ris und die Erkundung des *Neuen Mars*

Aufgabenstellung Projektseminar Automatisierungstechnik WiSe 23/24

Version 2.0 (7. November 2023) - Betreuender WiMi: Linus Groß, M. Sc.

REGELUNGSMETHODEN & INTELLIGENTE SYSTEME

Einleitung

In der fernen Zukunft im Jahre 3024 ist die Menschheit durch die Galaxien gereist, hat zahlreiche Planeten betreten und Kolonien auf diversen neuen Welten gegründet. Doch die Suche nach einer neuen Heimat geht weiter, und schließlich erscheint ein Hoffnungsschimmer am Horizont. Nach Jahren unermüdlicher Forschung wird ein vielversprechender Planet in einem weit entfernten Sonnensystem entdeckt.

Aufgrund seiner verblüffenden Ähnlichkeit mit dem roten Planeten wird er *Neuer Mars* genannt. Eine Flotte von zwölf Mutterschiffen vom Typ RIS (Riesig-Intergalaktische Schiffe), die jeweils Hunderte von Siedlern und alle notwendigen Ressourcen für die Gründung einer neuen Kolonie an Bord haben, werden in den Kosmos geschickt, um diese neue Welt zu erkunden und zu besiedeln. Es steht viel auf dem Spiel, und die Herausforderungen sind groß, aber die Verlockung eines Neuanfangs ist zu stark, um ihr zu widerstehen.

Nachdem die Mutterschiffe, die Reise zum *Neuen Mars* durch Wurmlöcher erfoglreich absolviert haben und eine Quantenkommunikation mittels Satelliten zwischen der Planetenoberfläche und den Mutterschiffen hergestellt wurde, steht als nächstes die Erkundung des Planeten *Neuer Mars* an. [1]

Die Menschheit setzt dabei auf die schlausten Köpfe ihrer Generation, um das Problem anzugehen. Deshalb hat man sich an das r.i.s. (Research Institute for Space) und deren Mitarbeiter, die sogenannten die sogenannten Ristronauten, gewendet. Ihre Aufgabe als Forschungsgruppe des r.i.s. ist es nun, der Menschheit zu helfen, den *Neuen Mars* besiedeln zu können. Getreu dem Motto *no ris - no fun* wagen Sie sich an das Projekt.

Abbildung 1: Mutterschiffe vor dem Neuen Mars [3]

Problemstellung

Oberflächensonden haben wichtige Proben für die Beurteilung der Bewohnbarkeit des *Neuen Mars* entdeckt. Es bleibt jedoch nur noch wenig Zeit, bis die Proben verfallen. Da eine bemannte Mission von Ristronauten zur Oberfläche noch zu gefährlich ist, hat die Flotte beschlossen, hochentwickelte autonome Rover zu entsenden, um die Proben so schnell wie möglich zu bergen.

Der neuartige Rover namens *ris* (Rover in Space) ist ein sogenannter Morphing Rover. Der Morphing Rover kann seine Form radikal zwischen vier verschiedenen, vordefinierten Formen ändern. Diese Formen sind an das jeweilige Terrain angepasst, so dass ihre Geschwindigkeit von den örtlichen Gegebenheiten abhängt, z. B. könnte eine Form kugelförmig sein, um bergab zu rollen. Die Rover werden von einem neuronalen Netz gesteuert, das auf der Grundlage der erfassten Daten über das lokale Gelände und die Einsatzbedingungen sowohl die Fahrtrichtung als auch den Zeitpunkt des Morphings bestimmt.

Abbildung 2: Morphing Rover ris auf dem Neuen Mars [3]

30 Proben befinden sich in 6 verschiedenen Regionen des *Neuen Mars*, wobei 5 Proben in jeder Region liegen. Der Landeplatz des Rovers für jede einzelne Probe wurde bereits bestimmt. Angesichts der Dringlichkeit der Mission muss eine einzige optimale Rover-Konfiguration entwickelt werden. Ein Exemplar dieser Konfiguration wird zu jedem Landeplatz geschickt und macht sich auf den Weg zu einer bestimmten Probe. Wenn die Rover landen, haben die Proben nur noch 500 Minuten Zeit, bevor sie verfallen. Wenn die Rover die Proben nicht rechtzeitig erreichen oder die Region verlassen, in der eine Kommunikation mit der Flotte möglich ist, scheitern sie mit ihrer Mission. Dieses optimale Roverdesign muss sofort entwickelt werden. Ihre Aufgabe ist es, sowohl die Morphing-Formen als auch die Konfiguration des neuronalen Netzes zu entwerfen, die es dem Rover ermöglichen, optimal durch das Terrain des neuen Mars zu navigieren und die Proben zu bergen. Ihr Ziel ist es, die Anzahl der gefundenen Proben zu maximieren und die dafür benötigte Zeit zu minimieren. [2]

Aufgabe des Projektseminars

Sie erhalten topologische Karten der $N_\text{K} = 6$ Regionen, in denen die $N_\text{P} = 30$ Proben/Samples liegen, wobei auf jeder Karte 5 Proben liegen. Darüber hinaus erhalten Sie die Koordinaten der Landeplätze sowie die Standorte der Proben für jedes Bergungsszenario. Die Koordinaten des Landeplatzes und der Proben werden als eine Liste der Form

$$[X, L_x, L_y, A_x, A_y]$$

gespeichert, wobei X die Nummer der Karte ist, in der die Bergung stattfindet, L_x, L_y die Koordinaten des Landeplatzes und A_x, A_y die Koordinaten der Proben sind.

Die topologische Karte für jede Region wird durch einen Namen gekennzeichnet, z. B. `Map1`, und als 8-Bit-JPG-Datei gespeichert. Für jeden Pixel ist hierbei eine Graustufe angegeben, welche die Höhe repräsentiert, sodass sich die topologische Karte als ein Höhenprofil in Graustufen darstellen lässt. Abbildung 3 zeigt die Karte 2 als Graustufenbild und als gefärbte Höhenfelder. Im Abschnitt *Karten* wird genauer auf die topologischen Karten eingegangen.

Ein Rover wird durch zwei Hauptkomponenten definiert: die *vier verschiedenen Formen*, in die er sich verwandeln kann, sowie ein *neuronales Netz*, das entscheidet, wann er seine Form ändert und in welche Richtung er fährt.

- Die Form des Rovers bestimmt hierbei die maximal fahrbare Geschwindigkeit und ist abhängig vom Terrain, auf dem sich der Rover aktuell befindet. Hierfür wird eine Maske $F^{(i)}$, welche das optimale Terrain für die Form $i \in [0, 3]$ beschreibt, mit dem Terrain T verglichen, auf dem der Rover aktuell steht. Diese $N_\text{R} = 4$ Roverformen können in Form der Masken $F^{(i)}$ konfiguriert werden. Das Höhenprofil der topologischen Karten bestimmt hierbei das Terrain um den Rover. Eine Geschwindigkeitsfunktion

  $$\nu = V(F^{(i)}, T)$$

  liefert den Geschwindigkeitsfaktor $\nu \in [0, 1]$ am aktuellen Standpunkt des Rovers für die Form i. Die Position r zum diskreten Zeitpunkt $t \in [0, 500]$ wird wie folgt bestimmt:

  $$r(t + 1) = r(t) + \nu \cdot v_\text{max} \cdot e_\alpha(t)$$

  mit $e_\alpha(t) = [\cos(\alpha(t)), \sin(\alpha(t))]$ als Einheitsvektor, der die Fahrtrichtung $\alpha(t)$ vorgibt, sowie der Maximalgeschwindigkeit v_max.

- Das neuronale Netz

  $$\Phi(M, h, s)$$

  steuert den Rover. Als Eingang erhält das neuronale Netz u. a. einen Ausschnitt M des Terrains um die aktuelle Position des Rovers, die Aktivität der letzten Schicht des Netzes h sowie einen Zustandsvektor s, welcher Informationen über den Rover, den Landeplatz sowie die Probe enthält. Mit seinem Ausgang steuert das neuronale Netzt zum einen die Form, in welche der Rover morphen soll, als auch die Fahrtrichtung. Hierfür wird eine Winkelgeschwindigkeit ω vorgegeben und somit die Richtung

  $$\alpha(t + 1) = \alpha(t) + \omega \tag{1}$$

  des Rovers bestimmt. Die insgesamt 18642 Parameter des neuronalen Netzes werden im Entscheidungsvektor x festgehalten.

Für jede Probe werden maximal $T_\text{max} = 500$ Zeitschritte simuliert, was die maximale Zeit zum Erreichen der Probe vom Landeplatz für den Rover darstellt. Die Simulation bricht zu einem Zeitpunkt $T \in [0, T_\text{max}]$ ab, sofern die Probe erreicht wird. Wird die Karte verlassen, so bricht die Kommunikation mit dem Rover ab (R.I.S. - rest in space) und der Versuch wird als ungültig gewertet ($T = T_\text{max}$ als ungültiger Versuch). Das Ziel ist es, in minimaler Zeit eine maximale Anzahl von Proben zu erreichen. Hierfür wird für die Probe p auf der Karte k die Fitnessfunktion

$$f_{p,k}(x) = \frac{T}{T_\text{min}} \cdot \left(1 + \frac{d(x)}{d_0}\right)$$

in Abhängigkeit vom Entscheidungsvektor x aufgestellt, welche es zu minimieren gilt. Hierbei beschreibt T_min die kürzest mögliche Zeit für den Rover, das Ziel zu erreichen (beispielsweise durch das Fahren einer geraden Linie bei maximaler Geschwindigkeit), d_0 ist der ursprüngliche Abstand vom Landeplatz zum Ziel und $d(x)$ ist der Abstand des Rovers zur Probe zum Endzeitpunkt der Simulation. Die Probe gilt bei einem Abstand von $d(x) \leq 6$ als erreicht. Ihre Aufgabe als Team im Projektseminar ist es, den optimalen Entscheidungsvektor x_opt zu finden, welcher die Fitnessfunktion für alle Proben p auf allen Karten k minimiert:

$$x_\text{opt} = \arg\min_x \frac{1}{N_\text{P}} \sum_{p,k} f_{p,k}(x).$$

Die 18642 freien Parameter des neuronalen Netzes im Entscheidungsvektor x müssen optimal getunt werden. [2]

Karten

Die Eingangsbilder sind 8-Bit-JPG-Dateien hinterlegt. Also JPG-Dateien sind es RGB-Bilder, wobei jede Farbe mit 8 Bit codiert wird. Die Kartenausschnitte haben eine Breite $M_x = 2800$ px und eine Höhe von $M_y = 2800$ px für `Map1` und $M_y = 3154$ px für `Map2-6`. O. B. d. A. wird vom zweiteren Fall ausgegangen. Das Bild der Karte lässt sich somit als 3D-Matrix

$$\mathcal{M} \in \mathbb{B}^{3\times M_x \times M_y} \qquad \text{mit} \qquad \mathbb{B} = \left\{ b \in \mathbb{Z} \,|\, 0 \leq b \leq 2^8 - 1 = 255 \right\}$$

darstellen. Hierbei gilt für die Matrixeinträge $m_{i,j,k} \in \mathbb{B}$, wobei $\mathbb{B}$ den ganzzahligen 8-Bit-Wertebereich angibt. Die Matrixdimensionen bzw. die Indizes $i \in [1, 2, 3]$, $j \in [1, M_x]$, $k \in [1, M_y]$ ergeben sich aus den drei RGB-Farbkanälen, sowie der Bildbreite M_x und der Bildhöhe M_y.

Da es sich bei den gegeben Bildern um Graustufenbildern handelt, ist der Farbkanal von Rot, Grün und Blau gleich (R=G=B). Dies ergibt verschiedene Grauabstufungen von weiß (R=G=B=255) bis schwarz (R=G=B=0). Für die Bildmatrix gilt somit $m_{1,j,k} = m_{2,j,k} = m_{3,j,k}$, das Bild wird also vollständig durch einen der Farbkanäle dargestellt. Es reicht also aus, die reduzierte Matrix

$$\widetilde{\mathcal{M}} = [m_{1,i,j}] \in \mathbb{B}^{M_x \times M_y}$$

im bekannten zweidimensionalen Raum anzuschauen.

Diese Karten werden entsprechend in `Python` als 2D-Matrizen geladen. Um Sprünge in der Höhe zwischen den einzelnen Einträgen zu glätten, wird die Matrix mit einem (zufälligen) Gaußfilter geglättet. Die Matrixeinträge sind nichtmehr ganzzahlig, aber weiterhin auf dem Wertebereich $0 - 255$ beschränkt.

Zum Plotten der Karten aus den gespeicherten Matrizen, kann die Funktion `imshow` benutzt werden. Hierbei kann man das Bild als Graustufenbild angezeigen, oder auch mit weiteren Colormaps versehen. Dies ist in Abbildung 3 dargestellt. Der Urpsrung der Karten befindet sich jeweils in der linken, unteren Ecke.

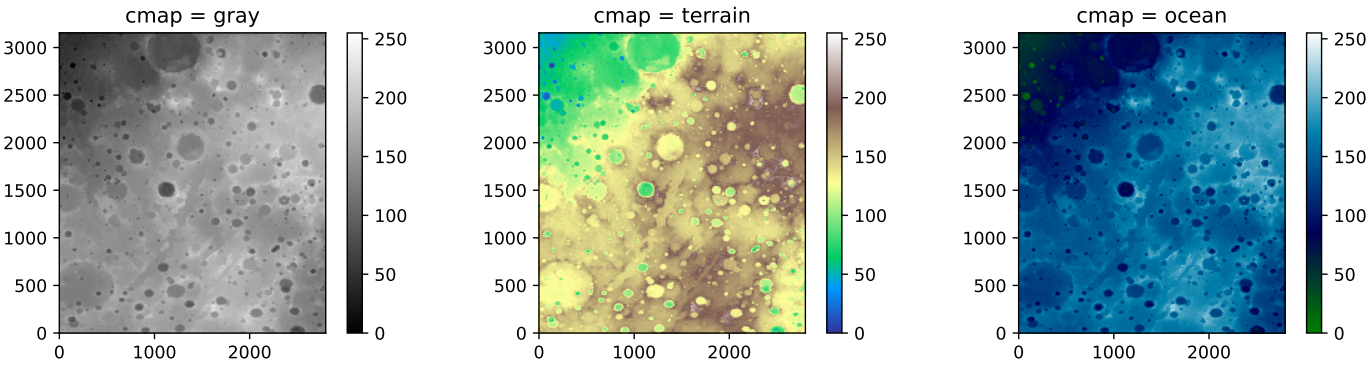

Abbildung 3: Darstellung der `Map2` in Colormaps. Der Graustufenwert $0 - 255$ wird unterschiedlich farblich codiert.

Neuronales Netz

Das neuronale Netz Φ besteht aus zwei convolutional Layern, drei dense Layern sowie einer rekurrenten Layer. Für das Arbeiten mit dem neuronalen Netz bietet es sich an, sich mit den Begrifflichkeiten vertraut zu machen. Hierfür gibt es diverse Tutorials im Internet, eine gute Anlaufstelle ist https://d2l.ai/ mit den Tutorials rund um Multilayer Perceptrons und Convolutional Neural Networks.

Das neuronale Netz Φ lässt sich als diskreter Regler interpretieren, welcher zu jedem diskreten Zeitschritt $t \in [0, T_{\text{max}}]$ aus den Eingängen M, h und s die Ausgänge m_s zur Formwandlung und ω zur Richtungsänderung berechnet.

- $M \subset \widetilde{\mathcal{M}}$ ist hierbei der 89×89 Pixel große Bildausschnitt der Karte um die aktuelle Roverposition herum. Der Bildausschnitt wird von oben genommen und nicht rotiert. Die Matrixeinträge von M (bzw. die Höheninformationen der Umgebung des Rovers) werden hierbei auf $[-1, 1]$ normiert relativ zur Höhe des Rovers (sprich der Höhe in der mitte des Bildausschnitts). Es handelt sich also bei M um eine 89×89 Matrix mit Einträgen $\in [-1, 1]$.
- Der Zustand/State s enthält mit seinen 9 Einträgen die Informationen des Rovers und seiner Position relativ zur einzusammelnden Probe. Hierbei teilt sich der Zustand wie folgt auf:

$$s = \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \\ s_{6,\dots,9} \end{bmatrix} = \begin{bmatrix} \text{Effizienz des aktuellen Rovermodus} \mathrel{\widehat{=}} \text{Geschwindigkeitsfaktor } \nu \\ \text{normierte Abklingzeit zum Morphen} \in [0, 1] \\ \text{normierte Winkeldifferenz der Roverausrichtung zur Probe} \\ \text{normierte Distanz des Rovers zur Probe} \\ \text{Ausrichtung des Rovers} \\ \text{aktueller Morphzustand als One-Hot-Codierung} \end{bmatrix} .$$

- Mit $h = x_{\text{lat}}$ wird der Ausgang von h_3 bezeichnet. Dieser wird über das rekurente Layer als Rückkopplung einen Zeitschritt später zurückgeführt.
- Mit dem Ausgang m_s gibt der Rover den Befehl zum Morphen. Gilt $m_s > 0$ und ist die Abklingzeit abgeklungen, so wird im nächsten Zeitschritt zur bestmöglichen/schnellsten Roverform gemorpht

$$i_{\text{best}} = \arg\max_i V(F^{(i)}, T) \quad .$$

  Nach einem Morphvorgang wird die Abklingzeit neu gesetzt, welche vor einem erneuten Morphvorgang abgewartet werden muss.
- Mit dem Ausgang $\omega \in [-\frac{\pi}{4}, \frac{\pi}{4}]$ wird die Winkeländerung des Rovers gesteuert und somit seine Orientierung. Der Winkel des Rovers wird nach Gleichung (1) aktualisiert.

Das neuronale Netz besteht aus insgesamt 6 Layern mit 146 Bias-Parametern und 18642 Gewichtungsparamter. Eine Auflistung der Parameter befindet sich auf der ESA-Website [2]. Zusätzlich gilt es 7 Hyperparameter, nämlich $a_{1,\dots,5}$ zur Auswahl der Aktivierungsfunktionen und $p_{1,2}$ zur Auswahl der Poolingfunktionen der convolutional Layer. Abbildung 4 zeigt die Übersicht über das neuronale Netz.

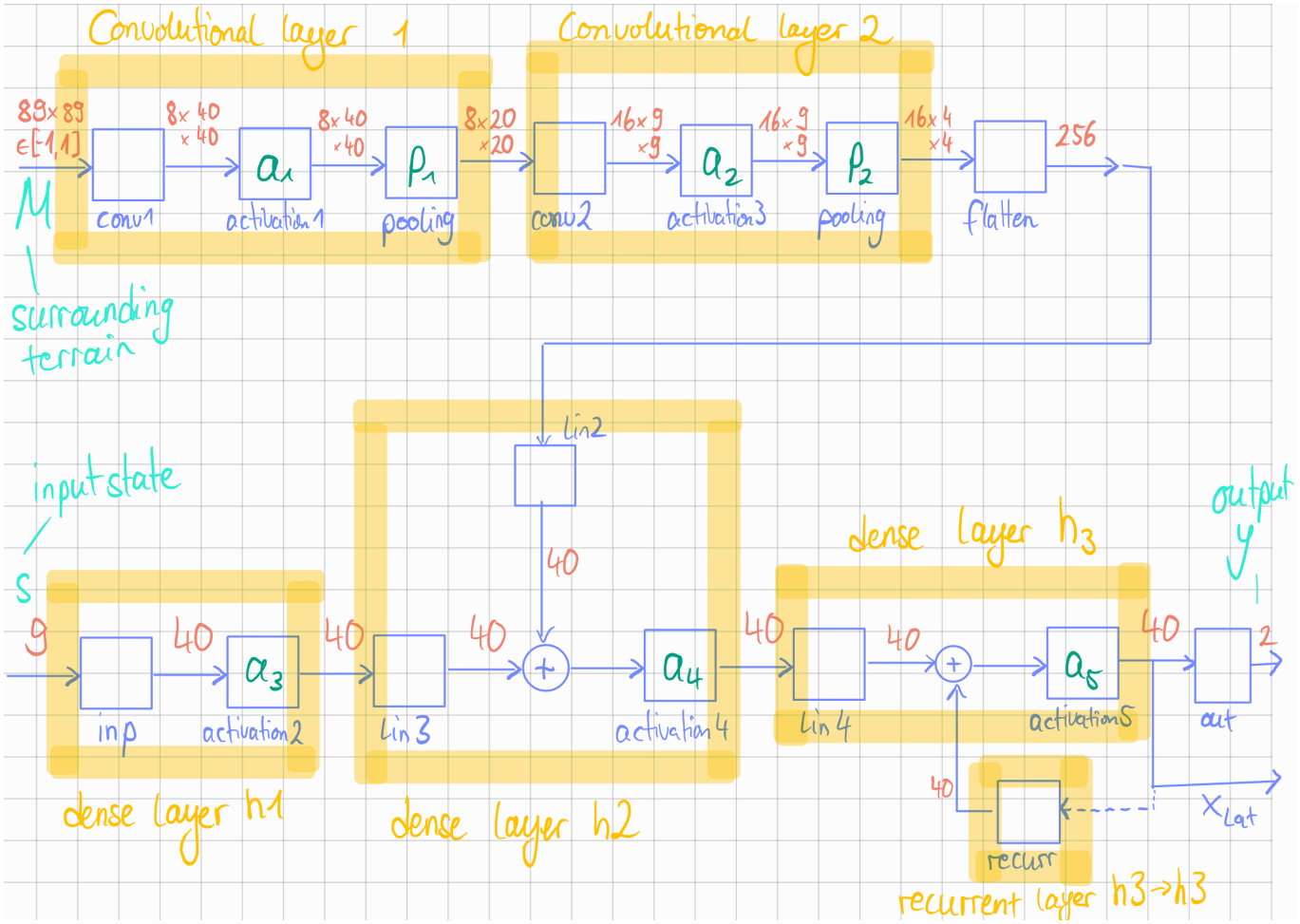

Abbildung 4: Aufbau des neuronalen Netzes. Als **Eingang** bekommt das Netz den **Kartenausschnitt** M sowie den aktuellen **Zustand** s. Dieser läuft zunächst durch das **dense Layer** h_1 in das **dense Layer** h_2. Der Kartenausschnitt durchläuft hierbei die zwei **convolutional Layer**, bevor er im **dense Layer** h_2 dazuaddiert wird. Anschließend folgt das **dense Layer** h_3, wobei hier der rekurrente Anteil des Netzes auftritt. Hierfür wird der letzte Ausgang $h = x_{\text{lat}}$ gespeichert und über das **rekurente Layer** zurückgeführt. Das Ausgangslayer bildet letzendlich den **Ausgang**, bestehend aus m_s und ω. Dargestellt sind zudem die **Dimensionen** der einzelnen Layer sowie die einstellbaren **Hyperparameter** mit den Poolingfunktionen $p_{1,2}$ und den Aktivierungsfunktionen $a_{1,\dots,5}$.

Jedes dense Layer besteht aus dem eigentlichen neuronalen Netz (inp, lin2, lin3, lin4, recurr), sowie einer anschließenden Aktivierungsfunktion[1]. Hierbei stehen verschiedene Aktivierungsfunktionen zur Auswahl (Sigmoid, ReLU, ...). Die Anzahl an zu wählenden Gewichten ergibt sich aus Eingangsdimensionen $\times$ Ausgangsdimesion, die Anzahl an zu wählenden Bias wird durch die Ausgangsdimensionen bestimmt. Tabelle 1 listet die Parameter für die dense Layer auf.

Tabelle 1: Paramter für das Neuronale Netz

Layer	Netz	Eingang	Ausgang	Anzahl Gewichte	Anzahl Bias
dense Layer h_1	inp	9	40	360	40
dense Layer h_2	lin2	256	40	10240	*None*
	lin3	40	40	1600	40
dense Layer h_3	lin4	40	40	1600	40
	recurr	40	40	1600	*None*
Ausgang	output	40	2	80	2
convolutional Layer 1	conv1	89×89	$8 \times 20 \times 20$	968	8
convolutional Layer 2	conv2	$8 \times 20 \times 20$	$16 \times 4 \times 4$	2048	16

Jede Convolutinoal Layer besteht aus drei Schritten: der eigentlichen Convolution, einer Aktivierungsfunktion sowie einem anschließenden Pooling. Es wird kein Padding verwendet, die Schrittweite beim Stride[2] beträgt 2. Das Pooling nutzt Fenster der Größe 2×2 und es kann über $p_{1,2}$ zwischen `AveragePooling` und `MaxPooling` gewählt werden. In Abbildung 5 und Abbildung 6 sind die verwendeten convolutional Layer im Detail beschrieben.

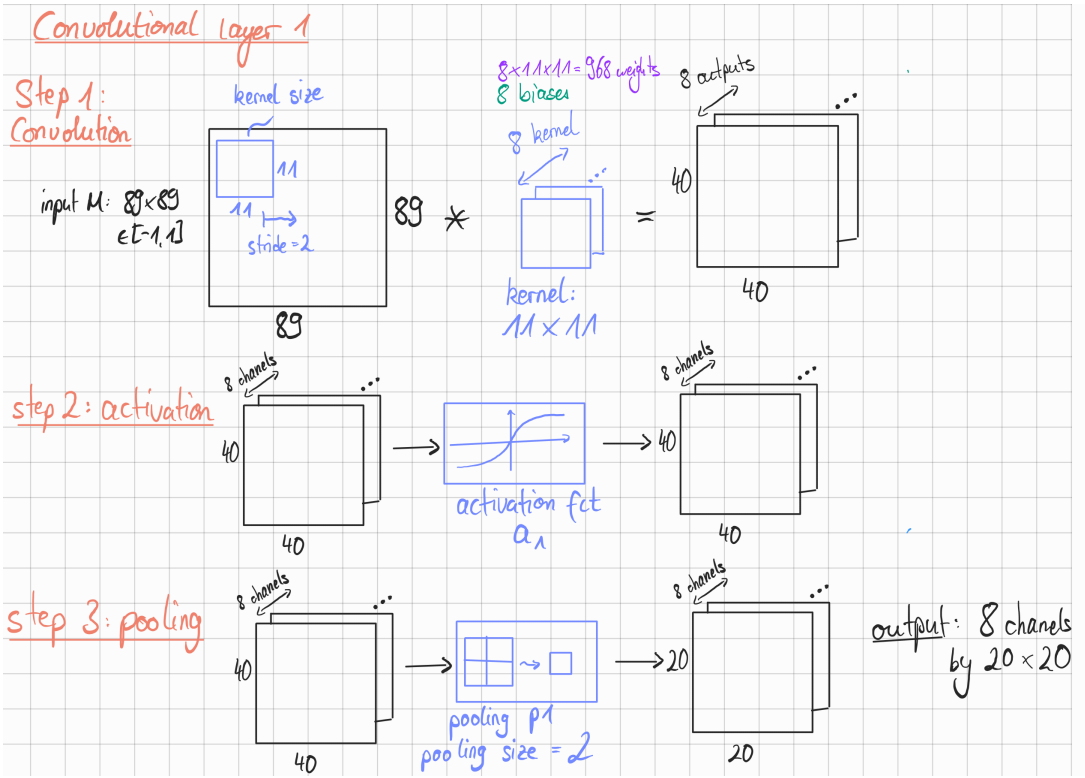

Abbildung 5: **Convolutional Layer 1**. Eingang ist der 89×89 große **Bildausschnitt M**, auf dem der Rover steht. Auf diesen Bildausschnitt werden insgesamt **8 Filter/Kernel** der Kernelgröße 11×11 angewandt. Dadurch ergeben sich die 968 Gewichte und 8 Biase des Layers. Durch den Stride von 2 (der Kernel bewegt sich immer um 2 Einträge zur Seite und nach Unten) ergibt sich somit der Ausgang der Dimension 40×40 mit insgesamt 8 Channels. Anschließend wird jeder Eintrag durch die **Aktivierungsfunktion** geschoben und es folgt das **Pooling** mit einem 2×2 Fenster. Der **Ausgang** besitzt somit eine Größe von 20×20 mit 8 Channels.

[1] **Achtung**: Im Beschreibungstext wird a_2 als Aktivierungsfunktion für das convolutional Layer 2 beschrieben und a_3 für das dense Layer h_1. Im Code sind diese beiden Aktivierungsfunktionen vertauscht!

[2] Eine schöne Visualisierung von Padding und Stride findet sich hier.

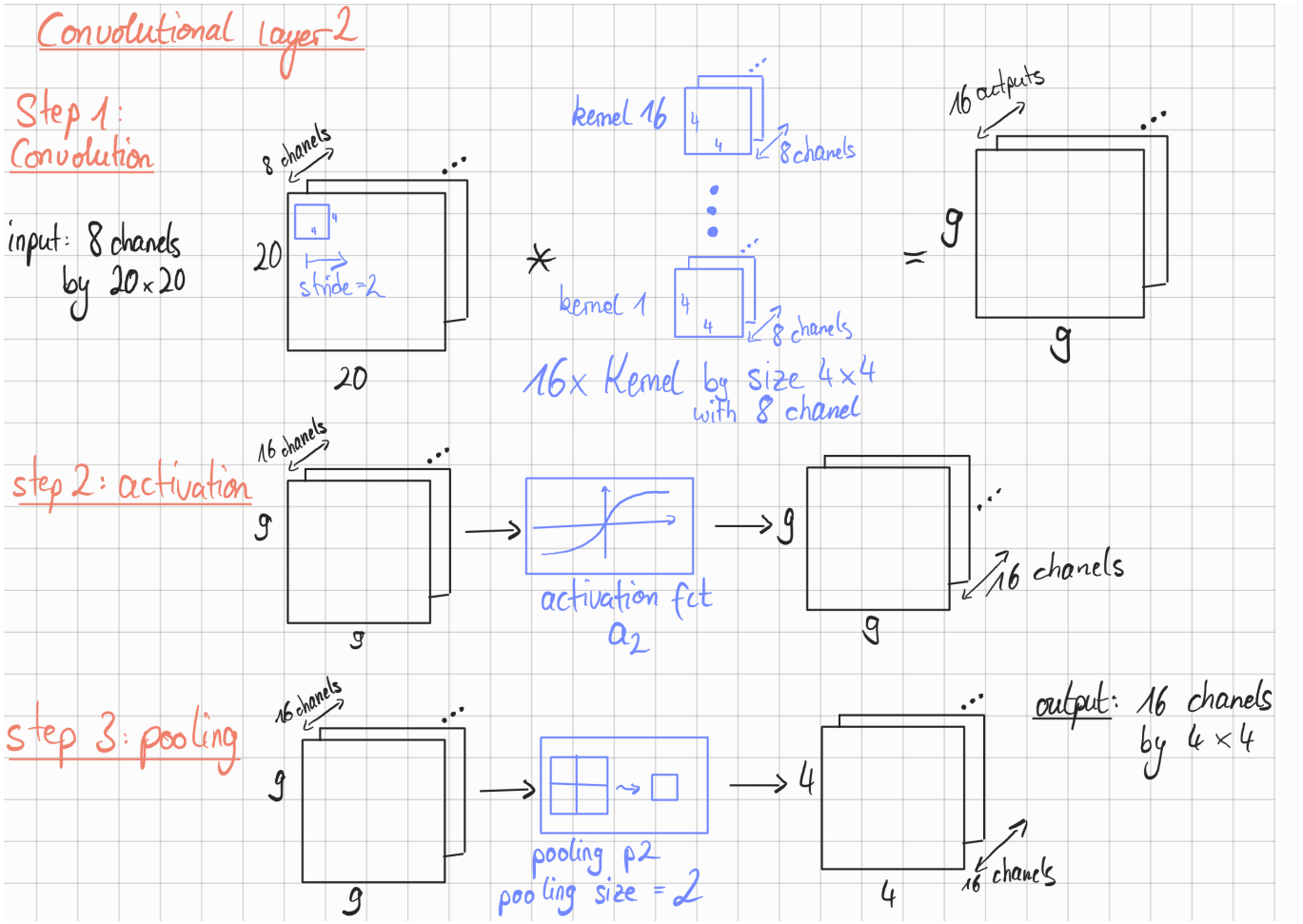

Abbildung 6: Convolutional Layer 2. **Eingang** ist der Ausgang des ersten Layer (20×20 mit 8 Channels). Auf diesen Eingang werden insgesamt **16 Filter/Kernel** der Kernelgröße 4×4 mit jeweils 8 Channels angewandt. Diese Kernel ergeben die 2048 Gewichte und 16 Biase des Layers. Mit dem Stride von 2 ergeben sich somit 16 Channel der Größe 9×9. Nach der **Aktivierungsfunktion** und dem **Pooling** mit 2×2 Fenster ergibt sich als der **Ausgang** mit einer Größe von 4×4 mit 16 Channels. Dieser wird abschließend noch zu einem Vektor aufeinandergestapelt (flatten), um als Eingang in das Netz h_2 zu fungieren.

Code

Der Code für die Aufgabenstellung kann über GitLab heruntergeladen werden. Hierfür kann sich auf GitLab mit der TU-ID eingeloggt werden. Das zugehörige Repository kann dort geklont werden, hier wurden schon einige Änderungen vorgenommen. Alternativ kann man das Repository auch aus [1] nehmen.

Für die Codeverwaltung bietet sich an, dass ihr in eurer Gruppe Git nutzt. Eine gute GUI für Git ist SmartGit. Mit der TU-Mailadresse kann man sich konstenlos eine Non-Commercial License besorgen, indem man sich hier unter `Educational Institution` mit der TU-Mailadresse registriert. Man erhält eine Lizenzdatei zugeschickt, welche man anschließend in SmartGit hinterlegen kann.

Eine gute IDE für die Programmierung von Python ist PyCharm. Auch hier bekommt man eine kostenlose Vollversion mit der TU-Mailadresse. Die Gratisversion ist aber ausreichend.

Macht euch zunächst mit dem Code sowie den einzelnen Programmteilen vertraut. Die Aufgabenstellung, die Beschreibung des Rovers sowie die Simulation ist soweit schon programmiert und durchkommentiert. Insbesondere die Simulation bei der Auswertung der Fitnessfunktion nimmt etwas Zeit in Anspruch. Mit dem gegeben Codegerüst sollte sich folgender Code direkt ausführen lassen. Als Ergebnis sollten die in Abbildung 7 dargestellten Plots erstellt werden. Diese zeigen die Trajektorien des Rovers auf der Suche nach den 30 Proben auf den 6 Kartenausschnitten. Für die Ausführung müssen `Python 3.8.8` sowie einige Libraries[3] installliert sein.

[3] `pytorch >= 1.13.1`, `torchvision >= 0.14.1`, `matplotlib >= 3.6.3`, `numpy >= 1.24.2`, `imageio >= 2.25.1`

```
### Custom Code
##################################

# define the UDP (User Defined Problem)
udp = morphing_rover_UDP()  # define the given UDP

x = udp.example() # load the example rover in ./data/example_rover.npy
f = udp.fitness(x) # calculates the fitness score for the chromosome x by simulating all scenarios
print(f)

udp.plot(x) # plot the results of the 4 rover masks, the trajectories on all samples
udp.pretty(x) # print the fitness for all scenarios
```

Weitere Informationen

Die von der ESA stammende Aufgabenstellung ist zu finden unter

- [1] https://github.com/esa/SpOC2 und
- [2] https://optimise.esa.int/challenge/spoc-2-morphing-rovers/About

sowie den weiterführenden Links. Die Bilder [3] entstammen https://hotpot.ai/.

Die Lehrveranstaltung ist als Wettbewerb organisiert, in dem kleine Projektgruppen (4 Teilnehmer) jeweils das gleiche Thema bearbeiten. Die erarbeiteten Lösungen werden hierbei im Rahmen der Abschlusspräsentationen miteinander verglichen. Der Wettbewerb besteht aus 3 Teilaufgaben:

- **Aufgabe 1 - Optimierte Bergung 30 Proben:** In den von der ESA vorgegeben Karten sowie Koordinaten der Proben + Landeplätze den besten Score erzielen. Die Lösung wird hierbei auf der ESA-Website hochgeladen. Jedes Team wählt einen *Teamnamen* und lädt die Ergebnisse unter dem Namen **TUDa_Space_*Teamname*** auf der ESA-Website hoch.
- **Aufgabe 2 - Verallgemeinerte Bergung auf einer Karte:** Für eine vorgegebene topografische Karte (diese wird noch bekannt gegeben) muss eine optimale Roverkonfiguration gefunden werden. Auf dieser Karte werden dann n noch unbekannte Koordinaten von Proben und Landeplätzen vorgegeben, welche der Rover erreichen muss. Abgegeben wird eine Roverkonfiguration, die Proben und Landeplätze werden erst zum Wettbewerb veröffentlicht. Der Rover sollte also in der Lage sein, für eine zufällige Auswahl von Probe und Landeplätzen das Ziel zu erreichen.
- **Aufgabe 3: Visualisierung:** Es soll eine GUI erstellt werden, mit der die Lösungen visualisiert werden. Hierbei soll zwischen Aufgabe 1 und Aufgabe 2 ausgewählt werden können. Die GUI soll hierbei als Teil der Abschlusspräsentation vorgestellt werden. Hierbei könnt ihr eurer Kreativität freien Lauf lassen, was genau mit der GUI angezeigt wird. (Einige Ideen: schwenkbare und hereinzoombare Topologische Karten in 2D- oder 3D-Ansicht, zeitliche Trajektorie des Rovers auf der Karte, Visualisierung der Roverformen, weitere Informationen des Rovers, etc.)

Hierbei gibt es während des Semesters festgelegte Termine, zu denen die Gruppen ihre Lösungen auf der ESA-Website hochladen müssen. Dies dient dazu, dass die Gruppen ihre Leistungen und den Fortschritt während des Semesters vergleichen können.

Es gibt keine individuelle fachliche Betreuung für die einzelnen Gruppen. Die Lösungsansätze müssen selbst erarbeitet werden. Organisatorische Fragen werden natürlich beantwortet. Pro Gruppe ist eine 10 seitige Ausarbeitung anzufertigen. Nähere Angaben zum Inhalt und der Struktur der Ausarbeitung werden während der Projektarbeit gegeben.

Das diesjährige Projektseminar wird vom Schreibcenter begleitet. Diese hilft euch beim Erstellen der wissenschaftlichen Ausarbeitung. Hierfür wird an das Schreibcenter während des Semesters eine verpflichtende, 5-seitige Vorabgabe der Ausarbeitung abgegeben und ihr erhaltet wertvolle Tipps und Rückmeldungen. Ziel ist es, die Qualität der Ausarbeitungen zu verbessern. Weitere Informationen erhaltet ihr im Laufe des Semesters.

Bitte beachtet, dass wir uns kleinere Änderungen der Aufgabenstellung vorbehalten. Schaut, dass ihr stets die aktuelle Version der Aufgabenstellung verwendet. Ansprechpartner:

- Linus Groß, M.Sc. - Kontaktseite
- linus.gross@tu-darmstadt.de

- Organisatorische Informationen während des Semesters: Moodle-Forum
- Weitere Infos: https://www.etit.tu-darmstadt.de/ris/lehre_ris/lehrveranstaltungen_ris/pro_auto_ris/index.de.jsp

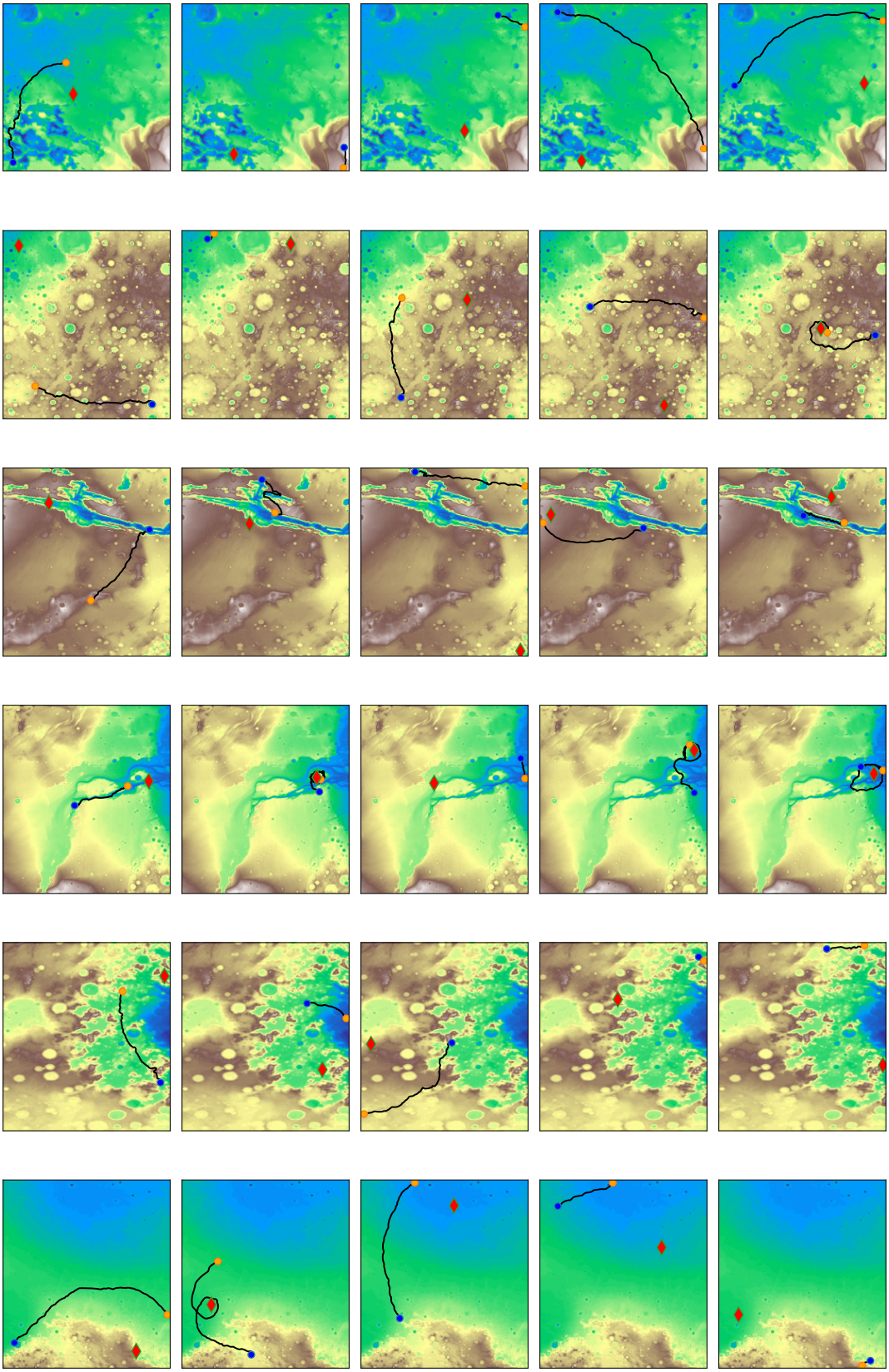

Abbildung 7: Trajektorien der Rover auf den 6 Kartenausschnitten, mit den Startpositionen (blau), Endpositionen (orange) sowie den Proben (rot)