (e.g. the ICs) have an internal structure (e.g. legs, die). Each of these sub-part was assigned its typical material (e.g. copper for the legs). To account for variations in the material properties, a physically

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 16 December 2021 doi:10.20944/preprints202112.0275.v1

plausible range was specified for each material type as the average value given in Tab. 1 ±25 %. The actual material properties of each part were randomly chosen (uniformly) from these ranges. The components were randomly placed on a fixed-size PCB of dimensions 25 mm × 25 mm × 2 mm (Fig. 1b). Such a random placing procedure does not produce functional electronic circuits. However, as long as the dataset used for the training of the NN captures the impact that different component placements have on the heat transfer, the NN is expected to generalise also to (unseen) functional electronic circuits.

Table 1. Average Material Properties: The Actual Values are Chosen Randomly from a Range of [0.75avg, 1.25avg].

Property Unit	**Avg.** k $(W/(mK))$	**Avg.** c_P $(J/(kgK))$	**Avg.** ρ (kg/m^3)
Silicon	148	705	2330
Copper	384	385	8930
Epoxy	0.881	952	1682
FR_4	0.25	1200	1900
Al_2O_3	35	880	3890
Aluminum	148	128	1930

The size of the dataset was increased by four via three 90 degree in-plane rotations of each system, so that the total dataset consisted of 1856 systems.

2.1.2. Finite-element simulations

Three types of boundary conditions (BCs) were imposed on the systems for the finite-element (FE) simulations: (1) a Dirichlet-type BC at the bottom of the PCB, which was set to a constant external temperature T_{ext}, taken randomly per system from the range 300 K ± 25 %, (2) a Neumann-type BC represented by a heat sink on the top of the large IC, modelled via a heat transfer coefficient α_{sink} randomly taken from the range 2538.72 $\text{W}/(\text{Km}^2)$ ± 25 % and the external (far-field) temperature (in the steady-state this corresponds to an implicit flux BC), and (3) on all other outer surfaces a heat transfer coefficient α was prescribed, taken randomly from the range 14 $\text{W}/(\text{Km}^2)$ ± 25 %. Electronic losses were modelled as constant, predefined volumetric heat sources located at the silicon die of the ICs and at the core of the capacitors, with values chosen randomly from the ranges specified in Tab. 2.

A conformal mesh was generated using `gmsh` [29], which consisted on average of 3 million elements. Steady-state solutions were obtained with the open-source FEM solver `ElmerSolver`[30,31]. The main advantage of `ElmerSolver` is that a scriptable interface between `FreeCAD` (for automatic geometry generation), to `ElmerGrid` / `gmsh` (for automatic tetrahedral mesh generation) and `ElmerSolver` exists which enables an automatised workflow for the system generation.

Table 2. Range of Applied Heat Sources for the Different Components (in W).

Component	**min**	**max**
Center of large capacitor	0.1	0.3
Silicon die of large chip	10	19
Silicon die of small chip	0.1	0.5

2.1.3. Voxelization

For the NN, the 3D-systems were represented by a stack of 3D-images. This enabled us to apply NN methods previously developed for vision tasks to our problem (Fig. 1d). Specifically, the input of the NN consisted of multiple channels where each channel was a 3D-image that represented one of the physical properties (heat conductivity, external temperature, density, heat capacity, BC values) of the system. To create these 3D-images, the geometries of the components were voxelized independently using a custom `FreeCAD`-python script. The voxel size was 0.19 × 0.19 × 0.05mm so that a batch of ten images fits in

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 16 December 2021 doi:10.20944/preprints202112.0275.v1

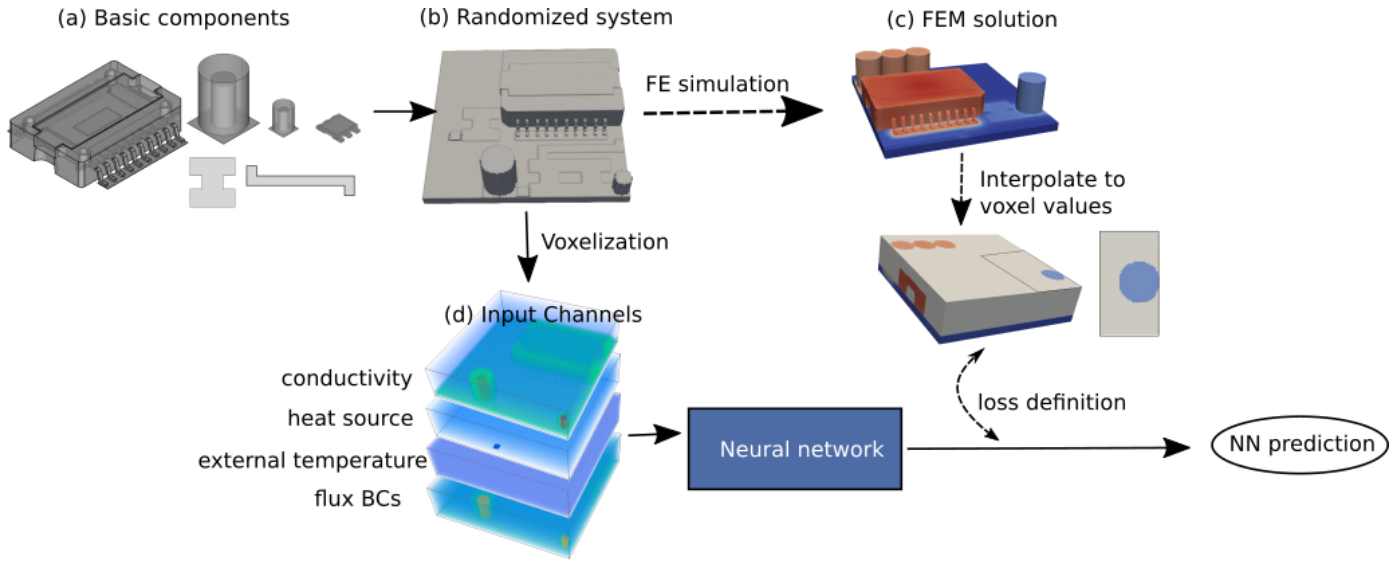

Figure 1. Illustration of the automatised workflow: Randomised systems are generated by randomly choosing and placing basic components. After assigning randomised material properties and BC values, the system is voxelized to create a stack of four 3D-images as input for the NN. Solutions for the supervised training procedure are created using FE simulations.

the GPU memory, while the voxel size still resolves sufficient structural detail. The smaller resolution in z-direction was chosen in order to resolve the thin copper layers. Then, each component in the original systems was replaced by the voxelized representations. To create the different channels, the voxelized geometry in the images was replaced part by part by the corresponding material parameter used in the FEM simulation. For our systems, this procedure led to images with $128 \times 128 \times 128$ voxels.

As labels for the supervised learning procedure the temperature solution obtained by the FEM solver was used. The 3D temperature field was directly voxelized during post-processing by `Elmer`. The FEM solution is only available within the geometry (where a mesh is available). The outer image regions (we call them "air" regions) were labelled as -1 and excluded in the loss definition.

2.2. NN architecture

The input of the NN consists of multi-channel 3D *images* and the output is a single-channel 3D temperature map. To this end, a 3D fully-convolutional NN is used in this work.

Note, first of all, that one of the design goals was to have a network with a relatively small number of parameters and, hence, low memory requirements. The reason is a technical one; the fastest implementations of NNs are currently optimised to run on GPU cards, where the memory available is limited (typically of the order of tens of GBs). Since the 3D systems of this work are represented as 3D images with several channels, which on their own require already a significant amount of the available memory, it is convenient that the network itself consumes a minimum amount of memory. The NN architecture should, thus, be chosen to match the physical problem.

2.2.1. Properties of heat propagation

For the choice of an appropriate NN, it is important to understand the main features of the heat propagation problem. Assuming that the material properties in the system do not change with temperature, the thermal behaviour of the system is determined by the heat equation

$$\rho(x)c_P(x)\left(\frac{\partial T(x,t)}{\partial t} + (\vec{u}\cdot\vec{\nabla})T(x,t)\right) - \vec{\nabla}\cdot\left(k(x)\vec{\nabla}T(x,t)\right) = \rho(x)h(x)\,, \qquad (1)$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 16 December 2021 doi:10.20944/preprints202112.0275.v1

where ρ is the density, c_P the constant-pressure specific heat, k the heat conductivity and h a heat source. All of them depend on the position x (although in the generated systems they are constant over the different parts of each component). The term $(\vec{u} \cdot \vec{\nabla})T$ represents convection, which is ignored for simplicity. A solution of the heat equation is fully determined only after imposing boundary conditions, which in this case are the Dirichlet and Neumann boundary conditions discussed in Sec. 2.1.2. As can be seen, density, heat capacity and heat source do not appear separately in the equation, but rather as the products ρc_P and ρh. This implies that any solution of the equation will depend on those products only and, thus, it is sensible to use them as input to the NN as explained below. Moreover, this work considers the stationary (thermal equilibrium) case only, where the temperature field $T(x)$ is no longer time dependent. If convection is ignored as well, the term ρc_P disappears from the equation, which implies that the solution will not depend on that factor either.

It is also instructive to explore exact analytical solutions of the heat equation, even though they apply in very idealised situations only. A fundamental solution of the heat equation, called a heat kernel solution, is a time dependent solution of the equation for an initial point-like heat source at a known position. This can be extended to a space and time dependent heat source by convolving point-like solutions under an integral. A 1D heat kernel solution for an extended heat source $h(x,t)$ reads

$$T(x,t) = \int_0^t dt' \int_0^\infty dy \frac{h(y,t')}{\sqrt{4\pi k(t-t')}} \left(\exp\left(-\frac{(x-y)^2}{4k(t-t')} \right) - \exp\left(-\frac{(x+y)^2}{4k(t-t')} \right) \right). \quad (2)$$

Formally, a general solution of the heat equation can be found in terms of the heat kernel solution. Inspecting the heat kernel solution, it is possible to anticipate the presence of exponentials as a general feature of solutions of the heat equation and the fact that the contributions from different spatial points are *aggregated* via an integral.

In order to fulfil the requirement of having a small network whilst still preserving key aspects of the heat propagation problem, a custom FCN was designed, as described in the following subsections.

2.2.2. Long-range correlations. Fusion blocks.

One of the most important aspects, that the NN must capture in order to successfully approximate the results of a standard simulation, of the heat problem in the steady-state regime is the presence of long-range effects. Most of the convolutional NNs (CNNs) available in the literature have been developed for the classification of images (in the case of 2D CNNs) or of video sequences (in the case of 3D CNNs). The performance of several of the standard low-resource 3D CNNs (e.g. [32]) on the dataset described in Sec. 2.1.2 was analysed but the temperature fields obtained with such networks were very sharp, not resembling a realistic situation, where around a hot spot a temperature gradient is to be expected. Moreover, heat propagation from a heat source to a distant but thermally connected point in the system was absent.

With the goal of recovering long-range effects in the system, another feature of convolutional layers called dilation was explored, see Eq. A2. In short, usually in a CNN the kernels act on contiguous pixels. The dilation parameters d_h, d_w and d_h in Eq. A2 reflect how many pixels (or voxels) are skipped when applying the kernel, with $d = 1$ meaning a standard convolution, $d = 2$ if one pixel is skipped, etc. Moreover, as stressed above with regards to the heat kernel solution, not only the solution takes long range effects into account, but it also aggregates them (via an integration in the case of the heat kernel solution)). We thus defined what we called Fusion block: Each fusion block is composed of N 3D convolutional layers of kernel size $[k_1, k_2, \ldots, k_N]$ and dilations $[d_1, d_2, \ldots, d_N]$ (equal in the width, height and depth dimensions, that is $d = d_w = d_h = d_d$) and output channels C/N with C the desired number of output channels from the Fusion block. A Fusion block may keep the size of the input or not, depending on the choice of the stride value s (also taking $s = s_w = s_h = s_d$; see Eq. A2). The result of those convolutions is concatenated

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 16 December 2021 doi:10.20944/preprints202112.0275.v1

along the channel dimension. An illustration of a simple (2D) Fusion block is shown in Fig. 2. The Fusion block turned out to be the decisive element to obtain realistic and accurate

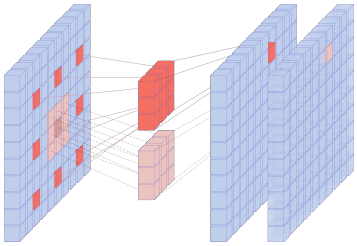

Figure 2. Schematic representation of the action of a 2D Fusion block consisting of two convolutional layers with 3x3 kernels and dilations 3 and 1.

enough predictions from the network.

2.2.3. Choice of activation functions

As explained in Sec. A, a NN is composed of a sequence of linear operations followed by a fixed non-linear operation called activation function. In a regression problem like the present one, a judicious choice of the type of non-linearities allows to keep the network small. From the heat kernel solution above, it can be seen that general solutions make use of the exponential function. It is thus convenient to use non-linearities that contain an exponential. Using too many of them, however, turns out to be damaging for the performance of the network. The reason is that, several consecutive layers containing an exponential function lead to a $\exp(-\exp(-\exp(...)))$-type functions acting on the input. After some exploration, a SELU activation function [33] was chosen for one of the layers only (see Fig. 3)/

$$\mathrm{SELU}(x) = \sigma\left(\max(0,x) + \min(0, \alpha\left(\exp(x) - 1\right))\right), \tag{3}$$

with $\alpha \sim 1.6733$ and $\sigma \sim 1.0507$. For the rest of the layers a LeakyReLU [34] was used

$$\mathrm{LeakyReLU}(x) = \max(0,x) + S\min(0,x), \tag{4}$$

with $S = 0.01$.

2.2.4. Input to the network

Another important aspect to take into account in the development of NN-based solutions is the choice of the input given to the network. As highlighted in Sec. 2.2.1, the density and heat source appear as products only in the heat equation. For the steady-state solutions the heat capacity does not contribute and, thus, will not be fed to the network in this work.

As discussed, on every system boundary conditions of the Dirichlet and Neumann types were imposed, a Dirichlet boundary condition on the bottom of the PCB, whose temperature is fixed to the external temperature value, and a Neumann temperature condition on every other exposed surface, where the heat flux is fixed via a heat transfer coefficient α

$$-k\frac{\partial T}{\partial n} = \alpha\left(T - T_{ext}\right), \tag{5}$$

with n the normal direction.

Moreover, the input of NNs is typically rescaled to be in the $[0,1]$ range since that makes the training of the networks more stable numerically. This can be easily achieved by dividing each of the channels by their maximum value. However, in this work, a more physically meaningful rescaling scheme was chosen. This comes at the expense that each of the input channels is not strictly in the $[0,1]$ range but still numerically small. Namely, characteristic values were introduced for each of the relevant dimensional quantities.

 doi:10.20944/preprints202112.0275.v1

These are the maximum heat power $P_{\text{max}} = 20\,\text{W}$, the maximum temperature $T_{\text{max}} = 1000\,\text{K}$, and a length scale $l_0 = 0.4\,\text{mm}$. Note, that the maximum values defined here do not represent a hard limit at which the network will fail to work since the network does not strictly require all values to be smaller than one. The dimensionless channels are then defined as

$$C_0 = k\frac{T_{\text{max}} l_0}{P_{\text{max}}} \tag{6}$$

$$C_1 = h\rho V_{\text{body}}\frac{1}{P_{\text{max}}} \tag{7}$$

$$C_2 = T_{\text{ext}}/T_{\text{max}} \tag{8}$$

$$C_3 = \frac{1}{\alpha A_{\text{body}}}\frac{P_{\text{max}}}{T_{\text{max}}}. \tag{9}$$

The heat source channel C_1 is defined in terms of the heat source density per mass h, the density ρ and the total volume V_{body} of the part where the heat density is applied. Similarly, the flux boundary condition channel C_3 is defined as the heat transfer coefficient α times the total surface area A_{body} over which the loss is specified. Note that the Dirichlet boundary condition is not fed to the network as an input channel, but softly imposed via the loss (see below).

2.2.5. Network architecture.

For this work, a fairly thorough exploration of possible network architectures that would contain the features discussed so far was performed. As a result the architecture shown in Fig. 3 is proposed. In the figure, k refers to the kernels size, s to the stride, d to the dilations and C to the output channels of each layer (in the three cases the same for all dimensions). This network has only ∼370K trainable parameters, which is a tiny number compared to standard architectures (e.g. the 3D version of SqueezeNet [32,35] has 2.15M parameters). A brief explanation of the choices made is in order:

- It was observed that too many downsampling layers have a damaging effect on the accuracy of the output. Downsampling in CNNs is used to extract useful features from images. In our case, the most relevant features are already part of the input, as discussed above. The main effects of downsampling in our case are to aggregate long-range effects in addition to the dilation in the Fusion blocks and also to reduce memory requirements. Thus, only two downsampling layers were used.
- As is well known in FCNs, skip connections help avoid the usual checkerboard artifacts in the output. In this work three additive skip connections were used in the network, from the output of the Fusion blocks to the input of the two upsampling layers (transpose convolutions; see Sec. A) and the final convolutional layer, respectively.
- An initial depthwise Fusion block (depthwise means that channels are not mixed) provides the necessary additional preprocessing of the input data.

2.2.6. Objective function

For the loss function a mean L_1 relative loss term was combined with a penalty term for the Dirichlet boundary condition at the bottom layer of the PCB. The air region was excluded from the loss evaluation because no FEM solution is available there (note, however, the values in the air region are indirectly influencing the temperature distribution within the system via the convolutions, which, as we will see, has a potentially negative impact on the accuracy of the result). A physics-informed loss as proposed by [13] did not improve the solutions. In fact, it drove the system to a uniform temperature distribution since in all parts of the system where no heat source is present, the differential heat equation error can be minimized by minimizing the temperature gradient. This is somewhat unexpected and further exploration on physics-informed losses for the present problem will be the subject of future work.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 16 December 2021 doi:10.20944/preprints202112.0275.v1

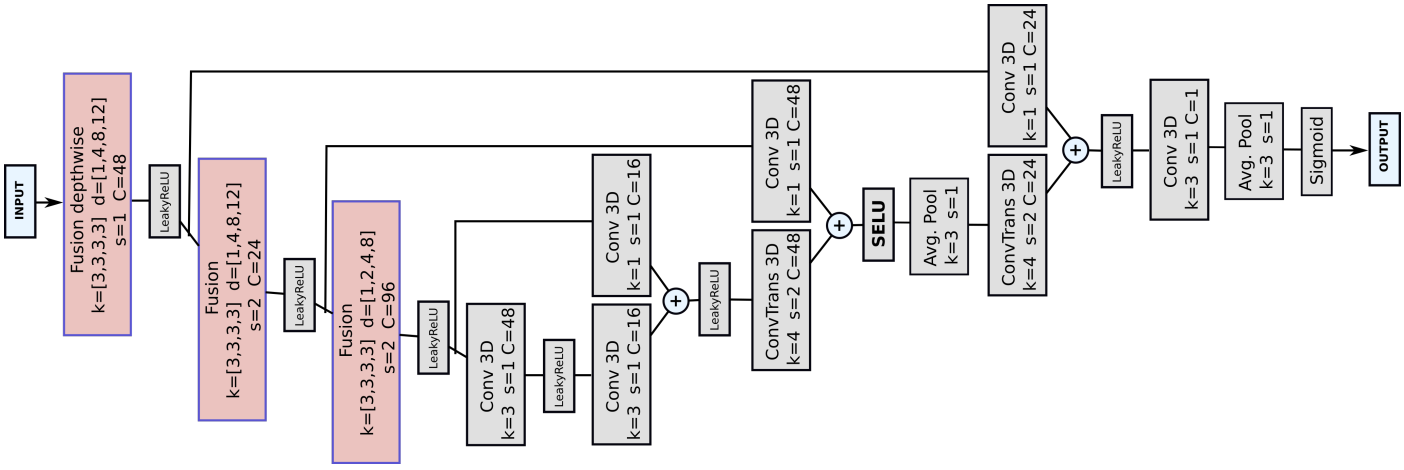

Figure 3. Architecture used in this work. In each block, k refers to the kernels size, s to the stride, d to the dilations and C to the output channels of each layer (in the three cases, the same for all dimensions). See Sec. 2.2 and App. A for further details.

3. Results

From the generated dataset formed by 3D systems and the corresponding FEM solution, only a fraction of it is used to train the NN. Specifically, 75 % of the dataset was used as training set and the remaining 25 % as test set. It is the accuracy in reproducing the latter that truly measures the quality of the network solution.

In Fig. 4 a histogram with the mean absolute value per voxel of the relative error between the FEM and the NN solutions is shown. As can be seen, the mean relative error is at the percent level, with values below 2 % in most systems in the test set. Additionally, the Dirichlet boundary condition was softly imposed via the loss function on the bottom of the PCB. The network reached here an average 0.1 % relative error. The present work was initiated as a proof of principle for an NN-based tool to evaluate the viability of potential system designs from a thermal point of view. In terms of accuracy we judge the values achieved as accurate enough for that goal.

The mean value of the relative error, however, does not fully represent the (in-)accuracy of the NN results, as large but very localised error values can be smeared out when averaging over the large number of voxels in a system. More insightful is thus to compare FEM and NN solutions directly in their 3D representations. In Fig. 4 a representative collection of 3D solutions and 3D maps of the relative temperature differences is shown, picking one sample from some of the bins in the histogram discussed above, as indicated in the figure. For the majority of systems the NN solution reproduces the FEM solution reasonably well. The main discrepancies appear in relation with the small chip and on the surfaces of the different components and PCB.

Further details are shown in Fig. 5, where a section of a selected system is presented. There, on the surface of the PCB and also around the small chip, the L_1 error is somewhat larger. Also, in the inner part of the large chip some localised discrepancies can be seen, which are a reminiscence of the checkerboard artefacts of FCNs. However, these are clearly localised and do not seem to affect the overall temperature of the chip.

The second aspect to consider when judging whether the NN solution presented herein is a valuable tool in the design workflow of electronic systems is the evaluation speed. In Tab. 3 a comparison of the evaluation speeds between the NN and the FEM solutions is shown. The NN was run on an NVIDIA Titan RTX GPU whilst the FEM solver was run on a single thread of an Intel Xeon W-2145 CPU. This implies that the FEM evaluation time in Tab. 3 does not correspond to the time a fully optimised and parallelised FEM simulation would require. Nevertheless, we see that the NN provides a solution in $\sim$ 35 ms (with most of the time, in fact, used in transferring the 3D system to the GPU

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 16 December 2021 doi:10.20944/preprints202112.0275.v1

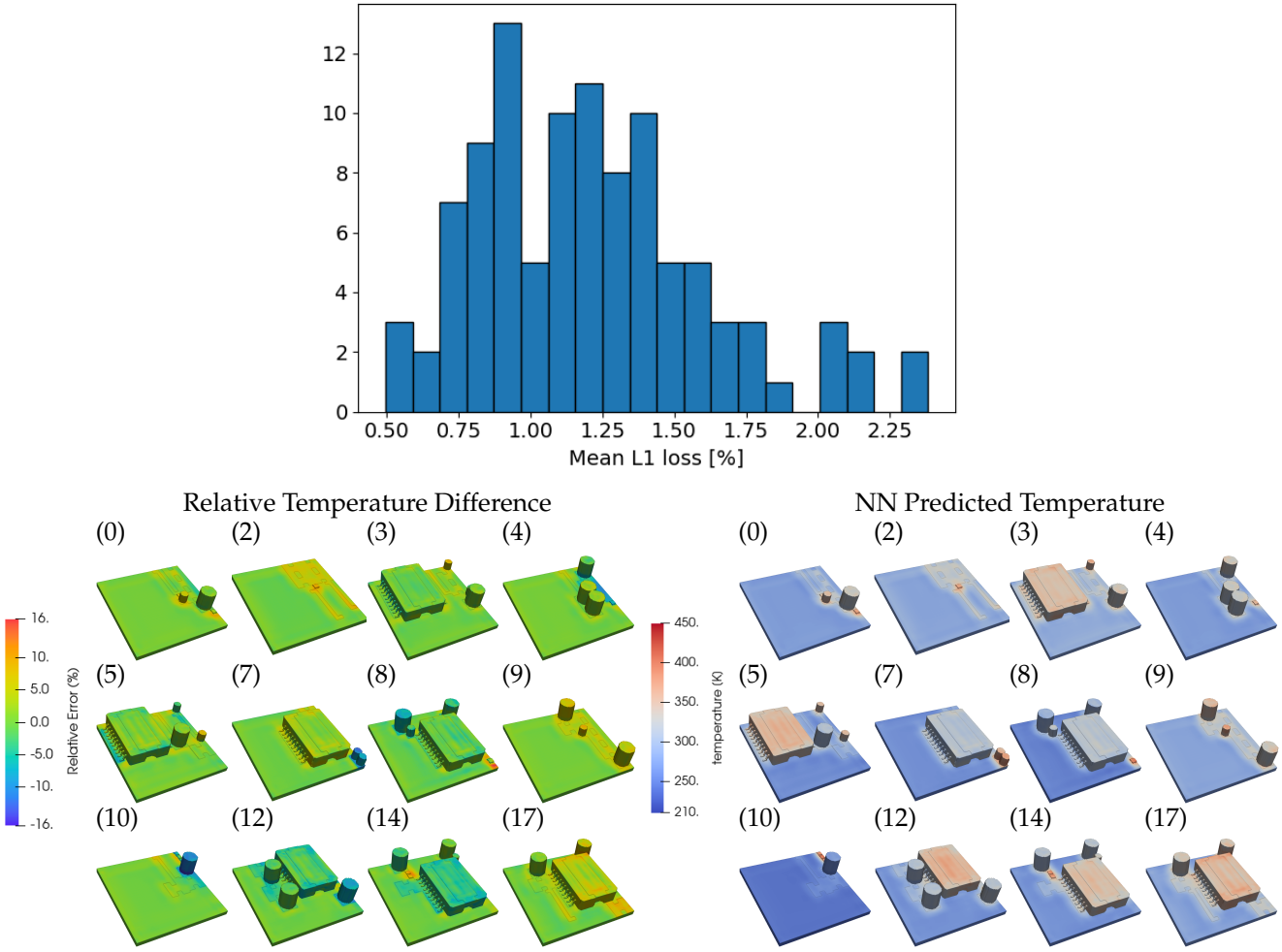

Figure 4. Histogram of the average relative L_1 error per test system (top). Below the temperature distributions estimated by the NN (right) and the relative temperature difference (left) for selected systems of the test dataset (the corresponding error bin of the histogram is indicated in brackets, from 0 indicating the lowest mean error to 19 for the worst mean error). The average relative L_1 error (top) is the mean of the absolute values of the relative temperature differences (bottom left) per sample.

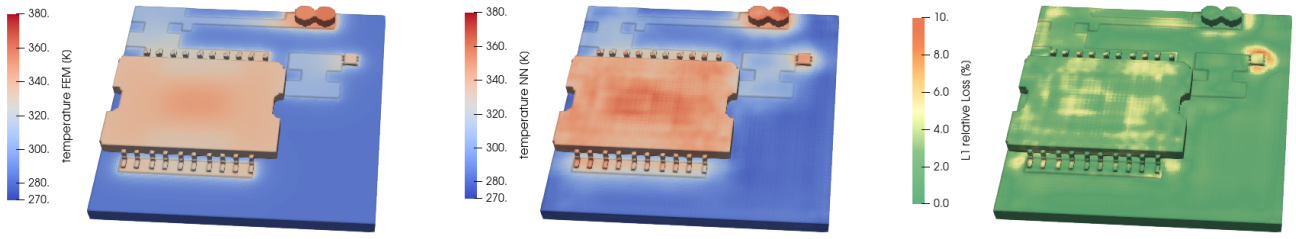

Figure 5. FE solution (left), NN prediction (center), and relative L_1 error of the temperature distribution (right) on a horizontal cut of a selected system. High predictive errors are mostly found on the surface of the system while the internal temperature distribution is well represented.

memory), which is certainly a significant speed-up over any conceivable full-field 3D FEM simulation.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 16 December 2021 doi:10.20944/preprints202112.0275.v1

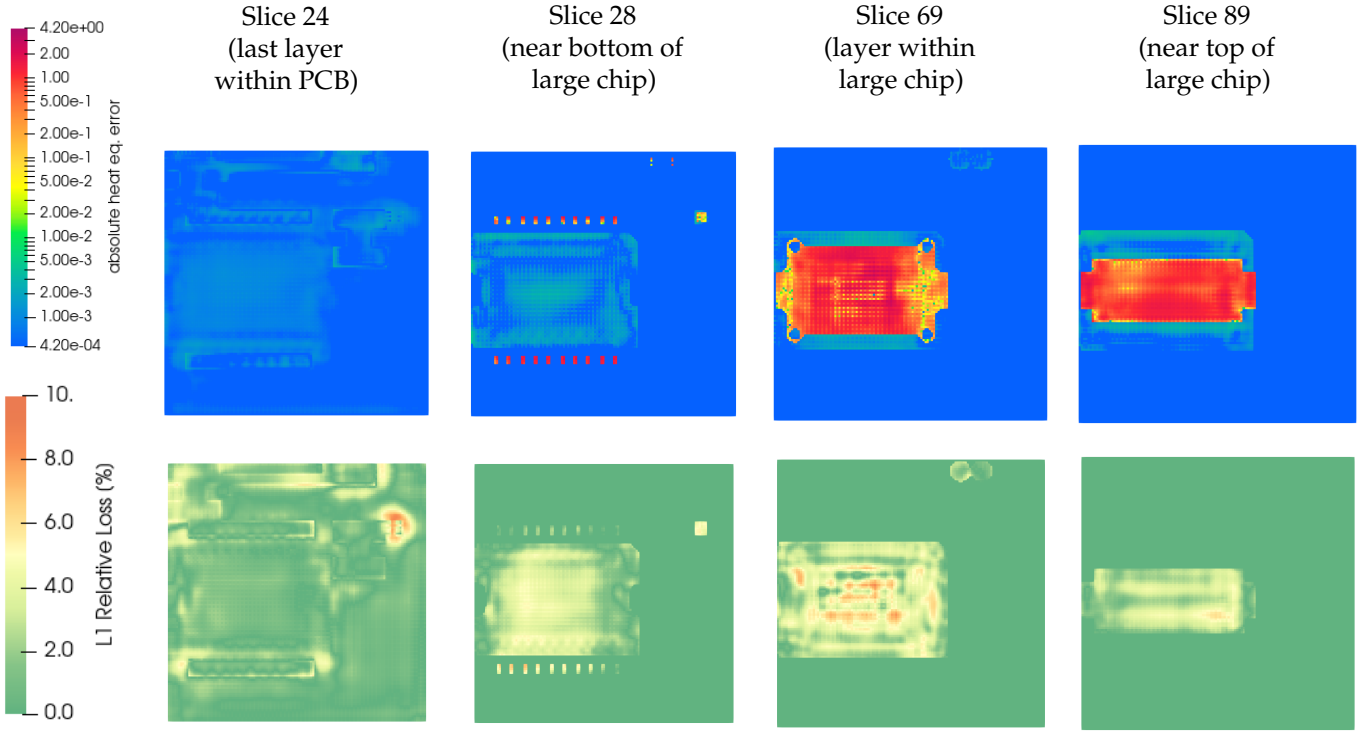

Figure 6. Comparison of the heat equation error (top row), which can be used as error predictor if no FE solution is available, and the L_1 error (bottom row) on selected slices from bottom to top (left to right). The heat equation error is able to indicate most of the regions with high error. It illustrates the checkerboard pattern expected from purely convolutional networks. Since the heat equation error is defined via the local imbalance of heat fluxes and sources, the detected errors can be slightly more localized compared to the actual error (e.g. orange points in slice 69, top compared to bottom).

Table 3. Average Evaluation Time for a NN Solution and a FEM Solution

NN, GPU transfer	NN inference	Total NN	FEM (single core)
0.033 s	0.002 s	0.035 s	160 s

It is worth discussing now some of the limitations of the FCN architecture suggested in this work. The first limitation stems from the fact that the architecture sets the maximum effective receptive field on the input images, which is determined by the combination of maximum dilation and number of downsamplings. Thus, even though the architecture is fully convolutional - which implies that it can immediately process larger systems than those it has been trained on - it will however not be able to capture any heat transfer over larger distances than those covered in the original receptive field. We have not tested what the implications of this limitation are. A second limitation arises from the typical length scales in the system. A minimum length scale is implicitly set by choosing the voxel resolution and, in principle, if a different voxel resolution has to be chosen the network would have to be retrained.

The main advantage of the approach presented here is that, once the network has been trained, it can approximate the temperature field of any system within the design space spanned by the randomised training dataset. This means that the network can be applied to systems with different number of components, different placement of those components, different material properties and different BCs than those it has been trained on. Therefore, the simulation time spent in generating the dataset is compensated by the gain in future simulations, as long as many of them are needed for a certain design task.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 16 December 2021 doi:10.20944/preprints202112.0275.v1

3.0.1. Confidence Estimation

Finally, we would like to discuss a possible confidence estimator for the NN solutions. The goal of a confidence estimator in this context is to assess the quality of the NN approximation when no FE result exists to compare with, as would be the case in any realistic use of our tool.

Our suggested confidence estimator is based on an integral form of the steady-state heat equation

$$\int_{\partial V} k \partial_i T n_i dA + \int_V \rho h dV = 0, \tag{10}$$

where Gauss' theorem was used to transform the integral over the system volume V to a surface integral in the first term. Since the NN prediction is available at each voxel, the equation is discretised to obtain an approximation per voxel e

$$\sum_{i=1}^{6} {}^e k (\partial_i T) n_i \, {}^e \Delta A_i + {}^e \rho^e h^e \Delta V \approx 0. \tag{11}$$

For the temperature gradient $(\partial_i T)$, the finite difference of the temperature (as predicted by the NN) of the voxel under consideration and its adjacent voxel (in the direction of the normal n_i) is inserted using a piece-wise linear interpolation of the temperature in cases where these two voxels have differing k values. The bottom of the PCB is kept at a fixed temperature. On all other outer surfaces the flux boundary condition is included by evaluating the first term with

$$-k \frac{\partial T}{\partial n} = \alpha (T - T_{\text{ext}}) \, . \tag{12}$$

This *heat equation error* should measure the local violation of the steady-state heat equation. If the net heat flux of a voxel is non-zero this implies that the temperature obtained from the NN does not represent a steady state.

The local L1 loss values are compared with the local heat equation error for a particular sample (Fig. 6) for different sections of the system in the z-direction. The proposed estimator appears to qualitatively identify many of the regions where the error of the NN estimation of the temperature map is high compared to the FEM solution. It is also apparent, however, that not all voxels with high error values are identified.

Moreover, the estimator does not provide a clear quantitative measure of the error. The absolute values of the heat equation error are much larger in regions with high k than in regions with lower k. This could indicate that the piece-wise linear interpolation used for the approximation of the temperature is not a good estimation of the actual temperature distribution. However, at the current state, it is not possible to use higher order approximations since they would require more voxels per uniform-k region, which in turn conflicts with the limited RAM available on the GPU.

More research in the direction of finding a confidence estimator is clearly needed.

4. Conclusions

In this article a neural network architecture was proposed to approximate the temperature distribution of a 3D system of electronic components with heat losses and heat transfer through their boundaries. The main goal was to assess the performance and accuracy of neural networks for this task.

The network proposed is able to provide a simulation result in an evaluation time of the order of milliseconds. The relative error achieved, when compared to a standard FEM simulation for the same system, is around 2% averaged over the whole system, with larger errors localised mostly on the surfaces of the components. Limitations of the architecture proposed in its application to systems with different sizes and/or resolutions have also been discussed.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 16 December 2021 doi:10.20944/preprints202112.0275.v1

Finally we motivated a possible criterion for estimating the confidence of a prediction based on an integral version of the heat equation. Even though the criterion proposed does not allow for a quantitative estimate of the error of a given prediction, it may be used to identify regions with large estimate errors.

Author Contributions: Both authors contributed substantially to the conceptualization, methodology, and writing—original draft preparation, review and editing. All authors have read and agreed to the published version of the manuscript.

Funding: This work has been supported by Silicon Austria Labs GmbH (SAL), owned by the Republic of Austria, the Styrian Business Promotion Agency (SFG), the federal state of Carinthia, the Upper Austrian Research (UAR), and the Austrian Association for the Electric and Electronics Industry (FEEI).

Acknowledgments: We thank L. Ratschbacher for a critical reading of an early version of the manuscript and C. Mentin for support.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

Appendix A. Introduction to ANNs

In this section we give an overview of neural networks, highlighting the aspects that are most relevant to understand the rest of the paper; see e.g. [36] for a detailed treatment. In this work, NNs are used as highly versatile and non-linear function approximators. Generally speaking, a neural network can be visualised as a stack of layers, each of them computing a set of linear operations on their input data, subsequently applying a fixed non-linear operation and passing the result to the next layer. The input to the network in this work is a voxelized representation of a 3D electronic system (i.e. a 3D image), where the different properties of the system are passed as different channels (in analogy to the RGB channels of a standard 2D image). Convolutional neural networks (CNNs), consisting of a special type of layer called convolutional layer, are used for processing images.

The most important feature in a convolutional layer of a CNN is that not every pixel in the input image of a layer is connected to every pixel in the output, but rather each output pixel is determined by a small set of input pixels (the receptive field), the number of them is given by the size of the so-called convolutional kernel (or filter). To be specific, the equation defining the action of a 3D convolutional layer is

$$z_{ijk;C} = b_C + \sum_{\alpha=0}^{k_h} \sum_{\beta=0}^{k_w} \sum_{\gamma=0}^{k_d} \sum_{F=0}^{n_{c'}} x_{i'j'k'F} \times w_{\alpha\beta\gamma;CF} \,, \tag{A1}$$

with

$$\begin{aligned} i' &= i \times s_h + d_h \times \alpha \\ j' &= j \times s_w + d_w \times \beta \\ k' &= k \times s_d + d_d \times \gamma \end{aligned} \,. \tag{A2}$$

where $z_{ijk;C}$ represents the value of the voxel at position i, j, k of the output 3D image, C labels the channel (or feature map) of the output image, s_h, s_w and s_d are natural numbers called strides and d_h, d_w and d_h are naturals called dilations (their meaning is explained in Sec. 2.2), k_h, k_w and k_d are the height, width and depth of the 3D convolutional kernels applied to the input image x, with $n_{c'}$ channels, b_C is a bias parameter for each feature map C and w are the weight parameters.

The next ideas to be presented are downsampling and upsampling. It is not necessarily the case that the size of the output and input images of a convolutional layer coincide. It can be inferred from Eq. A2, the output image size can be reduced by tuning the values of the stride parameters. For example, choosing $s_h = s_w = s_d = 2$, the size of the output image will be halved. One refers to this process as downsampling. In image processing,

the purpose of downsampling is to *force* the CNN to extract relevant features from an image (e.g. edges)but, in this work, extracting features is not so relevant since they are, to some extent, manually implemented. Nevertheless, some degree of downsampling is necessary to reduce the memory requirements of the network. The inverse action to downsampling is upsampling. One can think of it as an interpolation procedure, where from one input pixel a higher number of pixels are generated as output. In practice, this is achieved by an operation called transposed convolution, which is mathematically very similar to the standard convolution in Eq. A1, with different weights and biases to be learned. Upsampling is necessary if downsampling is used in some of the layers of the network but the output of a CNN must be another image of the same size of the input image, as is our case. CNNs of this type are called fully-convolutional neural networks (FCNs).

Finally there is a different type of layer called pooling layer. They are parameter-free layers (non-learnable) that apply a fixed operation on their receptive field. Typical pooling layers include max-pooling and average-pooling layers, which give as output the maximum or the average value, respectively, of the pixel (resp. voxel) values in their receptive field.

The training of a CNN consists in finding the optimal values of the parameters w and b of each of the layers. This is achieved by minimising a certain objective function (loss) on a labelled dataset. For this work, as discuss in the main text, the dataset consists of a collection of simplified 3D circuits, each *labelled* by the result of a standard thermal simulation. The training is performed on a subset of the whole dataset (training set), whilst the rest are used to test the network (test set).

References

1. Langbauer, T.; Mentin, C.; Rindler, M.; Vollmaier, F.; Connaughton, A.; Krischan, K. Closing the Loop between Circuit and Thermal Simulation: A System Level Co-Simulation for Loss Related Electro-Thermal Interactions. 2019 25th International Workshop on Thermal Investigations of ICs and Systems (THERMINIC). IEEE, 2019, pp. 1–6. doi:10.1109/THERMINIC.2019.8923595.
2. Zhao, S.; Blaabjerg, F.; Wang, H. An Overview of Artificial Intelligence Applications for Power Electronics. *Trans. Power Electron* **2015**, *30*, 6791–6803.
3. Wu, T.; Wang, Z.; Ozpineci, B.; Chinthavali, M.; Campbell, S. Automated heatsink optimization for air-cooled power semiconductor modules. *IEEE Transactions on Power Electronics* **2018**, *34*, 5027–5031.
4. Zhang, Y.; Wang, Z.; Wang, H.; Blaabjerg, F. Artificial Intelligence-Aided Thermal Model Considering Cross-Coupling Effects. *IEEE Transactions on Power Electronics* **2020**.
5. Delaram, H.; Dastfan, A.; Norouzi, M. Optimal Thermal Placement and Loss Estimation for Power Electronic Modules. *IEEE Transactions on Components, Packaging and Manufacturing Technology* **2018**, *8*, 236–243. doi:10.1109/TCPMT.2017.2781282.
6. Guillod, T.; Papamanolis, P.; W. Kolar, J. Artificial Neural Network (ANN) Based Fast and Accurate Inductor Modeling and Design. *IEEE Open Journal of Power Electronics* **2020**, *1*, 284–299. doi:10.1109/ojpel.2020.3012777.
7. Chiozzi, D.; Bernardoni, M.; Delmonte, N.; Cova, P. A Neural Network Based Approach to Simulate Electrothermal Device Interaction in SPICE Environment. *IEEE Transactions on Power Electronics* **2019**, *34*, 4703–4710. doi:10.1109/TPEL.2018.2863186.
8. Xu, Z.; Gao, Y.; Wang, X.; Tao, X.; Xu, Q. Surrogate Thermal Model for Power Electronic Modules using Artificial Neural Network. IECON 2019 - 45th Annual Conference of the IEEE Industrial Electronics Society. IEEE, 2019, Vol. 2019-Octob, pp. 3160–3165. doi:10.1109/IECON.2019.8927494.
9. Marie-Francoise, J.N.; Gualous, H.; Berthon, A. Supercapacitor thermal- and electrical-behaviour modelling using ANN. *IEE Proceedings - Electric Power Applications* **2006**, *153*, 255. doi:10.1049/ip-epa:20050096.
10. Dragicevic, T.; Wheeler, P.; Blaabjerg, F. Artificial Intelligence Aided Automated Design for Reliability of Power Electronic Systems. *IEEE Transactions on Power Electronics* **2019**, *34*, 7161–7171. doi:10.1109/TPEL.2018.2883947.
11. Kim, J.; Lee, C. Prediction of turbulent heat transfer using convolutional neural networks. *Journal of Fluid Mechanics* **2020**, *882*. doi:10.1017/jfm.2019.814.
12. Swischuk, R.; Mainini, L.; Peherstorfer, B.; Willcox, K. Projection-based model reduction: Formulations for physics-based machine learning. *Computers and Fluids* **2019**, *179*, 704–717. doi:10.1016/j.compfluid.2018.07.021.
13. Gao, H.; Sun, L.; Wang, J.X. PhyGeoNet: Physics-Informed Geometry-Adaptive Convolutional Neural Networks for Solving Parametric PDEs on Irregular Domain **2020**. [2004.13145].
14. Cai, S.; Wang, Z.; Lu, L.; Zaki, T.A.; Karniadakis, G.E. DeepM&Mnet: Inferring the electroconvection multiphysics fields based on operator approximation by neural networks **2020**. [2009.12935].

15. He, Q.Z.; Barajas-Solano, D.; Tartakovsky, G.; Tartakovsky, A.M. Physics-informed neural networks for multiphysics data assimilation with application to subsurface transport. *Advances in Water Resources* **2020**, *141*, 103610, [1912.02968]. doi:10.1016/j.advwatres.2020.103610.
16. Kadeethum, T.; Jørgensen, T.M.; Nick, H.M. Physics-informed neural networks for solving nonlinear diffusivity and Biot's equations. *PloS one* **2020**, *15*, e0232683, [2002.08235]. doi:10.1371/journal.pone.0232683.
17. Khan, A.; Ghorbanian, V.; Lowther, D. Deep Learning for Magnetic Field Estimation. *IEEE Transactions on Magnetics* **2019**, *55*, 1–4. doi:10.1109/TMAG.2019.2899304.
18. Breen, P.G.; Foley, C.N.; Boekholt, T.; Zwart, S.P. Newton vs the machine: solving the chaotic three-body problem using deep neural networks. *Monthly Notices of the Royal Astronomical Society* **2019**, *494*, 2465–2470, [1910.07291]. doi:10.1093/mnras/staa713.
19. Sanchez-Gonzalez, A.; Godwin, J.; Pfaff, T.; Ying, R.; Leskovec, J.; Battaglia, P.W. Learning to Simulate Complex Physics with Graph Networks. *ICML* **2020**, pp. 8459–8468, [2002.09405].
20. Blumberg, S.B.; Tanno, R.; Kokkinos, I.; Alexander, D.C. Deeper image quality transfer: Training low-memory neural networks for 3D images. Lecture Notes in Computer Science. Springer Verlag, 2018, Vol. 11070 LNCS, pp. 118–125, [1808.05577]. doi:10.1007/978-3-030-00928-1_14.
21. Ahmed, E.; Saint, A.; Shabayek, A.E.R.; Cherenkova, K.; Das, R.; Gusev, G.; Aouada, D.; Ottersten, B. A survey on Deep Learning Advances on Different 3D Data Representations **2018**. *1*, [1808.01462].
22. Liu, W.; Sun, J.; Li, W.; Hu, T.; Wang, P. Deep Learning on Point Clouds and Its Application: A Survey. *Sensors* **2019**, *19*, 4188. doi:10.3390/s19194188.
23. Bello, S.A.; Yu, S.; Wang, C. Review: deep learning on 3D point clouds. *Remote Sensing* **2020**, *12*, [2001.06280].
24. Rios, T.; Wollstadt, P.; Stein, B.V.; Back, T.; Xu, Z.; Sendhoff, B.; Menzel, S. Scalability of Learning Tasks on 3D CAE Models Using Point Cloud Autoencoders. 2019 IEEE Symposium Series on Computational Intelligence (SSCI). IEEE, 2019, pp. 1367–1374. doi:10.1109/SSCI44817.2019.9002982.
25. Riegler, G.; Ulusoy, A.O.; Geiger, A. OctNet: Learning Deep 3D Representations at High Resolutions. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017* **2016**, *2017-Janua*, 6620–6629, [1611.05009].
26. Karniadakis, G.E.; Kevrekidis, I.G.; Lu, L.; Perdikaris, P.; Wang, S.; Yang, L. Physics-informed machine learning. *Nature Reviews Physics* **2021**, *3*, 422–440. doi:10.1038/s42254-021-00314-5.
27. Raissi, M.; Perdikaris, P.; Karniadakis, G.E. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics* **2019**, *378*, 686–707. doi:10.1016/j.jcp.2018.10.045.
28. Battaglia, P.W.; Hamrick, J.B.; Bapst, V.; Sanchez-Gonzalez, A.; Zambaldi, V.; Malinowski, M.; Tacchetti, A.; Raposo, D.; Santoro, A.; Faulkner, R.; Gulcehre, C.; Song, F.; Ballard, A.; Gilmer, J.; Dahl, G.; Vaswani, A.; Allen, K.; Nash, C.; Langston, V.; Dyer, C.; Heess, N.; Wierstra, D.; Kohli, P.; Botvinick, M.; Vinyals, O.; Li, Y.; Pascanu, R. Relational inductive biases, deep learning, and graph networks **2018**. [1806.01261].
29. Geuzaine, C.; Remacle, J.F.; others. Gmsh: a three-dimensional finite element mesh generator with built-in pre-and post-processing facilities. *International journal for numerical methods in engineering* **2009**, *79*, 1309–1331.
30. Malinen, M.; Råback, P. Elmer finite element solver for multiphysics and multiscale problems. *Multiscale Model. Methods Appl. Mater. Sci.* **2013**, *19*, 101–113.
31. Råback, P.; Malinen, M.; Ruokolainen, J.; Pursula, A.; Zwinger, T. Elmer Models Manual. Technical report, 2020.
32. Köpüklü, O.; Kose, N.; Gunduz, A.; Rigoll, G. Resource efficient 3d convolutional neural networks. 2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW). IEEE, 2019, pp. 1910–1919.
33. Klambauer, G.; Unterthiner, T.; Mayr, A.; Hochreiter, S. Self-normalizing neural networks. Advances in neural information processing systems, 2017, pp. 971–980.
34. Maas, A.L.; Hannun, A.Y.; Ng, A.Y. Rectifier nonlinearities improve neural network acoustic models.
35. Iandola, F.N.; Han, S.; Moskewicz, M.W.; Ashraf, K.; Dally, W.J.; Keutzer, K. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size. *arXiv preprint arXiv:1602.07360* **2016**.
36. Goodfellow, I.; Bengio, Y.; Courville, A.; Bengio, Y. *Deep learning*; Vol. 1, MIT press Cambridge, 2016.