Einleitung

Das vorliegende Dokument befasst sich mit den in dieser Arbeit verwendeten Daten. Es enthält die Python-Skripte zur Erstellung der Datenpunkte, das physikalische Modell, die Nachbearbeitung sowie das Training des MLP-Modells und die Darstellung der Ergebnisse und Validierung des Modells. Diese Skripte befinden sich im Ordner „Scripts“.

Die Skripte sind wie folgt organisiert (Bitte die Skripte wie folgende Anordnung aufrufen):

1. **Create_new.py**: Enthält das physikalische Modell und den Algorithmus zur Erzeugung der Datensätze für das MLP-Modell.

2. **Neuronal_Network_new.py**: Enthält das MLP-Modell sowie den Code für das Training und die Hyperparameter-Optimierung.

3. **Validation_new.py**: Beinhaltet den Validierungscode zur Darstellung von Kurven- und Streudiagrammen, die Analyse der Datengröße und die thermodynamische Validierung der Netzwerke.

4. **Break_Event_new.py**: Enthält den Code für die zeitliche Analyse des MLP-Modells im Vergleich zum physikalischen Modell.

Zusammenfassung:

1. Öffnen Sie zunächst **Create_new.py**, um den Gesamtdatensatz zu erzeugen. Die .pkl-Datei wird als **nn_Merged_Dataset.**pkl im Ordner **Physical_Results** gespeichert. Sie können den Parameterraum am Ende des Codes nach **if __name__ == '__main__'** definieren.

2. Öffnen Sie dann **Neuronal_Network_new.py**, um ein neuronales Netzwerk zu trainieren. Geben Sie die Adresse von **nn_Merged_Dataset.pkl** als dataset_directory im Init_Dict an. Die Ergebnisse werden im Ordner NN_Results gespeichert. Das Modell wird als .pkl-Datei mit einem Namen wie **nn_L4_N256_relu_B256_a1e-4.pkl** gespeichert.

3. Geben Sie abschließend die Adresse der Modell-Datei, z.B. nn_L4_N256_relu_B256_a1e-4.pkl, in der Variable **MLP_Model_Address** im Skript **Validation_new.py** an und die Adresse von **nn_Merged_Dataset.pkl** in der Variable Dataset_Address. Stellen Sie fest, welche Art von Analyse durchgeführt werden soll, und starten Sie anschließend das Skript Validation_new.py. Die Ergebnisse werden als Abbildungen ausgegeben.

Wichtig: ändern Sie bitte die Skriptsname nicht , da sie als Bibliothek in jeweiligen Scripten verwenden werden.

Datensatzerzeugung mit dem physikalischen Modell in Create_new.py:

Zur Erzeugung der Trainingsdaten sollte zunächst das physikalische Modell für den gegebenen Parameterraum berechnet werden. Dieser Prozess ist automatisiert. Am Ende des Skripts, nach „if **name** == '**main**'“, werden die Eingangsparameter und deren Intervalle festgelegt. Der Code nutzt den Sobol-Algorithmus zur Erzeugung des Eingangsparameterraums und löst das physikalische Modell für jeden Parameter. Wenn die Ergebnisse den in der Masterarbeit festgelegten Rahmenbedingungen entsprechen, werden die Simulationen gespeichert und ein Excel-Bericht wird erstellt.

Eingangsparameter:

- sample_n_size: Sobol-Sampling-Größe, erhöht die Anzahl der Simulationen exponentiell (maximal bis 13 verwendet).
- Input Dictionary: Weitere Eingangsparameter können hier festgelegt werden.
    - 'WF': ('Propane*Isobutane*Pentane'): Arbeitsfluid für Refprop.
    - 'SF': 'water': Sekundärfluid für Refprop.
    - random_sample: (0.1 = 10%) Stichprobengröße der zufällig bei der Datensatzerzeugung verwendeten Datenpunkte.
    - p_w: Druck des Arbeitsfluids als Intervall (Liste).
    - p_s: Druck des Sekundärfluids am Austritt.
    - D_w: Innendurchmesser des äußeren Rohrs.
    - D_s: Innendurchmesser des inneren Rohrs.
    - m_r: Massenstromverhältnis.
    - m_w: Intervall des Massenstroms des Arbeitsfluids (Liste).
    - m_s: Intervall des Massenstroms des Sekundärfluids (Liste).

- save_directory: Adresse, wo die Ergebnisse gespeichert werden.
- Save_Outputs: Boolean, ob Zwischenergebnisse gespeichert werden sollen.
- L: Maximale Länge des Wärmeübertragers.
- t: Wandstärke des Innenrohrs.
- ode_atol: Absolute Konvergenztoleranz des Lösers.
- ode_rtol: Relative Konvergenztoleranz des Lösers.
- method: Lösermethode, z.B. RK45.
- n_samples: Sobol-Sampling-Größe.
- skip_list: Liste zur Angabe, welche Batches während der Datenerzeugung übersprungen werden sollen (für Debugging, sollte leer sein).
- cache_folder: Adresse zum Speichern von Zwischenergebnissen.

Das Programm führt alle notwendigen Schritte zur Erzeugung des Gesamtdatensatzes durch, der für das neuronale Netzwerk verwendet werden kann. Es müssen lediglich die gewünschten Intervalle im Input Dictionary festgelegt werden. Das Ergebnis wird als nn_Merged_Dataset.pkl im Ordner Physical_Results gespeichert. Mehr über die Struktur dieses Datensatzes wird im Kapitel "Datensätze" ausführlich erläutert.

Einige wichtige Funktionen:

- Model = Heat_Exchanger_Model(Input_Parameter_Dictionary): Definiert ein physikalisches Modell; dieser Schritt ist erforderlich, um weitere Funktionen zu nutzen.
- Model.simulate(computing_method='parallel'): Führt Simulationen mit den definierten Parametern im Input Dictionary durch; die Rechenmethoden „parallel“ und „serial“ sind verfügbar. Für größere Parameterbereiche wird die parallele Berechnung empfohlen.
- Model.Postprocessing(): Verwendet die erzeugten Simulationsdaten (h,p-Verläufe aus solve_ivp) zur Erstellung von künstlichen Datenpunkten für den Trainingsdatensatz des MLP-Modells; führt

alle in der Masterarbeit bezeichneten Nachbearbeitungsoperationen an den Datenpunkten durch.

- Model.merge_datasets(): Fügt alle mit Postprocessing() erzeugten Datensätze zusammen, um einen einzelnen Datensatz zu erstellen.
- Model.solve(p_w, p_s, m_w, m_s, X_A, X_B, X_C, D_w, D_s): Führt eine einzelne Simulation mit den festgelegten Eingangsparametern aus.

Training und Optimierung eines MLP-Modells:

Wenn ein Trainingsdatensatz (als `.pkl`-Datei) vorhanden ist, kann basierend auf diesem Datensatz ein MLP-Modell trainiert werden. Dazu wird das Programm `Neural_Network_new.py` verwendet. Sie können entweder die Hyperparameteroptimierung mit den festgelegten Hyperparameterintervallen verwenden, um automatisch das beste Netzwerk zu trainieren, oder Sie können das neuronale Netzwerk manuell mit den im `Init_Dict` Dictionary festgelegten Hyperparametern trainieren.

Wählen Sie die gewünschte Operation, indem Sie den Boolean `Perform_Hyperparameter_Tuning` auf `True` setzen oder `Perform_Training` auf `True` setzen. Beachten Sie, dass bei der Ausführung immer zuerst die Hyperparameteroptimierung durchgeführt wird. Daher wird empfohlen, jeweils nur eine dieser Operationen auszuwählen.

Hyperparameter:

Die Hyperparameter werden im `Init_Dict` Dictionary festgelegt:

- `max_epochs`: Maximale Trainings-Epochen.
- `activation`: Aktivierungsfunktion.

- `hidden_layer`: Anzahl der verdeckten Schichten.
- `neurons`: Anzahl der Neuronen pro Schicht.
- `batch_size`: Batchgröße.
- `alpha`: L2-Regularisierungskoeffizient.
- `lr`: Lernrate.
- `atol`: Konvergenztoleranz (entspricht dem minimalen Gradient des Validierungsverlustes (RMSE), um das Training zu stoppen).
- `epoch_to_converge`: Anzahl der Epochen, um das Training zu stoppen, falls keine Verbesserung auftritt.
- `dev_set_split_ratio`: Anteil des Dev-Datensatzes für den Early Stopping Algorithmus (z.B. 0.2).
- `test_set_split_ratio`: Anteil des Testdatensatzes, der im MLP-Modell für Tests verwendet wird (z.B. 0.2).
- `verbose`: Stufe der Informationen, die in der Konsole angezeigt werden.
- `save_directory`: Adresse, wo das trainierte MLP-Modell gespeichert wird.
- `cache_directory`: Adresse zum Speichern von Zwischenergebnissen.
- `dataset_directory`: Adresse des Trainingsdatensatzes.
- `optimization_space`: Ein Dictionary, das die Hyperparameterintervalle enthält.

Hyperparameterraum:

Um eine Hyperparameteroptimierung durchzuführen, sollten die Hyperparameterintervalle festgelegt werden:

- 'HL_space`: Liste der Anzahl der verdeckten Schichten.
- `N_space`: Liste der Anzahl der Neuronen pro Schicht.
- `AF_space`: Liste der Aktivierungsfunktionen.
- `Batch_space`: Liste der Batchgrößen.
- `Alpha_space`: Liste der L2-Regularisierungsfaktoren.
- `LR_space`: Liste der Lernraten.

- `sample_size`: Anzahl der Kombinationen aus den festgelegten Intervallen für die Zufallssuche.
- `method`: Für eine Zufallssuche geben Sie 'randomized_search' an, für eine Rastersuche geben Sie einen anderen String an.

Wichtige Funktionen:

Das Programm ist bereit, die festgelegte Operation durchzuführen. Hier sind einige der wichtigsten Funktionen:

- `Neural_Network_Model(Init_Dictionary)`: Definiert ein Ersatzmodell als Klasse.
  - Beispiel: `model = Neural_Network_Model(Init_Dict)`
- `model.search()`: Führt die Hyperparameteroptimierung durch.
- `model.train()`: Trainiert das festgelegte Modell.

Validierung mit Validation_new.py:

Stellen Sie die Adresse des Datensatzes und des MLP-Modells ein. Der Code erkennt die Adresse automatisch, wenn der Ordnername der Masterarbeitsdaten nicht geändert wird und alle Ordner mit der festgelegten Anordnung verwendet werden. Stellen Sie sicher, dass die Adresse von Refprop in `os.environ["REFPROP"]` richtig festgelegt ist.

Wählen Sie dann die Art der Analyse, die auf das MLP-Modell oder den Datensatz durchgeführt werden soll, indem Sie die entsprechenden Booleans auf `True` setzen:

- `Feature_Importance_Analysis = False`: Wenn aktiviert, wird eine Feature-Importance-Analyse durchgeführt.

- `Plot_Curve_Diagrams = False`: Wenn aktiviert, werden Interpolation und Extrapolation durchgeführt.

- `Size_Analysis = False`: Wenn aktiviert, wird eine Analyse der Datensatzgröße durchgeführt.

- `Perform_Validation = False`: Wenn aktiviert, wird das Modell mit dem Testdatensatz validiert.

- `Plot_Dataset_Distribution = False`: Wenn aktiviert, wird die Datensatzverteilung dargestellt.

`TSQ_index` dient dazu, die Anzahl der Datenpunkte bei der Berechnung der Temperatur und Entropie des Testdatensatzes festzulegen.

Starten Sie `Validation_new.py`. Die Ergebnisse werden als Abbildungen ausgegeben.

Datensätze

Die aus dem physikalischen Modell erzeugten Simulationsdaten befinden sich im Ordner "1-Simulation-Ergebnisse-aus-physikalischem_Modell". In diesem Ordner gibt es verschiedene Simulationen, die mit unterschiedlichen Kombinationen von Durchmessern unter Variation von Druck, Zusammensetzungen des Arbeitsfluids und Massenströmen beider Fluide ausgeführt wurden. Eine Excel-Datei stellt die Ergebnisse der Parameterräume dar, und die Simulationsdaten (solve-ivp Ergebnisse bzw. örtlich aufgelöste h- und p-Verläufe) befinden sich im Ordner "Raw_Data". Diese Datensätze sind als Numpy-Arrays gespeichert. Die Informationen über Eingangswerte wie Durchmesser, Zusammensetzung und Massenströme können aus dem Numpy-Dateinamen abgelesen werden. Dafür habe ich einen Algorithmus im Script Create_new.py unter der Klasse Heat_Exchanger_Model als Funktion "Get_File_Name(self, variable_parameter2)" entwickelt, der bei der Datenerzeugung automatisch verwendet wird.

Die Simulationsdaten, die für den Datenerzeugungsalgorithmus verwendet wurden, sind im Ordner "2-Verwendete-Simulation-Ergebnisse-für-Datenerzeugungsalgorithmus" verfügbar. Diese Daten wurden zur Erstellung der Datensätze für das MLP-Modell verwendet.

Die Ergebnisse aus dem Datenerzeugungsalgorithmus wurden im Ordner "3-Datensätze-aus-Datenerzeugungsalgorithmus" gespeichert. Sie enthalten die Datenpunkte, die aus den Simulationsergebnissen für das MLP-Modell künstlich vermehrt wurden. Zehn Prozent dieser Datenpunkte wurden jeweils als Pandas DataFrame gespeichert.

Die Ergebnisse aus dem Datenerzeugungsalgorithmus wurden zusammengefügt, um einen vereinten Datensatz zu bilden. Dieser Datensatz ("nn_Merged_Dataset.pkl") ist im Ordner "4-Gesamtdatensatz-zum-Training" gespeichert. Dieser Datensatz ist ein "Python-Dictionary" und sollte mit der Bibliothek "Joblib" geöffnet werden. Der Gesamtdatensatz enthält zwei Schlüssel, nämlich "inputs" (Eingabewerte) und "targets" (Zielwerte). Diese Datensätze sollten skaliert werden und können zum Training eines MLP-Modells verwendet werden.

Trainierte MLP_Modelle In der Arbeit:

Als Ergebniss dieser Masterarbeit wurden 2 beste MLP-Modelle entwickelt.

1.MLP-Modell mit X_C in Features:

Eingabevektoren:[dx, h_w, p_w, m_w, h_s, p_s, m_s, x_A, x_B, x_C, d_in, d_out]

dx: gewünschte Länge von Doppelrohrwärmeübertrager , an dieses Ort wird die spezifische Enthalpie und Druck vorhergesagt, die Einheit ist in Meter.

h_w: spezifische Enthalpie von Arbeitsfluid am Eintritt in J/kg

p_w: Druck von Arbeitsfluid am Eintritt in Pa

m_w: Massenstrom von Arbeitsfluid in Kg/s

h_s: spezifische Enthalpie von Sekundärfluid am Eintritt in J/kg

p_s: Druck von Sekundärfluid am Eintritt in Pa

m_s: Massenstrom von Sekundärfluide in Kg/s

x_A: Molenbruch von Propan

x_B: Molenbruch von Isobutan

x_C: Molenbruch von Pentan

d_in: Innendurchmesser vom inneren Rohr des Wärmeübertragers

d_in: Innendurchmesser vom äußeren Rohr des Wärmeübertragers

Ausgabevektoren: [h_w_aus, p_w_aus, h_s_aus, p_s_aus]

h_w_aus: spezifische Enthalpie von Arbeitsfluid am Austritt in J/kg

p_w_aus: Druck von Arbeitsfluide am Austritt in Pa

h_w_aus: spezifische Enthalpie von Arbeitsfluid am Austritt in J/kg

p_s_aus: Druck von Sekundärfluide am Austritt in Pa

Architektur:

Anzahl der Verdeckte Schichten:4

Anzahl der Neuronen pro Schichte: 256

Aktivierungsfunktion pro Schichte : ReLu

Batch-Größe: 256

Lernrate: 1e-4

Alpha : 1e-4

Hintergrund:

Dieses Modell ist die best mögliche Netzwerk , es ist das Netzwerk die mit dem Gesamtdatensatz (Datensätze\4-Gesamtdatensatz-zum-Training\nn_Merged_Dataset.pkl") traniert wurde und in Masterarbeit beschrieben , dieses Modell ist in Rahmen einer Coarse-to-fine Studie in einer Reihe von Irrtum und versuch der Hyperparameterraum durch Zufallsuche(RandomizedSearchCV)und Rastersuche(GridSearchCV) optimiert.

Anleitung:

Dieses Modell ist innerhalb einer pkl-Datei unter Adresse "Beste Modelle\ohne_x_c_in_Features\nn_L4_N256_relu_B128_a1e-4-ohne_xc.pkl" verfügbar, Pkl-Datei ist ein "python-Dictionary" , dies sollte mithilfe von Joblib bibliothek geöffnet werden.

dieses Dictionary hat folgende Schlüsseln:

'model': mit diesem Schlüssel kann auf das trainierte MLP-Modell zugriffen werden. nutze Datei['model'].predict(skalierte_Datensatz_Eingaben) um eine Vorhersage durchführen.

'scaler_inputs': Das ist die MinMax-Skaler-Klasse , damit kann die Eingabewerte zwischen 0 und 1 skaliert werden, nutze Datei['scaler_inputs'].transform(Datensatz_Eingabe) um die Eingabevektore zu skalieren.

'scaler_Target': Das ist die MinMax-Skaler-Klasse , damit kann die Zielwerte zwischen 0 und 1 skaliert werden, nutze Datei['scaler_inputs'].transform(Datensatz_Zielwerte) um die Zielwertevektoren zu skalieren.

'test_inputs':Eingabevektoren von Testdatensatz zur Bewertung des MLP-Modells, diese datenpunkte sollten skaliert werden.

'test_targets':Zielwertvektoren von Testdatensatz zur Bewertung des MLP-Modells , diese datenpunkte sollten skaliert werden.

2.MLP-Modell ohne X_C in Features:

Eingabevektoren:[dx, h_w, p_w, m_w, h_s, p_s, m_s, x_A, x_B, d_in, d_out]

dx: gewünschte Länge von Doppelrohrwärmeübertrager , an dieses Ort wird die spezifische Enthalpie und Druck vorhergesagt, die Einheit ist in Meter.

h_w: spezifische Enthalpie von Arbeitsfluid am Eintritt in J/kg

p_w: Druck von Arbeitsfluid am Eintritt in Pa

m_w: Massenstrom von Arbeitsfluid in Kg/s

h_s: spezifische Enthalpie von Sekundärfluid am Eintritt in J/kg

p_s: Druck von Sekundärfluid am Eintritt in Pa

m_s: Massenstrom von Sekundärfluide in Kg/s

x_A: Molenbruch von Propan

x_B: Molenbruch von Isobutan

d_in: Innendurchmesser vom inneren Rohr des Wärmeübertragers

d_in: Innendurchmesser vom äußeren Rohr des Wärmeübertragers

Ausgabevektoren: [h_w_aus, p_w_aus, h_s_aus, p_s_aus]

h_w_aus: spezifische Enthalpie von Arbeitsfluid am Austritt in J/kg

p_w_aus: Druck von Arbeitsfluide am Austritt in Pa

h_w_aus: spezifische Enthalpie von Arbeitsfluid am Austritt in J/kg

p_s_aus: Druck von Sekundärfluide am Austritt in Pa

Architektur:

Anzahl der Verdeckte Schichten:4

Anzahl der Neuronen pro Schichte: 256

Aktivierungsfunktion pro Schichte : ReLu

Batch-Größe: 128

Lernrate: 1e-4

Alpha : 1e-4

Dieses Modell ist sehr ähnlich wie vorheriges Modell ,es wurde mit gleichem Datensatz trainiert die einzige unterschied ist dass vor Training und Optimierung die x_c Molebruch von Pentan aus Gesamtdatensatz ausgeschlossen. deswegen das Modell benötigt keinen Molenbruch von pentan(x_c) zu verwenden. dieses Modell wurde nach Anforderung von Frau Welp trainiert, da es die richtiges Form hat die von Aufgabestellung gefragt wurde , aber es hat schlechtere Leistung im Vergleich zum ersten Modell.