Forschungsprojekt: Wärmepumpendemonstrator

Demonstratorapp zur Präsentierung der Ergebnisse eines Konvolutionsnetzwerks für Wärmepumpen

Ursula Krause	(st$X$),	Laurin Röseler	(st177288),
Christof Schuster	(st$X$),	Fabio Tucciarone	(st177167)

st$X$@stud.uni-stuttgart.de

Zusammenfassung

Neque porro quisquam est qui dolorem ipsum quia dolor sit amet, consectetur, adipisci velit... Neque porro quisquam est qui dolorem ipsum quia dolor sit amet, consectetur, adipisci velit... Neque porro quisquam est qui dolorem ipsum quia dolor sit amet, consectetur, adipisci velit...

1 Einführung

2 Projektarchitektur

Zu Begin möchten wir einen Überblick über die Projektarchitektur und die dazugehörigen Entwurfsentscheidungen geben.

- Diagramm: Zusammenspiel der Komponenten (grob)

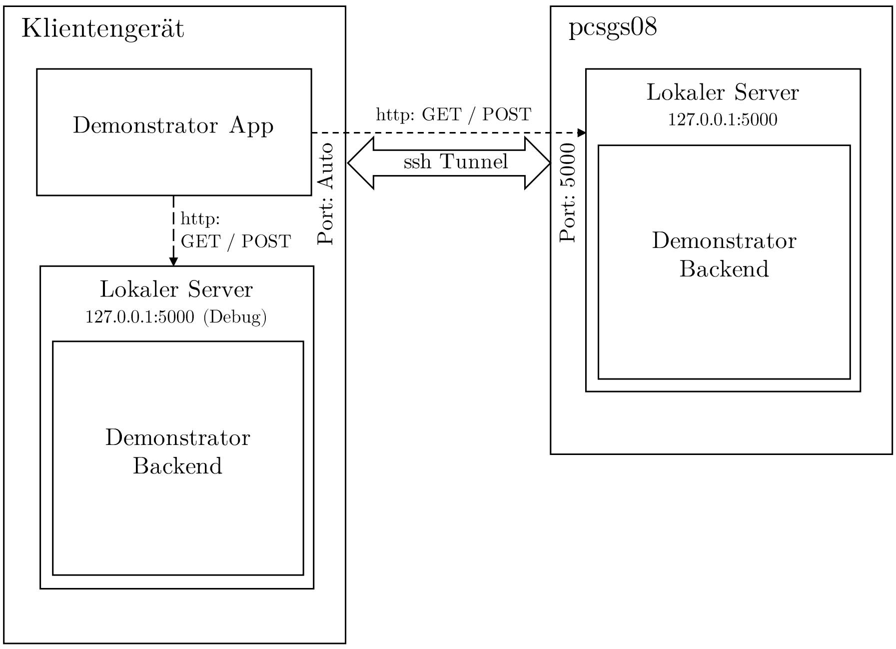

Abbildung 1: Grobe Projektarchitektur

- Backend

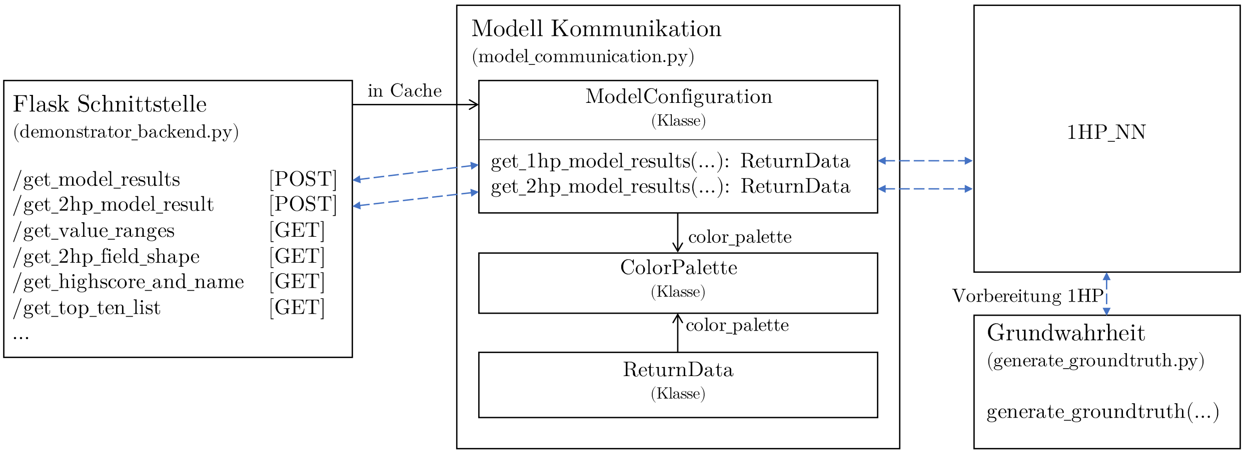

Abbildung 2: Grobe Projektarchitektur

- Frontend?

3 Backend

3.1 Stufe 1 (1HP NN)

- Hauptaufgabe: Generierung einer Grundwahrheit für eine Wärmepumpe
  - Algorithmus: Nächster Punkt
  - Algorithmus: Scipy Versuche (Vergleiche: Laufzeit, Fehler)
  - Algorithmus: Unser Algorithmus (Vergleiche ...)
- Kommunikation mit dem Modell
  - Pipeline / Ablauf einer Anfrage
  - Optimierungen / Engpässe

3.1.1 Triangulation

Vorüberlegung : Um die Farbe eines Pixels in einer zweidimensionalen Ebene zufällig verteilter Datenpunkte anzunähern, empfiehlt es sich über ein Dreieck aus Datenpunkten, das den Pixel einschließt zu interpolieren. Dabei, um möglichst viel von den Informationen der Datenpunkte zu nutzen, sollten die drei Datenpunkte einen möglichst geringen Abstand zum Pixelpunkt p besitzen. Die drei Datenpunkte, die das Dreieck bilden sollen, nennen wir c, c_1 und c_2. Bevor wir einen Algorithmus 3.1.4 konstruieren, der diese Eigenschaft erfüllt erst noch wichtige Hilfssätze:

SAG MIR BESCHEID, BEVOR DU WAS LÖSCHT, FABIO

3.1.2 Hilfssatz 1: Minimales-Abstands-Dreieck beinhaltet nächsten Punkt

Damit ein solches Dreieck eine minimale Punktabstandssumme besitzt, ist es notwendig, dass der, zu p nächste, Datenpunkt Teil dieses Dreiecks ist. Diesen Punkt nennen wir c.
Beweis : Angenommen, c wäre nicht Teil des p-umschließenden Dreiecks mit minimaler Punktabstandssumme. Dieses Dreieck bestehe aus den Eckpunkten a_1, a_2 und a_3 mit Abstandssumme:

$$S_{\text{AP}} \quad = \quad |\, p - a_1 \,| + |\, p - a_2 \,| + |\, p - a_3 \,|$$

Von den drei Punkten sei a_1 der nächste Punkt zu p. Dieser spannt, mit p als Mittelpunkt, einen Kreis auf. Innerhalb davon muss c liegen. Es gibt nun einen Punkt $\in \{a_1, a_2, a_3\}$, der sich durch c ersetzen lässt und mit den anderen beiden Punkten ein Dreieck aufspannt, das immer noch p umschließt (siehe 4).

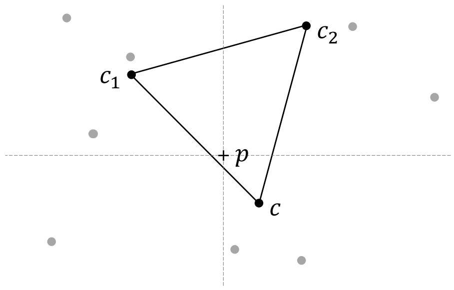

Abbildung 3: Ziel Triangulation

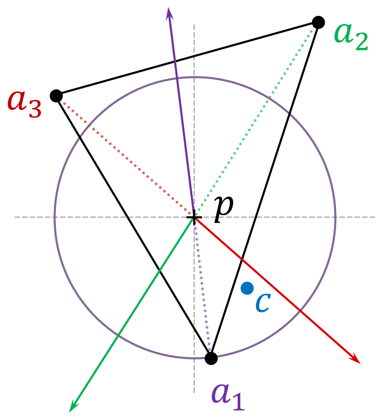

Abbildung 4: Dreiteilung mit Hilfslinien

Diesen, zu ersetzenden Punkt, kann man finden, indem man zuerst von jedem Punkt $\in \{a_1, a_2, a_3\}$ eine Linie durch p zieht (in Abb. 4 lila, grün, rot). Diese Linien, von p ausgehend (nur noch die durchgezogenen Linien), teilen die Ebene in drei Teile auf, wobei in jedem genau ein Punkt $\in \{a_1, a_2, a_3\}$ liegt. Wäre mehr als ein Punkt in einem Teil, so umschlössen die drei Punkte nicht mehr p (siehe 3.1.3).

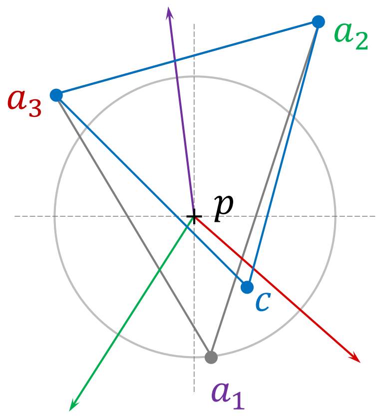

Abbildung 5: kleineres Dreieck

Sei a_1 der Punkt, der im selben Teil wie c liegt. Dieser lässt sich durch c ersetzen, da jeder Punkt innerhalb dieses Teils, ein p umschließendes Dreieck mit a_2 und a_3 aufstellen würde. Das liegt daran, dass die Mindestgröße der Winkel an jeweils a_2 und a_3 ausreicht, um p zu umschließen (siehe 3.1.3). Die Mindestgröße beider Winkel wird schließlich von dem Teil bestimmt, der a_1 beinhaltet. Das neue Dreieck (c,a_2,a_3) hat somit die Punktabstandssumme:

$$S^*_{\text{AP}} \quad = \quad |\, p - c \,| + |\, p - a_2 \,| + |\, p - a_3 \,|$$

O.B.d.A: S^*_{AP} ist kleiner als S_{AP}, da der Abstand von c zu p kleiner ist als der von a_1 zu p. Das ist ein

Widerspruch zur Annahme, dass (a_1, a_2, a_3) das Dreieck mit kleinster Punktabstandssumme ist. Daraus folgt, dass der nächste Punkt Teil dieses Dreiecks sein muss. □

3.1.3 Hilfssatz 2: Winkel-Punkteinschluss

Zwei Punkte a_2, a_3, die mit einem Punkt p eine Fläche f_{a_1} (blau, Abb. 6) hinter p projizieren, erzeugen mit jedem Punkt a_1 aus f_{a_1} ein Dreieck, das p umschließt.
Beweis : Seien a_2, a_3 und p Punkte auf einer zweidimensionalen Fläche und φ_2, φ_3 die kleineren Winkel (d.h. $\varphi_2, \varphi_3 < 90°$) zwischen den Linien $\{(a_2, a_3), (a_2, p)\}$ bzw. $\{(a_3, a_2), (a_3, p)\}$.

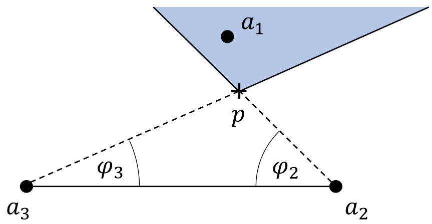

Abbildung 6: Winkel-Punkteinschluss

Um p nun in einem Dreieck mit a_2 und a_3 einzuschließen, muss der dreiecksvervollständigende Punkt a_1 so liegen, dass gilt: $\{(a_2, a_3), (a_2, a_1)\} \geqslant \varphi_2$ bzw. $\{(a_3, a_2), (a_3, a_1)\} \geqslant \varphi_3$, denn sonst würde mindestens eine Linie zwischen p und (a_3, a_2) verlaufen, was p ausschließt. Alle Punkte a_1, die $\{(a_2, a_3), (a_2, a_1)\} \geqslant \varphi_2 \wedge \{(a_3, a_2), (a_3, a_1)\} \geqslant \varphi_3$ erfüllen, liegen also per Definition in f_{a_1}. □

3.1.4 Konstruktion

Zuerst durchsuche man die Datenpunkte nach dem, an p nächstgelegenen. Dieser ist Teil des Dreiecks minimaler Punktabstandssumme und wir nennen ihn c (3.1.2). Wir teilen die Ebene mit einer Geraden, die durch c und p verläuft. Orthogonal zu dieser legen wir eine weitere Gerade, die durch p geht (in Abb. 7 rot).

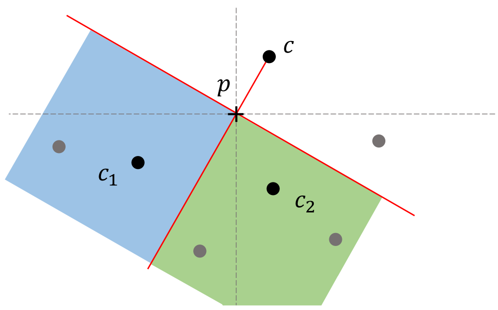

Abbildung 7: Konstruktion

Die beiden Quadranten, die nicht an c angrenzen (in Abb. 7 grün und blau), werden verwendet, um Datenpunkte c_1 bzw. c_2, in jedem der beiden Quadranten einer, zu finden, die mit c ein p-umschließendes Dreieck bilden. Dazu nimmt man in beiden Quadranten den nächsten Punkt zu c. Die Punkte c, c_1 und c_2 liefern so ein Dreieck mit einer einigermaßen kleinen Punktabstandssumme.

3.1.5 Laufzeit

Der Algorithmus läuft in linearer Laufzeit, da sich das k-kleinste Element einer unsortierten Liste in Linearzeit finden lässt.

3.1.6 Selbstkritische Beleuchtung der Konstruktion

Die Konstruktion erzeugt schlechte Ergebnisse, falls in mindestens einem der beiden Suchquadranten keine oder sehr weit von p entfernte Datenpunkte liegen, weil die an c angrenzenden Quadranten ignoriert werden, obwohl darin ebenfalls Punkte liegen könnten, die für eine kleinere Punktabstandssumme sorgen würden (siehe Abb. 7). Ein Lösungsansatz dafür bestünde, indem man beispielsweise zuerst den nächsten Punkt zu p von beiden Quadranten sucht, z.B. c_1 und mit diesem dann den möglichen Bereich für c_2 absteckt. Dies würde allerdings eine sequentielle Suche nach c_1 und c_2 verlangen, während unsere Konstruktion Parallelisierung zulässt.
Ein sehr guter Aspekt an der Abstandsminimierung des ersten Punkts c ist, dass, falls p nahe oder auf einem

Datenpunkt liegt, die Abweichung von den Messdaten sehr gering ist.
Eine weitere, hervorragende Eigenschaft ist, das sich die Konstruktion in Form eines Algorithmus mit linearer Laufzeit implementieren lässt.

3.2 Stufe 2 (2HP NN)

- Generierung einer Grundwahrheit? Warum nicht?
- Kommunikation mit dem Modell
  - Pipeline / Ablauf einer Anfrage
  - Optimierungen / Engpässe

4 Nutzeroberfläche

Unterteilung in Kinderversion und wissenschaftliche Version? Farbliche Gestaltung

4.1 Wissenschaftliche Version

- Abstriche in der Darstellung

4.2 Kinderversion

- Kinderversion: Nutzernamenvergabe
- Einführung und Vereinfachung des Themas für Kinder
- Anreize / Spielifizierung: KI Charakter
- Frage: Wie stellt man eine KI dar?

5 Diskussion

5.1 Nutzeroberfläche

5.2 Backend

5.3 Weiterführende Ideen

6 Fazit