Thermal-Net: Convolutional neural network for 3D printing

Francesco Colibazzi[1], Andrea Samorè[1], Giulia Scalet[2], Serena Morigi[1]

[1]Department of Mathematics, University of Bologna

[2]Department of Civil Engineering and Architecture (DICAr), University of Pavia

"Mathematical Methods for Object Reconstruction: from 3D Vision to 3D Printing"

11 February 2021

Additive Manufacturing – 3D printing

- Additive Manufacturing (AM), also known as 3D printing, is the process of joining materials in a layer-by-layer manner to build an object from 3D model data.

- Advantages with respect to other manufacturing techniques:

  very rapid prototyping cycles, flexible designs, waste reduction.

- As the mechanical properties of a metal manufactured part depend on thermal history,

the simulation of the thermal history of the AM procedure is a key aspect to ensure the quality of the resulting product.

OUR GOAL:

Simulate the AM thermal history (modeled by a parabolic PDE) by a convolutional neural network

Metal AM: Selective Laser Melting (SLM)

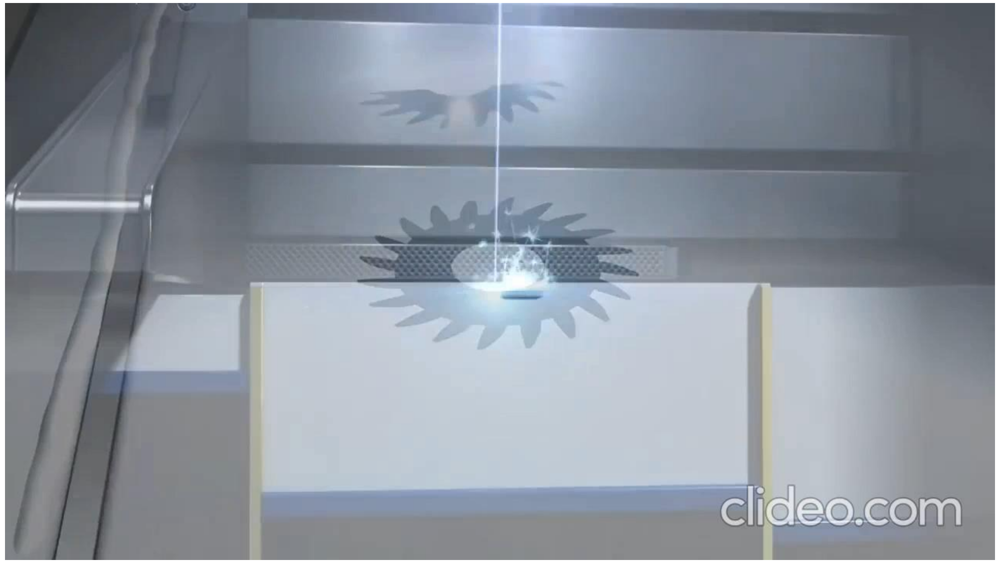
SLM PROCESS

1. A thin layer of metal powder is spread over a build platform

2. A high power laser scans the cross-section of the object, selectively melting and fusing the metal powders together and creating the next layer

3. Once the scanning process is complete, the build platform moves downwards by one layer thickness and another thin layer of metal powder is spread

4. The process is repeated until the whole object is complete



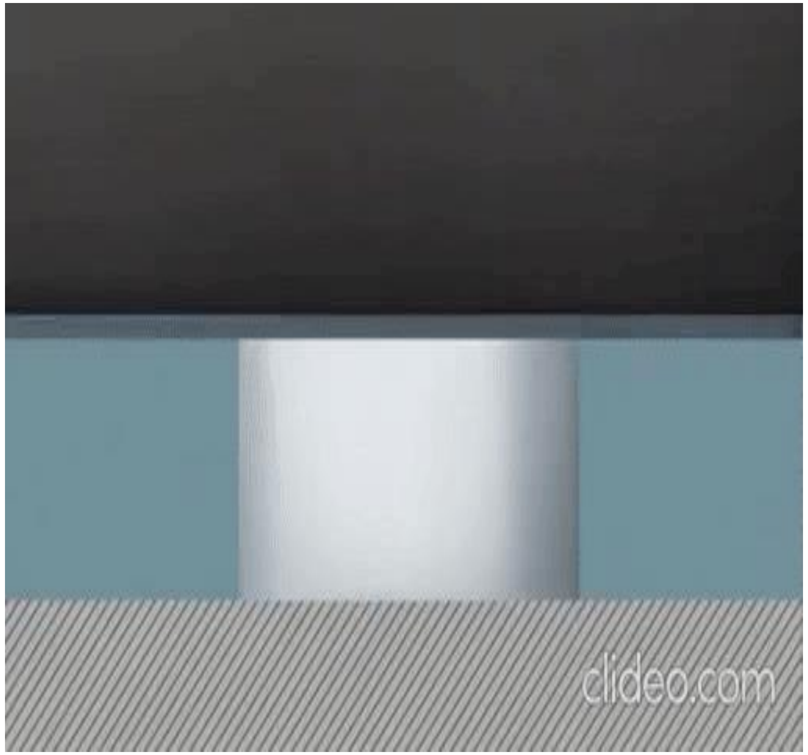
Modeling approach

- The thermal analysis starts at the point when the first layer of powder is laid upon the machine table and the laser head begins to offer energy to the powder.

- The analysis continues over time, by simulating the movement of the laser scanner head.

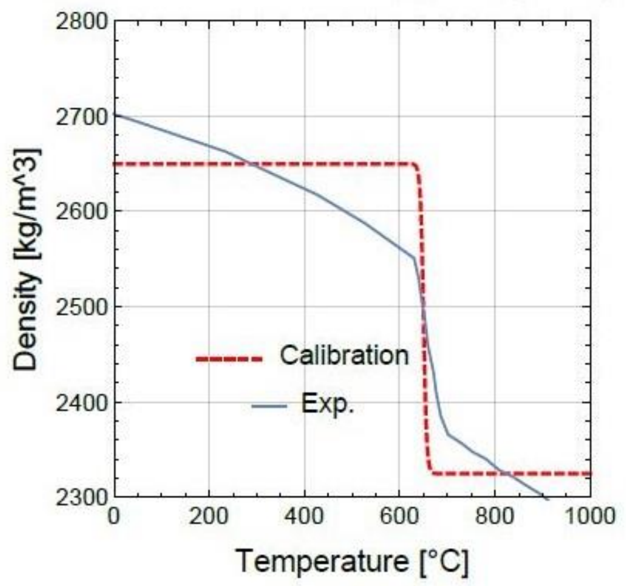
- A further increase of the accuracy is achieved by assuming temperature dependent material properties:

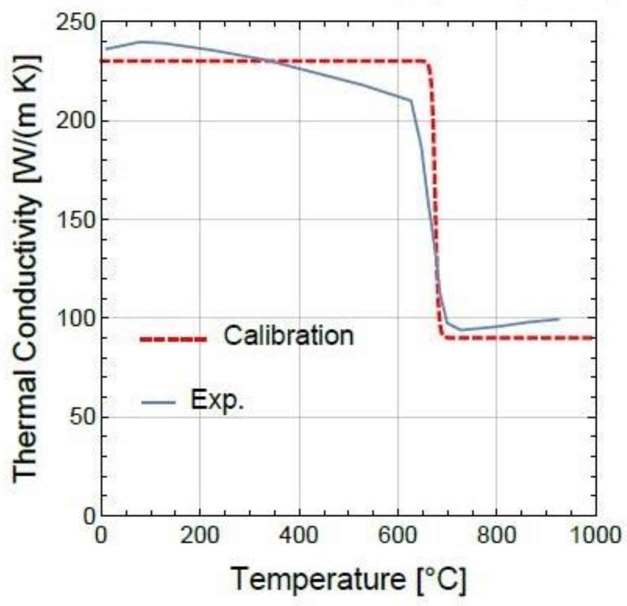
  ✓ thermal conductivity ***k*** ,
  ✓ specific heat capacity ***c***,
  ✓ density ***ρ***.

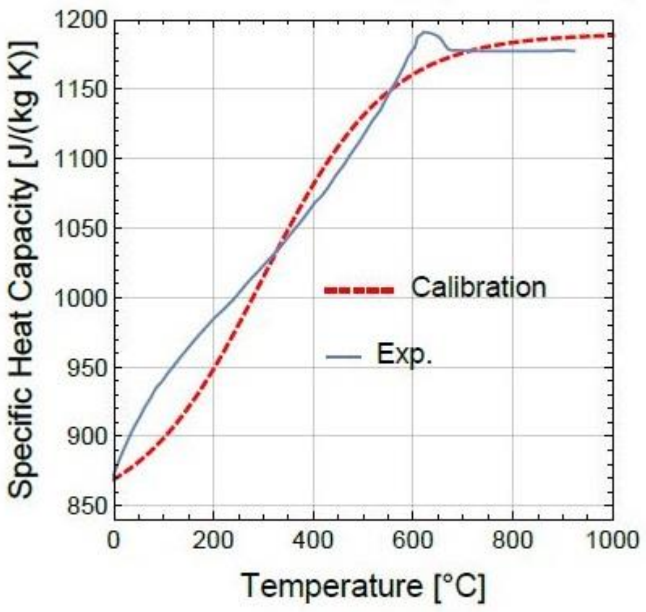


Thermal properties:

$$\zeta(T) = klow + \frac{|khigh - klow|}{1 + e^{\alpha(T-\hat{T})}}$$



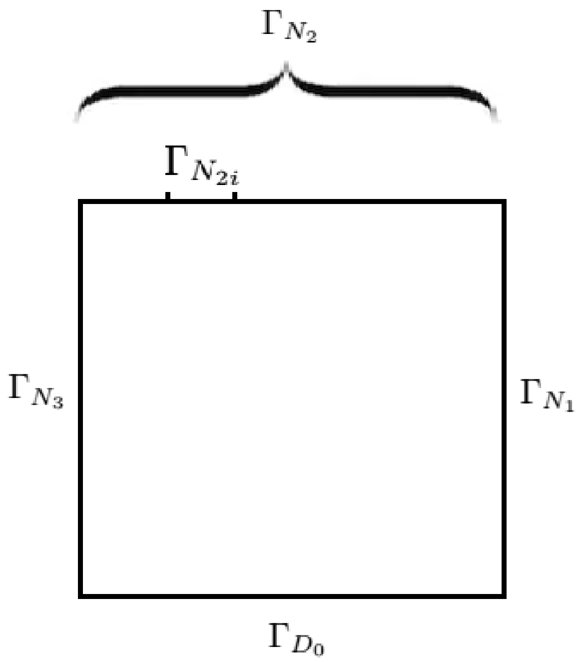




	Thermal Conductivity	Specific Heat Capacity	Density
$\hat{T}$	675	300	650
α	0.3	-0.008	0.3
$klow$	90	840	2325
$khigh$	230	1190	2650

Table 1: Constant values that fit the thermal properties.

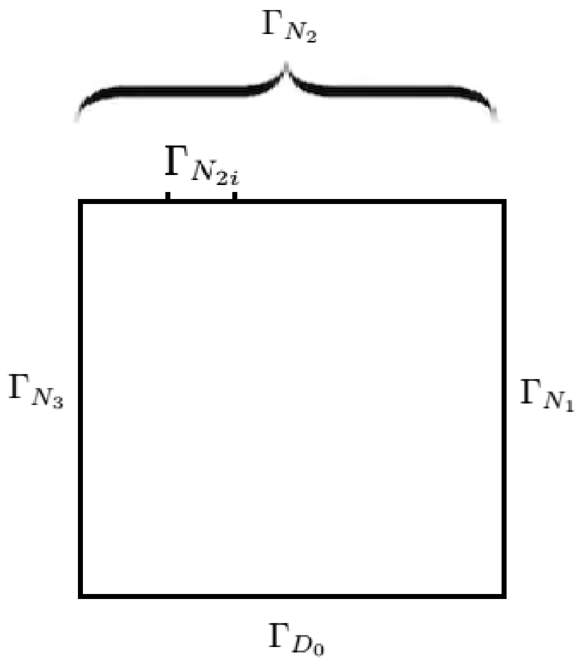
Modeling a 2D section



- A square domain Ω_x represents a layer of powder.
- The top boundary Γ_{N_2} is split into 10 subdomains representing all the possible positions assumed by the laser.
- $\xi \in \mathbb{R}^{10}$ position vector with values 0 or 1 depending on whether the laser fires heat or not.

$$
\begin{cases}
\rho(\bar{x},T)c(\bar{x},T)T_t(\bar{x},t) = \nabla \cdot (k(\bar{x},T)\nabla T(\bar{x},t)) & \text{in } \Omega_x \times [0,\bar{t}] \\
-k(\bar{x},T)\frac{\partial T(\bar{x},t)}{\partial n} = \psi(\bar{x},T) & \text{on } \Gamma_{N_1} \cup \Gamma_{N_2} \backslash \Gamma_{N_{2i}} \cup \Gamma_{N_3} \\
k(\bar{x},T)\frac{\partial T(\bar{x},t)}{\partial n} = \phi(\bar{x},\xi) & \text{on } \Gamma_{N_{2i}} \\
T(\bar{x},t) = T_b & \text{on } \Gamma_{D_0} \\
T(\bar{x},0) = T_{env}
\end{cases}
$$

Modeling a 2D section



- $\Gamma_{N_{2i}}$ *Heating*

$$k(\bar{x},T)\frac{\partial T(\bar{x},t)}{\partial n} = \phi(\bar{x},\Gamma_{N_{2i}}) = \begin{cases} I_0 e^{(-2(\frac{x-x_c}{r})^2)} & \text{, if } \quad \xi(i)=1 \\ 0 & \text{, otherwise} \end{cases}$$

$I_0 = \frac{2P}{\pi r^2}$, P is the laser power.

- $\Gamma_{N_2} \backslash \Gamma_{N_{2i}}$ *Convection*

$$-k(\bar{x},T)\frac{\partial T(\bar{x},t)}{\partial n} = \psi(\bar{x},T) = h(T_{env} - T(\bar{x},t))$$

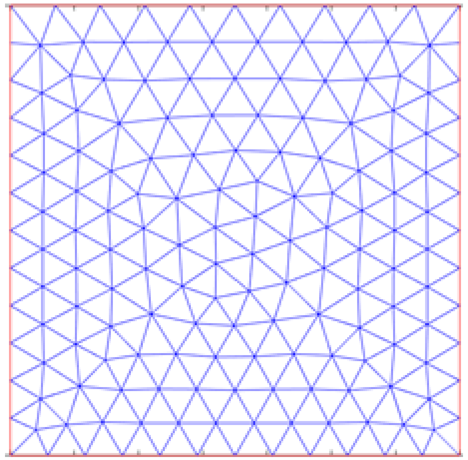
- Γ_{D_0}

$$T = T_b.$$

$x1$-axes	$0.0007m$
$x2$-axes	$0.0007m$
Spot size	$0.00007m$
Scanner head speed	$0.2m/s$
T_{env} Ambient temperature	$20°C$
T_b	$100°C$
h Convective heat transfer coefficient	$10W/m^2K$
P Laser power	$200W$
Absportance	9%

Finite Element Method (FEM) Solver

Simulation by the FEM method has some limitations:



- The mesh:
  the stability of the method strongly depends on mesh quality;

- Computational cost:
  FEM can be computationally expensive for complex problems

Semi-implicit Euler scheme to solve time-dependent PDEs

$$\rho(\bar{x},T)c(\bar{x},T)T_t(\bar{x},t) = \nabla \cdot (k(\bar{x},T)\nabla T(\bar{x},t)) \qquad (2)$$

Apply backward finite difference scheme in time:

$$\frac{\partial T}{\partial t} \approx \frac{T^{t+1} - T^t}{\Delta t} \quad t \in [0, \bar{t}]$$

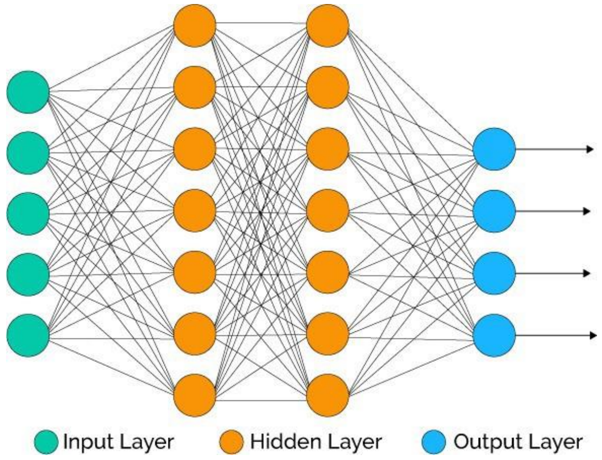
Replace in (2):

$$\rho(\bar{x},T^{t+1})c(\bar{x},T^{t+1})T_t^{t+1}(\bar{x},t) = \nabla \cdot (k(\bar{x},T^{t+1})\nabla T^{t+1}(\bar{x},t))$$

Linearize the PDE considering the non-linear quantities *k*, *ρ*,*c* as functions of the previous time-step solution. The result is a sequence of (stationary) step for $\boldsymbol{T^{n+1}}$, assuming $\boldsymbol{T^n}$ is known at the previous time step:

$$\underbrace{(\rho^t c^t - \Delta t(k^t \nabla^2))}_{\mathbf{L}} \underbrace{T^{t+1}}_{T} = \underbrace{T^t}_{f} \qquad t \in [0, \bar{t}]$$

What's it neural network?



- The goal is to approximate some function f^*
- A feedforward network f defines a mapping

$$y = f(x;w)$$

which learns the value of the parameters w that result in the best function approximation.

Supervised Training

- Both the inputs $[x_1, \ldots, x_n]$ and the outputs $[y_1, \ldots, y_n]$ are provided during training.
- The network then processes the inputs and compares the resulting outputs against the desired outputs.
- Errors are then propagated back through the system, causing the system to adjust the weights which control the network.

Mathematical inspiration

- Considering the Poisson equation with Dirichlet boundary conditions, Ω unit disk in $\mathbb{R}^2$, $f \in C^{\infty}(\bar{\Omega})$

$$\begin{cases} \Delta u = f & \text{in} \quad \Omega \\ u = 0 & \text{on} \quad \partial\Omega \end{cases}$$

- The solution of the homogeneous Dirichlet problem can be represented as:

$$u(x) = \int_{\Omega} G(x, y)\, f(y)\, dy \quad \forall x \in \Omega$$

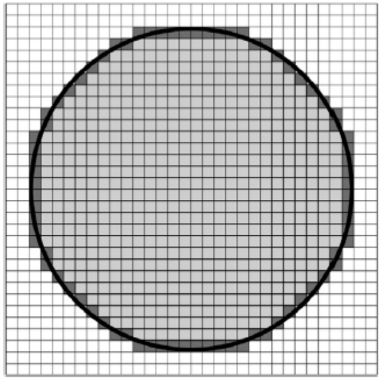
where *G(x,y)* is the Green's function for the unit disk in $\mathbb{R}^2$

- An explicit solution mapping between a source term *f* and the associated solution *Gf=u:*

$$\begin{aligned} \mathcal{G} : C^{\infty}(\overline{\Omega}) &\longrightarrow C^{\infty}(\overline{\Omega}) \\ f &\mapsto \int_{\Omega} G(-, y) f(y) dy \end{aligned}$$

Mathematical inspiration

-



Introduce a uniform grid ∑ which covers the *Ω* domain with fixed resolution *R*
Let Ω_d the collection of mesh points in ∑, we can define the associated:

- Interpolation:

$$\mathcal{I} : \left\{ (x,\ f(x)) \right\}_{x \in \Omega_d} \mapsto \mathrm{interp}_{\overline{\Omega}} \left\{ (x,\ f(x)) \right\}$$

- Projection mappings:

$$\mathcal{P} :\ u\,|_{\overline{\Omega}} \mapsto \left\{ (x,\ u(x)) \right\}_{x \in \Omega_d}$$

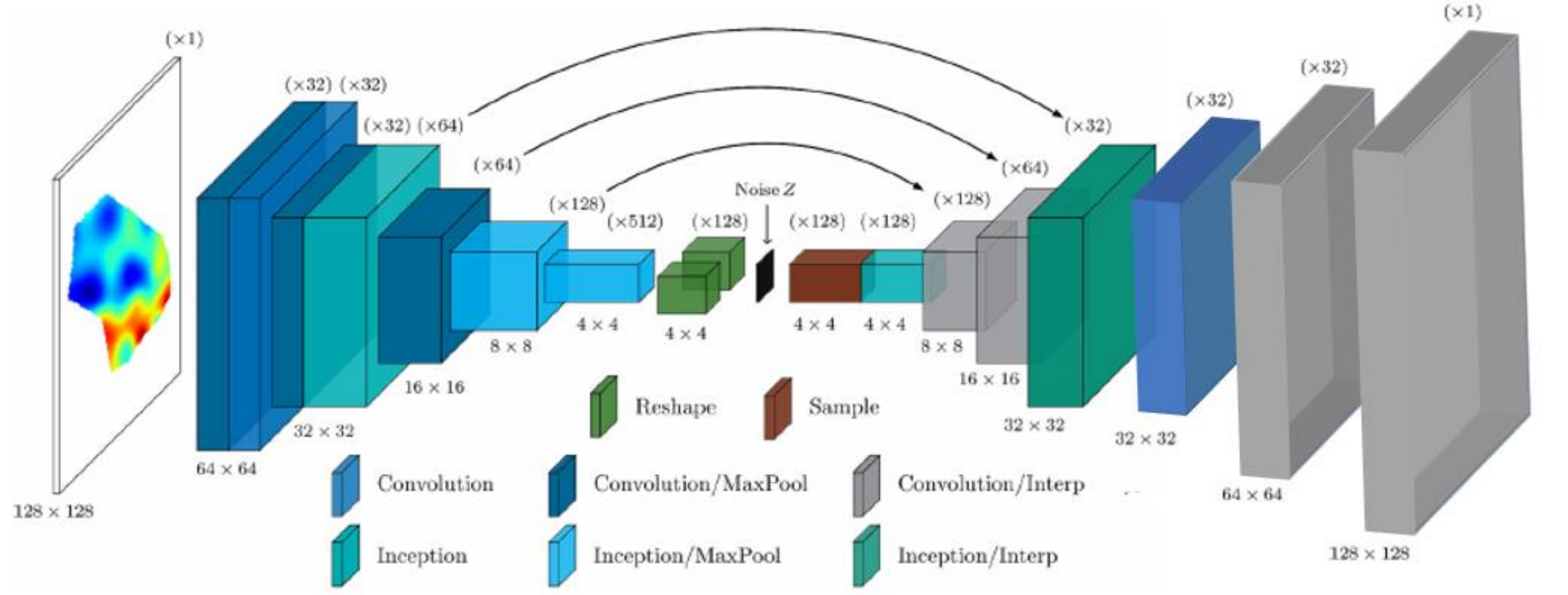
Then the discretized composite mapping:

$$\left\{ (x,\ f(x)) \right\}_{x\in\Omega_d} \xrightarrow{\ \mathcal{I}\ } f\,|_{\overline{\Omega}} \xrightarrow{\ \mathcal{G}\ } u\,|_{\overline{\Omega}} \xrightarrow{\ \mathcal{P}\ } \left\{ (x,\ u(x)) \right\}_{x\in\Omega_d}$$

can be approximate by a multi-layer feedforward network.

✓ **The convolutional form of the integral operator *G* inspires the use of convolutional layers within this network approximation.**

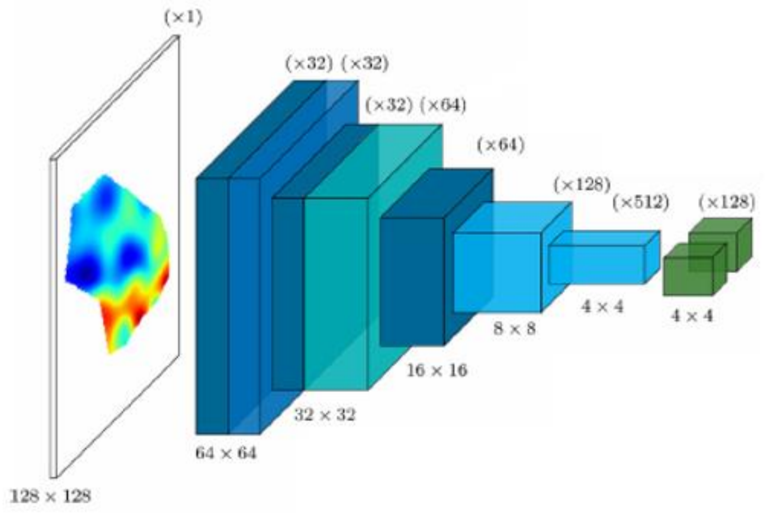
Thermal-Net: *Lu=f*

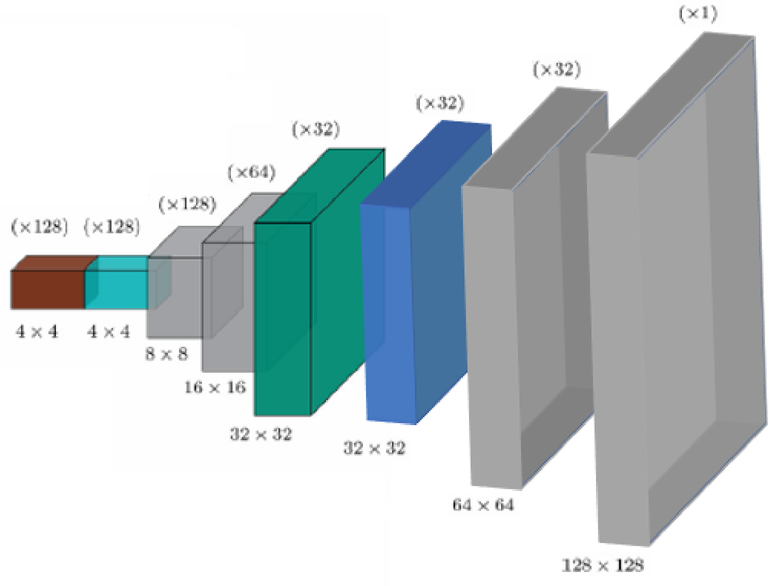


The network has two primary components:

- **encoder** designed to map high-level input functions to low-dimensional latent features
- **decoder** used to map these latent features to approximate solutions

- Encoder consists of a series of convolutional layers which reduce the resolution of input features
- The encoder features with spatial resolutions 32×32, 16×16, and 8×8 are concatenated with the features of the same resolution in the network's decoder





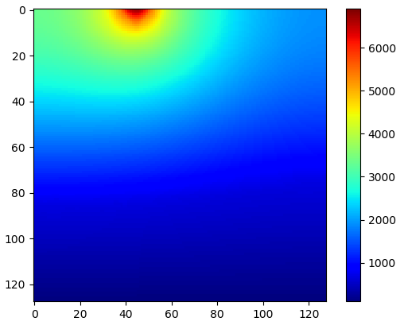
- The decoder maps the sampled latent features back to the original resolution using a series of convolutional layers followed by bilinear interpolation.
- Some layers have been split into inception blocks: max-pooling layer along with 1×1, 3×3, and two stacked 3×3 convolutional layers implemented in parallel
- Dropout layers with drop-rate 0.045 have also been included before and after the first inception block in the decoder.

Input-Output

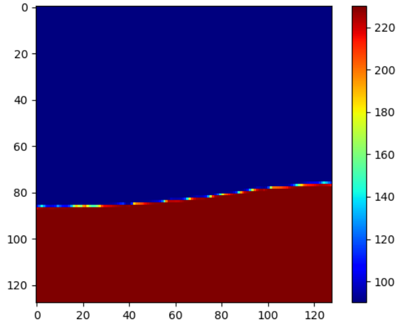
$$\underbrace{(\rho^t c^t - \Delta t(k^t \nabla^2))}_{\mathbf{L}} \underbrace{T^{t+1}}_{T} = \underbrace{T^t}_{f}$$

- *Input*: images of size 128 x 128 x 7
  - Solution at time *t*: $\boldsymbol{T^t}$
  - Neumann boundary conditions: ***Nbc***
  - Dirichlet boundary condition
  - Thermal conductivity: $\boldsymbol{k^t}$
  - Specific heat capacity: $\boldsymbol{c^t}$
  - Density: $\boldsymbol{\rho^t}$
  - Domain

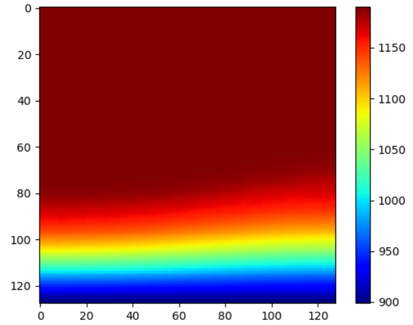
$\boldsymbol{T^t}$



$\boldsymbol{k^t}$

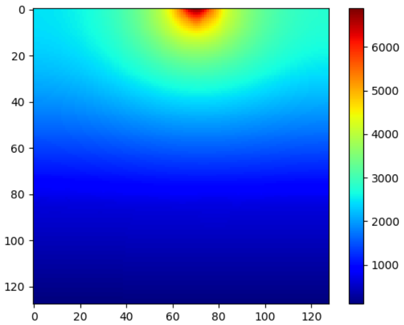


$\boldsymbol{c^t}$

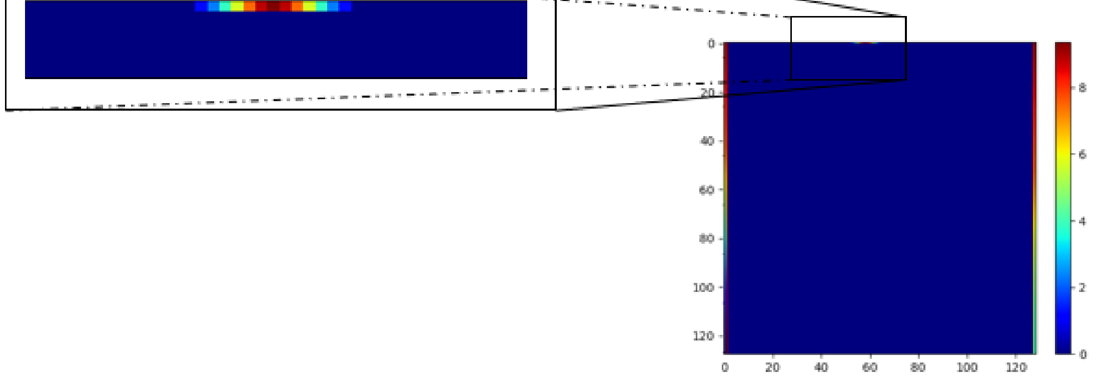


- *Output*:
  - Solution at time *t+1:* $\boldsymbol{T^{t+1}}$

$\boldsymbol{T^{t+1}}$

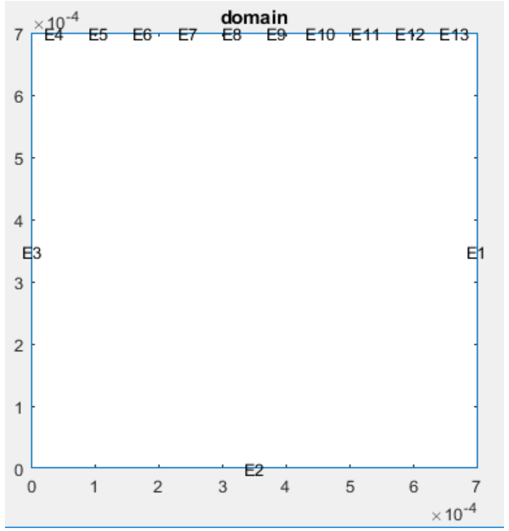


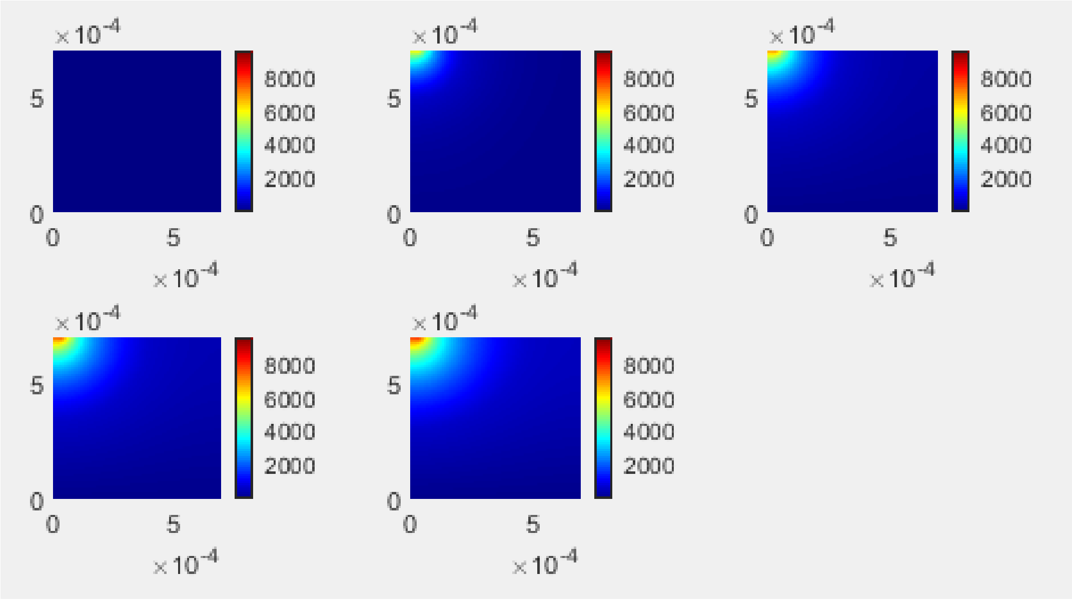
Nbc



Dataset Generation

- The dataset is generated using a FEM solver:
- Divide the top boundary domain in 10 position $\boldsymbol{\Gamma_{N_{2i}}}$, i={1,…,10}
- $t \in [0, \bar{t}]$, final time $\bar{t}$=0,035
- Time-step Δt_{pos} = n*0,0035 n={0,1,2…}
- For each position $\boldsymbol{\Gamma_{N_{2i}}}$ we obtain 5 solutions Δt = 0,000875 distant

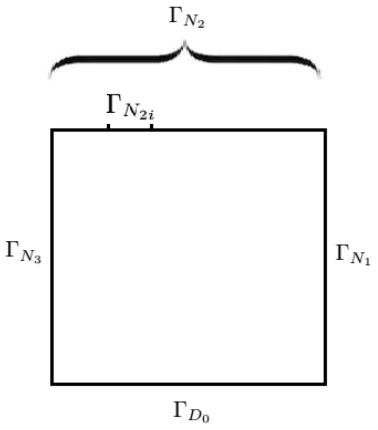




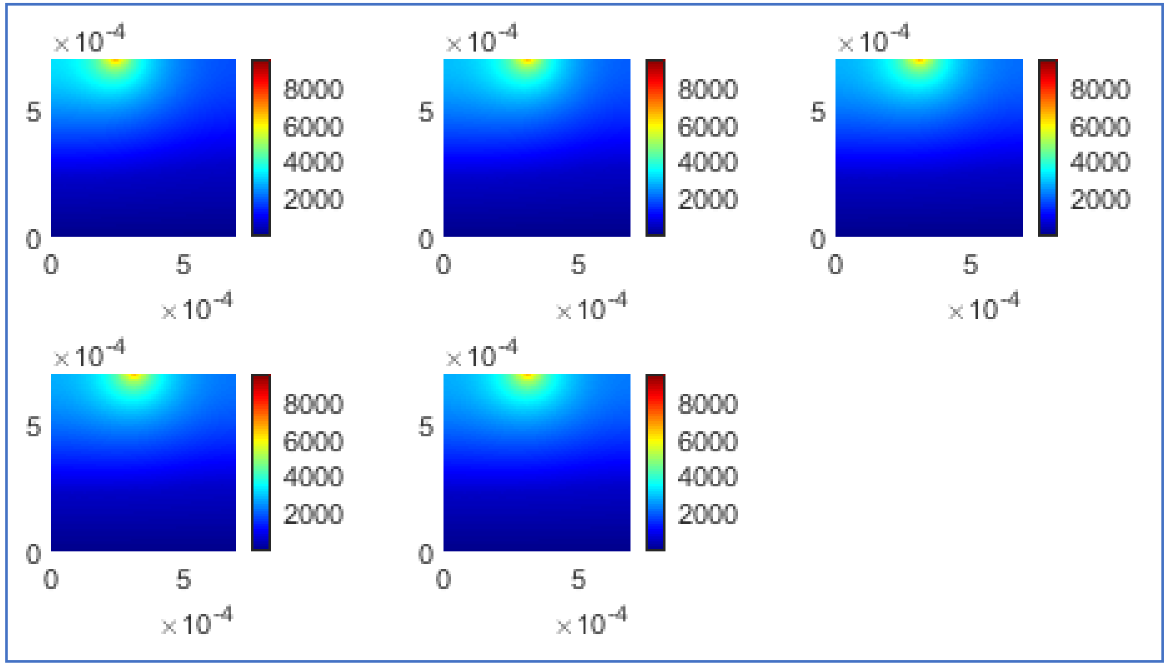
- 80% Training set
- 10% Validation set
- 10% Test set

Example

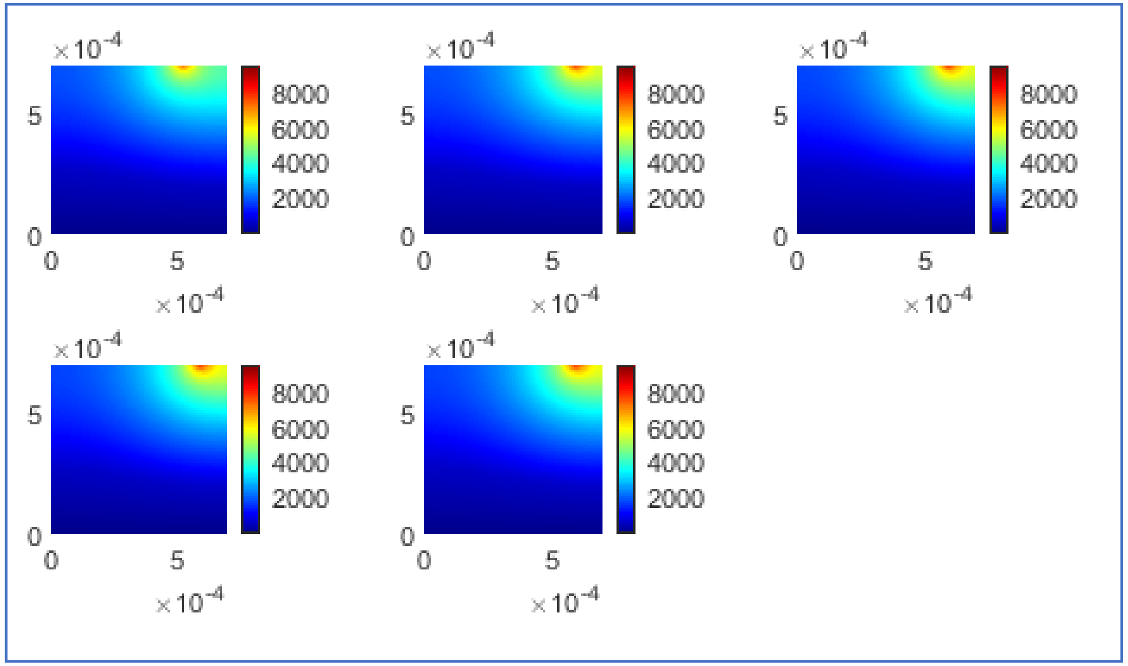
two groups of 5 solutions each in two successive positions $\Gamma_{N_{2i}}$



i = 4



i = 9



Training phase

- **Loss Function (MSE):**

$$Loss_{MSE}(\hat{y}; y, \Omega) = \frac{1}{|\Omega|} \sum_{i,j=1}^{R} \mathbf{1}_{\Omega}[i,j] \cdot (\hat{y}[i,j] - y[i,j])^2 + \frac{\lambda}{|\partial\Omega|} \sum_{i,j=1}^{R} \mathbf{1}_{\partial\Omega}[i,j] \cdot (\hat{y}[i,j] - y[i,j])^2$$

- Number of epochs: 600,000
- Optimizer: Adam optimization algorithm, with batch size of 64 samples
- Learning rate : 0.000075 with exponential decay applied every 10,000 steps by a factor of 0.95

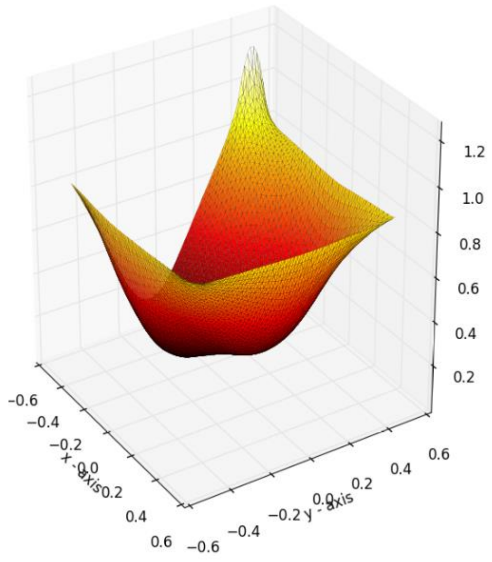
Numerical results:

- stationary state $\boldsymbol{L}T=f$

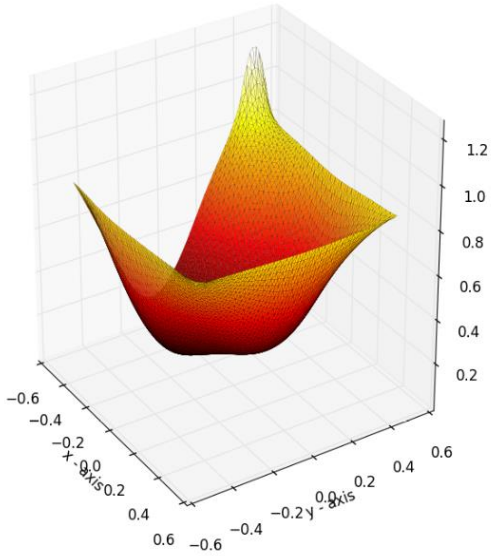
$\boldsymbol{L} = (I\text{-}\Delta t(\bar{k}\nabla^2))$

$$\begin{cases} k = k(\bar{x}) = \bar{k} = \text{cost} & \\ \bar{k}\dfrac{\partial T(\bar{x},t)}{\partial n} = \phi(\bar{x}, \Gamma_{N_{2i}}) & \text{on } \Gamma_{N_{2i}} \\ -\bar{k}\dfrac{\partial T(\bar{x},t)}{\partial n} = 0 & \text{on } \Gamma_{N_2} \backslash \Gamma_{N_{2i}} \\ T = \bar{T} & \text{on } \Gamma_{D_0} \cup \Gamma_{N_1} \cup \Gamma_{N_3} \end{cases}$$

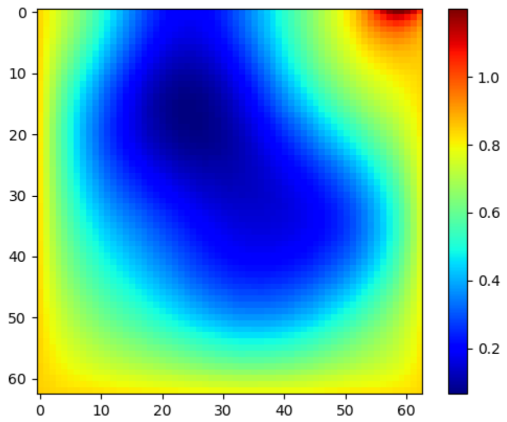
Network



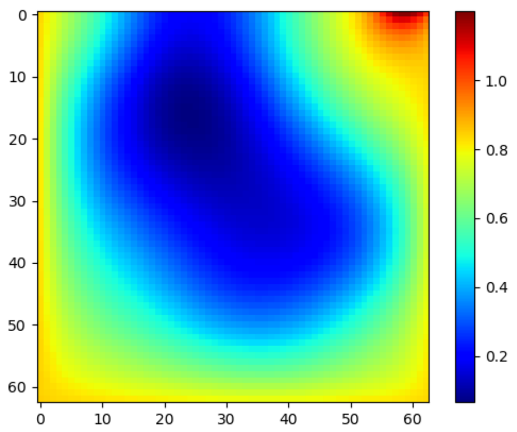
FEM



Network



FEM



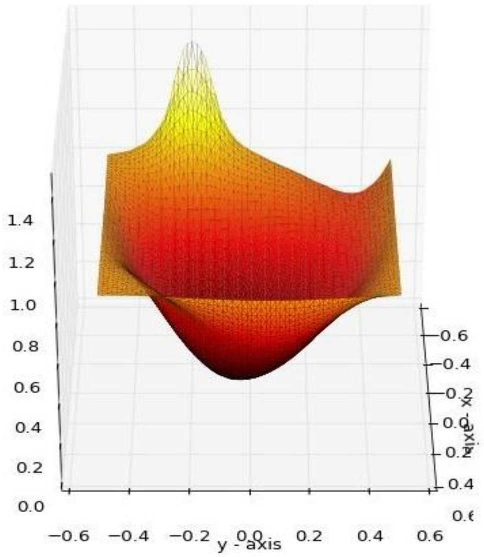
$\|T_{fem} - T_{net}\|_2^2$ on $\partial\Omega_x$	$\|T_{fem} - T_{net}\|_2^2$ on $\partial\Omega_x$
0,00434	0,00934

Numerical results:

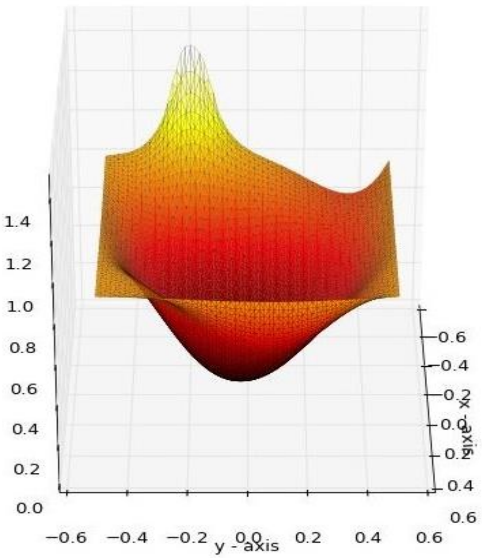
- stationary state $\boldsymbol{L}T=f$

$$\begin{cases} k = k(\bar{x}) = \bar{k} = \text{cost} & \\ \bar{k}\dfrac{\partial T(\bar{x},t)}{\partial n} = \phi(\bar{x},\Gamma_{N_{2i}}) & \text{on } \Gamma_{N_{2i}} \\ -\bar{k}\dfrac{\partial T(\bar{x},t)}{\partial n} = 0 & \text{on } \Gamma_{N_2} \backslash \Gamma_{N_{2i}} \\ T = \bar{T} & \text{on } \Gamma_{D_0} \cup \Gamma_{N_1} \cup \Gamma_{N_3} \end{cases}$$

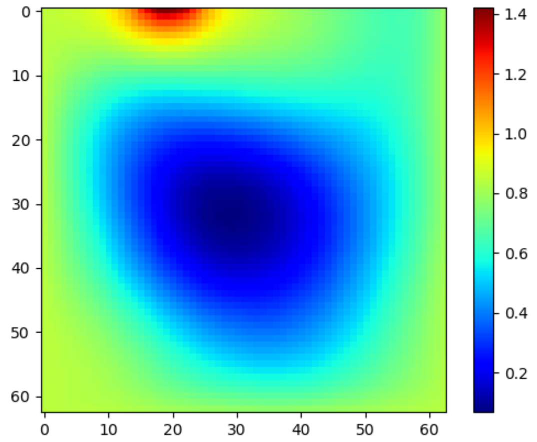
Network



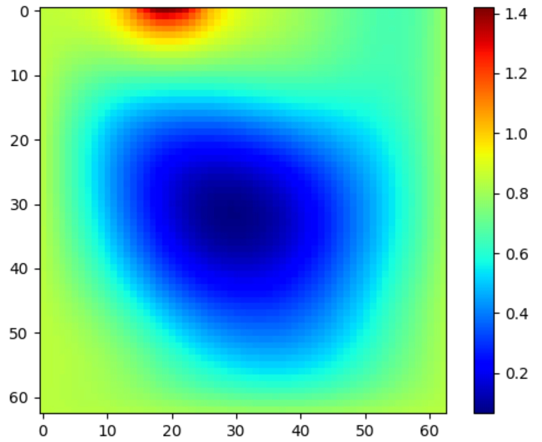
FEM



Network



FEM

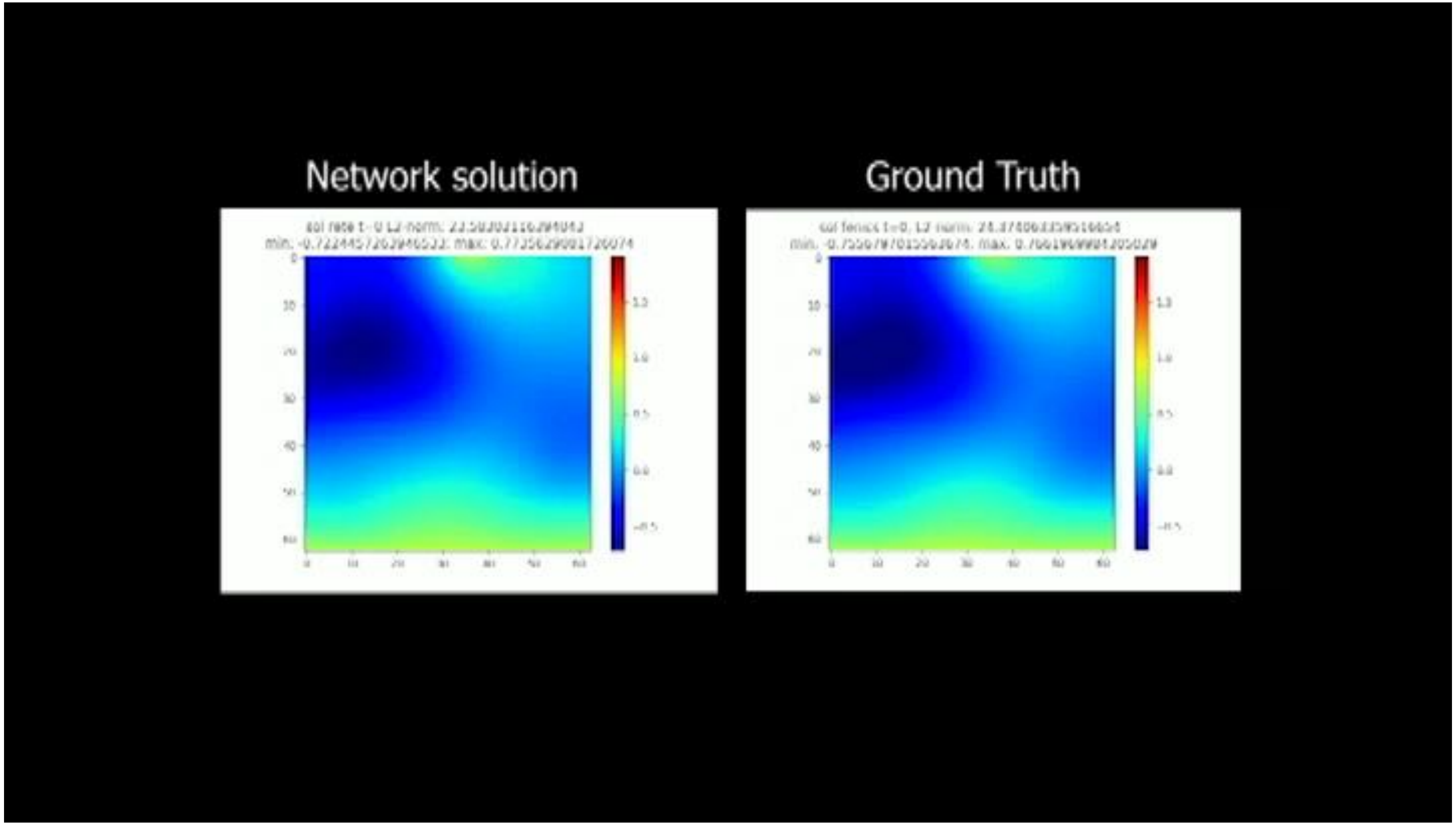


$\|T_{fem} - T_{net}\|^2_2$ on $\partial\Omega_x$	$\|T_{fem} - T_{net}\|^2_2$ on $\partial\Omega_x$
0,00476	0,001019

Numerical results:

- evolution $T_t = \nabla \cdot (\bar{k} \nabla T)$

$$
\begin{cases}
k = k(\bar{x}) = \bar{k} = \text{cost} & \\
\bar{k}\frac{\partial T(\bar{x},t)}{\partial n} = \phi(\bar{x}, \Gamma_{N_{2i}}) & \text{on } \Gamma_{N_{2i}} \\
-\bar{k}\frac{\partial T(\bar{x},t)}{\partial n} = 0 & \text{on } \Gamma_{N_1} \cup \Gamma_{N_2} \backslash \Gamma_{N_{2i}} \cup \Gamma_{N_3} \\
T = \bar{T} & \text{on } \Gamma_{D_0}
\end{cases}
$$

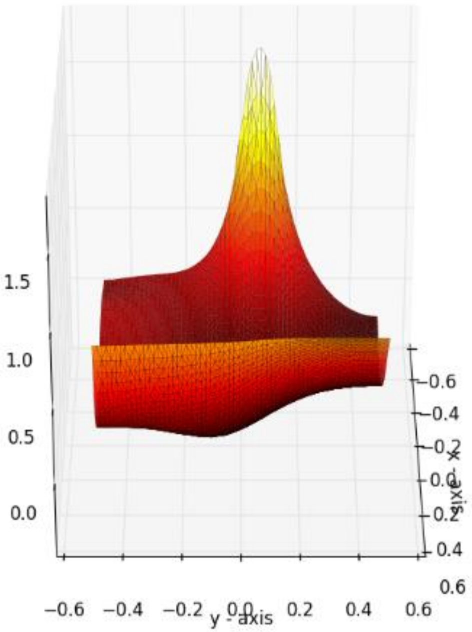


Numerical results:

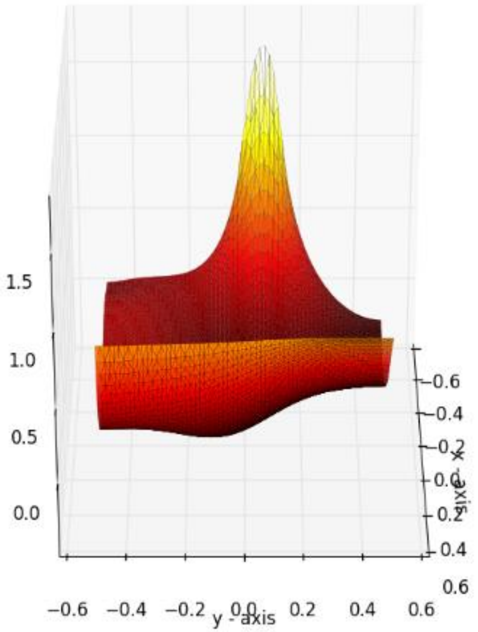
- stationary state $\boldsymbol{L}T=f$

$$\begin{cases} k = k(\bar{x}, T) & \\ k(\bar{x}, T)\dfrac{\partial T(\bar{x},t)}{\partial n} = \phi(\bar{x}, \Gamma_{N_{2i}}) & \text{on } \Gamma_{N_{2i}} \\ -k(\bar{x}, T)\dfrac{\partial T(\bar{x},t)}{\partial n} = \bar{\psi} & \text{on } \Gamma_{N_1} \cup \Gamma_{N_2} \backslash \Gamma_{N_{2i}} \cup \Gamma_{N_3} \\ T = \bar{T} & \text{on } \Gamma_{D_0} \end{cases}$$

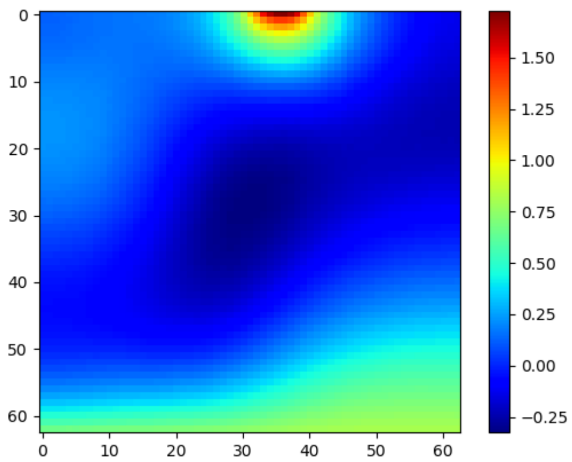
Network



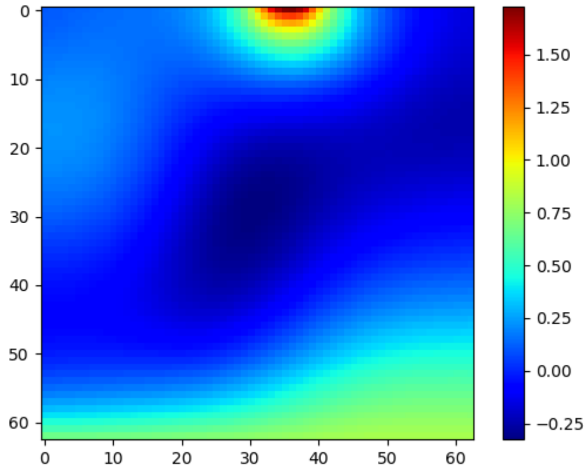
FEM



Network



FEM



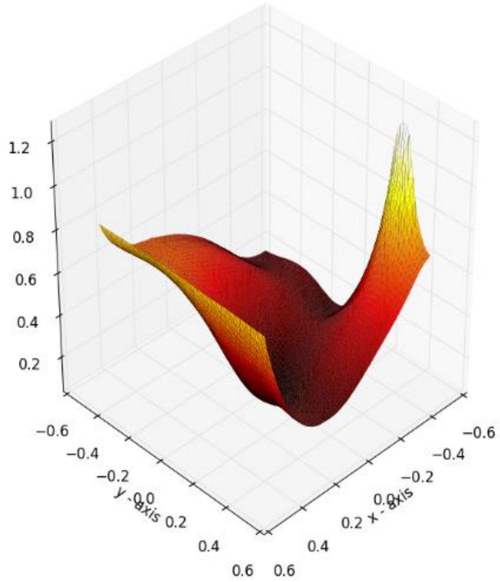
$\lVert T_{fem} - T_{net} \rVert_2^2$ on $\partial\Omega_x$	$\lVert T_{fem} - T_{net} \rVert_2^2$ on $\partial\Omega_x$
0,01829	0,02359

Numerical results:

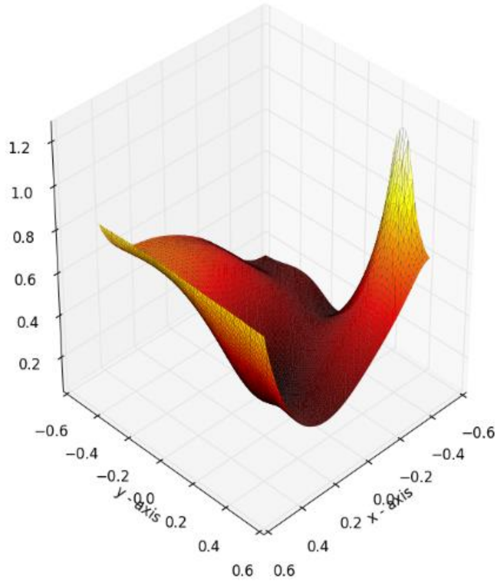
- stationary state $\boldsymbol{L}T=f$

$$
\begin{cases}
k = k(\bar{x}, T) & \\
k(\bar{x}, T)\frac{\partial T(\bar{x},t)}{\partial n} = \phi(\bar{x}, \Gamma_{N_{2i}}) & \text{on } \Gamma_{N_{2i}} \\
-k(\bar{x}, T)\frac{\partial T(\bar{x},t)}{\partial n} = \bar{\psi} & \text{on } \Gamma_{N_1} \cup \Gamma_{N_2} \backslash \Gamma_{N_{2i}} \cup \Gamma_{N_3} \\
T = \bar{T} & \text{on } \Gamma_{D_0}
\end{cases}
$$

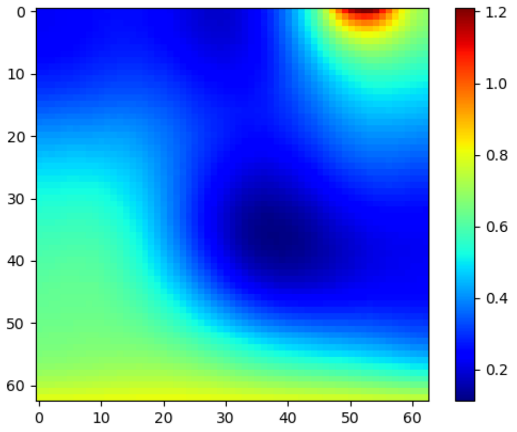
Network



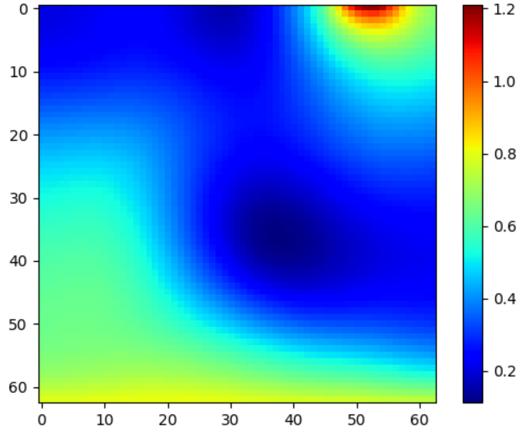
FEM



Network



FEM



$\lVert T_{fem} - T_{net} \rVert^2_2$ on $\partial\Omega_x$	$\lVert T_{fem} - T_{net} \rVert^2_2$ on $\partial\Omega_x$
0,01727	0,01832

Conclusion and future work

- Degrees of freedom:
  - position of the laser
  - training with different materials
  - spatial domain

- Execution time (resolution of image)

Stationary step

FEM	7.3592s
Network	0,28s