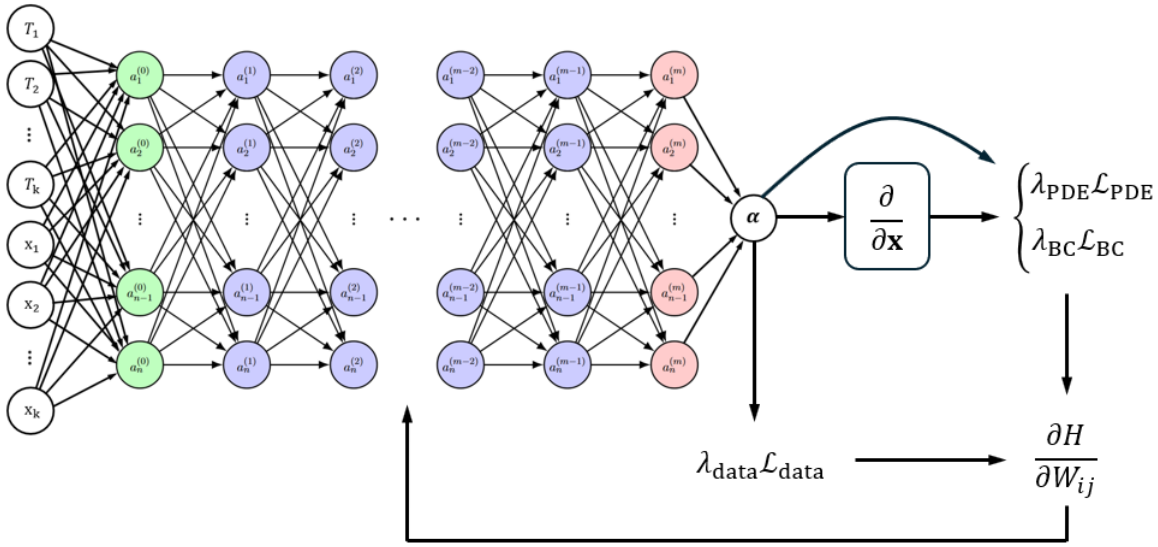
CHALMERS



Inversa värmetransportproblem med Physics-Informed Neural Networks (PINN)

Kandidatarbete vid Institutionen för Mekanik och Maritima vetenskaper

Alvin Andreasson
Nils Blomstrand
Hilma Claesson
Joel Ekeberg
Hjalmar Käll
Anton Lindqvist

INSTITUTIONEN FÖR MEKANIK OCH MARITIMA VETENSKAPER
Chalmers Tekniska Högskola
Göteborg, Sverige 2026
www.chalmers.se

Kandidatarbete vid Institutionen för Mekanik och Maritima vetenskaper

Inversa värmetransportproblem med Physics-Informed Neural Networks (PINN)

Alvin Andreasson
Nils Blomstrand
Hilma Claesson
Joel Ekeberg
Hjalmar Käll
Anton Lindqvist

CHALMERS

Institutionen för Mekanik och Maritima vetenskaper
Avdelningen för Strömningslära
Chalmers Tekniska Högskola
Göteborg 2026

Inversa värmetransportproblem med Physics-Informed Neural Networks (PINN)

Alvin Andreasson
Nils Blomstrand
Hilma Claesson
Joel Ekeberg
Hjalmar Käll
Anton Lindqvist

Handledare: Lars Davidson, Institutionen för Mekanik och Maritima vetenskaper
Examinator: Niklas Andersson, Institutionen för Mekanik och Maritima vetenskaper

Kandidatarbete 2026
Institutionen för Mekanik och Maritima vetenskaper
Chalmers Tekniska Högskola
SE-412 96 Göteborg
Telefon +46 31 772 1000

Omslagsbild: Förenklad illustration av nätverksarkitekturen för ett Physics-Informed Neural Network av godtycklig storlek.

Typsatt i LaTeX
Göteborg 2026

Inversa värmetransportproblem med Physics-Informed Neural Networks (PINN)

Alvin Andreasson, Nils Blomstrand, Hilma Claesson,
Joel Ekeberg, Hjalmar Käll, Anton Lindqvist

Institutionen för Mekanik och Maritima vetenskaper
Avdelningen för Strömningslära
Chalmers Tekniska Högskola

Sammandrag

Den partiella differentialekvationen (PDE) för värmeledning beskriver hur värme sprids genom ett medium. Att lösa det stationära värmeledningsproblemet inverst, det vill säga lösa ut den bakomliggande värmediffusiviteten från temperaturfältet, kan bli beräkningstungt och känsligt för brus. Syftet med detta kandidatarbete är att utreda möjligheterna till att i stället träna ett neuronnät, mer specifikt ett Physics-Informed Neural Network (PINN), till att prediktera värmediffusiviteten. Vidare har samma metod implementerats på PDE:n som beskriver värmetransport i biologisk vävnad, Pennes bioheat-ekvation, där dess parametrar har predikterats med PINN. Att kunna bestämma parametrarna i Pennes bioheat-ekvation kan potentiellt ha medicinska tillämpningar som exempelvis hypertermibehandling av cancertumörer. Resultatet av arbetet visar att det är möjligt att träna PINN till att med viss precision kunna prediktera värmediffusiviteten från en given temperaturdistribution, både i en och två dimensioner. Vidare har det visats att PINN presterar bättre än den mer konventionella finita differensmetoden (FDM) då brus infördes i temperaturdata. Dessutom visar resultatet att det är möjligt att träna PINN att prediktera parametrarna i Pennes bioheat-ekvation med viss precision. För framtida studier hade ytterligare tillämpningar kunnat undersökas samt att optimera neuronnätens arkitektur och struktur. Dessutom hade problemet kunnat expanderas till tre dimensioner.

Nyckelord: PINN, neuronnät, maskininlärning, värmeledning, värmetransport, inversa problem, hypertermibehandling.

Inverse Heat Transfer Problems with Physics-Informed Neural Networks (PINNs)

Alvin Andreasson, Nils Blomstrand, Hilma Claesson,
Joel Ekeberg, Hjalmar Käll, Anton Lindqvist

Department of Mechanics and Maritime Sciences
Division Fluid Dynamics
Chalmers University of Technology

Abstract

The partial differential equation (PDE) for heat conduction describes the propagation of heat through a medium. Inversely solving the stationary heat transfer problem, i.e. determining the underlying thermal diffusivity from the given temperature field, may be computationally expensive and sensitive to measurement noise. The purpose of this Bachelor's thesis is to investigate the feasibility of instead training a neural network, more specifically a Physics-Informed Neural Network (PINN), to predict the thermal diffusivity. Furthermore, the same method was applied to the PDE describing heat transfer through biological tissue, Pennes' bioheat equation, where its governing parameters were determined using a PINN. Predicting the parameters of Pennes' bioheat equation could potentially have medical applications such as hyperthermia treatments of cancerous tumours. The results of the project show that it is possible to train a PINN to, with some precision, correctly predict the thermal diffusivity from a given temperature field — both in one and two dimensions. Further, the results also show that a PINN performs better than the more conventional finite difference method (FDM) when noise is introduced to the temperature data. Similarly, the results show that it is feasible to train a PINN to predict the parameters of Pennes' bioheat equation with some precision. Future studies on the topic could investigate additional applications of the method, as well as optimize the architecture and structure of the neural networks. Lastly, the problem could also be investigated in three dimensions.

Keywords: PINN, neural network, machine learning, heat conduction, heat transfer, inverse problems, hyperthermia treatment

Förord

Denna rapport presenterar resultatet av vårt kandidatarbete, utfört vid Institutionen för Mekanik och Maritima vetenskaper under våren 2026. Uppdraget beställdes av handledare Lars Davidson som vi i projektgruppen vill rikta ett stort tack till för all stöttning och hjälp längs vägen. Vi vill även tacka examinator Niklas Andersson för granskning och bedömning av arbetet.

Begreppslista

Nedan är en lista över akronymer och begrepp som använts i rapporten, listade i alfabetisk ordning. Hänvisningar till begreppslistan kommuniceras löpande genom texten med hjälp av *kursiv stil*.

Aktiveringsfunktion	En funktion som beräknar utvärdet för en nod baserat på dess vikter och tröskelvärden.
Avkodande nivå	Den del av nätverket där den rumsliga upplösningen byggs upp igen. (eng. decoding level).
Bakåtpropagering	Användande av kedjeregeln för att beräkna derivatan av kostnadsfunktionen med avseende på vikterna.
Batchning	Uppdelning av träningsdatan i delmängder som alla används för att uppdatera vikterna.
Beräkningsnät	En uppsättning punkter för att approximera lösningar till differentialekvationer med numeriska beräkningar (eng. mesh).
Bias	Se *Tröskelvärde* (eng. bias).
Brygga	Den centrala och mest komprimerade delen av U-Net-strukturen, där upplösningen är som lägst och antalet kanaler som störst.
CNN	Faltande nätverk. Typ av neuronnät.
Direkt problem	Ett problem där den styrande ekvationen och dess parametrar är kända och lösningen, exempelvis temperaturfältet, beräknas.
Dirichlet-randvillkor	Randvillkor av typen $f(x) = D(x)$.
Djupa neuronnät	Ett neuronnät med ett eller flera gömda lager mellan indata och utdata.
Epok	En fullständig genomgång av hela träningsdatasetet genom inlärningsalgoritmen.
FDM	Finita differensmetoden. Numerisk beräkningsmetod för PDE:er.

FEM	Finita elementmetoden. Numerisk beräkningsmetod för PDE:er.
Framåtriktat neuronnät	Ett neuronnät där informationen går i en riktning, från indata till utdata, utan återkoppling.
FVM	Finita volymmetoden. Numerisk beräkningsmetod för PDE:er.
Förlustfunktion	Se *kostnadsfunktion*.
Försvinnande gradienter	När gradienten av kostnadsfunktionen med avseende på en vikt blir 0 eller mycket nära 0, vilket resulterar i att vikten inte längre uppdateras.
Hypertermibehandling	Behandlingsmetod där vävnad värms upp för att skada tumörceller i samband med cancerbehandling.
Inlärningshastighet	En hyperparameter som beskriver hur mycket gradienten av kostnadsfunktionen påverkar viktuppdateringar.
Inversa problem	Problem där orsaker eller parametrar bestäms utifrån observerade resultat, exempelvis bestämma materialegenskaper utifrån ett temperaturfält.
Kanal	En kanal i ett faltande nätverk är ett helt tvådimensionellt fält. Varje kanal innehåller alltså ett värde i varje rutnätspunkt.
Kodande nivå	En del av nätverket där den rumsliga upplösningen reduceras samtidigt som antalet kanaler ofta ökar. (eng. encoding level).
Kostnadsfunktion	Skalär funktion som mäter hur bra nätverket presterar.
Gradientnedstigning	En algoritm som numeriskt försöker hitta kostnadsfunktionens minimum genom att motsatt följa funktionens gradient.
Neuronnät	En modell bestående av sammankopplade lager av noder med träningsbara parametrar, som används för att approximera samband mellan indata och utdata (eng. neural network).
MSE	Vanligt förekommande kostnadsfunktion som består av medelvärdet av kvadraten av skillnaden mellan modellprediktionen och det korrekta värdet.
Neumann-randvillkor	Randvillkor av typen $f'(x) = N(x)$ på randen av området där PDE:n löses.
PDE	Partiell differentialekvation. Differentialekvation innehållandes derivator med avseende på fler än 1 variabel.
Perfusion	Genomströmning av blod i biologisk vävnad.

PINN	Physics-Informed Neural Network. Neuronnät som tar hänsyn till fysikaliska samband.
Pooling	En operation som minskar den rumsliga upplösningen i ett fält för att skapa en mer kompakt representation.
Regularisering	Ett bidrag till kostnadsfunktionen bestående av en norm av samtliga vikter i nätverket för att minska risken för överanpassning.
Robin-randvillkor	Randvillkor av typen $a(x)f(x)+b(x)\partial_n f(x) = R(x)$ på randen av området där PDE:n löses, där ∂_n är normalderivatan.
Skip connection	En direkt koppling mellan två nivåer i ett neuronnät, exempelvis mellan den kodande och avkodande delen i ett U-nät.
Stokastisk gradientnedstigning	Stokastisk gradientnedstigning är en iterativ process där gradienten approximeras för en delmängd av träningsdatan.
Tröskelvärde	Ett numeriskt värde som kan justera utvärdet för en neuron oberoende från kopplingar i föregående lager.
U-nät	En faltande neuronnätsstruktur med en kodande del, en brygga och och en avkodande del, där skip connections används mellan kodande och avkodande del.
Uppsampling	En operation som ökar den rumsliga upplösningen i ett fält i syfte att återskapa ett utdatafält med önskad storlek.
Vikt	Det numeriska värde som definierar styrkan på kopplingen mellan neuroner i olika lager.
Värmediffusivitet	En materialparameter som beskriver hur fort temperaturförändringar sprids i ett material.
Värmeledning	Transport av värme i ett material på grund av temperaturskillnader.
Överanpassning	När modellen inte längre lär sig generella mönster och i stället memorerar träningsdatan.

Nomenklatur

Nedan är nomenklaturen för index, uppsättningar, parametrar och variabler som har använts genom hela rapporten.

Indexering

i,j,k,l,m,n	Generella diskreta index
μ	Superskript som betecknar ett godtyckligt träningsmönster

Parametrar och variabler

Allmänt

t	Tid [s]
N	Totala antalet noder
(x, y)	Kartesiska koordinater i domän
h	Avstånd mellan noder i beräkningsnät

Neuronnät

w_{ij}	Vikt för neuron i, i lager j
$\mathbf{W}$	Viktmatris för neuroner
$\mathcal{L}$	Förlust för neuronnät
λ	Viktning av förlust för neuronnät
θ_i	Tröskelvärde för neuronaktivering
s	Signaler i neuronnät
$\boldsymbol{s}$	Tillståndsvektor för signaler i neuroner
g	Aktiveringsfunktion
b_i	Lokalt fält för neuron i

η	Inlärningshastighet
$O_i^{(\mu)}$	Utsignal från utsignalsneuron i för mönster μ
V_j	Utsignal från dold neuron j
$t_i^{(\mu)}$	Önskat målvärde för neuron i och mönster μ
$x^{(\mu)}$	Indatamönster μ
$\Delta_m^{(\mu)}$	Viktat fel för utsignalsneuron m vid mönster μ
$\delta_m^{(\mu)}$	Bakåtpropagerande fel till dold neuron m
$a_j^{(l)}$	Aktivering för neuron j i lager l
H	Energifunktion tillika kostnadsfunktion
p	Antal lagrade mönster
$r(\mathbf{x})$	PDE-residual
N_c	Antal punkter för utvärdering av MSE
γ	Fysikaliska parametrar i PDE
$\mathcal{N}$	Differentialoperator i PDE
$\hat{u}$	Nätverkets approximation av fältvariabeln u

Värmeledning

α	Värmediffusivitet [m^2/s]
T	Temperatur [K]
Q	Värmeproduktion [W/m^3]
k	Värmeledningskoefficient [W/(m K)]
β	Perfusionsparameter [$\mathrm{W}/(\mathrm{m}^3$ K)]
ρ_b	Blodets densitet [kg/m^2]
c_b	Specifik värmekapacitet i blod [J/(kg K)]
w_b	Perfusionshastighet i blodet [s^{-1}]
T_a	Blodets temperatur i artärerna [K]
ρ	Densitet [kg/m^2]
Q_{met}	Värmeproduktion från metabolism [W/m^3]
q	Värmeflödestäthet [W/m^2]
σ_T	Standardavvikelsen av temperaturfältet
μ_T	Medelvärdet av temperaturfältet

Innehåll

Figurer

Tabeller

1
Inledning

Partiella differentialekvationer (PDE:er) är matematiska samband som kan användas för att beskriva allt från naturliga processer som fluidströmning, värmeöverföring och vågutbredning till finansiella modeller och populationsdynamik [1] [2]. Med tanke på den centrala roll som PDE:er spelar i många tillämpningar är utvecklingen av effektiva lösningsmetoder av stor betydelse. På grund av komplexiteten i många av de system som kan beskrivas av PDE:er går det inte att lösa dessa ekvationer analytiskt, utan olika numeriska beräkningsmetoder behövs. Dessa metoder är ofta beräkningstunga och tidskrävande. Därför är metoder som kan göra beräkningsprocessen lättare efterfrågade. En sådan metod skulle kunna vara *Physics-Informed Neural Networks* (PINN).

Ett *neuronnät* är grunden i en maskininlärningsmetod som bygger på att neuroner aktiverar varandra, likt hur ett biologiskt nervsystem fungerar [3]. I ett artificiellt neuronnät sammankopplas neuroner av funktioner med olika *vikter* och *tröskelvärden*, även kallade *biaser* (från engelskans bias). Genom att strukturera dessa neuroner i lager kan nätverket, genom att justera vikter och tröskelvärden, generera ett resultat utifrån given indata. För att skatta kvaliteten på resultatet från neuronnätet används en *kostnadsfunktion* som också ligger till grund för hur vikter och tröskelvärden justeras under iterationerna. En kostnad, eller förlust, är ett mått på hur långt ifrån det faktiska värdet resultatet från neuronnätet är och beräknas med kostnadsfunktionen. [4].

Ett PINN skiljer sig från ett vanligt neuronnät på sättet att det innehåller ytterligare bidrag till förlusten utifrån kända fysikaliska egenskaper om systemet [5]. Matematiskt kan förlusten beskrivas som

$$\mathcal{L} = \lambda_{\text{data}}\mathcal{L}_{\text{data}} + \lambda_{\text{PDE}}\mathcal{L}_{\text{PDE}} + \lambda_{\text{BC}}\mathcal{L}_{\text{BC}} \tag{1.1}$$

där $\mathcal{L}_{\text{data}}$ är kostnaden från att nätverkets resultat avviker från den korrekta datan. $\mathcal{L}_{\text{PDE}}$ och $\mathcal{L}_{\text{BC}}$ i sin tur skattar fel mot den fysik som man vill att systemet ska uppfylla, där $\mathcal{L}_{\text{PDE}}$ är kostnaden från PDE:n och $\mathcal{L}_{\text{BC}}$ är kostnaden från randvillkoren. λ är vikter för respektive kostnad för att kunna reglera deras bidrag till den totala kostnaden $\mathcal{L}$. Genom att introducera fysikinformationen i nätverksträningen kan man skapa ett neuronnät som konvergerar till ett fysikaliskt stringent resultat och kan därför tillämpas för att lösa komplexa fysikaliska problem.

För att lösa PDE:er inom bland annat fluidmekanik [6], kvantmekanik [7] och värmetransport [8] har PINN sett allt fler tillämpningar de senaste åren. En fördel med PINN jämfört med traditionella numeriska metoder såsom *finita elementmetoden* (FEM) är att PINN kan producera en kontinuerlig lösning utan att vara

beroende av ett diskretiserat beräkningsnät [9]. Vidare kan PINN hantera situationer med brusig eller ofullständig data, framför allt om PINN kombineras med andra nätverksmetoder som *faltande nätverk* (CNN) [10]. FEM kan exempelvis inte heller användas för att lösa *inversa problem* (utan måste kombineras med andra metoder) [11]. För dessa problem är det därför lämpligt att utnyttja PINN. Ett inverst problem definieras som processen att från observerad data bestämma faktorerna som har orsakat den [9]. Detta kan exempelvis innebära att bestämma okända parametrar i en PDE utifrån data, vilket är vad detta arbete ämnar göra.

Att kunna ta reda på parametrarna i de differentialekvationer som styr vår omgivning har många applikationer, inte minst för värmetransport. Att kunna bestämma exempelvis värmeledningskoefficienten kan inom materialteknik vara intressant för att bättre förstå ett material [11]. Invers PINN har också tillämpningar inom biomedicin där det kan användas för att bestämma egenskaper hos biologisk vävnad som i sin tur kan användas för behandling av cancer. *Hypertermibehandling* är en behandlingsmetod för cancer där tumörvävnad värms upp så att den dör, utan att skada frisk vävnad [12]. Precisionen i att åstadkomma rätt temperatur, på rätt plats, under rätt tid är dock svår [13]. Denna behandling skulle därför kunna förbättras genom mer kännedom om de parametrar som styr värmetransporten i vävnaden. Värmetransport i biologisk vävnad kan modelleras som en summa av värmeledning och *perfusion*, en typ av konvektion som orsakas av att värme transporteras med blodflödet [14].

1.1 Syfte

Syftet med detta arbete är att testa och utreda huruvida PINN kan användas för att inverst bestämma okända parametrar i PDE:er som beskriver värmetransport från en temperaturfördelning. Initialt ska användningen av PINN för att prediktera värmediffusivitet för värmeledningsproblem i en och två dimensioner utforskas. Sedan utforskas om nätverket kan modifieras för att även kunna prediktera perfusionstermen i PDE:n som beskriver värmetransport i biologisk vävnad.

1.2 Mål

Målet med detta arbete är att konstruera PINN-modeller som utifrån en temperaturfördelning kan prediktera den bakomliggande värmediffusiviteten i både en och två dimensioner. Ett ytterligare mål har varit att använda denna metodik för att bestämma både värmediffusivitet och perfusionsterm för den PDE som beskriver värmetransporten för en förenklad modell av en cancertumör i biologisk vävnad.

1.3 Avgränsningar

För att göra projektet genomförbart inom ramen för ett kandidatarbete kommer ett flertal avgränsningar att göras. En avgränsning kommer vara att alla PDE:er som

ska undersökas under projektets gång kommer att begränsas till en- och tvådimensionella problem i syfte att minimera datorbaserad beräkningstid. Även randvillkoren kommer hållas enkla för att minimera beräkningstiden, därav utforskas endast Dirichlet-, Neumann- och Robin-randvillkor.

Projektet begränsas dessutom på grund av att inversa problem för stationära temperaturfält ofta är dåligt konditionerade. Vad detta betyder är att små störningar i mätdata, exempelvis mätbrus eller diskretiseringsfel, kan ge stora förändringar i det rekonstruerade α-fältet. Av denna anledning kommer projektet först och främst hålla sig till att endast utveckla PINN-modeller som predikterar α för ett stort antal datapunkter opåverkade av brus.

Neuronnät av den typ och storlek som har använts kräver mycket datorresurser och tid för att tränas. Av denna anledning har arbetet behövt begränsas till enklare geometrier som rektangulära defekter eller matematiska funktioner. Hyperparameteroptimering för PINN-modellerna kommer endast att göras i en begränsad utsträckning för att uppnå stabil konvergens snarare än maximal precision eftersom målet ligger mer i riktningen av att undersöka genomförbarheten än att maximera prestandan.

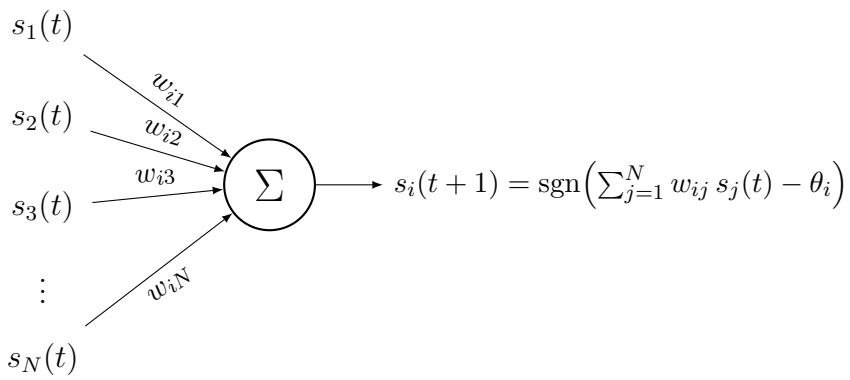
Slutligen kommer inga praktiska experiment att genomföras för att validera modellerna mot verkligheten på grund av att tillgång till rätt utrustning saknas.

2
Teori

I följande avsnitt presenteras den relevanta teorin för projektet. Inledningsvis förklaras grundläggande hur neuronnät är uppbyggda och dess ingående delar. Därefter beskrivs numeriska metoder för lösning av PDE:er och till sist förklaras fysikaliska samband så som olika typer av värmetransport.

2.1 Neuronnät

Att artificiellt försöka återskapa det biologiska neuronnätet har varit centralt för utvecklingen av maskininlärning och är från början helt inspirerat av hur det biologiska nervsystemet bearbetar information. En av de första och mest framstående modellerna för att försöka återskapa en biologisk neuron är den så kallade McCulloch-Pitts-neuronen 2.1 [15].



Figur 2.1: Schematisk illustration av en McCulloch-Pitts-neuron, baserad på [15]. Neuronen summerar insignalerna $s_j(t)$ vid det diskreta tidssteget t som viktas med faktorerna w_{ij} vilka ämnas motsvara de synaptiska kopplingarnas olika styrkor i ett biologiskt nervsystem. Därefter avges en aktiv eller inaktiv respons beroende på om summan överstiger tröskelvärdet θ_i med hjälp av aktiveringsfunktionen sgn.

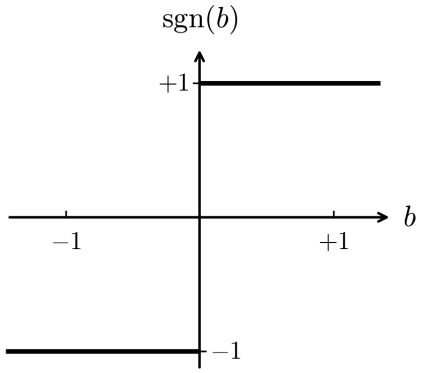
Modellen som McCulloch och Pitts tog fram återger hur den utgående signalen från en given neuron i kan beskrivas som en viktad summa av utsignalerna från neuronerna j, där $j \in 1, 2, ..., N$ för ett godtyckligt antal neuroner N. Efter att en signal $s_j(t)$ multiplicerats med motsvarande vikt w_{ij} subtraheras ett tröskelvärde θ_i som också är specifikt för den givna neuronen i. Summan av alla produkter av insignaler $s_j(t)$ och vikter w_{ij} efter subtraktion av tröskelvärdet θ_i körs sedan genom en så

kallad *aktiveringsfunktion*. I McCulloch-Pitts ursprungliga modell gjordes detta för att återskapa det biologiska fenomenet att neuroner antingen är aktiva eller inaktiva med en tydlig distinktion. Fenomenet kan observeras genom att studera pyramidala neuroner hos fiskar med hjälp av elektromagnetiska sensorer och kallas för ”spike train” på engelska [15]. McCulloch och Pitts valde därför att använda signumfunktionen som aktiveringsfunktion för att försöka återskapa just det fenomenet. Det finns många olika aktiveringsfunktioner som passar olika bra till olika applikationer men signumfunktionen var den som bäst tycktes återspegla de biologiska neuronernas beteende. Funktionen är styckvis konstant och returnerar +1 om argumentet är positivt och -1 om argumentet är negativt. Med andra ord gäller:

$$\operatorname{sgn}(b) = \begin{cases} -1, & b < 0, \\ +1, & b \geq 0. \end{cases} \tag{2.1}$$

som appliceras på ett argument b vilket definieras som summan av alla vikter och insignaler till neuronen i efter subtraktion med tröskelvärdet.

$$b_i(t) = \sum_{j=1}^{N} w_{ij} s_j(t) - \theta_i, \tag{2.2}$$



Figur 2.2: Signum-funktionen applicerad på argumentet b som motsvarar den viktade summan av alla signaler från föregående neuroner j subtraherat med tröskelvärdet θ_i.

Tillståndet i neuronen i beräknas sedan rekursivt över diskreta tidssteg $t \in 1, 2, ..., t_n$ och med endast en uppsättning neuroner, eller ett lager neuroner i enlighet med McCulloch och Pitts modell enligt följande [15].

$$s_i(t+1) = \operatorname{sgn}\left(\sum_{j=1}^{N} w_{ij} s_j(t) - \theta_i\right) = \operatorname{sgn}[b_i(t)]. \tag{2.3}$$

Förhoppningen var sedan att ett antal sammankopplade McCulloch-Pitts-neuroner skulle ha förutsättningarna att återskapa samma beteende som kunde observeras hos de biologiska neuronerna. En central fråga som återstod att besvara för McCulloch-Pitts var dock hur vikterna w_{ij} ska väljas och hur nätverket faktiskt skulle kunna lära sig. År 1949 föreslog den kanadensiske neuropsykologen Donald Hebb en princip som kan sammanfattas med att neuroner som aktiveras samtidigt stärker sina

inbördes kopplingar (”Neurons that fire together, wire together”) [15]. Tillämpat på McCulloch-Pitts-neuroner kunde detta formuleras som Hebbs regel:

$$w_{ij} = \frac{1}{N} \sum_{\mu=1}^{p} x_i^{(\mu)} x_j^{(\mu)}, \tag{2.4}$$

där $x^{(\mu)} \in \{-1, +1\}^N$ betecknar det lagrade mönstret μ av totalt p lagrade mönster och N är antalet neuroner. Regeln implicerar att vikten w_{ij} blir positiv om neuronerna i och j tenderar att ha samma tillstånd i de lagrade mönstren och negativ om de tenderar att ha olika tillstånd. Vikten w_{ij} beräknad med Hebbs regel (2.4) kan därefter placeras på rad i och kolumn j i en viktmatris $\mathbf{W}$ av dimension $N \times N$:

$$\mathbf{W} = \frac{1}{N} \sum_{\mu=1}^{p} \boldsymbol{x}^{(\mu)} \left(\boldsymbol{x}^{(\mu)}\right)^{\mathsf{T}} = \frac{1}{N} \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1N} \\ w_{21} & w_{22} & \cdots & w_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ w_{N1} & w_{N2} & \cdots & w_{NN} \end{bmatrix}, \tag{2.5}$$

där varje term $\boldsymbol{x}^{(\mu)} \left(\boldsymbol{x}^{(\mu)}\right)^{\mathsf{T}}$ är en yttre produkt som bildar en $N \times N$-matris. Uppdateringsregeln (2.7) kan då skrivas kompakt som en matrismultiplikation följd av en komponentvis tillämpning av signum-funktionen:

$$\boldsymbol{s}(t+1) = \operatorname{sgn}[\mathbf{W}\, \boldsymbol{s}(t)]\,, \tag{2.6}$$

där $\boldsymbol{s}(t) = \left[s_1(t),\, s_2(t),\, \ldots,\, s_N(t)\right]^{\mathsf{T}}$ är tillståndsvektorn vid tidssteg t och signumfunktionen appliceras elementvis på den resulterande vektorn $\mathbf{W}\boldsymbol{s}(t)$.

Notera att efter att vikterna är satta förblir de konstanta under hela inlärningsprocessen och väljs enligt Hebbs regel för att få neuronerna i nätverket att sträva efter att återgå till närmaste lagrade mönster. Vikterna är dessutom symmetriska enligt (2.4) eftersom $w_{ij} = w_{ji}$ och nätverket är fullt kopplat vilket innebär att varje neuron i princip är kopplad till alla andra. Med vikterna satta enligt Hebbs regel och med uppdateringsregeln (2.3) erhålls det som John Hopfield populariserade 1982 under namnet Hopfield-nätverket [15].

Om tillstånden eller aktiveringarna $s_i(t) \in \boldsymbol{s}(t)$ uppdateras samtidigt med hjälp av den ovan fastställda uppdateringsregeln (2.3) kallas det för en synkron uppdatering. För ett nätverk med ett enda lager av N neuroner kan tillstånden även uppdateras asynkront vilket innebär att enskilt utvalda neuroner m kan uppdateras enligt:

$$s_m(t+1) = \operatorname{sgn}\left(\sum_{j=1}^{N} w_{mj}\, s_j(t)\right), \tag{2.7}$$

medan övriga neuronerns tillstånd förblir oförändrade. För hela neuronnätet innebär det att:

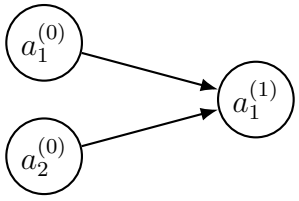
$$s_i(t+1) = \begin{cases} g\left(\sum_j w_{mj} s_j(t) - \theta_m\right), & \text{för } i = m, \\ s_i(t), & i \neq m. \end{cases} \tag{2.8}$$

Nätverkets dynamik kan sedan tolkas med hjälp av en *kostnadsfunktion*:

$$H(\boldsymbol{s}) = -\frac{1}{2}\sum_{i,j} w_{ij}\, s_i s_j, \tag{2.9}$$

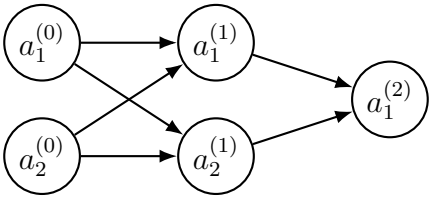
som Hopfield kunde bevisa var garanterat icke-ökande under den asynkrona uppdateringsregeln (2.7). Nätverkets dynamik konvergerar därmed alltid mot ett lokalt minimum för H vilket svarar mot ett lagrat mönster eller en förvrängd version av mönstret.

Med ett enda lager neuroner finns dock en fundamental begränsning. Eftersom alla neuroner delar samma lager och kopplas tillbaka till varandra finns det ingen distinktion mellan in- och utlager vilket innebär att nätverket är rekursivt. I praktiken innebär det att nätverket kan lagra och återkalla enkla binära mönster men aldrig genomföra fullständiga mappningar från en insignal till en utsignal som biologiska neuroner kan. Lagringskapaciteten för ett Hopfield-nätverk är också kraftigt begränsad och börjar misslyckas med återkallning av inprogrammerade mönster om antalet mönster $p \gtrsim 0,14\,N$ [15]. Eftersom vikterna aldrig uppdateras uteblir också ett fundamentalt fenomen som återfinns hos de biologiska neuronerna, nämligen möjligheten att variera styrkan i kopplingarna mellan sig. Det är egentligen precis det vikterna är ämnade att återskapa och i naturen är dessa högst plastiska och ändras ständigt i respons till ny information vilket spelar en betydande roll i hur biologiska neuronnät lär sig [15]. Dessa begränsningar motiverade utvecklingen av *djupa neuronnät* vilket ledde fram till den första fundamentala komponenten till artificiella neuronnät: perceptronen , vilken består av totalt tre neuroner i två olika lager.



Figur 2.3: En perceptronenhet, enklaste formen av neuronnät framtagen av McCulloch och Pitts. Kunde användas för att lösa enkla binära klassificeringsproblem [15]

Lägg märke till notationen som används för att beteckna positionen för varje neuron och dess aktivering $a_j^{(l)}$. Indexeringen j definierar positionen vertikalt och superskriptet l fastställer vilket lager neuronen och aktiveringen tillhör. För att ett neuronnät ska kunna kategoriseras som djupt behöver det ha minst ett lager av dolda neuroner vilket innebär att det resulterande neuronnätet har ett inlager, ett mellanlager (dolt) och ett utlager. Djupa neuronnät kan konstrueras genom att kombinera perceptroner.



Figur 2.4: En kombination av två perceptronenheter skapar i den ovanstående konstellationen ett av de första exemplen på djupa neuronnät [15].

Med introduceringen av perceptronen och kombinationer av perceptroner kunde en helt ny kategori av problem lösas eftersom neuronnätets insignaler nu kunde skiljas från utsignalerna. Mer specifikt kunde nu mappningar och klassificeringar göras vilket blev startskottet för maskininlärningens explosionsartade utveckling. Grundidén är att ett givet mönster eller indata $\mathbf{x}^{(\mu)}$ bearbetas av nätverket för att försöka producera ett väldefinierat önskat resultat $\mathbf{t}^{(\mu)}$. Notera här att superskriptet μ används för att kommunicera vilket specifikt träningsmönster indatan, det önskade resultatet och sedan också de producerade utsignalerna hör till. Gör därför skillnad på den notationen och superskriptet som används för att referera till en specifik neuron $a_j^{(l)}$. I litteraturen är det denna konvention som används så därför används den även här [15].

$$x^{(\mu)} = \begin{bmatrix} x_1^{(\mu)} \\ x_2^{(\mu)} \\ \vdots \\ x_N^{(\mu)} \end{bmatrix}, \qquad t^{(\mu)} = \begin{bmatrix} t_1^{(\mu)} \\ t_2^{(\mu)} \\ \vdots \\ t_M^{(\mu)} \end{bmatrix}. \tag{2.10}$$

Baserat på specifika insignaler kommer neuronnätet att producera utsignaler $\mathbf{O}^{(\mu)}$ som sedan jämförs mot det önskade resultatet. Med andra ord eftersträvas att alla vikter och tröskelvärden w_{ij}, θ_i ska resultera i utsignaler $O_i^{(\mu)} = t_i^{(\mu)}$ för alla olika indata μ och neuroner i. Olika indata $x^{(\mu)}$ kan alltså ha olika önskade resultat $t^{(\mu)}$ för samma antal datapunkter N och M och därför är det viktigt att göra den distinktionen. Notera att det endast är de slutgiltiga neuronernas utsignaler som betecknas med $O_i^{(\mu)}$ eftersom dessa blir de resulterande utsignalerna för hela neuronnätet. Alla utsignaler för alla dolda neuroner betecknas istället med V_j och ges av (2.11)

$$V_j = g(b_j) \quad \text{där} \quad b_j = \sum_k w_{jk} x_k - \theta_j, \tag{2.11}$$

med vikter w_{jk} och tröskelvärden θ_j. Funktionen $g(b)$ motsvarar aktiveringsfunktionen och argumentet b_j beräknas fortfarande precis på samma sätt som i (2.2) och kallas generellt för det lokala fältet. Utsignalerna O_i kan därefter beräknas genom:

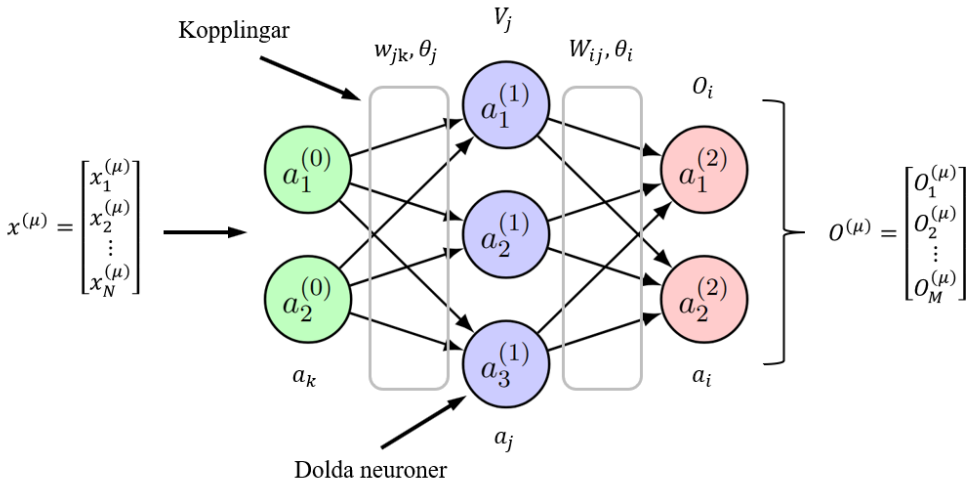
$$O_i = g(b_i) \quad \text{där} \quad b_i = \sum_j w_{ij} V_j - \theta_i. \tag{2.12}$$

där numreringen $i = 1, 2, ..., M$ indexerar utsignalsneuronerna med vikter w_{ij}, $j = 1, 2, ..., P$ indexerar de dolda neuronerna med vikter w_{jk} och $k = 1, 2, ..., N$ indexerar insignalsneuronerna, som naturligt inte har några associerade vikter eftersom de är

de första neuronerna i sekvensen. Notera att vikten w_{ij} som kopplar samman neuron i och j indexeras från utlagret och bakåt, se figur 2.5.

2.1.1 Gradientnedstigning

För att ett djupt neuronnät ska kunna lära sig kan inte längre Hebbs regel användas eftersom det består av mer än ett lager. Vikterna w kommer inte heller vara statiska längre så det krävs en systematisk metod för att justera alla vikter och tröskelvärden i nätverket. Härledningen av denna metod kommer göras med hjälp av ett exempelnätverk 2.5.



Figur 2.5: Genom att kombinera fler perceptronenheter kan mer komplexa neuronnät konstrueras. De gröna neuronerna motsvarar insignalsneuronerna och indexeras med k, de blåa kategoriseras som dolda och indexeras med j och de röda motsvarar utsignalsneuronerna, indexerade med i.

Den metod som ligger till grund för i princip all modern träning av neuronnät kallas *gradientnedstigning* eller gradient descent på engelska. Idén är att definiera en *kostnadsfunktion* H som kvantifierar hur långt nätverkets utsignaler $O_i^{(\mu)}$ ligger från de önskade målvärdena $t_i^{(\mu)}$ som definierades genom 2.10. Skillnaden mellan den producerade utsignalen $O_i^{(\mu)}$ och det önskade resultatet $t_i^{(\mu)}$ kan sedan summeras över alla träningsmönster $\mu = 1, \ldots, p$ och alla utsignalsneuroner $i = 1, \ldots, M$ för att erhålla sambandet:

$$H = \frac{1}{2} \sum_{\mu=1}^{p} \sum_{i=1}^{M} \left(t_i^{(\mu)} - O_i^{(\mu)} \right)^2 . \tag{2.13}$$

Varje vikt kan sedan justeras sekventiellt i riktning mot den negativa gradienten av H med avseende på den aktuella vikten vilket resulterar i att H minskar. Genom att också definiera en *inlärningshastighet* $\eta > 0$ kan viktuppdateringar definieras som:

$$\delta W_{mn} = -\eta \frac{\partial H}{\partial W_{mn}}, \qquad \delta w_{mn} = -\eta \frac{\partial H}{\partial w_{mn}}, \tag{2.14}$$

där W_{mn} betecknar de vikter som kopplar ihop utlagret med det dolda lagret och w_{mn} de vikter som kopplar ihop det dolda lagret med inlagret som etablerat i figur 2.5.

2.1.2 Bakåtpropagering

Derivatorna i (2.14) beräknas med hjälp av kedjeregeln. För samma nätverk 2.5 med index k, j och i för de olika lagren beräknas:

$$V_j^{(\mu)} = g\left(\sum_{k=1}^{2} w_{jk}\, x_k^{(\mu)} - \theta_j\right) = g\left(b_j^{(\mu)}\right), \tag{2.15}$$

$$O_i^{(\mu)} = g\left(\sum_{j=1}^{3} W_{ij}\, V_j^{(\mu)} - \theta_i\right) = g\left(b_i^{(\mu)}\right), \tag{2.16}$$

där g återigen är aktiveringsfunktionen, $b_j^{(\mu)}$ och $b_i^{(\mu)}$ är de lokala fälten i respektive lager och θ_j, θ_i är de tillhörande tröskelvärdena.

För vikterna W_{mn} som kopplar utlagret till det dolda lagret tillämpas kedjeregeln en gång:

$$\frac{\partial H}{\partial W_{ij}} = -\sum_{\mu}\sum_{i}\left(t_i^{(\mu)} - O_i^{(\mu)}\right)\frac{\partial O_i^{(\mu)}}{\partial W_{ij}}. \tag{2.17}$$

Utsignalen ges av $O_i^{(\mu)} = g(b_i^{(\mu)})$ och det dolda lagrets tillstånd V_j beror inte på w_{ij} eftersom information endast flödar från vänster till höger. Därför kallas denna typ av neuronnät för *framåtriktat* (feed-forward på engelska). Notera att felet summeras över alla mönster μ för att uttrycka gradienten. Det ger vidare att:

$$\frac{\partial O_i^{(\mu)}}{\partial W_{mn}} = g'\left(B_i^{(\mu)}\right)\,\delta_{im}\, V_n^{(\mu)}, \tag{2.18}$$

där g' betecknar derivatan av aktiveringsfunktionen och δ_{im} definieras som Kroneckerdeltat. Om (2.17) och (2.18) kombineras erhålls uppdateringsregeln för utsignalsneuronernas vikter W_{mn}.

$$\delta W_{mn} = \eta \sum_{\mu=1}^{p} \underbrace{\left(t_m^{(\mu)} - O_m^{(\mu)}\right)\, g'\left(B_m^{(\mu)}\right)}_{\equiv\ \Delta_m^{(\mu)}} V_n^{(\mu)}, \tag{2.19}$$

där $\Delta_m^{(\mu)}$ definierar det viktade felet för utsignalsneuron m vid mönster μ. Om $O_m^{(\mu)} = t_m^{(\mu)}$ är felet noll och vikten förblir oförändrad. Precis som för gradienten summeras de viktade felen för alla träningsmönster μ [15].

För vikterna w_{mn} som kopplar insignalerna till det dolda lagret appliceras kedjeregeln ytterligare gånger eftersom felet som evalueras med $O_m^{(\mu)} - t_m^{(\mu)}$ inte beror direkt på w_{mn} utan indirekt via de dolda neuronernas tillstånd V_j:

$$\frac{\partial O_i^{(\mu)}}{\partial w_{mn}} = \sum_j \frac{\partial O_i^{(\mu)}}{\partial V_j^{(\mu)}}\frac{\partial V_j^{(\mu)}}{\partial w_{mn}} = \sum_j g'\left(B_i^{(\mu)}\right) W_{ij}\; g'\left(b_j^{(\mu)}\right)\delta_{im}\, x_n^{(\mu)}. \tag{2.20}$$

Den resulterande uppdateringsregeln för vikterna tillhörande det dolda lagret w_{mn} blir därmed:

$$\delta w_{mn} = \eta \sum_{\mu=1}^{p} \underbrace{\left[\sum_i \Delta_i^{(\mu)} W_{im} \right] g'\left(b_m^{(\mu)}\right)}_{\equiv\ \delta_m^{(\mu)}} x_n^{(\mu)}, \tag{2.21}$$

där $\delta_m^{(\mu)}$ är det bakåtpropagerande felet till den dolda neuronen m. Det resulterar i att neurontillstånden beräknas framåt enligt (2.15), (2.16) medan felen propagerar bakåt genom (2.21). Därav namnet *bakåtpropagering* (eng. backpropagation) [15]. Tröskelvärdena uppdateras på motsvarande vis enligt:

$$\delta\Theta_m = -\eta \sum_{\mu=1}^{p} \Delta_m^{(\mu)}, \qquad \delta\theta_m = -\eta \sum_{\mu=1}^{p} \delta_m^{(\mu)}. \tag{2.22}$$

Notera särskilt att uppdateringsformlerna (2.19) och (2.21) har exakt samma struktur. Det gör algoritmen enkel att generalisera till ett godtyckligt antal lager genom att iterera bakåtpropageringsalgoritmen rekursivt. Detta är i stora drag vad som görs för näst intill all form av maskininlärning. I praktiken summeras felet sällan över alla träningsmönster μ samtidigt utan istället väljs enstaka mönster slumpmässigt och summan över μ i (2.19), (2.21) utelämnas. När viktuppdateringen sedan görs så pekar inte förändringen nödvändigtvis i den exakta globala gradientminimerande riktningen men i genomsnitt minskar H ändå över tillräckligt många iterationer och *epoker*. Metoden kallas då snarare för *stokastisk gradientnedstigning* (eng. stochastic gradient descent). Fördelen med den är att benägenheten att fasta i lokala minima minskar. En epok definieras som en fullständig genomgång av träningsdatan med tillhörande viktuppdateringar och är ett disciplinspecifikt mått på träningstid. Efter en epok körs neuronnätet genom träningen igen i hopp om att felet ska minska ännu mer och i slutändan leda till en konvergerad modell. Hur konvergens avgörs framgår i avsnitt 3.2.3.

2.1.3 Olika aktiveringsfunktioner

I avsnittet 2.1 introducerades signumfunktionen (2.1) som en så kallad aktiveringsfunktion och användes under hela den efterföljande härledningen av både gradientnedstigning och bakåtpropagering. Valet motiverades ursprungligen av att den ger neuronnätet karaktäristik som efterliknar de biologiska motsvarigheternas binära beteende. För Hopfield-nätverket, där vikterna är fasta och nätverket arbetar genom att iterativt konvergera mot lagrade mönster är signumfunktionen tillräcklig. För djupa neuronnät som tränas med gradientnedstigning uppstår dock ett fundamentalt problem med bakåtpropageringsalgoritmen (2.19). Mer specifikt kräver (2.21) att derivatan $g'(b)$ av aktiveringsfunktionen kan beräknas eftersom den ingår som en faktor i varje viktuppdatering. Signumfunktionens derivata är noll överallt förutom i $b = 0$ där den istället är odefinierad vilket innebär att alla viktuppdateringar blir noll och nätverket inte kan lära sig. Ett djupt neuronnät som ska tränas med gradientnedstigning behöver därför en aktiveringsfunktion som är kontinuerlig och deriverbar [15].

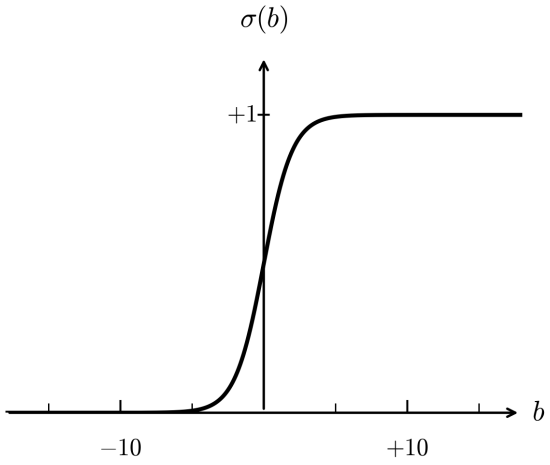
En av de tidigaste och mest studerade kontinuerliga aktiveringsfunktionerna är sigmoidfunktionen, definierad som

$$\sigma(b) = \frac{1}{1 + e^{-b}}, \tag{2.23}$$

som avbildar hela den reella tallinjen på intervallet (0,1). Sigmoidfunktionen kan ses som en mjuk version av signumfunktionen som dels har den efterfrågade egenskapen att dess derivata är väldefinierad och skild från noll men också att den kan uttryckas i termer av funktionen själv

$$\sigma'(b) = \sigma(b)(1 - \sigma(b))\,, \tag{2.24}$$

vilket gör den beräkningsmässigt effektiv. Sigmoidfunktionens utdata kan dessutom tolkas som en sannolikhet vilket gör den särskilt lämplig i utlager för binära klassificeringsproblem.

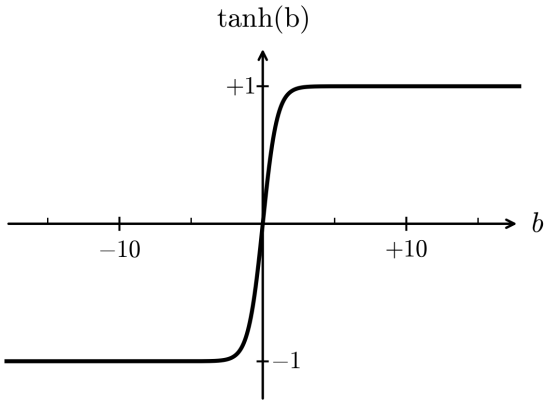


Figur 2.6: Aktiveringsfunktionen sigmoid, traditionellt sett betecknad med $\sigma(b)$ med det lokala fältet b.

En nära besläktad funktion är hyperbolisk tangens, eller tanh:

$$\tanh(b) = \frac{e^{b} - e^{-b}}{e^{b} + e^{-b}}, \tag{2.25}$$

som avbildar den reella tallinjen på intervallet (-1,1) och är centrerad kring origo till skillnad från sigmoidfunktionen. Funktionerna är relaterade genom $\tanh(b) = 2\sigma(2b) - 1$ och delar därför samma grundläggande karaktäristik [15].



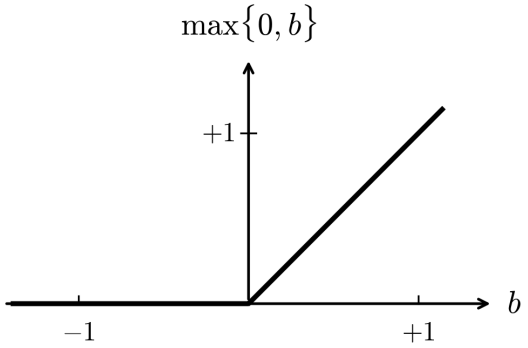
Figur 2.7: Aktiveringsfunktionen tanh(b) som likt sigmoid är en av de historiskt mest studerande aktiveringsfunktionerna och har varit avgörande för maskininlärningens utveckling [15].

Trots sina fördelar lider båda dessa funktioner av ett gemensamt problem som blir särskilt påtagligt i djupa nät. När det lokala fältet $|b|$ blir stort går $g'(b)$ mot noll vilket kallas för att funktionerna mättas. Eftersom bakåtpropageringsalgoritmen multiplicerar faktorer av $g'(b)$ genom varje lager enligt (2.21) innebär det att felprodukterna $\delta_m^{(\mu)}$ krymper exponentiellt med antalet lager. Detta fenomen kallas för *försvinnande gradienter* (eng. vanishing gradient problem) och var länge ett avgörande hinder för att framgångsrikt träna djupa neuronnät [15]. Vikterna i lager nära inlagret uppdateras knappt medan lager nära utlagret lär sig normalt vilket resulterar i att djupa nät med sigmoid- eller tanh-aktivering i praktiken beter sig som om de vore betydligt grundare. Neuronnätet blir därmed ineffektivt och kan inte dra nytta av sin storlek.

En aktiveringsfunktion som till stor del löser problemet med försvinnande gradienter är ReLU (Rectified Linear Unit) och definieras som

$$\mathrm{ReLU}(b) = \max(0, b). \tag{2.26}$$

ReLU-funktionen introducerades i samband med djupa neuronnät av Glorot et al. [16] och är styckvis linjär med en derivata som antar värdet 1 för $b > 0$ och 0 för $b < 0$ [15]. Eftersom derivatan inte mättas för positiva argument undviks det exponentiella krympandet av gradienterna som ställer till problem för implementeringar med sigmoid och tanh. Dessutom är ReLU beräkningsmässigt snabbare att evaluera än de exponentialbaserade funktionerna.

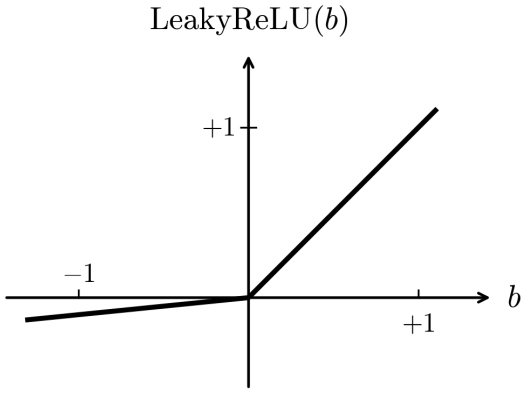


Figur 2.8: Aktiveringsfunktionen ReLU, som genom att returnera argumentet b för alla positiva värden hanterar problemet med försvinnande gradienter.

En nackdel med ReLU är dock fenomenet med döda neuroner (eng. dying neurons). Om det lokala fältet b blir negativt för alla träningsmönster μ sätts derivatan permanent till noll och neuronen slutar bidra till inlärningen. En variant som adresserar detta problem är Leaky ReLU,

$$\text{LeakyReLU}(b) = \begin{cases} b, & b \geq 0, \\ \gamma \cdot b, & b < 0, \end{cases} \tag{2.27}$$

där γ är en liten positiv konstant, typiskt $\gamma \approx 0.01$. Genom att tillåta ett litet bidrag för negativa argument som är skilt från noll säkerställs att derivatan aldrig blir exakt noll vilket eliminerar problemet med döda neuroner.



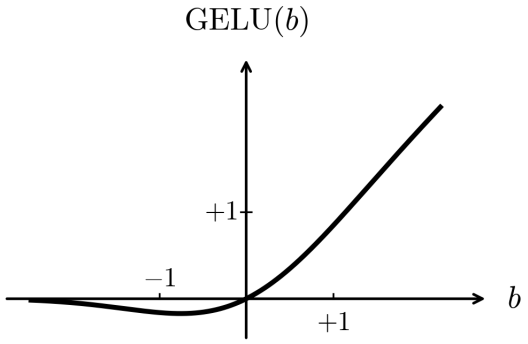
Figur 2.9: Aktiveringsfunktionen *Leaky ReLU* är en adaption av ReLU som adresserar problemet med döda neuroner genom att returnera ett nollskilt värde även för negativa argument b.

En ytterligare variant som har blivit allt mer populär inom modern maskininlärning är GELU (Gaussian Error Linear Unit) och den definieras som

$$\text{GELU}(b) = b\,\Phi(b)\,, \tag{2.28}$$

där $\Phi(b)$ betecknar den kumulativa fördelningsfunktionen för standardnormalfördelningen. Till skillnad från ReLU som har en skarp övergång vid $b = 0$ ger GELU en

mjuk och kontinuerligt deriverbar övergång mellan negativa och positiva argument. För stora positiva b antar GELU samma linjära beteende som ReLU och för stora negativa värden likaså. I området kring origo avtar funktionen mer gradvis. GELU är den aktiveringsfunktion som används i implementeringen och lösningen av det tvådimensionella värmeledningsproblemet, se avsnitt 3.2.1.



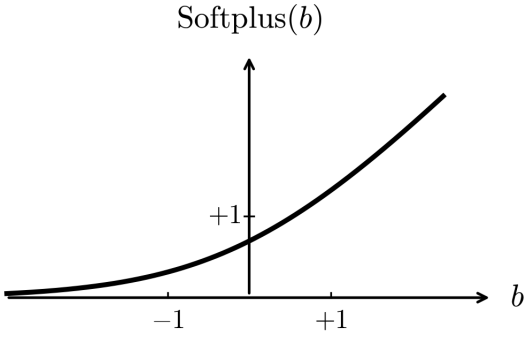
Figur 2.10: GELU är en aktiveringsfunktion som delar samma karaktäristik som ReLU för stora positiva och negativa argument men som också bidrar med en kontinuerligt deriverbar övergång mellan $b < 0$ och $b \geq 0$.

Värt att notera är också varför en rent linjär aktiveringsfunktion $g(b) = b$ inte är tillräcklig trots att den är fullkomligt kontinuerlig och deriverbar överallt. Om alla lager i ett neuronnät använder linjära aktiveringsfunktioner kan hela nätverkets mappning från insignaler till utsignaler reduceras till en enda linjär transformation oavsett hur många lager nätverket har [15]. Icke-linjäriteten i aktiveringsfunktionerna är med andra ord avgörande för neuronnätets förutsättningar att approximera godtyckligt komplexa funktioner.

De ovan introducerade aktiveringsfunktionerna behöver inte användas entydigt genom hela neuronnätet utan kan med fördel kombineras för att dra nytta av deras unika fördelar och karaktäristik. Individuella lager kan alltså ha olika aktiveringsfunktioner. För specifika lager kan exempelvis aktiveringsfunktioner som är specifikt anpassade till problemets natur användas istället, utan att nödvändigtvis vara de mest effektiva valen vad gäller beräkningstid. I det tvådimensionella neuronnätet i detta arbete används till exempel Softplus-funktionen i utlagret

$$\text{Softplus}(b) = \ln\Big(1 + e^b\Big), \tag{2.29}$$

vilken kan ses som en mjuk approximation av ReLU med egenskapen att dess utdata alltid är strikt positiv. Valet motiveras av att den predikterade värmediffusiviteten α är en fysikalisk storhet som per definition måste vara positiv.

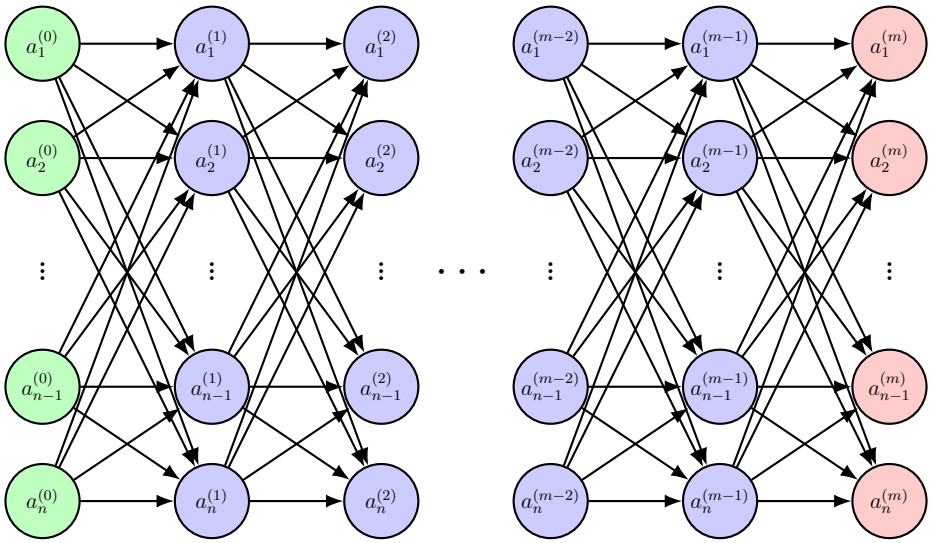


Figur 2.11: Aktiveringsfunktionen Softplus, som med fördel kan användas i specifika applikationer där utvärdena från neuronlagret inte ska kunna anta negativa värden.

Valet av aktiveringsfunktion avgör vilka förutsättningar det resulterande neuronnätet får för sin inlärning och inferens. Genom att välja aktiveringsfunktion med hänsyn till problemets omfattning, specifik karaktäristik och kända fysikaliska begränsningar och förutsättningar kan det producerade resultatet finjusteras och optimeras.

2.1.4 Nätarkitekturer

De neuronnät som hittills beskrivits i avsnitten 2.1 och 2.1.3 från perceptronen i figur 2.3 till det mer allmänna nätverket i figur 2.5 har alla tillhört kategorin fullt kopplade framåtriktade nät (eng. fully connected feed-forward networks). I ett sådant nät är varje neuron i ett givet lager kopplad till samtliga neuroner i det efterföljande lagret och informationen flödar uteslutande i en riktning från inlager till utlager. De vikter och tröskelvärden som härletts genom bakåtpropagering i avsnitt 2.1.2 beskriver precis denna typ av arkitektur. I praktiken kan nätverkets dimensioner väljas godtyckligt vilket illustreras figur 2.12.



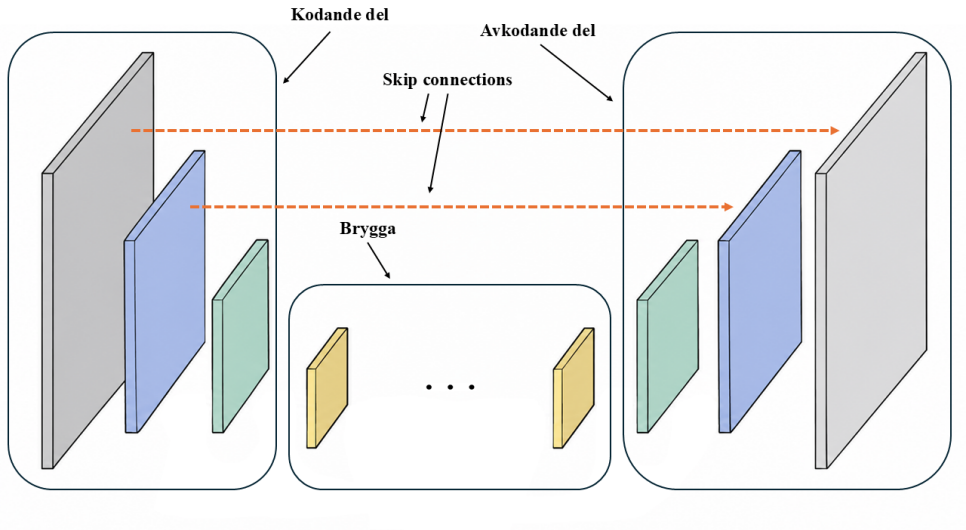
Figur 2.12: Generaliserad nätstruktur för ett godtyckligt antal lager m och neuroner n. Notera att antalet neuroner för ett specifikt lager också kan väljas godtyckligt. Olika lager kan alltså innehålla olika många neuroner.

Utöver valet av aktiveringsfunktion finns ytterligare designparametrar som avgör neuronnätets egenskaper, framför allt antalet lager (nätverkets djup) och antalet neuroner per lager (nätverkets bredd).

Generellt kan ett djupare nät lära sig mer hierarkiska representationer av data där eftersom tidiga lager kan extrahera enklare mönster för att senare lager ska kunna kombinera dessa till allt mer abstrakta strukturer [15]. Samtidigt innebär fler lager att bakåtpropageringsalgoritmen måste multiplicera fler faktorer av $g'(b)$ enligt (2.21) vilket förvärrar problemet med försvinnande gradienter som beskrevs i avsnitt 2.1.3. Ett bredare nät med fler neuroner per lager kan samtidigt fånga en rikare uppsättning mönster inom varje abstraktionsnivå men ökar också antalet fria parametrar w_{ij}^m och θ_i^m avsevärt. Ett alltför brett eller djupt nät riskerar dessutom att *överanpassa* (eng. overfit) träningsdatan vilket innebär att nätverket även lär sig brus och tillfälliga mönster i träningsdatan snarare än den underliggande strukturen och därmed presterar sämre på ny data [15]. Med ny data avses mönster som neuronnätet inte har stött på under sin träning och används för att evaluera dess prestanda. Val av djup och bredd innebär därför alltid en avvägning mellan expressiv kapacitet och generaliseringsförmåga.

En annan kategori av arkitektur som är av central betydelse för detta arbete är den så kallade *kodande* och *avkodande* strukturer. Idén är att neuronnätet delas upp i två delar där en kodande del successivt reducerar den rumsliga eller dimensionella upplösningen av indatan och en avkodande del som expanderar representationen tillbaka till samma dimension som indatan. Mellan dem finns en så kallad *brygga* (eng. bridge) som utgör nätverkets mest komprimerade representation. Den kodande delen tvingar nätverket att extrahera de viktigaste övergripande strukturerna i indatan medan den avkodande delen rekonstruerar detaljerad information från den komprimerade representationen. Denna typ av arkitektur har visat sig särskilt effektiv för problem där in- och utdata har samma rumsliga dimensionalitet men representerar olika storheter, exempelvis bildsegmentering [15]. Då används också en speciell typ av neuronnät som kallas för *faltande nätverk* (CNN) och denna struktur förklaras i mer detalj under sitt eget avsnitt 2.1.5 CNN.

En specifik och inflytelserik variant av kodande och avkodande arkitekturer är *U-Nät* (eng. U-Net), som utvecklades av Ronneberger et al. [17] för segmentering av biomedicinska bilder. Det som skiljer U-nät från en enkel kodande och avkodande struktur är introduktionen av *skip connections* mellan motsvarande nivåer i den kodande och avkodande delen. Skip connections för direkt vidare lokal information från den kodande delens tidiga lager till den avkodande delens motsvarande lager genom att sammanfoga dem kanalvis. Mer om kanaler finns också att läsa under avsnitt 2.1.5. Utan skip connections skulle detaljerad rumslig information gå förlorad i bryggan eftersom den komprimerade representationen inte kan bevara all lokal detalj. Med skip connections kan den avkodande delen istället kombinera den övergripande strukturen från den komprimerade representationen med de lokala detaljerna från den kodande delen. Skip connections bidrar dessutom till att reducera problemet med försvinnande gradienter eftersom de tillhandahåller kortare vägar genom nätverket för felsignalerna under bakåtpropagering [15]. En visualisering av U-Nät-strukturen finns att se i figur 2.13



Figur 2.13: En visualisering av den specifika neuronnätsarkitekturen ”U-Nät” med den kodande och avkodande delen markerade. Även bryggan och exempel på skip connections är inritade.

2.1.5 CNN

I de fullt kopplade nät som beskrevs i föregående avsnitt är varje neuron i ett givet lager kopplad till samtliga neuroner i det efterföljande lagret. För endimensionella problem med måttligt antal inparametrar fungerar detta väl men för tvådimensionella fält som temperaturfältet $T(x, y)$ växer antalet vikter snabbt. Om exempelvis ett tvådimensionellt fält representeras på ett 32×32-rutnät har inlagret 1024 neuroner och ett enda fullt kopplat dolt lager med lika många neuroner skulle kräva över en miljon vikter. Utöver att detta är beräkningsmässigt kostsamt saknar ett fullt kopplat lager en uppfattning om rumslig närhet, det vill säga att punkter som ligger intill varandra i fältet typiskt är starkare korrelerade med varandra än punkter långt ifrån varandra. *Faltande neuronnät* (Convolutional Neural Networks, CNN) adresserar båda dessa problem genom att ersätta de fullt kopplade lagren med faltande lager [15].

Idén bakom ett faltande lager är att en liten matris av vikter, en så kallad kärna eller filter (eng. kernel) appliceras lokalt och systematiskt över indatan. Filtret har typiskt en liten rumslig utsträckning och sveper stegvis över hela det tvådimensionella indatafältet. Exempelvis hade filtret kunnat representeras med en 3×3 matris innehållande vikterna. I varje position beräknas en viktad summa av de lokala indataelementen genom att de elementvis multipliceras med filtrets vikter för att produkterna sedan ska kunna summeras och resultatet körs genom en aktiveringsfunktion. Mer formellt kan detta skrivas som

$$V_{ij} = g\left(\sum_{p=1}^{P}\sum_{q=1}^{Q} w_{pq}, x_{p+i-1,q+j-1} - \theta\right), \tag{2.30}$$

där V_{ij} är utsignalen för neuronen vid position (i, j) i den resulterande egenskapsmappningen (eng. feature map), w_{pq} är filtrets vikter med dimension $P \times Q$, x betecknar indatamatrisen, θ är ett tröskelvärde och g är aktiveringsfunktionen [15].

Notera att indexeringen $p+i-1$, $q+j-1$ beskriver hur filtret förskjuts stegvis och att samma vikter w_{pq} och tröskelvärde θ används i varje position. Antalet steg som kärnan förskjuts per beräkning kallas för stride.

Den centrala principen är alltså att vikterna delas (eng. weight sharing) mellan alla positioner i fältet. Om indatan har dimension $N \times N$ och filtret har dimension $P \times Q$ krävs därför bara $P \cdot Q + 1$ fria parametrar (vikter och tröskelvärde) för en enda egenskapsmappning, oberoende av indatans storlek. Jämfört med ett fullt kopplat lager som hade krävt $N^2 \times N^2$ vikter innebär detta en drastisk reduktion av antalet parametrar. Parameterdelningen innebär dessutom att samma lokala mönster kan detekteras oavsett var i fältet det förekommer vilket är en egenskap som kallas translationsinvarians [15]. För att illustrera faltningsoperationen konkret kan en 4×4-indatamatris $\mathbf{X}$ och ett 2×2-filter $\mathbf{K}$ betraktas:

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & x_{13} & x_{14} \\ x_{21} & x_{22} & x_{23} & x_{24} \\ x_{31} & x_{32} & x_{33} & x_{34} \\ x_{41} & x_{42} & x_{43} & x_{44} \end{bmatrix}, \qquad \mathbf{K} = \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix}. \tag{2.31}$$

Filtret placeras först i det övre vänstra hörnet av $\mathbf{X}$ och beräknar den viktade summan $w_{11}x_{11} + w_{12}x_{12} + w_{21}x_{21} + w_{22}x_{22}$. Filtret förskjuts därefter ett steg åt höger och beräknar $w_{11}x_{12} + w_{12}x_{13} + w_{21}x_{22} + w_{22}x_{23}$ och så vidare. Efter att filtret har svept över hela raden förskjuts den ett steg nedåt och processen upprepas. Med stride [1, 1] och utan utfyllnad (eng. padding) ger ett 2×2-filter applicerad på en 4×4-matris en resulterande egenskapsmappning av dimension 3×3 vars element ges av (2.30). De fyra hörnelementen blir specifikt:

$$\begin{cases} V_{11} = g(w_{11}x_{11} + w_{12}x_{12} + w_{21}x_{21} + w_{22}x_{22} - \theta) \\ V_{13} = g(w_{11}x_{13} + w_{12}x_{14} + w_{21}x_{23} + w_{22}x_{24} - \theta) \\ V_{31} = g(w_{11}x_{31} + w_{12}x_{32} + w_{21}x_{41} + w_{22}x_{42} - \theta) \\ V_{33} = g(w_{11}x_{33} + w_{12}x_{34} + w_{21}x_{43} + w_{22}x_{44} - \theta) \end{cases} \tag{2.32}$$

där indexen p och q antar värdena 1 och 2 och aktiveringsfunktionen g appliceras elementvis. Varje element i $\mathbf{V}$ sammanfattar alltså den lokala informationen i ett litet område av indatan. I praktiken används flera olika filter parallellt i samma faltande lager där varje filter producerar en egen egenskapsmappning. De olika filtren har egna uppsättningar vikter och kan därför lära sig att detektera olika typer av lokala mönster som exempelvis gradienter i olika riktningar eller lokala extrempunkter. Varje individuell egenskapsmappning kallas för *kanal* och deras vikter tränas med bakåtpropagering på samma sätt som vikterna i fullt kopplade lager. Bidraget från varje kanal summeras sedan innan elementvis applicering av aktiveringsfunktionen [15].

Faltande lager kombineras ofta med *pooling*-lager som reducerar den rumsliga upplösningen genom att ersätta ett lokalt område med ett enda värde. Om värdena i ett lokalt område ersätts med det maximala kallas detta för max-pooling. Pooling minskar antalet parametrar i efterföljande lager och bidrar till att göra nätverket mer robust mot små rumsliga förskjutningar i indatan [15]. Det är faltande lager av denna typ som utgör de centrala byggstenarna i den U-Nät-arkitektur som beskrevs i avsnitt 2.1.4. I den kodande delen av U-nätet appliceras faltande lager följda

av pooling för att successivt extrahera allt mer övergripande strukturer ur indatan medan den avkodande delens faltande lager rekonstruerar den fulla rumsliga upplösningen. Parameterdelningen och den lokala karaktären hos faltningsoperationen gör CNN särskilt lämpliga för de tvådimensionella värmetransportproblem som studeras i detta arbete där det predikterade α-fältet har samma rumsliga struktur som det observerade temperaturfältet.

2.1.6 PINN

De neuronnät som hittills beskrivits i detta kapitel har alla tränats uteslutande på data. Både fullt kopplade framåtriktade nät och faltande arkitekturer tränas på samma sätt utan någon större variation. Nätverket justerar sina vikter och tröskelvärden med gradientnedstigning så att kostnadsfunktionen H i (2.13) minimeras så att skillnaden mellan neuronnätets utsignaler $O_i^{(\mu)}$ och de önskade målvärdena $t_i^{(\mu)}$ blir så liten som möjligt. Ett sådant rent datadrivet nät har dock ingen inbyggd kunskap om de fysikaliska lagar som styr det system som datan baseras och hämtas från. Om tillgänglig träningsdata är begränsad, brusig eller ofullständig kan nätverket därför konvergera mot lösningar som visserligen passar datan men som nödvändigtvis inte följer de underliggande fysiska lagarna.

Physics-Informed Neural Networks (PINN) adresserar detta genom att utöka kostnadsfunktionen med ytterligare termer som tvingar nätverkets lösning att respektera kända fysikaliska samband. Genom att implementera *partiella differentialekvationer* (PDE:er) i kostnadsfunktionen får neuronnätet de förutsättningar som krävs för säkerställa att resultatet uppfyller gällande fysiska lagar [18]. Den totala kostnadsfunktionen för ett PINN kan då skrivas som

$$\mathcal{L} = \lambda_{\text{data}}\mathcal{L}_{\text{data}} + \lambda_{\text{PDE}}\mathcal{L}_{\text{PDE}} + \lambda_{\text{BC}}\mathcal{L}_{\text{BC}}, \tag{2.33}$$

där $\mathcal{L}_{data}$ kvantifierar avvikelsen mellan nätverkets producerade resultat och observerad data, $\mathcal{L}_{PDE}$ mäter hur väl nätverkets lösning uppfyller den styrande differentialekvationen och $\mathcal{L}_{BC}$ kvantifierar avvikelser mot kända randvillkor. Vikterna λ_{data}, λ_{PDE} och λ_{BC} avgör hur stor påverkan de olika termerna ska ha på den totala kostnaden och valet av dessa skalningsfaktorer kan ha stor inverkan på nätverkets konvergens och resultat.

Notera att termen $\mathcal{L}_{data}$ är analog med den kostnadsfunktion som definierades i (2.13) och mäter hur väl nätverkets utsignaler överensstämmer med tillgängliga mätningar eller numeriskt genererad data. Det som gör ett PINN till mer än ett vanligt neuronnät är just tillägget av PDE-termen $\mathcal{L}_{PDE}$. Betrakta en allmän PDE på formen

$$\mathcal{N}[\, u(\mathbf{x})\,;\,\boldsymbol{\gamma}\,] = 0, \tag{2.34}$$

där $\mathcal{N}$ är en differentialoperator, u är den primärt okända, $\mathbf{x}$ betecknar de oberoende variablerna som rumskoordinater och tid och γ representerar de fysikaliska parametrarna i ekvationen. Om nätverket predikterar en approximation $\hat{u}(\mathbf{x})$ av u kan PDE-residualen definieras som

$$r(\mathbf{x}) = \mathcal{N}[\, \hat{u}(\mathbf{x})\,;\,\boldsymbol{\gamma}\,], \tag{2.35}$$

som är noll om och endast om $\hat{u}$ uppfyller differentialekvationen exakt. PDE-bidraget till kostnadsfunktionen kan sedan beräknas som exempelvis *medelkvadratfelet* av residualen (eng. Mean Squared Error (MSE)) utvärderad i en uppsättning punkter $\{\mathbf{x}_c\}$ i domänen:

$$\mathcal{L}_{\text{PDE}} = \frac{1}{N_c} \sum_{c=1}^{N_c} [\, r(\mathbf{x}_c) \,]^2 , \tag{2.36}$$

Under träningen drivs residualen mot noll i alla punkter där den beräknats vilket innebär att nätverkets lösning inte bara anpassas till tillgänglig data utan också tvingas vara konsekvent med den styrande fysiken. Beräkningen av residualen (2.35) kräver att partiella derivator av nätverkets utdata med avseende på dess indata kan beräknas. I moderna ramverk som PyTorch görs detta med hjälp av automatisk differentiering (eng. automatic differentiation, även autograd) som beräknar exakta derivator genom att utnyttja kedjeregeln på de elementära operationer som utgör nätverkets beräkningsgraf. Automatisk differentiering är exakt till maskinprecision till skillnad från numeriska differensmetoder som introducerar trunkeringsfel.

PDE-termen fungerar som en fysikalisk *regularisering* av nätverket. I ett rent datadrivet nät kan många olika viktuppsättningar ge upphov till approximativt samma utsignaler på träningspunkterna men med vitt skilda beteenden mellan dessa punkter. Genom att kräva att lösningen också ska uppfylla en differentialekvation begränsas mängden tillåtna lösningar avsevärt. Lösningar som inte uppfyller de underliggande fysikaliska lagarna kan med andra ord sållas bort för att ge rum åt lösningar som tillfredställer både datan och fysiken. Detta är särskilt värdefullt när tillgänglig data är gles, brusig eller koncentrerad till en del av domänen [18].

I ett *direkt problem* är PDE:ns parametrar γ kända och den sökta kvantiteten är fältvariabeln $u(\mathbf{x})$. Nätverket tar de oberoende variablerna $\mathbf{x}$ som indata och predikterar $\hat{u}(\mathbf{x})$ där PDE-residualen (2.35) driver lösningen mot enlighet med den kända ekvationen eller sambandet. Det direkta problemet motsvarar alltså vad traditionella numeriska metoder som FDM och FVM löser, vilka beskrivs i avsnitt 2.2.

I ett *inverst problem* är det fältvariabeln $u(\mathbf{x})$ som antas vara känd medan en eller flera av parametrarna γ i PDE:n är okända och ska bestämmas. Det är för inversa problem som PINN visar en av sina mest utmärkande styrkor jämfört med traditionella numeriska metoder. Inversa problem kan i allmänhet inte lösas direkt med metoder som FDM eller FEM utan kräver att dessa kombineras med andra optimeringstekniker [11]. I ett PINN hanteras det inversa problemet naturligt inom samma ramverk genom att de okända parametrarna γ antingen predikteras direkt av nätverket eller behandlas som fria parametrar som optimeras tillsammans med nätverkets vikter under träningen. Kostnadsfunktionens dataterm $\mathcal{L}_{data}$ ser till att den predikterade lösningen överensstämmer med de observerade fältvärdena medan PDE-termen $\mathcal{L}_{PDE}$ säkerställer att de predikterade parametrarna γ följer den fysik som genererat träningsdatan.

2.2 Numeriska metoder för lösning av PDE:er

De flesta PDE:er är mycket svåra eller omöjliga att lösa analytiskt och därför är numeriska metoder viktiga. Det finns ett flertal olika metoder, där de vanligaste är *finita differensmetoden* (FDM), *finita volymmetoden* (FVM) och *finita elementmetoden* (FEM). Både FDM och FVM har använts under arbetet men FEM noteras bara som en möjlig numerisk metod utan att utredas vidare.

2.2.1 FDM

Den simplaste numeriska metoden är finita differensmetoden (FDM). Den utgår från den PDE som ska lösas på differentialform $\mathcal{L}u = f$, där $\mathcal{L}$ är en linjär differentialoperator, f en källterm, och u är den sökta funktionen [19]. Metoden kräver också en stencil för beräkning av derivator och ett beräkningsnät bestående av de punkter där funktionsvärdet u ska approximeras. Därefter approximeras derivatorna i varje nod i beräkningsnätet med differenskvoter som angivits av stencilen, exempelvis $\frac{\partial u}{\partial x} \approx \frac{u(x+h)-u(x)}{h}$, för att erhålla en uppsättning linjära ekvationer. Med hjälp av dessa ekvationer erhålls ett stort linjärt ekvationssystem för funktionsvärdet i varje nod, vilket kan uttryckas som $Mx = b$, där M är en gles matris, x är vektorn med de sökta funktionsvärdena och b är en term som beror på randvillkoren och eventuella källtermer i PDE:n.

2.2.2 FVM

Finita volymmetoden (FVM) utgår från integralformen på PDE:n, istället för differentialformen som i FDM. Tillvägagångssättet för att erhålla diskretiseringen sker likt det för FDM. Ett beräkningsnät skapas och integralformen appliceras på varje cell i beräkningsnätet [20]. Därefter approximeras integralen med de noder som valts i beräkningsnätet och en linjär ekvation av funktionsvärdena i noderna erhålls för varje cell. Därifrån erhålls ett linjärt ekvationssystem som kan lösas som tidigare.

Fördelar med FVM gentemot FDM är till exempel att skarpa diskontinuiteter, såsom chockvågor eller fronter, kan hanteras bättre och att differentialformen inte beskriver dem entydigt [21]. Det beror på att FVM utgår från integralformen av PDE:n, där flöden mellan celler beskrivs, i stället för att enbart approximera derivator i enskilda punkter. Dessutom är FVM konservativt vilket innebär att lokala storheter som energi, massa och värme bevaras. Detta beror på att flödet över cellväggar är bevarat.

2.3 Värmeledning

Värmeledning är en av de tre grundläggande formerna av värmetransport, tillsammans med konvektion och strålning. Processen beskriver hur termisk energi sprids inom ett material till följd av temperaturgradienter. Värmeledning kan beskrivas med den paraboliska partiella differentialekvationen

$$\frac{\partial T}{\partial t} = \nabla \cdot (\alpha \nabla T) + Q, \tag{2.37}$$

vanligtvis benämnd värmeledningsekvationen, där α står för *värmediffusiviteten*, Q en allmän källterm som representerar intern värmegenerering per volymsenhet, t tid och T temperatur. Värmeledning kan också studeras utan tidsberoende. Genom sätta $\frac{\partial T}{\partial t} = 0$ erhålls istället den stationära värmeledningsekvationen som blir:

$$\nabla \cdot (\alpha \nabla T) + Q = 0. \tag{2.38}$$

Den stationära värmeledningsekvationen beskriver ett tillstånd där temperaturfältet stabiliserat sig och inte längre förändras med tiden. Fördelen med att betrakta det stationära sambandet är att man då endast behöver studera enskilda mätningar av temperaturfältet. För direkta problem, där materialparametrar som exempelvis α är kända och temperaturfältet T ska bestämmas, är den stationära värmeledningsekvationen i allmänhet välställd i Hadamards mening [22]. Välställd i detta sammanhang innebär att en lösning existerar och att den är entydig samt beror kontinuerligt på data med givet randvillkor. En svårighet uppkommer dock bland inversa problem där man bestämmer värmediffusiviteten från temperaturfältet då lösningen kan bli icke-entydig. Ett exempel på icke-entydighet är om man har en homogen platta med en kant som värms till en viss temperatur medan motstående kant hålls vid en annan temperatur. Eftersom plattan är homogen måste värmediffusionskoefficienten vara konstant överallt, vilket leder till att det stationära temperaturfältet blir helt linjärt. Problemet är att alla konstanta α skulle ge upphov till exakt detta stationära temperaturfält, oavsett värde.

För att se till att inversa lösningar till stationära problem blir unika, alltså att problemet är inverterbart, krävs i allmänhet ytterligare information. Ofta gäller det kännedom om randvillkor, förekomst av källor eller att ha flera oberoende temperaturmätningar. Vad som i vissa fall uppstår om man inte kan uppfylla dessa krav är att problemet blir icke-välställt och lösningar blir osäkra och känsliga för störningar i data.

2.3.1 Analytisk lösning i 1D

I 1D kan värmeledningsekvationen skrivas som

$$\frac{\partial}{\partial x}\left(\alpha(x)\frac{\partial T(x)}{\partial x}\right) + Q(x) = 0, \tag{2.39}$$

vilket kan utvecklas och skrivas om till

$$\frac{\partial \alpha}{\partial x}\frac{\partial T}{\partial x} + \alpha\frac{\partial^2 T}{\partial x^2} + Q = 0. \tag{2.40}$$

Låt $f = \frac{\partial T}{\partial x}$, då erhålls

$$\frac{\partial \alpha}{\partial x} f + \alpha\frac{\partial f}{\partial x} + Q = 0, \tag{2.41}$$

vilket är en symmetrisk differentialekvation med avseende på utbyte mellan f och α. Med hjälp av kedjeregeln kan det skrivas om till

$$(\alpha f)' = -Q. \tag{2.42}$$

Genom att integrera båda led erhålls det att

$$\alpha f = -\int_0^x Q(\xi)d\xi + \alpha(0)f(0). \tag{2.43}$$

Därmed blir lösningen för α respektive f som funktioner av varandra

$$\begin{aligned} \alpha(x) &= -\frac{1}{\frac{\partial T}{\partial x}(x)}\int_0^x Q(\xi)d\xi + \frac{\alpha(0)\frac{\partial T}{\partial x}(0)}{\frac{\partial T}{\partial x}(x)}, \\ f(x) = \frac{\partial T}{\partial x}(x) &= -\frac{1}{\alpha(x)}\int_0^x Q(\xi)d\xi + \frac{\alpha(0)\frac{\partial T}{\partial x}(0)}{\alpha(x)}. \end{aligned} \tag{2.44}$$

Genom att integrera $\frac{\partial T}{\partial x}$ erhålls det att

$$T(x) = -\int_0^x \frac{1}{\alpha(\xi)}\int_0^\xi Q(\zeta)d\zeta d\xi + \int_0^x \frac{\alpha(0)\frac{\partial T}{\partial x}(0)}{\alpha(\xi)}d\xi + T(0). \tag{2.45}$$

Även om ekvation 2.44 och 2.45 är analytiska lösningar till 2.39 är det inte nödvändigtvis så att de kan uttryckas i elementära funktioner då det är möjligt att välja T, α och Q så att integralerna inte kan uttryckas med elementära funktioner.

2.4 Medicinsk tillämpning av PINN

Termisk behandling av tumörer är en växande behandlingsmetod inom onkologi som bygger på att hetta upp vävnad för att försvaga eller döda den [23]. För att veta hur värme sprider sig under uppvärmning av ett område behövs termisk modellering som utnyttjar ekvationer för värmeledning. Inom modellering av biologisk vävnad användes ofta en modifierad värmeledningsekvation utformad för att ta hänsyn till vävnadens struktur och funktion.

2.4.1 Värmetransport i biologisk vävnad

För att modellera värmetransport i levande biologisk vävnad kan Pennes bioheat-ekvation användas [24]. Den har formen

$$\nabla \cdot (k\nabla T) + \beta(T_a - T) + Q_{\text{met}} = 0, \tag{2.46}$$

där T är vävnadens temperatur, k är värmeledningskoefficienten, T_a är temperaturen på blodet i artärerna och Q_{met} är en källterm som beskriver värmeproduktionen från metabolism. β beskriver så kallad *perfusion* och definieras som

$$\beta = \rho_b c_b \omega_b, \tag{2.47}$$

där ρ_b och c_b är blodets densitet respektive specifik värmekapacitet och ω_b är perfusionshastigheten. Perfusionshastighet är ett mått på blodflödet som når ett organ eller vävnadsvolym och är definierat som volymflöde av blod per vävnadsvolym och tidsenhet [14].

2.4.2 Hypertermibehandling

Hypertermibehandling är en medicinsk metod där cancerceller försvagar genom att värma upp dem, vanligtvis till 42-45 °C [23]. Hypertermibehandling kan delas upp i tre kategorier baserat på hur stort område som värms upp: helkropp, regional och lokal. Helkroppshypertermi innebär att hela kroppen värms upp i syfte att försvaga tumörceller i samband med annan typ av cancerbehandling. Kroppen värms oftast upp uniformt och därav har man ingen nytta av termisk modellering inom denna hypertermibehandling. Vid regional och lokal hypertermibehandling används däremot riktad uppvärmning av vävnad. I dessa fall har man ett bestämt område som ska värmas upp och då är det en stor fördel att veta hur värme sprider sig och vilket temperaturfördelning man får vid behandlingen för att säkerställa att all tumörvävnad behandlas. Dessutom kan det vara viktigt att se till att temperaturerna i frisk vävnad inte blir för höga och ger onödiga biverkningar.

Genom att kombinera en modell för värmetransport i biologisk vävnad, exempelvis Pennes bioheat-ekvation, med PINN kan man i teorin lösa ett inverst problem och rekonstruera termiska parametrar som värmeledning och perfusion från temperaturmätningar. När dessa parametrar är kända reduceras problemet till ett välställt framåtriktat problem, där en given uppvärmning ger en entydig och stabil temperaturfördelning i vävnaden. Att veta exakt hur värmen sprider sig och vilka temperaturer vävnaden når upp till är extra viktigt vid tumörbehandling med termoablation, vilket innebär att man istället värmer upp mot 100 °C [23]. Istället för att försvaga tumörcellerna som i hypertermibehandlingarna försöker man alltså döda dem med termoablation. I och med de höga temperaturerna skulle PINN kunna vara användbart för att se till att man inte också dödar frisk vävnad.

3
Metod

Metodiken som använts för att lösa det inversa värmetransportproblemet med hjälp av neuronnät kan brytas upp i ett antal delmoment. Dessa delmoment har till stor del varit samma för projektets olika delproblem. För delproblemen i två dimensioner har AI använts som hjälpmedel för produktion och felsökning av kod. I följande kapitel kommer delproblemens metodiker för datagenerering, nätverkarkitekturens utformning och validering att presenteras.

Datagenerering har inkluderat generering av temperaturfältet genom att lösa det *direkta problemet* av värmeledningsproblemet och framöver benämns denna färdiga data som den exakta lösningen. Datagenerering har generellt utgått från någon form av diskretiseringsmetod.

Nätverkarkitekturens utformning syftar till att bestämma nätverkets initiala parametrar för att maximera dess förutsättningar att lära sig datans ingående mönster och karaktäristik, exempel på dessa parametrar är lagerstrukturer, antal neuroner och aktiveringsfunktioner. För träning av nätverk inkluderas även att göra datan kompatibel mellan olika Python-bibliotek såsom NumPy och PyTorch.

Validering har utförts genom att grafiskt följa kostnadsfunktionens ingående delar eftersom insynen i nätverkets faktiska metod för att beräkna sitt resultat är begränsad. För att förhindra en övertro på systemets pricksäkerhet, *överanpassning* (eng. overfitting), har nätverket även validerats på data som inte varit träningsexempel.

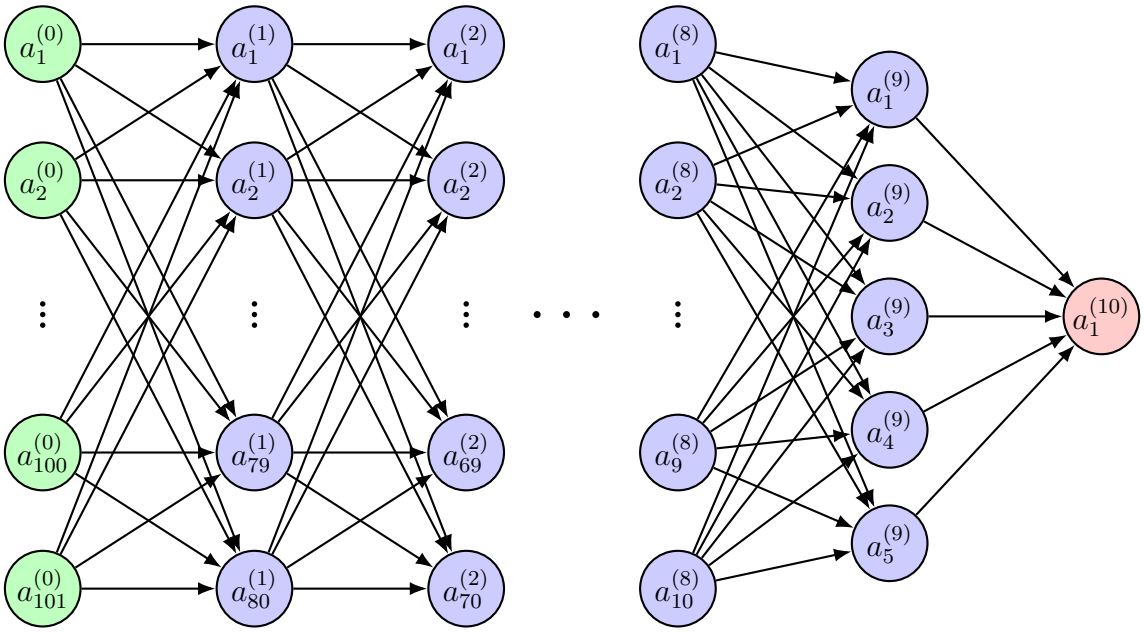
3.1 Metod för värmeledning i 1D

För att kunna lösa det inversa värmeledningsproblemet i en dimension (1D) måste ekvationen först lösas för det direkta problemet för att erhålla temperaturfältet.

Ett Python-ramverk skapades för att numeriskt lösa för temperaturfältet givet ett analytiskt α med hjälp av *finita volymmetoden*, FVM. Värmeledningsekvationen löstes på detta vis för α som tilläts variera linjärt, kvadratiskt, gaussiskt och sinusformigt med varierande koefficienter, totalt över 300 olika fall. Inga linjärkombinationer eller andra funktionskompositioner mellan de ovan nämnda typer av α ingick i träningsdatan. I värmeledningsekvationen läts källtermen $Q \equiv 0$ medan T hade *Dirichlet-randvillkoren* $T(0) = 0$ och $T(1) = 1$. Även α normerades så att $\alpha(0) = 0{,}01$. Den numeriska data som sparades från *FVM*-lösaren var T, T', T'', α samt vilket x-värde de var beräknade i, vilka alla är nödvändig information för att kunna beräkna PDE-bidraget i kostnadsfunktionen i PINN:et. De x-punkter som träningsexemplen samplades i var $x_i = \frac{i}{74}$ för $i = 1, 2, \ldots, 73$.

Det slutgiltiga PINN:et hade en arkitektur bestående av inlagret, nio gömda lager och utlagret. Den sammanlagda arkitekturen för lagerens antal noder ser ut enligt följande: $[101, 80, 70, 60, 50, 40, 30, 20, 10, 5, 1]$. En visualisering av arkitekturen visas i figur 3.1. Utparametern från nätverket är det predikterade $\alpha_{PINN}(x)$, där x är den första av de 101 inparametrarna till nätverket. De resterande 100 inparametrarna var temperaturfältet samplat i 100 punkter. De exakta x-värdena för punkterna är $x_i = \frac{i}{101}$, för $i = 1, 2, \ldots, 100$, vilket innebär att randnoderna har exkluderats. *Aktiveringsfunktionen* som nätverket använder är en kombination av tanh och LeakyReLU och visas i ekvation 3.1.

$$\sigma(x) = \left(1 + \frac{1}{10}\left|\text{LeakyReLU}(x)\right|\right) \cdot \tanh(x) \tag{3.1}$$



Figur 3.1: Visualisering av lagerna i arkitekturen för det PINN som användes för att lösa det inversa problemet i 1D.

Den *kostnadsfunktion* som användes bestod av fyra bidrag, alla med en viktning av $\lambda = 1$. Det första bidraget var en term som bestod av summan av det absoluta och relativa felet för varje träningsexempel.

$$\mathcal{L}_{data} = \sqrt{\sum_i (\alpha_{i,PINN} - \alpha_{i,exakt})^2} + \sqrt{\sum_i \left(\frac{\alpha_{i,PINN} - \alpha_{i,exakt}}{\alpha_{i,exakt}}\right)^2}, \tag{3.2}$$

där $\alpha_{i,PINN}$ och $\alpha_{i,exakt}$ är modellprediktionen respektive det exakta värdet för α för träningsexempel i.

Den andra termen i kostnadsfunktionen var PDE-förlusten. Den beräknades enligt

$$\mathcal{L}_{PDE} = \sqrt{\sum_i \frac{9}{10}\left(\alpha_{i,PINN}T_i'' + \alpha_{i,PINN}'T_i'\right)^2 + \frac{1}{10}\left(\alpha_{i,exakt}T_i'' + \alpha_{i,PINN}'T_i'\right)^2}. \tag{3.3}$$

där T_i' och T_i'' är temperaturens derivator i punkten x_i för träningsexempel i. Det till synes märkliga valet av $\mathcal{L}_{PDE}$ motiveras med att tvinga PINN:et att finna en lösning till PDE:n som är icke-trivial.

Det tredje bidraget till kostnadsfunktionen bestod av att summera $|\alpha_{i,PINN}|$ för de i där $\alpha_{i,PINN}$ var negativt, och därmed icke-fysikaliskt. Det implementerades med `torch.sum(torch.where(prediction>=0,0,-prediction))`. Detta har nästan enbart en påverkan i början av träningen i och med att alla α är positiva.

Det fjärde och sista bidraget är en regulariseringsterm för att minska problem med eventuell överanpassning. Den beräknades enligt

$$\mathcal{L}_{reg} = 10^{-3} \cdot \frac{\|\omega\|}{N}, \tag{3.4}$$

där ω är modellvikterna, N är antalet vikter och $\|\cdot\|$ är L^2-normen.

På grund av storleken på träningsdatan har *batchning* använts vid träning av modellen. Det innebär att träningsdatan delas upp i delmängder där kostnadsfunktionen beräknas och modellvikterna uppdateras en gång per delmängd, eller batch. PyTorch tillhandahåller funktioner för detta och träningsexemplens ordning randomiserades för att få en bra fördelning mellan olika α i varje batch.

Ingen dedikerad datamängd avsattes för validering. Istället testades manuellt ett antal olika α som inte ingått i träningen, bland annat linjärkombinationer och funktionskompositioner av de olika typerna av α som ingått i träningen.

3.2 Metod för värmeledning i 2D

I det tvådimensionella delproblemet studerades den stationära värmeledningsekvationen på en kvadratisk domän. Domänen valdes som enhetskvadraten

$$\Omega = [0, 1] \times [0, 1],$$

där varje punkt i domänen beskrivs av koordinaterna (x, y). Målet var att undersöka om ett neuronnät kunde rekonstruera ett rumsligt varierande värmediffusivitetsfält $\alpha(x, y)$ utifrån det temperaturfält $T(x, y)$ som fältet ger upphov till. Problemet formulerades därmed som ett *inverst problem*, där temperaturfältet betraktades som känt och α-fältet som okänt.

Temperaturfältet antogs uppfylla den stationära värmeledningsekvationen, se ekvation (2.38). I samtliga genererade exempel användes samma källterm,

$$Q(x, y) = 0.20 + 0.65 \sin(\pi x) \sin(2\pi y) - 0.25 \cos(2\pi x) \sin(\pi y). \tag{3.5}$$

Dirichlet-randvillkor användes på vänster och höger rand, där temperaturen sattes till $T = 1$ vid $x = 0$ och $T = 0$ vid $x = 1$. På den nedre och övre randen användes *Neumann-randvillkor* i form av givna värmeflöden. Det nedre randflödet sattes till

$$q_{\text{botten}}(x) = 0.25 + 0.10 \sin(2\pi x), \tag{3.6}$$

och det övre randflödet sattes till

$$q_{\text{toppen}}(x) = -0.15 + 0.08 \cos(\pi x). \tag{3.7}$$

För varje genererat α-fält löstes först det direkta värmeledningsproblemet numeriskt för att erhålla motsvarande temperaturfält. För att göra detta konstruerades en specialanpassad rutnätsbaserad lösare för stationär värmeledning på ett strukturerat kartesisk nät. Först skapas ett regelbundet 2D-rutnät över enhetskvadraten med `meshgrid` i NumPy. Därefter byggs ett glest linjärt ekvationssystem upp, punkt för punkt, där varje inre nod kopplas till sina fyra grannar i öst-, väst-, nord-, sydled. Diffusionskoefficienten mellan två angränsande punkter approximeras med harmoniskt medelvärde, vilket ger en mer robust behandling när $\alpha(x, y)$ varierar spatialt. Randvillkoren implementeras direkt i matrisen och det resulterande glesa systemet löses sedan med SciPys glesa lösare `spsolve`. Metoden är en variant av FDM för diffusion på strukturerade nät, men är här skräddarsydd för problemets enkla geometri och för effektiv generering av data till PINN-modellen.

3.2.1 Arkitektur för neuronnätet i 2D

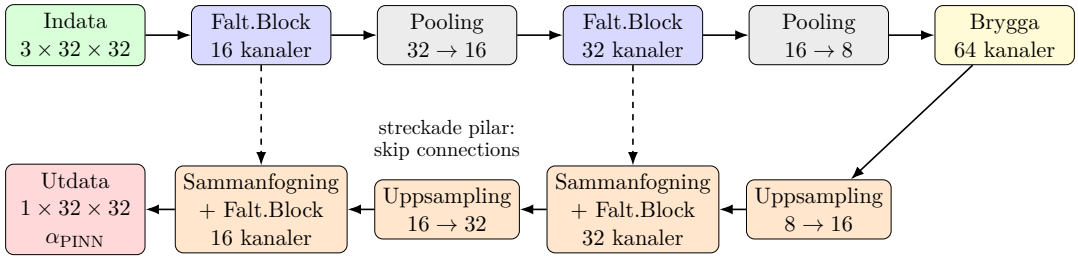
För det tvådimensionella inversa värmeledningsproblemet användes ett PINN av typen *U-Nät*. Eftersom nätverket ska rekonstruera värmediffusivitetsfälet $\alpha(x, y)$ från det stationära temperaturfältet $T(x, y)$ passar U-Nät särskilt bra som nätarkitektur. Problemet är i sin karaktäristik mycket likt ett bildsegmenteringsproblem på sättet att indatan är ett tvådimensionellt fält T och utdatan är ett annat tvådimensionellt fält (α) av samma storlek. U-Nätet i kombination med ett *CNN* sammanställer lokal detaljinformation med en övergripande förståelse av fältets struktur och blir därför lämplig för denna typ av invers rekonstruktion. Det resulterande nätverket kan då identifiera lokala rumsliga strukturer i temperaturfältet och samtidigt bevara information om var i domänen dessa strukturer förekommer. Modellen implementerades i PyTorch och definierades som en `SmallUNet`-modell.

Indatan till nätverket som användes bestod av tre *kanaler*: det normaliserade temperaturfältet T_{norm}, se ekvation (3.9), samt två koordinatkanaler som innehöll x- respektive y-koordinaten för varje punkt i rutnätet. Detta gav varje träningsexempel formen $3 \times N \times N$, där N är antalet element på vardera sidan av domänen. Arkitekturen av antalet kanaler i de olika lagren kan beskrivas enligt:

$$3 \to 16 \to 16 \to 32 \to 32 \to 64 \to 64 \to 32 \to 32 \to 16 \to 16 \to 1, \tag{3.8}$$

med två *kodande nivåer*, en *brygga* och två *avkodande*. Den första kodande nivån använde 16 kanaler, den andra 32 kanaler och bryggan 64 kanaler. I den avkodande delen minskades antalet kanaler åter till 32 och därefter 16 innan ett avslutande 1×1-faltningslager med aktiveringsfunktionen Softplus gav ett enkanaligt utdatafält motsvarande α_{PINN}. Varje faltande block bestod av två faltningslager med GELU-aktivering efter respektive lager. I den kodande delen reducerades den rumsliga upplösningen successivt genom *pooling*, samtidigt som antalet kanaler ökade. I den avkodande delen ökades upplösningen igen, så kallad *uppsampling*, samtidigt som antalet kanaler minskade, så att modellen kunde producera ett α-fält med samma rumsliga dimensioner som indatan. Mellan motsvarande kodande och avkodande nivåer användes dessutom *skip connections*, det vill säga direkta kopplingar som för vidare lokal information från den kodande delen till den avkodande delen. Denna information sammanfogades kanalvis med den uppsamplade representationen innan

nästa faltande block. Skip connections är viktiga eftersom de gör att detaljerad rumslig information kan bevaras, trots att upplösningen reduceras i den kodande delen. Figur 3.2 visar en schematisk översikt av nätverkets arkitektur. Figuren är förenklad och visar främst hur upplösning och antal kanaler förändras genom modellen.



Figur 3.2: Schematisk visualisering av den U-nät-arkitekturen som användes i 2D-fallet. Indatan bestod av tre kanaler: det normaliserade temperaturfältet T_{norm} samt koordinatkanalerna x och y. I den kodande delen reducerades den rumsliga upplösningen genom pooling, samtidigt som antalet kanaler ökade från 16 till 32 och därefter 64 i bryggan. I den avkodande delen återställdes upplösningen genom uppsampling. De streckade pilarna visar skip connections, vilka för över lokal information från den kodande delen till den avkodande delen. Vid sammanfogning kombineras informationen från skip connection med den uppsamplade representationen innan nästa faltande block. Det avslutande lagret ger ett enkanaligt fält motsvarande α_{PINN}.

Temperaturfältet normaliserades enligt

$$T_{\text{norm}} = \frac{T - \mu_T}{\sigma_T}, \tag{3.9}$$

där μ_T och σ_T är medelvärden respektive standardavvikelser beräknade över hela datasetet. Koordinatkanalerna lades till för att modellen inte enbart skulle känna till temperaturens värde, utan också var i domänen värdet uppträder.

3.2.2 Datagenerering

α-fälten genererades för att skapa varierade men kontrollerade träningsdata. Varje fält byggdes upp som en kombination av flera typer av rumsliga strukturer, bland annat Gaussiska variationer, sinus- och cosinusmönster, rektangulära områden, cirkulära områden samt långsamma gradienter i x- och y-led. Efter att dessa bidrag summerats skalades fältet om till intervallet $0{,}15 \leq \alpha \leq 2{,}50$. På så sätt erhölls α-fält med både mjuka variationer och mer lokala strukturer, vilket gav modellen varierade exempel att träna på. Det genererades 2000 exempel varav 85 % var träningsdata och resterande var valideringsdata.

3.2.3 Förlustfunktion och träningsstrategi

Det neuronnät som användes i 2D-fallet hade en *förlustfunktion* som bestod av flera bidrag med olika roller. Den totala förlustfunktionen, se ekvation (2.33), bestod av tre delar: Datatermen $\mathcal{L}_{\text{data}}$ mätte skillnaden mellan det predikterade och det exakta α-fältet. PDE-termen $\mathcal{L}_{\text{PDE}}$ byggde på residualen till den stationära värmeledningsekvationen och beräknades diskret från det predikterade α-fältet tillsammans med

det givna temperaturfältet. Randvillkorstermen $\mathcal{L}_{\text{BC}}$ säkerställde att lösningen var förenlig med de använda Dirichlet- och Neumannvillkoren. Träningen genomfördes batchvis med gradientbaserad optimering, där batchvis träning innebär att modellen uppdaterar sina vikter efter att ha bearbetat en mindre delmängd av träningsdata i taget, vilket gör träningen mer minneseffektiv och ofta stabilare än att använda hela datasetet på en gång. Batchstorleken som användes var 6 träningsexempel per batch. Efter varje *epok* beräknades motsvarande förlust på valideringsmängden. Om modellen, det vill säga värdena på samtliga parametrar, gav bättre resultat än den föregående bästa modellen sparades samtliga parametrar som en ny modell. För att undvika onödigt långa körningar infördes en stoppfunktion, vilket innebar att träningen avbröts när valideringsmåttet *Mean Squared Error* (MSE) för dataförlusten inte förbättrades mer än 10^{-5} på 60 epoker. Inlärningshastigheten reducerades också automatiskt när när förbättringen stannade av, vilket gav en mer stabil konvergens.

3.3 Metod för värmetransport i biologisk vävnad

Det neuronnät som användes för att bestämma perfusionstermen β i Pennes bioheatekvation 2.46 grundade sig i det nätverk som användes i tidigare steg för att bestämma värmeledningskoefficienten i två dimensioner. Nätverket ändrades så att PDE:n som skattas är ekvation 2.46 samt att β användes för att beräkna kostnadsfunktionerna. I det sista lagret användes en sigmoidfunktion som aktiveringsfunktion och 20% av datan användes för validering. I övrigt var nätverkets arkitektur samma.

Den data som genererades ska efterlikna en tumör i frisk vävnad där tumören har andra värmeledings- och perfusionsegenskaper än omgivningen. I domänen med storlek 1×1 generades en cirkel eller oval på en slumpmässig plats som kunde ha diameter mellan 0,15 och 0,18. Domänen var uppdelad i 64 × 64 element, istället för de 32 × 32 som användes i föregående delproblem. k i bakgrunden sattes till 0,2 och i tumören var det 0,3. β i bakgrunden sattes till 0,1. I tumören, det värde som nätverket ska bestämma, slumpades β mellan 0,6 och 0,8. Källtermen Q_{met} sattes till 0,1 över hela domänen. Nätverket tränades på totalt 500 tränings- och valideringsexempel. Samma randvillkor som det generella värmeledningsproblemet använde implementerades även här förutom på den vänstra randen. Där byttes Dirichlet-randvillkoret till *Robinvillkor*. Robin-randvillkor valdes för att efterlikna exempelvis värmeutbytet mellan hud och omgivningen och hade formen

$$-k\frac{\partial T}{\partial x} = 1(1 - T). \tag{3.10}$$

Dessutom tränades ett nätverk för att hitta både k och β om båda dessa varierade samtidigt. β slumpades som tidigare mellan 0,6 och 0,8. Därutöver introducerades ett slumpmässigt k som varierade mellan 0,3 och 0,5. I denna data användes inte Robin-randvillkor till höger, utan Dirichlet. Detta för att göra problemet något enklare då både k och β varierade. I övrigt gick datagenerering och träning till på samma sätt som då endast β predikterades.

3.4 FDM-lösare för värmeledningsekvationen

För att kunna jämföra PINN mot andra metoder skrevs en finit differenslösare som alternativ metod. *FDM* valdes på grund av sin enkelhet. Den generella PDE:n som ska diskretiseras är värmeledningsekvationen:

$$\nabla \cdot (\alpha(x)\nabla T(x)) = f(x). \tag{3.11}$$

Betrakta inledningsvis värmeledning i 1D. Då kan PDE:n skrivas som

$$\frac{\partial}{\partial x}\left(\alpha(x)\frac{\partial T}{\partial x}(x)\right) = f(x). \tag{3.12}$$

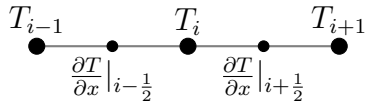
Derivatan av T med avseende på x kan approximeras med en differenskvot bakåt

$$\frac{\partial T}{\partial x} \approx \frac{T(x) - T(x-h)}{h}, \tag{3.13}$$

där h är ett litet avstånd. Om stencilen i figur 3.3 används erhålls då att

$$\left.\frac{\partial T}{\partial x}\right|_{i-\frac{1}{2}} \approx \frac{T_i - T_{i-1}}{h} \tag{3.14}$$

där indexet $i + \frac{1}{2}$ betecknar att derivatan har approximerats mellan nod i och nod $i+1$. En visualisering av var derivatan har approximerats i förhållande till T visas i figur 3.3. En stencil kan också ge ledtrådar kring hur derivatorna har approximerats. Exempelvis är stencilen i figur 3.3 symmetrisk, vilket tyder på att andraderivatan har approximerats med en central differenskvot, vilket också är fallet här.



Figur 3.3: Stencilen som beskriver nodernas placering och var värdena för T samt $\frac{\partial T}{\partial x}$ är beräknade i förhållande till varandra. De större punkterna motsvarar faktiska noder i beräkningsnätet, medan de mindre punkterna endast visualiserar var derivatan beräknas och existerar inte i beräkningsnätet.

För att inte introducera fler fel än nödvändigt bör α utvärderas vid samma position som $\frac{\partial T}{\partial x}$. För att göra det kan ett godtyckligt medelvärde användas för att finna $\alpha|_{i+\frac{1}{2}}$, men vanligt är att använda det aritmetiska eller harmoniska medelvärdet:

$$\alpha_{i+\frac{1}{2}} = \frac{\alpha_i + \alpha_{i+1}}{2}$$

eller

$$\alpha_{i+\frac{1}{2}} = 2\frac{\alpha_i \alpha_{i+1}}{\alpha_i + \alpha_{i+1}}.$$

Sammantaget kan hela PDE:n utvecklad kring nod i approximeras med

$$\begin{aligned}\frac{\partial}{\partial x}\left(\alpha(x)\frac{\partial T}{\partial x}(x)\right)\Bigg|_i &\approx \frac{\alpha\frac{\partial T}{\partial x}\Big|_{i+\frac{1}{2}} - \alpha\frac{\partial T}{\partial x}\Big|_{i-\frac{1}{2}}}{h} \\ &\approx \frac{\alpha_{i+\frac{1}{2}}(T_{i+1}-T_i) - \alpha_{i-\frac{1}{2}}(T_i - T_{i-1})}{h^2} \\ &\approx \frac{\alpha_{i+\frac{1}{2}}T_{i+1} - (\alpha_{i+\frac{1}{2}} + \alpha_{i-\frac{1}{2}})T_i + \alpha_{i-\frac{1}{2}}T_{i-1}}{h^2}\end{aligned}$$

Om approximationerna görs om till likheter erhålls ett linjärt ekvationssystem där ekvation i ser ut som:

$$\frac{\alpha_{i+\frac{1}{2}}T_{i+1} - (\alpha_{i+\frac{1}{2}} + \alpha_{i-\frac{1}{2}})T_i + \alpha_{i-\frac{1}{2}}T_{i-1}}{h^2} = f_i. \tag{3.15}$$

Ekvation 3.15 gäller dock bara för de interna noderna i beräkningsnätet. Om Dirichlet-randvillkor används för T erhålls det att det för randnoderna gäller att

$$\begin{aligned}T_1 &= D_1, \\ T_N &= D_N,\end{aligned}$$

om det är N noder i beräkningsnätet, och för Neumann-randvillkor erhålls det istället att randnoderna ska uppfylla

$$\begin{aligned}\frac{T_2 - T_1}{h} &= N_1, \\ \frac{T_N - T_{N-1}}{h} &= N_N.\end{aligned}$$

För att lösa för T givet α erhålls ett linjärt ekvationssystem $M\overline{T} = b$, där

$$\begin{aligned}M_{ii} &= -\frac{\alpha_{i+\frac{1}{2}} + \alpha_{i-\frac{1}{2}}}{h^2}, \\ M_{i,i\pm1} &= \frac{\alpha_{i\pm\frac{1}{2}}}{h^2}, \\ b_i &= f_i,\end{aligned}$$

för $i = 2, 3, \ldots, N-1$ och där rad 1 och N i M och b beror på vilka randvillkor som används. Samma diskretisering av PDE:n kan också användas för att lösa för α. Om det aritmetiska medelvärdet används för att interpolera α till mellan noderna erhålls det att

$$\frac{(T_{i+1} - T_i)\alpha_{i+1} + (T_{i+1} - 2T_i + T_{i-1})\alpha_i - (T_i - T_{i-1})\alpha_{i-1}}{2h^2} = f_i. \tag{3.16}$$

Då erhålls ett ekvationssystem $M\overline{\alpha} = b$, där

$$\begin{aligned}M_{ii} &= -\frac{T_{i+1} - 2T_i + T_{i-1}}{2h^2}, \\ M_{i,i+1} &= \frac{T_{i+1} - T_i}{2h^2}, \\ M_{i,i-1} &= -\frac{T_i - T_{i-1}}{2h^2}, \\ b_i &= f_i\end{aligned}$$

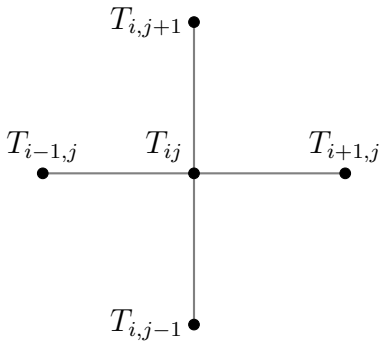
för $i = 2, 3, \ldots, N-1$ och matriselementen tillhörande randnoderna beror på vilka randvillkor som används för α. Något som är värt att notera är att denna matris inte längre är approximativt symmetrisk, som när PDE:n löstes för T. Därför är matrisekvationen ett mer instabilt system där små variationer i matriselementen kan resultera i stora variationer i lösningen till systemet. Därför blir det mycket svårare att hitta en bra numerisk lösning för α än när PDE:n löses för T.

Något som är värt att notera är att båda dessa matriser är tridiagonala, vilket innebär att det är lätt att representera dem effektivt i minnet i en dator och att det finns en simpel algoritm, Thomas algorithm eller tridiagonal matrix algorithm, för att lösa denna typ av matrisekvationer.

Med motsvarande tillvägagångssätt som för 1D-fallet erhålls det att diskretiseringen för värmeledning i 2D blir

$$f_{ij} = \frac{\alpha_{i+\frac{1}{2},j}(T_{i+1,j} - T_{ij}) - \alpha_{i-\frac{1}{2},j}(T_{ij} - T_{i-1,j})}{h_x^2} + \frac{\alpha_{i,j+\frac{1}{2}}(T_{i,j+1} - T_{ij}) - \alpha_{i,j-\frac{1}{2}}(T_{ij} - T_{i,j-1})}{h_y^2} \tag{3.17}$$

där stencilen nedan i figur 3.4 användes. I detta falla erhålls ett linjärt ekvationssystem med 5 diagonaler i stället för 3, som i 1D, där 2 av dem inte ligger angränsande till huvuddiagonalen. Då denna matris inte är tridiagonal kan inte ”Thomas algorithm” användas, men det finns ett flertal iterativa lösare som kan användas på godtyckliga matrisekvationer och därmed även i detta fall.



Figur 3.4: Stencilen som användes för diskretiseringen av värmeledningsekvationen i 2D.

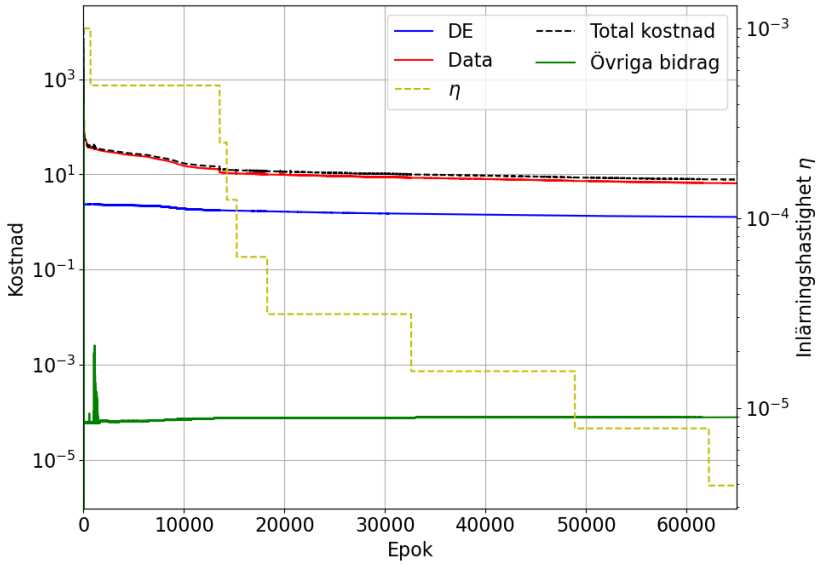
4

Resultat

I följande kapitel presenteras resultaten från de tre olika delproblemen. Resultaten består av träningshistorik för de olika nätverken samt några testexempel.

4.1 Resultat för värmeledning i 1D

Ett nätverk med 101 inparametrar, bestående av ett x-värde och temperaturen samplat i 100 jämt fördelade interna punkter, och ett utvärde har tränats. Modellen består av inlagret, nio gömda lager och utlagret med en sammanlagd arkitektur med lager med antal noder enligt följande: $[101, 80, 70, 60, 50, 40, 30, 20, 10, 5, 1]$. Nedan i figur 4.1 visas hur *kostnadsfunktionen* för modellen utvecklades över tid.



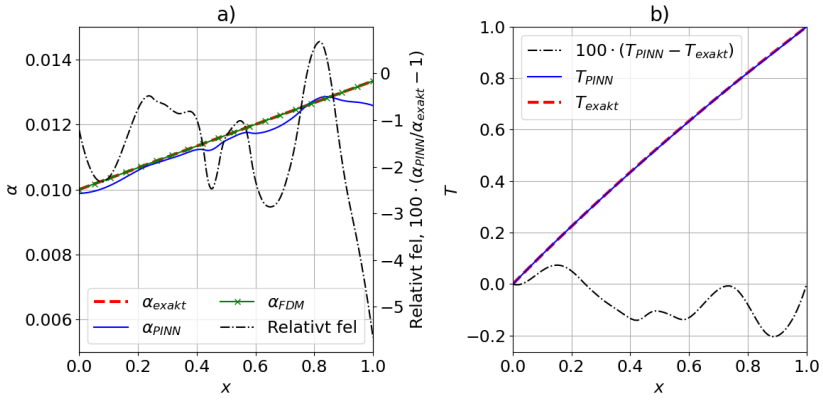
Figur 4.1: Historik över kostnadsfunktionen för PINN:et. ”DE” motsvarar PDE:ns bidrag till kostnadsfunktionen, ”Data” motsvarar kostnadsbidraget från summan av det relativa och absoluta felet för modellprediktionen och ”Övriga bidrag” består av en regulariseringsterm.

Under följande delavsnitt jämförs PINN-modellen med den *FDM*-lösare, som beskrevs i avsnitt 3.4. Jämförelsen görs för testfall som inte finns i träningsdatan, både för exakta värden på temperatur-inparametrarna och när brus adderats.

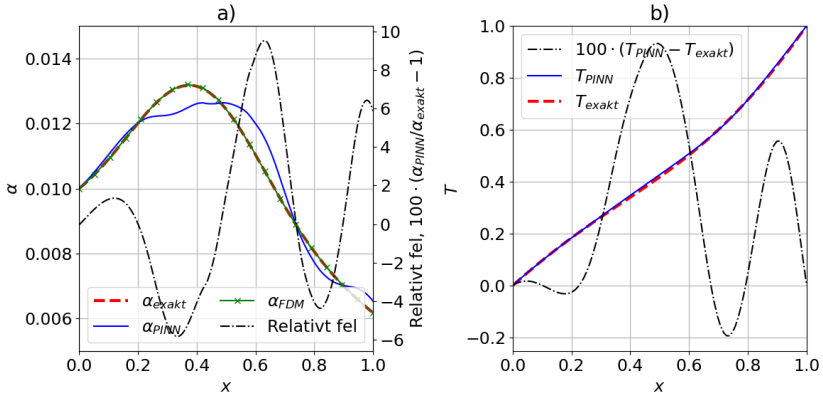
Det α^* som presenteras under respektive graf har normerats så att $\alpha(x) = \frac{10^{-2}}{\alpha^*(0)}\alpha^*(x)$. Modellen har tränats på $\alpha(x)$ och inte $\alpha^*(x)$ och det är $\alpha(x)$ som presenteras i graferna.

4.1.1 Utan brus

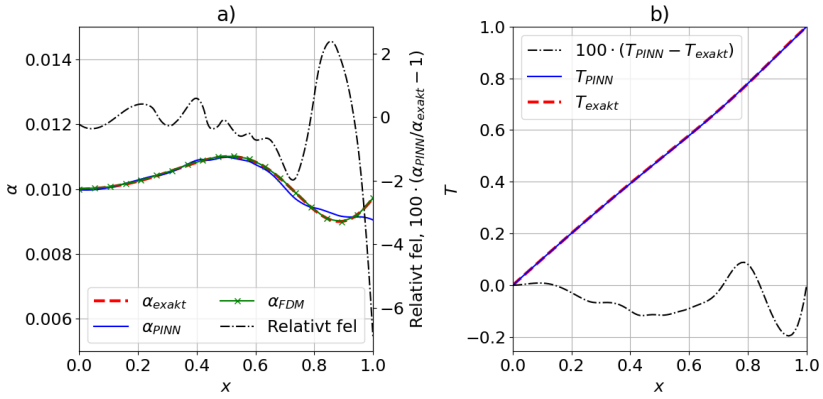
I de efterföljande graferna har 20 noder använts för FDM-lösaren.



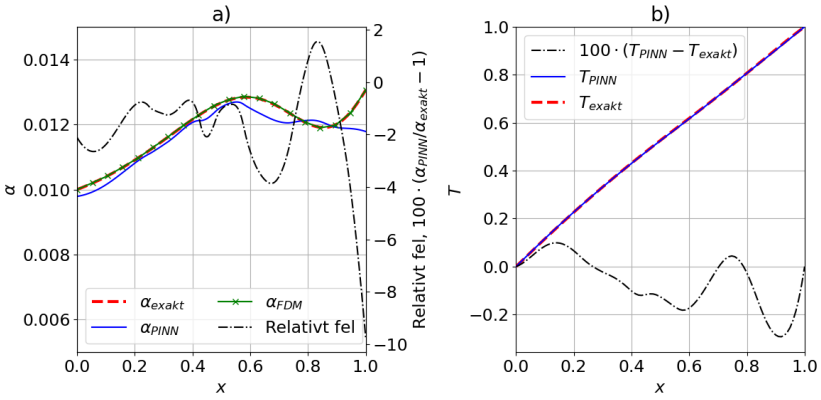
Figur 4.2: Resultatet från PINN och FDM för $\alpha^* = 1 + x/3$.



Figur 4.3: Resultatet från PINN och FDM för $\alpha^* = 1 + \exp\big(-10(x - 0{,}4)^2\big)/2 - x^2/3$.



Figur 4.4: Resultatet från PINN och FDM för $\alpha^* = 1 + \sin\big(6x^2\big)/10$.

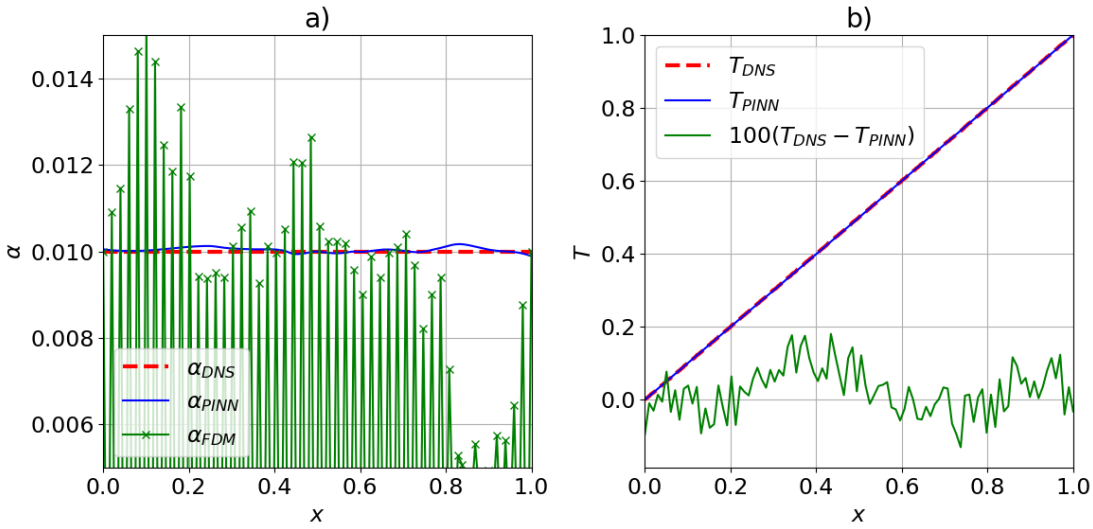


Figur 4.5: Resultatet från PINN och FDM för $\alpha^* = 1 + \sin(6x^2)/10 + x/3$.

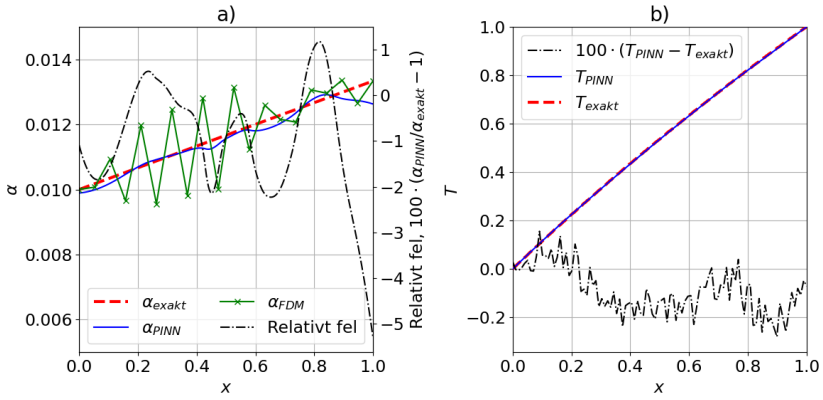
4.1.2 Med brus

Nedan har modellen jämförts mot FDM efter att likformigt fördelat brus mellan -10^{-3} och 10^{-3}, vilket motsvarar $\pm 0{,}1$ % av temperaturskillnaden.

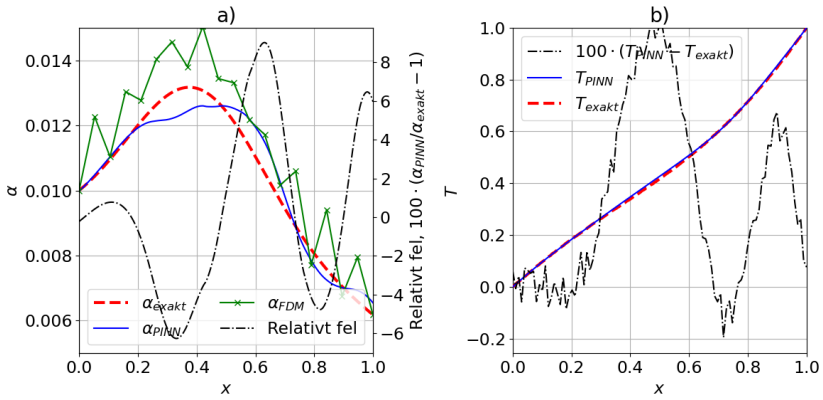
I figur 4.6 nedan visas att även för det simplaste fallet för α presterar FDM mycket dåligt med 100 noder när brus introducerats. För färre noder presterar FDM något bättre och 20 noder används i efterföljande grafer.



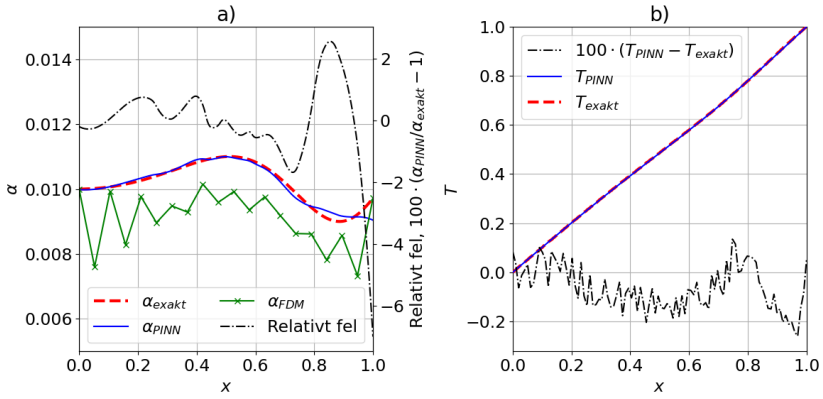
Figur 4.6: Konstant $\alpha^* = 0{,}01$ för FDM med 100 noder.



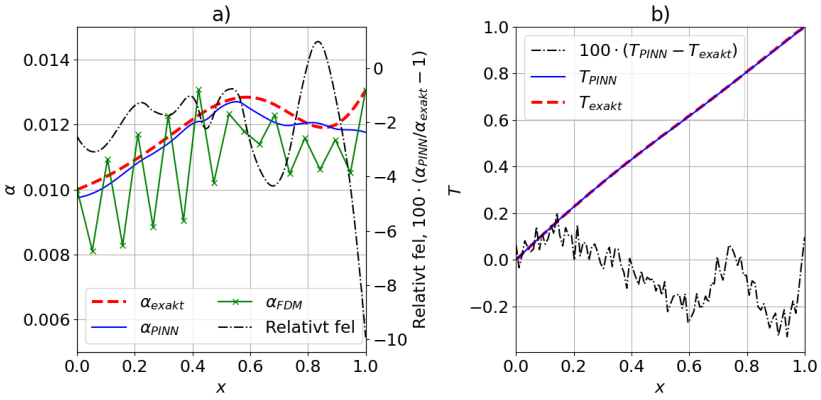
Figur 4.7: Resultatet från PINN där $\alpha^* = 1 + x/3$.



Figur 4.8: Resultatet från PINN där $\alpha^* = 1 + \exp\big(-10(x - 0{,}4)^2\big)/2 - x^2/3$.

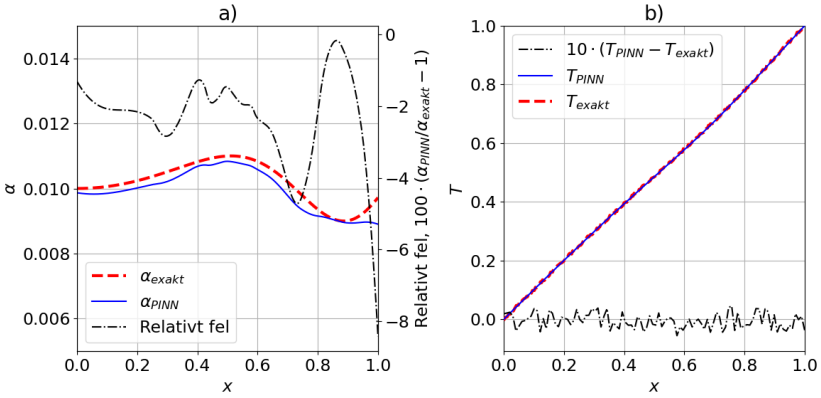


Figur 4.9: Resultatet från PINN där $\alpha^* = 1 + \sin\big(6x^2\big)/10$.

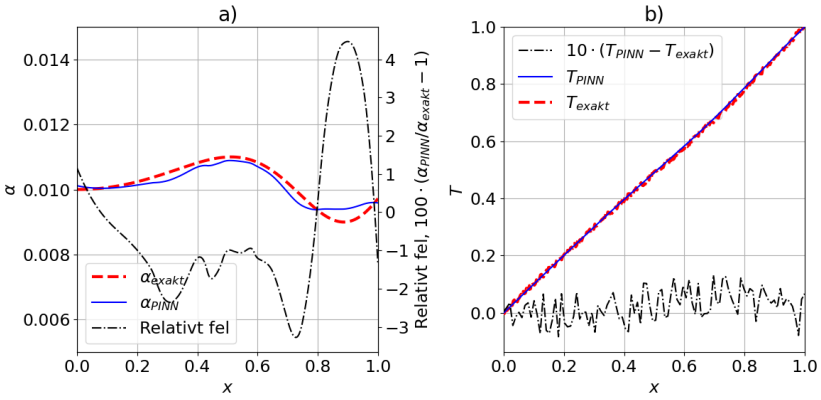


Figur 4.10: Resultatet från PINN där $\alpha^* = 1 + \sin(6x^2)/10 + x/3$.

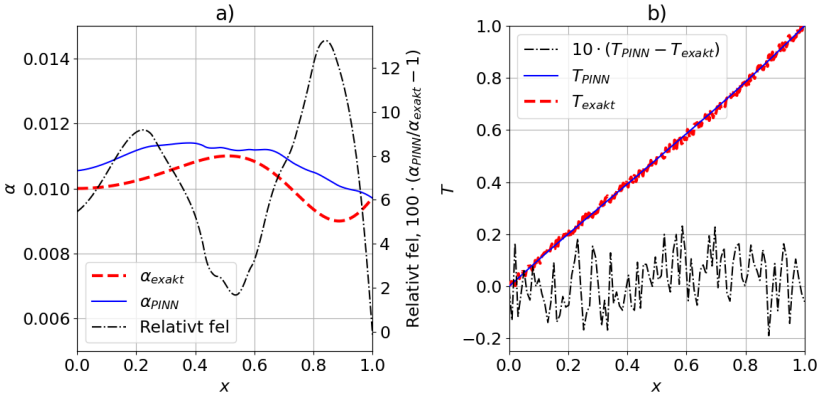
Även olika brusnivåer testades. Modellprediktionerna för brusnivåer på ±0,5 %, ±1 % och ±2 % visas nedan. För högre brusnivåer varierar T_{exakt} så mycket att det knappt finns några likheter med motsvarande temperaturfält utan brus. Anledningen till att 2 % är det högsta pålagda bruset som redovisas nedan är att resultaten från PINN-modellen då bruset är >2 % inte visar någon större likhet med det riktiga värdet på α och felet blir allt större. α_{FDM} har inte inkluderats i figurerna nedan då den mestadels producerar brus där felet är flera storleksordningar större än PINN-modellens fel. Notera att skillnaden $T_{exakt} - T_{PINN}$ nu bara multiplicerats med en faktor 10 istället för 100, som användes tidigare.



Figur 4.11: Resultatet från PINN där $\alpha^* = 1 + \sin(6x^2)/10$ och likformigt brus på upp till ±0,5 % adderats.



Figur 4.12: Resultatet från PINN där $\alpha^* = 1 + \sin(6x^2)/10$ och likformigt brus på upp till ±1 % adderats.



Figur 4.13: Resultatet från PINN där $\alpha^* = 1 + \sin(6x^2)/10$ och likformigt brus på upp till ±2 % adderats.

4.2 Resultat för värmeledning i 2D

I följande tabeller och figurer redovisas resultaten från två olika modellträningar av ett PINN som predikterar värmediffusiviteten α över enhetskvadraten utifrån temperaturfältet som indata enligt metoden i avsnitt 3.2. Det genomfördes flera modellträningar med olika λ_{PDE} och i tabell 4.1 nedan visas den bäst presterande modellen ($\lambda_{\text{PDE}} = 0{,}50$) samt modellen med $\lambda_{\text{PDE}} = 0{,}00$ för att visa hur PDE:en hjälper neuronnätet. λ_{data} och λ_{BC} sattes till 3,0 respektive 0,05 för samtliga.

I figur 4.14, 4.15 och 4.16 visas tre olika valideringsexempel för modellen utan PDE-term i kostnadsfunktionen och i figur 4.17 visas träningshistoriken för samma modell. Dessa tre valideringsexempel valdes för att visa hur modellen presterar i olika typer av α-fördelningar, en mycket varierande, en relativt mjuk och en med skarpa kanter. Samma information för modellen med $\lambda_{\text{PDE}} = 0{,}50$ visas i figur 4.18, 4.19, 4.20 och 4.21.

Tabell 4.1: Sammanfattning av de modellträningar som visas i rapporten.

Modellvariant	Rutnät	Träningsexempel [#]	Epoker [#]	Batch [#]
$\lambda_{\text{PDE}} = 0{,}00$	32x32	2000	146	6
$\lambda_{\text{PDE}} = 0{,}50$	32x32	2000	473	6

Tabell 4.2 visar att modellen med PDE-term ger lägre globalt valideringsfel än modellen utan PDE-term. *Mean Squared Error* (MSE) minskar från $2{,}56 \cdot 10^{-2}$ till $1{,}53 \cdot 10^{-2}$, medan det relativa valideringsfelet minskar från 9,87 % till 7,26 %. Detta innebär att modellen med $\lambda_{\text{PDE}} = 0{,}50$ ger en mer träffsäker rekonstruktion av α över hela valideringsmängden.

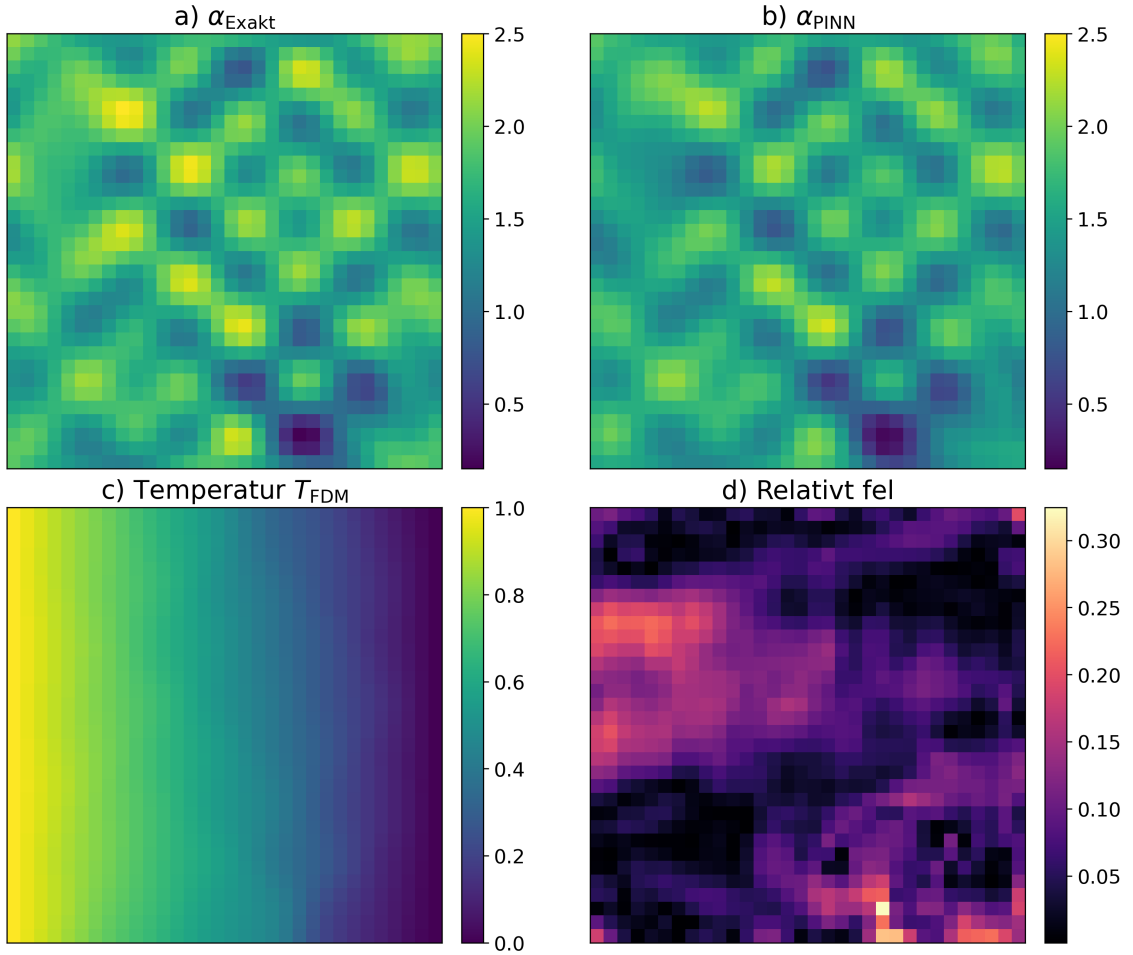
Tabell 4.2: Globala valideringsresultat för modellvarianterna över hela valideringsdatan.

Modellvariant	Valideringsfel MSE	Relativt valideringsfel
$\lambda_{\text{PDE}} = 0{,}00$	$2,56 \cdot 10^{-2}$	9,87 %
$\lambda_{\text{PDE}} = 0{,}50$	$1,53 \cdot 10^{-2}$	7,26 %

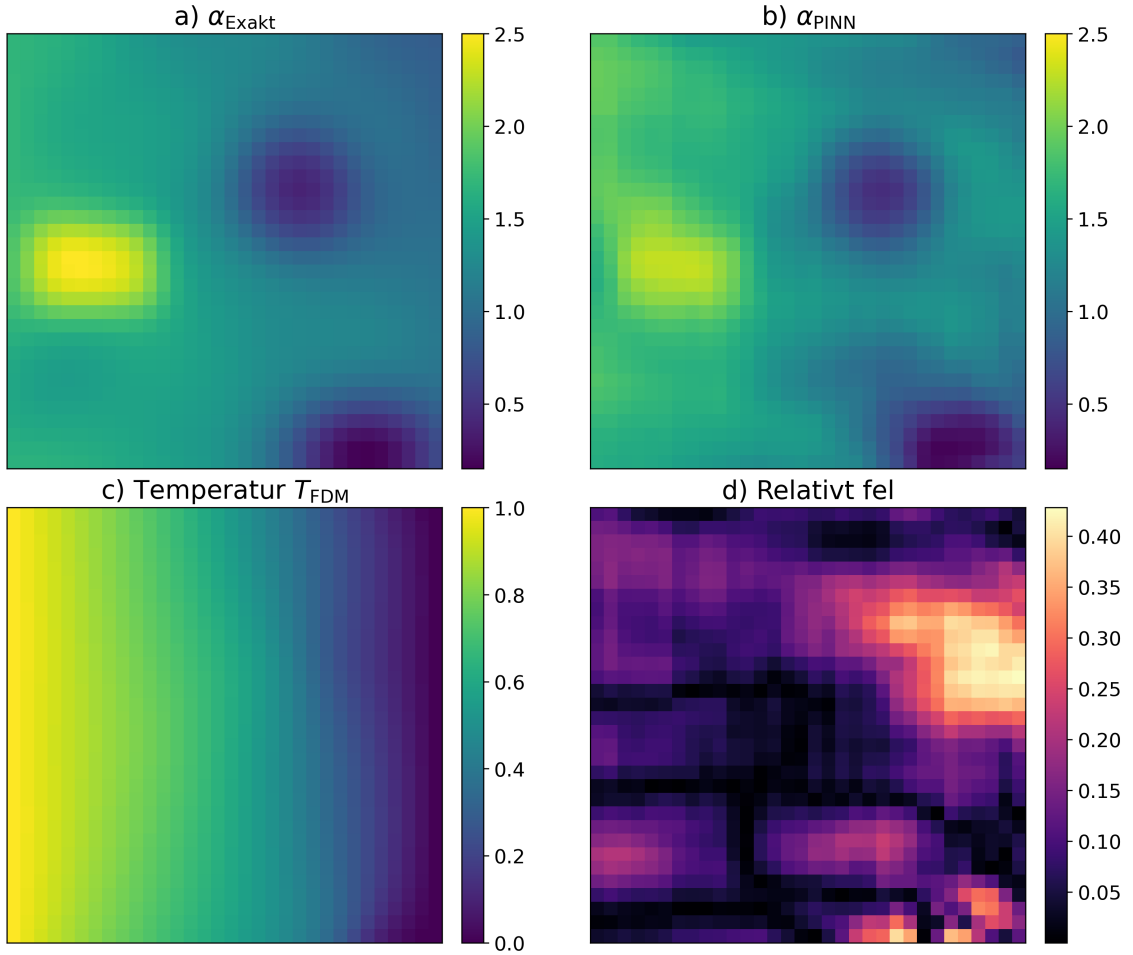
De visuella resultaten i figurerna visar samtidigt att felet inte är jämnt fördelat över domänen. De största avvikelserna uppstår framför allt i områden där α varierar snabbt eller där fältet innehåller skarpare strukturer. I mjukare områden följer det predikterade α-fältet i regel det exakta fältet bättre.

4.2.1 Modell med $\lambda_{\text{PDE}} = 0{,}00$

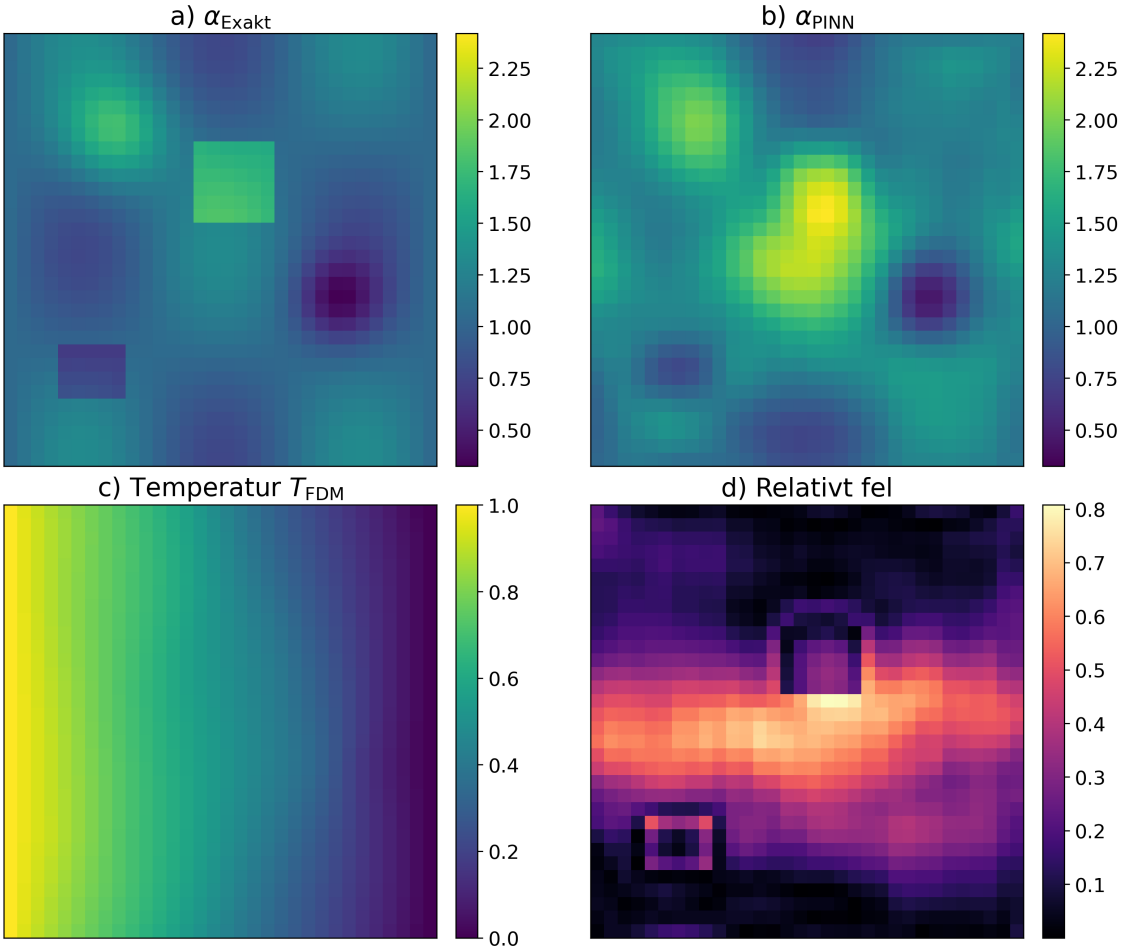
För modellen utan PDE-term återges de övergripande strukturerna i α-fältet, men prediktionen blir delvis utslätad jämfört med det exakta fältet. Detta syns särskilt i områden där α har lokala variationer eller skarpare övergångar. Det relativa felet är därför störst i dessa delar av domänen, medan modellen presterar bättre i områden där α varierar långsammare.



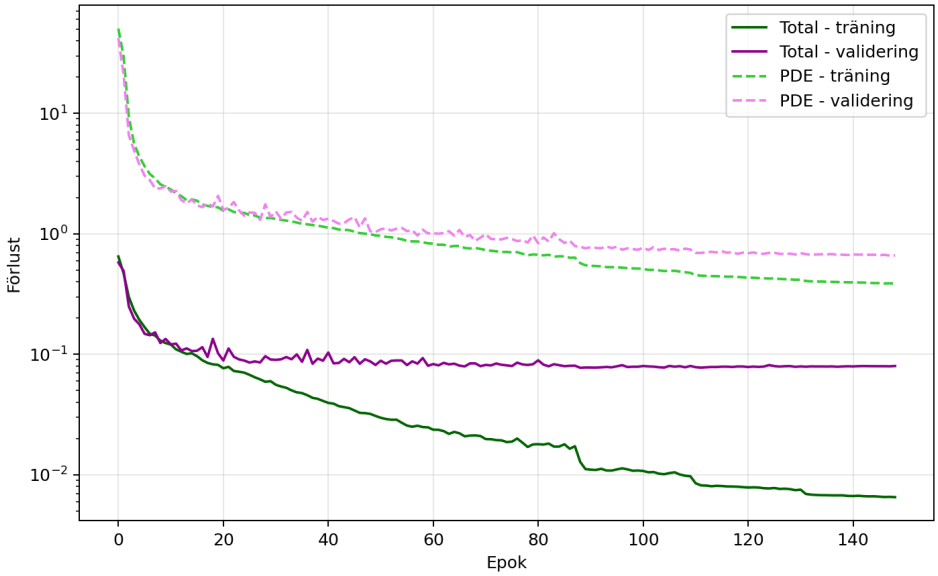
Figur 4.14: Valideringsexempel 1 för modellen med $\lambda_{\text{PDE}} = 0{,}00$. Delfigur a) visar det exakta värmediffusivitetsfältet α_{Exakt} och b) visar modellens predikterade fält α_{PINN}. Dessa två delfigurer visas med samma färgskala för att underlätta jämförelse. Delfigur c) visar temperaturfältet T_{FDM} som användes som indata till modellen, och d) visar det relativa felet mellan α_{Exakt} och α_{PINN}.



Figur 4.15: Valideringsexempel 2 för modellen med $\lambda_{\text{PDE}} = 0{,}00$. I a) visas det exakta α-fältet från den genererade datan och i b) visas motsvarande PINN-prediktion. Delfigur c) visar det tillhörande temperaturfältet T_{FDM}, vilket är modellens huvudsakliga indata tillsammans med koordinatinformation. Delfigur d) visar det relativa felet och synliggör var i domänen avvikelsen mellan exakt och predikterat α-fält är störst.



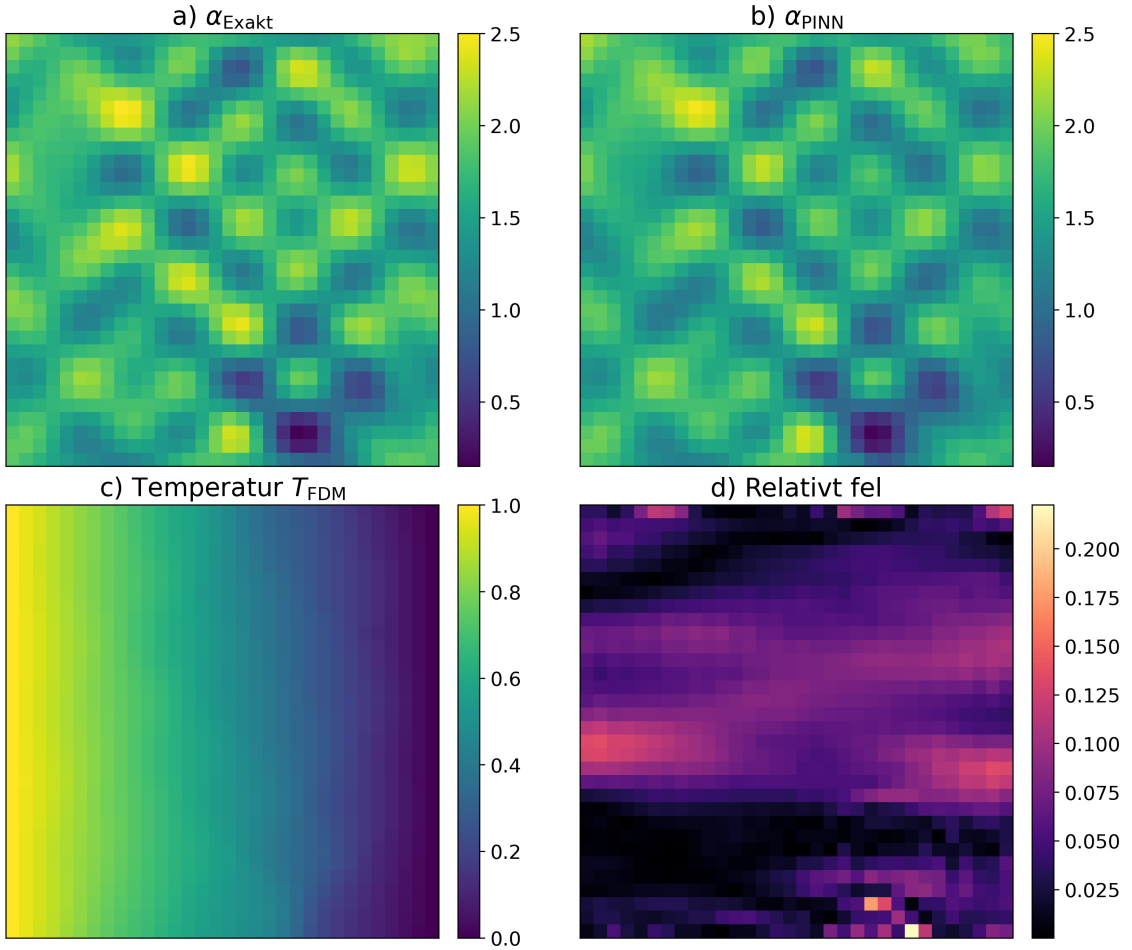
Figur 4.16: Resultat för det specialkonstruerade testfallet `blandat` för modellen med $\lambda_{\text{PDE}} = 0{,}00$. Delfigur a) visar det exakta värmediffusivitetsfältet α_{Exakt}, medan b) visar modellens prediktion α_{PINN}. Delfigur c) visar temperaturfältet T_{FDM} som beräknats från det exakta α-fältet, och d) visar det relativa felet. Testfallet innehåller både mjuka variationer och skarpare lokala strukturer, vilket gör det lämpligt för att bedöma modellens generaliseringsförmåga.



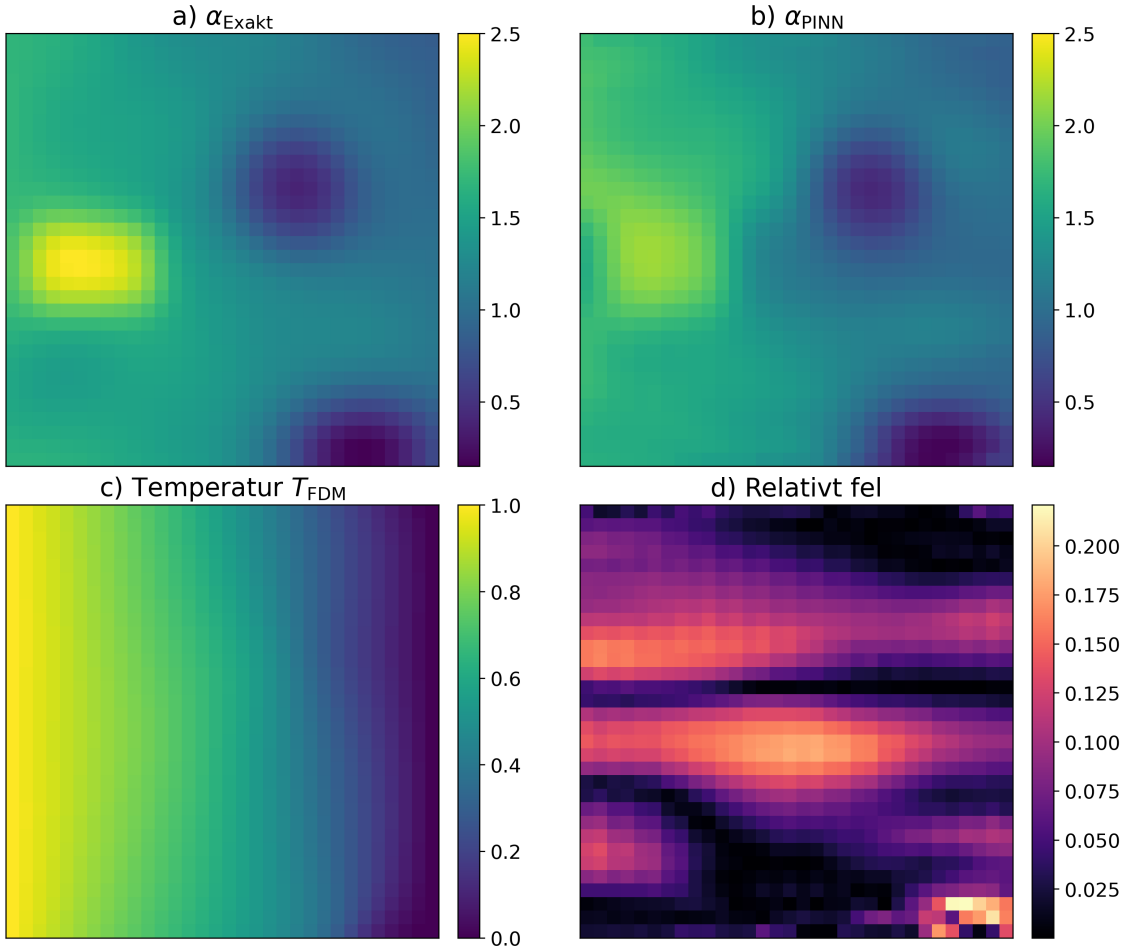
Figur 4.17: Träningshistorik för modellen med $\lambda_{\text{PDE}} = 0{,}00$.

4.2.2 Modell med $\lambda_{\text{PDE}} = 0{,}50$

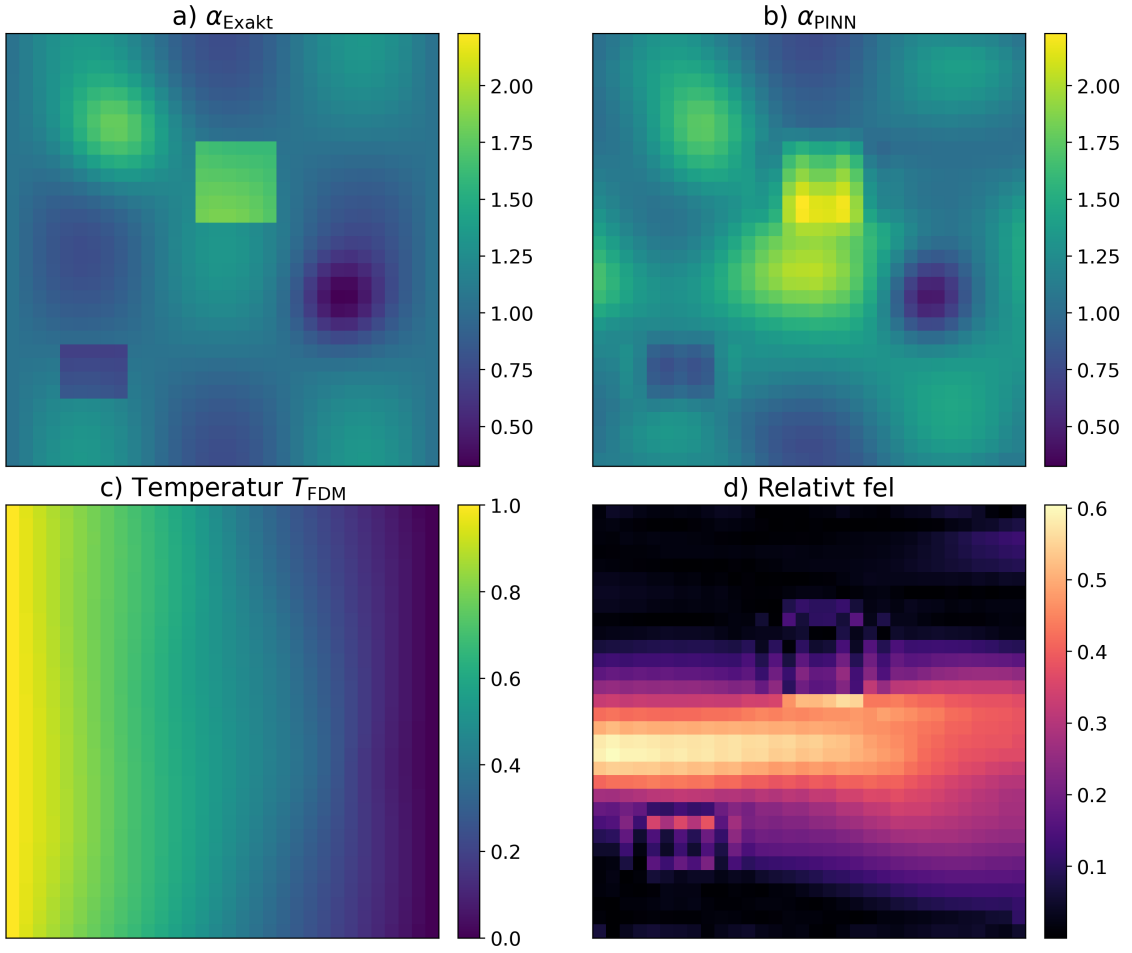
För modellen med $\lambda_{\text{PDE}} = 0{,}50$ ligger det predikterade α-fältet närmare det exakta fältet i både de globala felmåtten och de visuella exemplen. Jämfört med modellen utan PDE-term är strukturerna i α tydligare återgivna, och det relativa felet är generellt lägre. Även för testfallet `blandat` fångar modellen huvuddragen i α-fördelningen, även om avvikelser fortfarande finns nära skarpa övergångar.



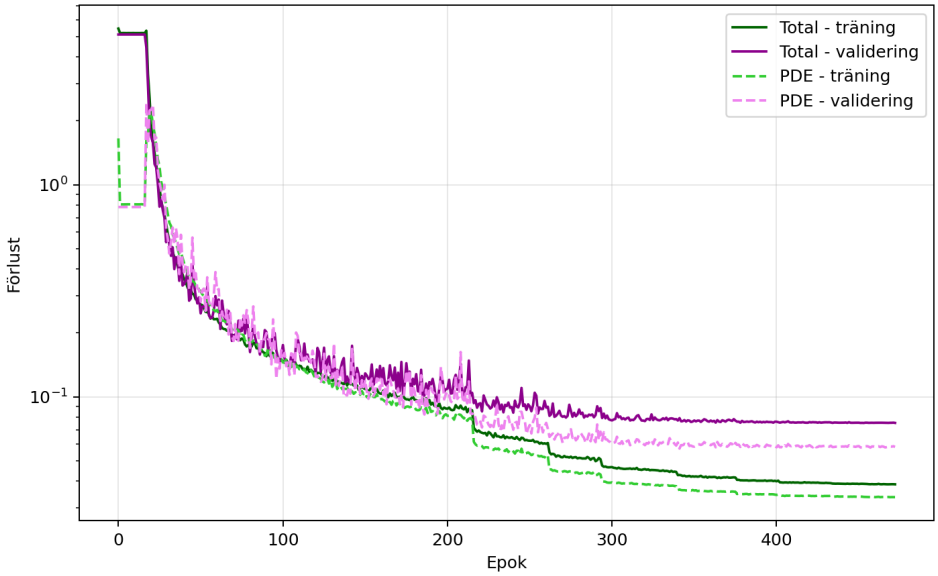
Figur 4.18: Valideringsexempel 1 för modellen med $\lambda_{\text{PDE}} = 0{,}50$. Delfigur a) visar det exakta värmediffusivitetsfältet α_{Exakt} och b) visar det av PINN-modellen predikterade fältet α_{PINN}. Färgskalan är densamma i a) och b), vilket gör det möjligt att direkt jämföra fältens storlek och rumsliga struktur. Delfigur c) visar temperaturfältet T_{FDM} som användes som indata, och d) visar det relativa felet mellan exakt och predikterat α-fält.



Figur 4.19: Valideringsexempel 2 för modellen med $\lambda_{\text{PDE}} = 0{,}50$. I a) visas referensfältet α_{Exakt} och i b) visas modellens prediktion α_{PINN}. Delfigur c) visar det temperaturfält T_{FDM} som uppstår från det exakta α-fältet, medan d) visar det relativa felet i rekonstruktionen. Figuren illustrerar hur väl modellen kan återskapa värmediffusiviteten för ett valideringsexempel som inte ingick i träningsdatan.



Figur 4.20: Resultat för det specialkonstruerade testfallet `blandat` för modellen med $\lambda_{\text{PDE}} = 0{,}50$. Delfigur a) visar det exakta värmediffusivitetsfältet α_{Exakt} och b) visar det predikterade fältet α_{PINN}. Delfigur c) visar det tillhörande temperaturfältet T_{FDM}, och d) visar det relativa felet. Testfallet används för att undersöka hur väl modellen generaliserar till ett sammansatt α-fält med både mjuka variationer och mer lokala förändringar.



Figur 4.21: Träningshistorik för modellen med $\lambda_{\text{PDE}} = 0{,}50$, tränad i 473 epoker.

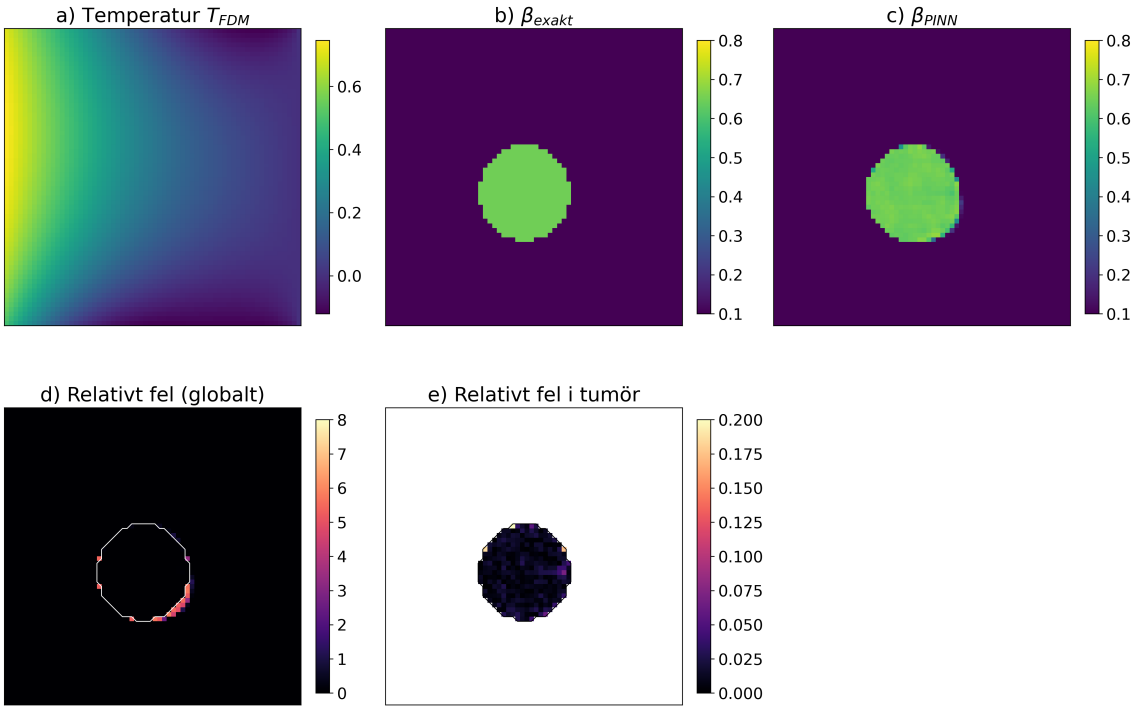
4.3 Resultat för värmetransport i biologisk vävnad

I detta avsnitt presenteras resultaten från de nätverk som tränats för att prediktera parametrerna i Pennes bioheat-ekvation 2.46. Först redovisas resultaten från nätverket som endast predikterade perfusionstermen β och sedan redovisas resultatet från nätverket som predikterade både k och β.

4.3.1 Prediktion av β

För att träna neuronnätverket för att prediktera perfusionstermen β i Pennes bioheatekvation behövdes vikterna för respektive förlustterm i ekvation 2.33 bestämmas. Eftersom inga av randvillkoren beror på β kan förlusten från randvillkoren $\mathcal{L}_{bc}$ bortses från, så den termen sattes till 0,0. För att bestämma storleken på λ_{PDE} jämfört med λ_{data} gjordes några tester där det observerades att nätverket generellt presterade bättre när λ_{PDE} höjdes. Om dock λ_{PDE} blev för stort i förhållande till λ_{data} konvergerade nätverket tidigt vid en lösning där β var i princip konstant över hela domänen. Ett värde på $\lambda_{data} = 1$ och $\lambda_{PDE} = 0,02$ användes därför. Dock är dessa värden inte optimerade då det ligger utanför detta arbetets omfattning.

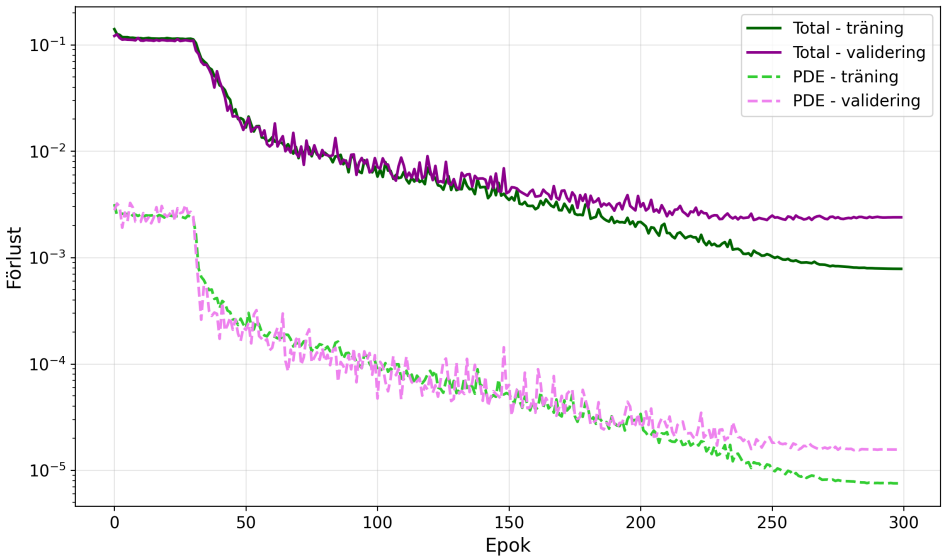
Nätverket tränades i 300 epoker, då konvergens observerades (se figur 4.23). Resultatet av denna träning illustreras med ett exempel från valideringsdatan och visas i figur 4.22. I figur 4.22b visas β-fördelningen som använts för att generera temperaturfördelningen med FDM och i 4.22c syns nätverkets prediktion av β. För att skatta resultatets kvalitet beräknades medelvärdet av det relativa felet över hela domänen för valideringsdatan. För denna träning blev det relativa felet för valideringsexemplen 2,31 %.



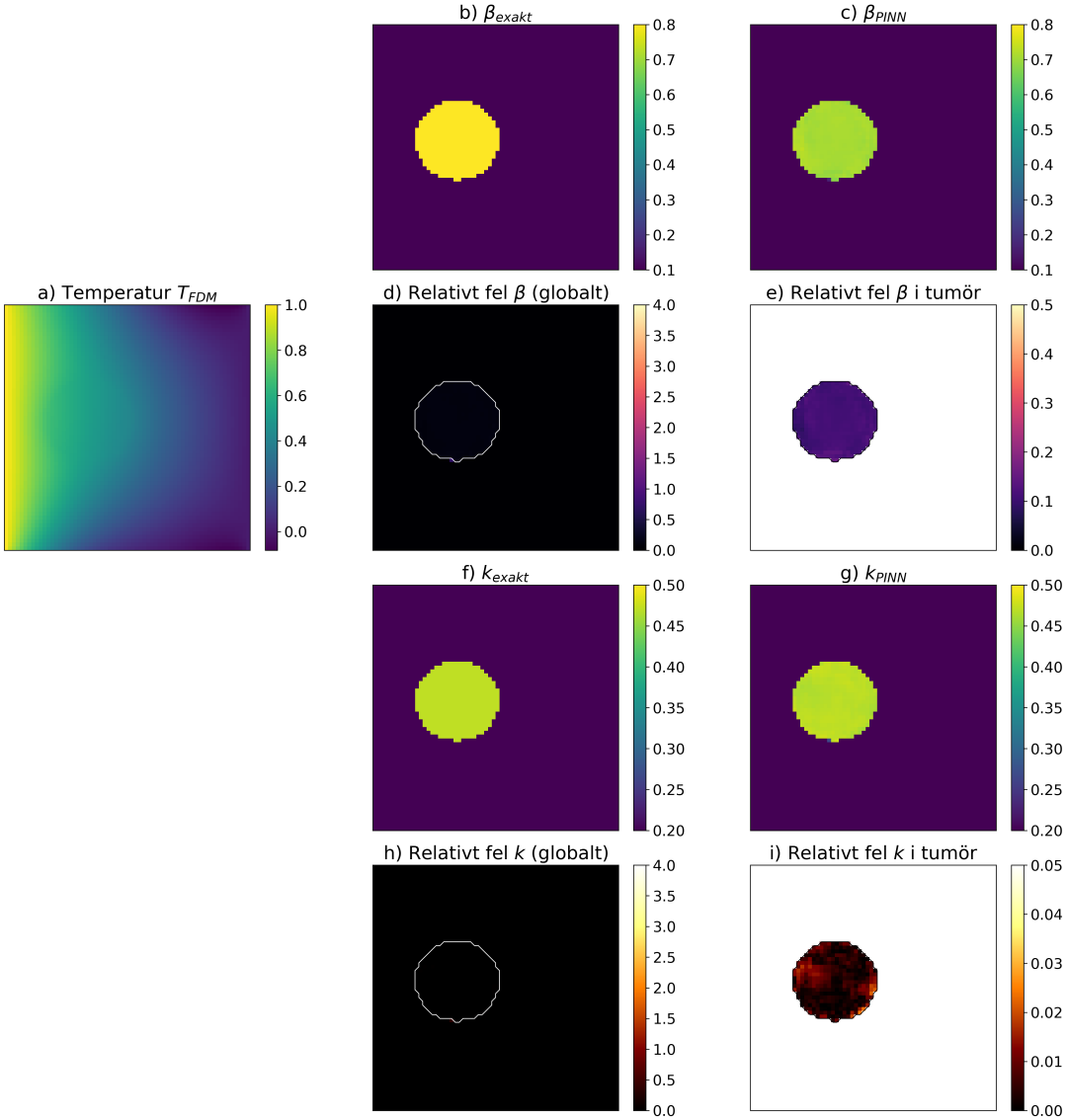
Figur 4.22: Exempel för resultat från neuronnätverk som predikterar β i Pennes bioheatekvation (ekvation 2.46). En tumör i form av en cirkel eller oval genereras med slumpmässigt värde på β som syns i b). När randvillkoren appliceras på domänen ger denna tumör med FDM upphov till en temperaturfördelning a) som används som indata till PINN:et. Predikteringen av β syns i c). Felet i hela domänen syns i d) och når ca 700 % precis vid tumörens kant. Om endast i element innanför tumörens gränser studeras e) når felet ca 8 % förutom några element nära kanten där felet blir större, uppåt 20 %.

4.3.2 Prediktion av β och k

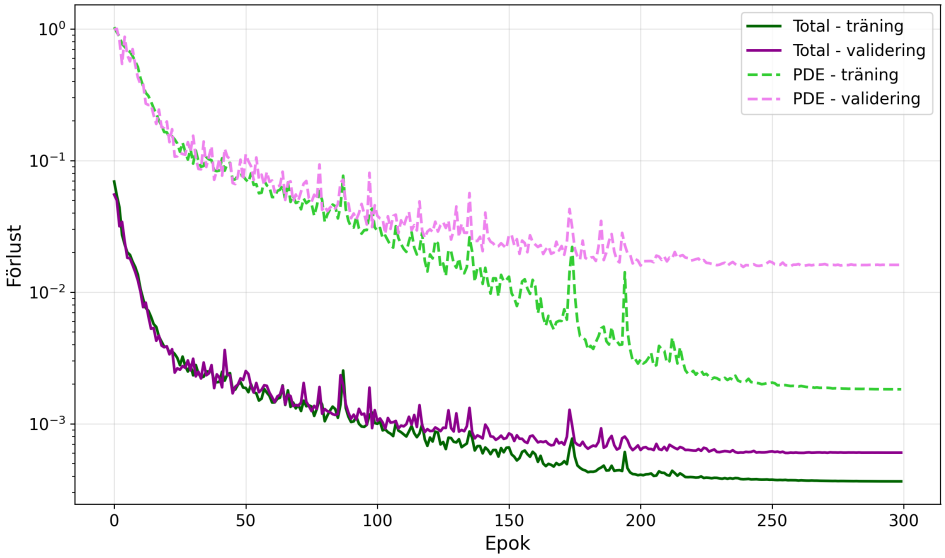
För neuronnätet som predikterar både värmeledningskoefficienten k och perfusionskoefficienten β återintroducerades förlusttermen för randvillkoren eftersom det kommer påverka k. λ_{BC} valdes till samma värde som λ_{PDE} och en träning startades med båda dessa värden som 0,02. Då konvergerade nätverket tidigt på ett felaktigt resultat så λ_{BC} och λ_{PDE} sänktes båda till 0,01. Resultatet från denna träning kan ses i figur 4.24. Även för detta problem tränades nätverket i 300 epoker och träningshistoriken visas i figur 4.25. Medelvärdet för det relativ felet för k över hela domänen var 0,084 % och för β var det 0,78 %. Utanför tumören syns för detta träningsexempel inga fel för varken k eller β. För några exempel syntes en eller ett par enstaka element som nätverket predikterat till ett högre värde men generellt är modellen träffsäker när det kommer till att identifiera tumörens gränser. I figur 4.24f syns att felet för β-fältet i tumören är konstant ca 15 %. Däremot för k ligger felet mellan 0 och 2,5 %.



Figur 4.23: Träningshistorik för neuronnätet som predikterar β i Pennes bioheat-ekvation. Den totala förlusten för träningsdatan visas i lila och den totala förlusten för valideringsdatan i grönt. Den totala förlusten inkluderar förlusten från både datan och PDE:n. Förlusten för endast PDE:n syns i lila- och grönstreckat för träningsdata respektive valideringsdata.



Figur 4.24: Valideringsexempel för PINN:et som har uppskattat både k och β. Tumörplaceringen med motsvarande k och β som använts för att ta fram temperaturfältet i a) visas i f) respektive b). Nätverkets prediktion av k och β syns i c) respektive g). Felen utanför tumören syns i figur d) för β och för k i h). Det relativa felet i tumören syns i e) för β och i) för k. Notera att dessa inte visas i samma skala för att eventuella avvikelser ska synas för respektive parameter.



Figur 4.25: Träningshistorik för neuronnätet som predikterar både k och β i Pennes bioheat-ekvation. Den totala förlusten för träningsdatan visas i lila och den totala förlusten för valideringsdatan i grönt. Den totala förlusten inkluderar förlusten från både datan och PDE:n. Förlusten för endast PDE:n syns i lila- och grönstreckat för träningsdata respektive valideringsdata.

5
Diskussion

5.1 Resultatdiskussion

I detta kapitel diskuteras de resultat som presenterades i föregående kapitel. En genomgående frågeställning är huruvida *PINN* kan användas för att rekonstruera värmediffusivitetsfältet α från ett känt temperaturfält. Att detektera att en avvikelse existerar i ett material är i grunden ett enklare problem än att kvantifiera dess storlek och exakta placering. Det senare kräver en fullständig rekonstruktion av α-fältet, vilket generellt sett är ett svårlöst inversionsproblem [22]. Resultaten visar att PINN har potential att utföra denna rekonstruktion, men att prestandan beror på en rad faktorer såsom problemets inverterbarhet, dess komplexitet samt hur mycket indata liknar träningsdatan. Dessa aspekter diskuteras mer ingående i de följande avsnitten.

5.1.1 Resultatdiskussion för värmeledning i 1D

Resultaten visar att PINN:et till stor del presterar bra för att återskapa de huvudsakliga mönster som finns i värmediffusiviteten givet en temperaturfördelning. Som tidigare nämnts är det ett svårlöst *inverst problem*, vilket också går att se i bland annat från figur 4.2 till 4.5. I b)-delfigurerna visas att den relativa skillnaden mellan T_{exakt} och T_{PINN} är mindre än 1 % för de presenterade testfallen, och oftast inom $\pm$0,2 %. Från figur 4.7 till 4.10 går det att se att motsvarande även gäller för PINN:et trots att brus mellan $\pm$0,1 % har adderats till mätdatan. Då det relativa felet mellan α_{PINN} och α_{exakt} är cirka 10 gånger större än motsvarande relativa skillnad mellan T_{PINN} och T_{exakt} visar det på svårigheten i att lösa det inversa problemet. Detta bekräftas även av att *FDM*-lösaren nästan helt upphör att fungera efter att bruset adderats till temperaturfördelningen. Det visar också på att det är mycket svårt att förbättra prediktionen av α eftersom även små variationer i T resulterar i större förändringar av α.

En anledning till att PINN:et presterar bättre kring $x = 0$ än $x = 1$ beror sannolikt på hur träningsdatan har genererats. I träningsdatan är $\alpha(0) = 0{,}01$ för alla olika träningsexempel, medan $\alpha(x)$ inte har någon sådan begränsning. Även antalet x-värden och deras placering kan spela en roll. Eftersom datagenereringen samplade 73 punkter uniformt mellan $(0, 1)$ har modellen fått mindre träning kring randnoderna än de inre noderna eftersom randnoderna har färre grannoder.

Även om PINN-modellen inte tränats med brus i träningsdatan motstår den brus bra, som kan ses i figur 4.7 till 4.10. En möjlig förklaring kan vara att nätverket inte specifikt kollar på någon individuell nods värden, utan i stället alltid tar alla nodernas värden i hänsyn till varandra och på så sätt bättre hittar det generella

mönstret. En annan möjlig förklaring till varför PINN-modellen presterade bättre än FDM då brus introducerats är att PINN-modellens begränsade storlek på vikterna begränsar den kraftigt oscillerande derivatan, som uppstår mellan två noder på grund av bruset.

När brusnivån ökades visade PINN-modellen sig vara motståndskraftig mot brus, vilket kan ses i figur 4.9, 4.11, 4.12 och 4.13. För brusnivå på 2 % avvek modellprediktionen mer och visade inte alla de generella svängningar som fanns i det egentliga α_{DNS}. På grund av hur känsligt det inversa problemet är för variationer i T är det rimligt att prediktionen för α-fältet försämras från och med dessa nivåer.

5.1.2 Resultatdiskussion för värmeledning i 2D

Resultaten för det tvådimensionella värmeledningsproblemet visar att PINN-modellen kan återskapa de huvudsakliga strukturerna i värmediffusivitetsfältet α utifrån ett givet temperaturfält. I båda valideringsexemplen och i det specialkonstruerade testfallet `blandat` följer det predikterade α-fältet i stora drag det exakta fältets rumsliga variationer. Detta är det viktigaste resultatet för 2D-delen, eftersom det visar att modellen har lärt sig ett användbart samband mellan temperaturfältet T_{FDM} och det bakomliggande värmediffusivitetsfältet α_{Exakt}.

De visuella resultaten, i figur 4.18 till 4.20, visar också att modellen generellt har lättare att återskapa mjuka och långsamt varierande α-fält än områden med skarpa övergångar. I fält med tydliga kanter eller mer lokala variationer blir det relativa felet större, särskilt nära gränser mellan områden med olika värmediffusivitet. Detta är väntat eftersom skarpa strukturer är svårare att rekonstruera från ett stationärt temperaturfält, där informationen om lokala variationer i α ofta är indirekt och utsmetad i temperaturfältet. Felet koncentreras därför ofta till de delar av domänen där α förändras snabbt.

Testfallet `blandat` är särskilt intressant eftersom det kombinerar flera typer av strukturer, bland annat mjuka variationer och skarpare lokala förändringar. Att modellen med PDE-term även där kan fånga huvuddragen i α visar att den har viss generaliseringsförmåga utanför de enklaste valideringsexemplen. Samtidigt framgår det att prediktionen inte är perfekt, vilket visar att modellen fortfarande har begränsningar när den möter mer komplexa fält som exempelvis flera sammansatta rumsliga strukturer.

Modellen utan PDE-term i *förlustfunktionen* presterade sämre än modellen med $\lambda_{\text{pde}} = 0{,}50$. Det relativa valideringsfelet var 9,87 % för modellen med $\lambda_{\text{pde}} = 0{,}00$, jämfört med 7,26 % för modellen där PDE-termen inkluderades. Skillnaden på 2,61 procentenheter visar att den fysikinformerade delen av förlustfunktionen bidrog till en mer träffsäker rekonstruktion av α-fältet. Att modellen utan PDE-term ändå uppnår relativt goda resultat visar samtidigt att mycket av sambandet mellan temperaturfältet och det bakomliggande α-fältet kan läras direkt från den genererade träningsdatan. Eftersom modellen tränades övervakat med tillgång till exakta α-fält kunde datatermen ensam ge en användbar approximation. Detta innebär att modellen med $\lambda_{\text{pde}} = 0{,}00$ snarare fungerar som ett rent datadrivet referensfall, där nätverket lär sig statistiska samband mellan T_{FDM} och α_{Exakt} utan att explicit straffas för avvikelser från värmeledningsekvationen.

Samtidigt blev förbättringen av PDE-termen mindre än vad som möjligen hade kunnat förväntas. En möjlig förklaring är att träningsdatan var brusfri och genererad med samma fysikaliska modell som användes i förlustfunktionen. Under sådana förhållanden innehåller datatermen redan mycket stark information om sambandet mellan temperaturfält och α-fält. PDE-termen får därför främst en kompletterande roll snarare än att helt förändra modellens beteende. En annan möjlig förklaring är att valet av λ_{pde} inte optimerades systematiskt. Det är därför möjligt att andra viktningar av PDE-termen hade gett en större förbättring.

Sammantaget visar resultaten att ett PINN kan användas för att approximativt lösa det inversa värmeledningsproblemet i två dimensioner för den artificiella problemklass som studerats. Ett relativt valideringsfel på 7,26 % för den bästa modellvarianten visar att metoden har potential att identifiera större strukturer och variationer i rumsligt varierande α-fält. Däremot bör slutsatser om verkliga tillämpningar dras med försiktighet, eftersom resultaten bygger på artificiella och brusfria data med idealiserade randvillkor. För mer realistiska tillämpningar behöver modellen därför testas på data med brus, osäkerheter i randvillkor och mer varierade fysikaliska situationer.

5.1.3 Resultatdiskussion för värmetransport i biologisk vävnad

Det nätverk som konstruerades för att bestämma perfusionskoefficienten β lyckades hitta β-fördelningen i domänen med ett medelfel på 2,3 %. För den friska vävnaden var PINN:ets prediktion exakt i nästan hela domänen. De stora avvikelserna förekom i kanten av tumörområdet. Där kunde det relativa felet nå uppemot 700 % i vissa fall. Denna avvikelse anses rimlig då det innebär att nätverket har gissat att tumören ligger utanför där den egentligen är. Inuti tumören är felen mindre. I stora delar av tumören var felet under 1 %, men några element i kanten har ett fel uppemot 20 %. Problemen i kanten beror antagligen åter igen på att nätverket inte hittar tumörens kanter särskilt bra. Att felen i övrigt är låga tyder på att modellen kan identifiera β-fältet i stora drag. Även innanför tumören förekommet dock variationer i β-värdet som når fel på cirka 8 % vilka tyder på att PINN:et inte kan förutsäga fältet med hög precision i nuvarande utformning.

PINN:et som predikterar både k och β hade ett medelfel på valideringsdatan gentemot den verkliga fördelningen på 0,084 % för k och 0,78 % för β. Att modellen presterar bättre för k än för β syns också i figur 4.24. Detta är väntat då värmediffusionskoefficienten bör ha större påverkan på temperaturfältet än den konvektiva termen (som här är perfusion) när de är i samma storleksordning som i detta fall. Inom själva tumören är felet på β större. Trots att det verkliga värdet på β i exemplet är cirka 0,8 har nätverket predikterat ett värde på 0,7. Eftersom datan genererats i intervallet 0,6 till 0,8 antas därför att PINN:et bara identifierat ett medelvärde av alla möjliga värden på β. Detta förstärks av att de andra valideringsexemplen visar samma sak, alltså en prediktion på 0,7, oavsett β_{exakt}. PINN:et kan alltså inte i sin nuvarande utformning uppskatta både k och β samtidigt utifrån den givna temperaturfördelningen.

Trots att felet för β är högt för PINN:et som uppskattar både k och β i figur 4.24e så är det relativa medelfelet lägre än för nätverket som endast predikterar β.

Detta tros bero på att PINN:et som bara predikterar β är sämre på att identifiera tumörens gränser. De element som PINN:et felaktigt identifierar som tumörceller får ett väldigt högt fel eftersom skillnaden mellan tumörens och bakgrundens perfusionskoefficient är så pass stor. Därför kommer dessa celler att bidra mycket till det totala felet. Att nätverket har tillgång till både α- och β-fördelningen verkar alltså göra det lättare att identifiera tumörens gränser. Detta anses vara rimligt eftersom värmeledningen påverkar temperaturfältet mer, som nämndes ovan. Dessutom användes Dirichlet-randvillkor för temperaturen på den vänstra randen för att skapa temperaturfältet, vilket observerades vara lättare för nätverket att hantera jämfört med Robin-randvillkor. För eventuella implementationer bör enstaka element som är felaktigt identifierade inte påverka den resulterande temperaturfördelningen avsevärt. Dessutom visar resultatet från PINN:et som predikterar båda parametrarna att detta problem kan elimineras om nätverket får tillgång till mer information.

5.2 Metoddiskussion

I följande avsnitt diskuteras de metoder som använts och metodval som gjorts i detta arbete samt potentiella delmetoder att vidareutveckla eller införa.

5.2.1 Metoddiskussion för värmeledning 1D

Det finns ett flertal förbättringar som skulle kunna implementeras och undersökas. En enkel sak som potentiellt hade kunnat förbättra PINN:et är att istället för att sampla alla α_{exakt} i precis samma 73 x-värden, så skulle α kunna samplas kring de 73 punkterna, men att de har en förskjutning som varierar slumpmässigt kring ett litet intervall centrerat kring respektive punkt för att öka variationen i träningsexemplen på ett enkelt sätt. Det hade även varit intressant att se hur mycket bättre PINN:et hade kunnat prestera om träningsdatan hade innehållit ännu fler exempel för α_{exakt} med tillhörande T_{exakt}. Potentiellt hade ett större nätverk krävts för att PINN:et ska lära sig den ökade komplexiteten i träningsdatan. Generellt sett borde en ökad variation i träningsexemplen ge en PINN-modell som kan prediktera fler olika α.

Ytterligare en enkel sak att testa hade varit att undersöka hur andra viktningar i kostnadsfunktionen hade påverkat resultatet. Som visas i figur 4.1 är det $\mathcal{L}_{data}$ som dominerar kostnadsfunktionen. En konsekvens av att storleksordningen för α valdes till $\sim 0{,}01$ är att bidragen från det absoluta felet och PDE:n naturligt viktas 100 gånger mindre än om α valts till ~ 1 då båda dessa bidrag är linjära med avseende på α. Anledningen till att $\alpha \sim 0{,}01$ är att det är en vanlig storleksordning på α i många material.

Anledningen till att begränsningen $\alpha(0) = 0{,}01$ gjordes var för att värmeledningsekvationen studerades i steady-state, och därför är α inte unik utan randvillkor eftersom om α löser PDE:n i ekvation 2.38, så löses PDE:n även av 10α och $s\alpha$, för godtycklig konstant $s \in \mathbb{R}$. Om tidsberoende i PDE:n hade introducerats och värmeledningen studeras inte längre i steady-state, så hade olika storleksordningar på α kunnat särskiljas eftersom modellen då hade fått information om hur snabbt temperaturgradienter utjämnats. Det hade dock ställt större krav på beräkningskraft

då fler inparametrar, bestående av temperaturfältet vid flera olika tidpunkter, hade behövts till modellen och en större mängd träningsdata.

5.2.2 Metoddiskussion för värmeledning i 2D

Metoden som användes för värmeledning i 2D bygger på att först generera α-fält och därefter lösa det direkta värmeledningsproblemet för att få motsvarande temperaturfält. Detta är en styrka eftersom det ger full kontroll över både indata och den exakta lösningen, vilket gör det möjligt att kvantitativt utvärdera hur väl modellen rekonstruerar α. Samtidigt innebär det att modellen tränas och valideras på data som kommer från samma typ av generator och samma numeriska lösare. Resultaten visar därför främst hur väl metoden fungerar inom den artificiella problemklass som konstruerats, snarare än hur väl modellen direkt skulle fungera på experimentella mätdata. För framtida applikationer kan dock träningsdatan väljas för det specifika problem och situation man ska använda PINN för just då.

Det *inversa problemet* är i grunden svårt eftersom ett stationärt temperaturfält inte alltid innehåller tillräckligt med information för att entydigt bestämma ett bakomliggande α-fält. Detta innebär att modellen felaktigt kan lära sig att alltid återge en rekonstruktion av α-fältet oavsett givet temperaturfält, trots att denna rekonstruktion inte är rätt för något enskilt temperaturfält. Metoden är därför beroende av att träningsdatan innehåller tillräckligt informativa variationer i α, randvillkor och källterm. I detta arbete användes fasta randvillkor och en fast källterm, vilket gör problemet mer kontrollerat men också begränsar variationen i den fysikaliska situation modellen tränas på.

Stoppfunktionen som implementerats resulterar i att modeller tränas i olika antal epoker, och de två som visas här tränas i 146 respektive 473 epoker vilket på ett sätt problematiserar jämförelsen mellan dessa två modeller eftersom skillnaden i resultat inte enbart beror på PDE-termen. Men genom att studera träningshistoriken för de två modellerna, figur 4.17 och 4.21, så syns att de båda visar tecken på *överträning* vilket alltså tyder på att resultatet inte hade varit särskilt annorlunda utan stoppfunktionen. Dock ska överträning ej ses som något positivt, då det betyder att nätverket i sin nuvarande utformning inte kan lösa problemet bättre med den givna datamängden. Det är dock inte omöjligt att nå ett bättre resultat med den givna datan, men då måste exempelvis nätverkets arkitektur och/eller förlustvikterna ändras för att optimera modellen. Dessutom skulle överträning påverka resultatet negativt om brus introduceras i datan.

Sammanfattningsvis är metoden lämplig för att undersöka om ett PINN kan användas för att lösa ett kontrollerat inverst problem för stationär värmeledning i två dimensioner. De viktigaste begränsningarna är att all data är artificiell, att randvillkor och källterm hålls fasta samt att resultatet är känsligt för valet av förlustvikter och träningsparametrar. För framtida studier vore det därför relevant att undersöka fler värden på samtliga förlustvikter, inkludera brus i temperaturfälten, variera randvillkor och källterm samt testa modellen på mer realistiska eller experimentellt framtagna data. Ytterligare en möjlig utveckling hade varit att undersöka om värmediffusivitetens upplösning hade kunnat vara högre än temperaturfältets upplösning. Detta hade i så fall åstadkommits genom någon form av interpolation,

men en sådan metod kräver dock att modellen tränas mot högupplösta referensfält och bör tolkas med försiktighet, eftersom detaljer som inte finns representerade i indatan inte nödvändigtvis kan bestämmas entydigt.

5.2.3 Metoddiskussion för värmetransport i biologisk vävnad

Resultaten från PINN:et som predikterar β visar att ett PINN kan användas för att identifiera β-fördelningen från en temperaturfördelning med relativt hög precision. Att β kan identifieras är intressant eftersom *perfusion* ofta är den dominerande termen som påverkar temperaturfördelningen, och att det därför kan vara viktigt vid exempelvis *hypertermibehandlingar* [13]. Däremot är de numeriska värdena som använts i denna undersökning orealistiska kopplat till tillämpning i människokroppen utan är dimensionslösa numeriska värden endast valda för att fungera med det befintliga PINN:et som använts till tidigare problem. Lozano et al. [25] har sammanställt värmeledningsegenskaperna för bröstcancervävnad och frisk bröstvävnad. Enligt dessa värden kan de relevanta koefficienterna vara ungefär $k = 0,322$ W/mK och β är ungefär mellan 10^3 och 10^4 W/m^{3}K för frisk vävnad. För cancervävnad är $k = 0,564$ W/mK och β variera mellan 10^4 och 10^5 W/m^{3}K. Från dessa värden kan man också se att β bör ha en större påverkan på temperaturfältet än i de testfall som undersökts av denna studie. Dessutom är skillnaderna mellan frisk vävnad och cancervävnad stora vilket bör innebära att det är lättare för PINN:et att identifiera ändringarna än om den friska och sjuka vävnaden hade haft mer liknande värmetransportsegenskaper. Därför hade det varit intressant att vidare modifiera nätverket för att behandla mer verklighetsförankrade modeller av tumörerna. Även de temperaturer som använts i randvillkoren och blodtemperaturen T_b hade kunnat ändras för att mer efterlikna ett realistiskt fall.

Den metaboliska värmeproduktionen har också antagits vara konstant över både frisk vävnad och tumörvävnad. Cancervävnad har dock förändrad metabolism vilket kan leda till en ökad värmeproduktion [26]. För att göra modellen mer realistisk hade denna skillnad också kunnat implementeras genom att använda ett högre värde på Q_{met} i tumören. Dessutom hade det kunnat underlätta identifieringen av tumörens gränser vilket hade kunnat förbättra resultatet för både k och β vid kanterna.

När både k och β predikterades av PINN:et lyckades inte nätverket uppskatta båda dessa parametrar, utan β_{PINN} hade ingen koppling till det verkliga värdet. Detta var väntat då β hade en mindre påverkan på temperaturfältet och nätverket inte hade något sätt att skilja på påverkan från k- eller β-termen och därför tillskriver allt till k. Resultaten tyder alltså på att det inte finns någon entydig lösning för både k och β utifrån endast temperaturfördelningen. För att PINN:et möjligen ska kunna identifiera både k och β föreslås två förbättringar på nuvarande modell. Delvis hade ett större β i förhållande till k kunnat användas för att β ska få en större påverkan på temperaturfältet, likt som föreslogs tidigare i detta avsnitt för att göra modellen mer realistisk. Denna förändring bör dock inte lösa att temperaturfördelningen inte motsvarar en entydig fördelning av k och β. Sannolikt hade bara all variation tillskrivits β istället om β är tillräckligt mycket större än k. Det är möjligt att en sådan modell är tillräckligt bra på att beskriva hur värmetransporten

ser ut i domänen för vissa tillämpningar om k ändå bara har en försumbar påverkan. Om precisionen för en sådan modell är för låg eller om man av någon annan anledning skulle vilja ta reda på båda parametrar hade någon annan förbättring behövts. En sådan skulle kunna vara att studera Pennes bioheat-ekvation med tidsberoende, detta diskuteras mer i avsnitt 5.2.4.

En begränsning i denna undersökning är tillgången till beräkningskraft. Det har gjort att större träningar med fler träningsexempel och högre upplösning inte varit möjliga. Mer omfattande träningar hade kunnat leda till bättre resultat samt möjliggjort för större variationer i flera parametrar som tumörens storlek, form och variationer i koefficienternas värden. Dessutom hade modellen kunnat utökas till 3D för att återigen komma närmre det verkliga fallet.

Resultatet från avsnitt 4.1.2 (där data med brus har använts i PINN:et) tyder på att PINN är en bra metod för att hantera data med brus. För en applikation för hypertermibehandling krävas att metoden är tolerant mot brus då det alltid kommer förekomma i verkliga mätningar och eftersom toleranserna är små krävs det att nätverket fortfarande presterar bra även om brus förekommer. För att säkerställa att resultatet från avsnitt 4.1.2 också gäller i applikationen för cancerbehandling skulle även detta PINN behöva tränas med brusig data.

5.2.4 Jämförelse av delproblem samt generell metoddiskussion

Trots att liknande metoder användes för alla delproblem presterade de olika bra. I 1D-fallet användes ingen valideringsdata så ett totalt relativt fel beräknades inte men för de fall som illustreras i avsnitt 4.1.1 verkar det relativa felet vara i storleksordningen 1-2 % för de allra flesta fall. För värmeledningsproblemet i 2D var det relativa felet 7,26 % och för det biologiska värmetransportproblemet var det relativa felet mellan 2,31 % ner till 0,084 % beroende på vilken modell och vilken parameter som predikterades. Att nätverket som behandlar det endimensionella problemet presterar bättre än det tvådimensionella var väntat eftersom det är ett enklare problem. Skillnaden är dock inte så stor, och jämfört med det biologiska värmetransportproblemet presterade den i vissa fall sämre. Det beror förmodligen på att nätverket för 1D-problemet tränats på en stor variation av α-fördelningar. Datan i det biologiska värmetransportproblemet bestod av enklare samband (styckvis konstanta fördelningar) och variationen mellan fallen var liten i jämförelse. Därför anses det rimligt att nätverket som applicerades på det biologiska värmetransportproblemet presterade bättre trots att detta i grunden är ett svårare problem. I naturen förekommer i största utsträckning enklare värmeledningskoefficienter än de som använts i värmeledningsproblemen i 1D och 2D. Därför bör modeller som presterar bättre kunna utvecklas om de ska implementeras på mer praktiska problem. I problem där modellen ska tillämpas kan också mer kännedom om systemet finnas så att träningsdatans omfattning kan begränsas till att till största del omfatta den typ av fördelning som är av intresse att kunna rekonstruera.

Det inversa problemet att bestämma α från ett temperaturfält är, som förklarat i teoriavsnittet, generellt sett inte ett välställt problem. Eftersom datan som används för att bestämma α-fältet är diskret uppstår möjligheten att flera α-fält kan

ge samma temperaturfördelning. Upplösningen på indatan är således direkt kopplad till hur små och kraftiga variationer som går att finna med PINN. Att arbeta med det stationära fallet innebär en förenkling jämfört med det tidsberoende, men medför samtidigt en informationsförlust. Temperaturens tidsutveckling, som direkt återspeglar delar av diffusiviteten och andra parametrars rumsliga variation går förlorad. Denna förlust är en orsak till att de problem som har betraktats ibland är svåra att invertera. För biologisk vävnad där önskan är att kunna göra en prediktion av värmeledningskoefficienten k och perfusionen β samtidigt är tidsberoende sannolikt extra viktigt. Eftersom både k och β påverkar värmeledning är det rimligt att anta att en separation av bidragen under en stationär förenkling kan vara komplicerad.

Det kommer aldrig att gå att rekonstruera α på hela intervallet $[0, 1]$ perfekt för ett godtyckligt, eller nästan godtyckligt, α från ett ändligt antal punkter från temperaturfältet eftersom α är definierat i ett oändligt antal punkter, men modellen endast tar ett finit antal punkter som inparametrar. Om α varierar någorlunda snällt går det att återskapa det i stora drag, som visats i resultaten. Denna begränsning utgör en av svårigheterna i att lösa det inversa problemet.

Ytterligare en svårighet med att lösa det inversa problemet kommer från hur α beror på T. Som visades i avsnitt 2.3.1 finns ett $\left(\frac{\partial T}{\partial x}\right)^{-1}$-beroende i den analytiska lösningen för α i 1D. Det gör att α varierar mycket känsligt på ändringar i T' och det blir särskilt problematiskt om T' byter tecken, då det orsakar en division med 0 och därmed inte går att säga något om α i den punkten. Rent fysikaliskt är det rimligt att det inte går att säga något om α där $T' = 0$ eftersom det innebär att ingen värme flödar. Även om det inte finns någon analytisk lösning till värmeledningsekvationen 2.38 i 2D för det generella fallet kommer det att finnas likartade beroenden på T-derivatorna, vilket kan orsaka liknande problem som i 1D-fallet.

6
Slutsatser

Värmeledning genom ett medium är sällan helt homogen. I detta arbete användes ett *Physics-Informed Neural Network* (PINN) för att med hjälp av endast temperaturfältet och dess randvillkor, inverst hitta de bakomliggande parametrarna värmeledningsfält och perfusionsfält.

I en dimension tyder resultaten på att det är möjligt att prediktera lätta varierande α med lågt relativt fel från endast ett temperaturfält med hjälp av PINN. Ett intressant resultat är att PINN kan bestämma α då brus introduceras, vilket mer traditionella numeriska metoder som FDM har svårt för. Denna fördel kan göra PINN intressant i mer praktiska applikationer. Vidare hade det varit intressant att vidareutveckla metoden genom att införa slumpmässig sampling av α inom intervallet samt att införa fler träningsexempel i syfte att förbättra nätverkets prediktering av svårare och mer godtyckliga α.

För värmeledning i två dimensioner visar resultaten att ett PINN kan användas för att rekonstruera huvuddragen i ett rumsligt varierande värmediffusivitetsfält utifrån ett stationärt temperaturfält. Modellen med PDE-term presterade bättre än modellen utan, vilket tyder på att fysikalisk information kan förbättra rekonstruktionen. Samtidigt kvarstår problem vid skarpa övergångar i α-fältet, där det relativa felet blev störst. Studien visar därför att PINN har potential för inversa problem inom värmeledning i två dimensioner, men att vidare studier krävs för mer realistiska fall med exempelvis brus, varierande randvillkor och mer komplexa materialfält.

I den förenklade modellen för värmetransport i biologisk vävnad kunde nätverket uppskatta perfusionsparametern β med relativt hög noggrannhet. När endast β predikterades blev det relativa felet 2,31 % över valideringsdatan. Även när både värmeledningskoefficienten k och perfusionsparametern β predikterades samtidigt erhölls låga relativa fel, 0,084 % för k och 0,78 % för β. Detta visar att metoden har potential för att identifiera flera okända parametrar i en förenklad modell av biologisk vävnad. Samtidigt bör resultatet tolkas med försiktighet eftersom modellen bygger på en enkel geometri och genererade temperaturfält snarare än experimentella mätdata.

Sammantaget visar arbetet att PINN är en lovande metod för inversa problem inom värmetransport. Resultaten bör dock tolkas inom ramen för arbetets avgränsningar. All data som använts är genererad, randvillkor och källtermer har varit idealiserade och mätbrus har endast behandlats i begränsad omfattning. För att metoden ska kunna användas i mer realistiska tillämpningar krävs därför vidare studier med mer verklighetsnära geometrier, materialparametrar, randvillkor och brusnivåer.

Litteraturförteckning

[1] J. D. Logan, *Applied Partial Differential Equations*, 3:e. Springer, 2015.

[2] F. Black och M. Scholes, "The Pricing of Options and Corporate Liabilities," *Journal of Political Economy*, årg. 81, s. 637–654, 1973.

[3] C. Forssén, "Neural Networks," i *Machine Learning and Neural Networks*, Chapter 13, 2024. hämtad 23 febr. 2026. URL: https://cforssen.gitlab.io/tif385-book/content/MachineLearning/NeuralNet.html.

[4] S. Thakur, E. Esmaili, S. Libring, L. Solorio och A. M. Ardekani, "Inverse resolution of spatially varying diffusion coefficient using physics-informed neural networks," *Physics of Fluids*, årg. 36, nr 8, aug. 2024. DOI: 10.1063/5.0207453.

[5] Z. Ren, S. Zhou, D. Liu och Q. Liu, "Physics-Informed Neural Networks: A Review of Methodological Evolution, Theoretical Foundations, and Interdisciplinary Frontiers Toward Next-Generation Scientific Computing," *Applied Sciences*, årg. 15, nr 14, 2025. DOI: 10.3390/app15148092.

[6] T. Botarelli, M. Fanfani, P. Nesi och L. Pinelli, "Using Physics-Informed neural networks for solving Navier-Stokes equations in fluid dynamic complex scenarios," *Engineering Applications of Artificial Intelligence*, årg. 148, 2025, ISSN: 0952-1976. DOI: 10.1016/j.engappai.2025.110347.

[7] J. Pu, J. Li och Y. Chen, "Solving localized wave solutions of the derivative nonlinear Schrödinger equation using an improved PINN method," *Nonlinear Dynamics*, årg. 105, s. 1723–1739, 2021. DOI: 10.1007/s11071-021-06554-5.

[8] H. M. Abdelaal, M. Wahba och H. A. El-Dawy, "Physics-informed neural networks (PINN) for heat transfer: a comprehensive analysis," *Neural Computing and Applications*, årg. 38, s. 250, 2026. DOI: 10.1007/s00521-026-12019-w.

[9] D. Kim och J. Lee, "A Review of Physics Informed Neural Networks for Multiscale Analysis and Inverse Problems," *Multiscale Science and Engineering*, årg. 6, s. 1–11, 2024. DOI: 10.1007/s42493-024-00106-w.

[10] J. Chu, B. Li och G. Yang, "Convolutional physics-informed neural networks for heat flux prediction under noisy data conditions," *International Journal of Heat and Mass Transfer*, årg. 254, 2026, ISSN: 0017-9310. DOI: 10.1016/j.ijheatmasstransfer.2025.127727.

[11] Z. He, F. Ni, W. Wang och J. Zhang, "A physics-informed deep learning method for solving direct and inverse heat conduction problems of materials," *Materials Today Communications*, årg. 28, 2021, ISSN: 2352-4928. DOI: 10.1016/j.mtcomm.2021.102719.

[12] National Cancer Institute. "Hyperthermia to Treat Cancer," hämtad 17 april 2026. URL: https://www.cancer.gov/about-cancer/treatment/types/hyperthermia.

[13] G. Cappellini, A. Cristofaro, E. De Santis, E. Staffetti, G. Trappolini och M. Vendittelli, "Adaptive Estimation of Pennes' Bio-Heat Equation: Observer Design and PINNs-Based Implementation," *IEEE Transactions on Control Systems Technology*, s. 1–16, 2026. DOI: 10.1109/TCST.2026.3656581.

[14] R. E. McClelland, R. Dennis, L. M. Reid, J. P. Stegemann, B. Palsson och J. M. Macdonald, "Chapter 6 - Tissue Engineering," i *Introduction to Biomedical Engineering (Third Edition)*, ser. Biomedical Engineering, J. D. Enderle och J. D. Bronzino, utg., Third Edition, Boston: Academic Press, 2012, s. 273–357, ISBN: 978-0-12-374979-6. DOI: 10.1016/B978-0-12-374979-6.00006-X.

[15] B. Mehlig, *Machine Learning with Neural Networks: An Introduction for Scientists and Engineers*. Cambridge, United Kingdom: Cambridge University Press, 2022, ISBN: 978-1-108-49493-9. DOI: 10.1017/9781108860604.

[16] X. Glorot och Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," i *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, PMLR, 2010, s. 249–256.

[17] O. Ronneberger, P. Fischer och T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," i *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, Springer, 2015, s. 234–241.

[18] M. Raissi, P. Perdikaris och G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational Physics*, årg. 378, s. 686–707, 2019.

[19] R. Fitzpatrick, *Computational Physics*. CreateSpace Independent Publishing Platform, 2015, ISBN: 1506189806.

[20] G. Biswas och S. Mukherjee, *Computational Fluid Dynamics*. Alpha Science International, 2012.

[21] R. J. LeVeque, *Finite Volume Methods for Hyperbolic Problems* (Cambridge Texts in Applied Mathematics). Cambridge University Press, 2002.

[22] V. Isakov, *Inverse Problems for Partial Differential Equations*, 3. utg. Cham: Springer, 2017, ISBN: 978-3-319-51658-5. DOI: 10.1007/978-3-319-51658-5.

[23] X. Yuan m. fl., "Advances in bioheat transfer models for hyperthermia: A comprehensive review and future directions," *Results in Engineering*, årg. 28, 2025, ISSN: 2590-1230. DOI: 10.1016/j.rineng.2025.107499.

[24] R. K. Jain, ”Analysis of Heat Transfer and Temperature Distributions in Tissues during Local and Whole-Body Hyperthermia,” i *Heat Transfer in Medicine and Biology: Analysis and Applications. Volume 2*, A. Shitzer och R. C. Eberhart, utg., New York, NY: Springer, 1985, s. 3–54.

[25] A. Lozano, J. C. Hayes, L. M. Compton, J. Azarnoosh och F. Hassanipour, ”Determining the thermal characteristics of breast cancer based on high-resolution infrared imaging, 3D breast scans, and magnetic resonance imaging,” *Scientific Reports*, årg. 10, nr 1, 2020. DOI: 10.1038/s41598-020-66926-6.

[26] F. J. González, ”Non-invasive estimation of the metabolic heat production of breast tumors using digital infrared imaging,” *Quantitative InfraRed Thermography Journal*, årg. 8, nr 2, s. 139–148, 2011. DOI: 10.3166/qirt.8.139-148.

A

Källkod för respektive delproblem

Källkoden för respektive delproblem är inbäddade som ZIP-filer i PDF:en och kan bland annat extraheras i PDF-läsare som Acrobat Reader och Okular. Även kommandot

```
pdfdetach -saveall Kandidatarbete_MMSX21_VT26_17A.pdf
```

kan användas för att extrahera samtliga inbäddade filer. Vanligtvis kan inte webbläsare som exempelvis Chrome, Safari eller Firefox extrahera inbäddade filer.

INSTITUTIONEN FÖR MEKANIK OCH MARITIMA VETENSKAPER
CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige 2026
www.chalmers.se

CHALMERS