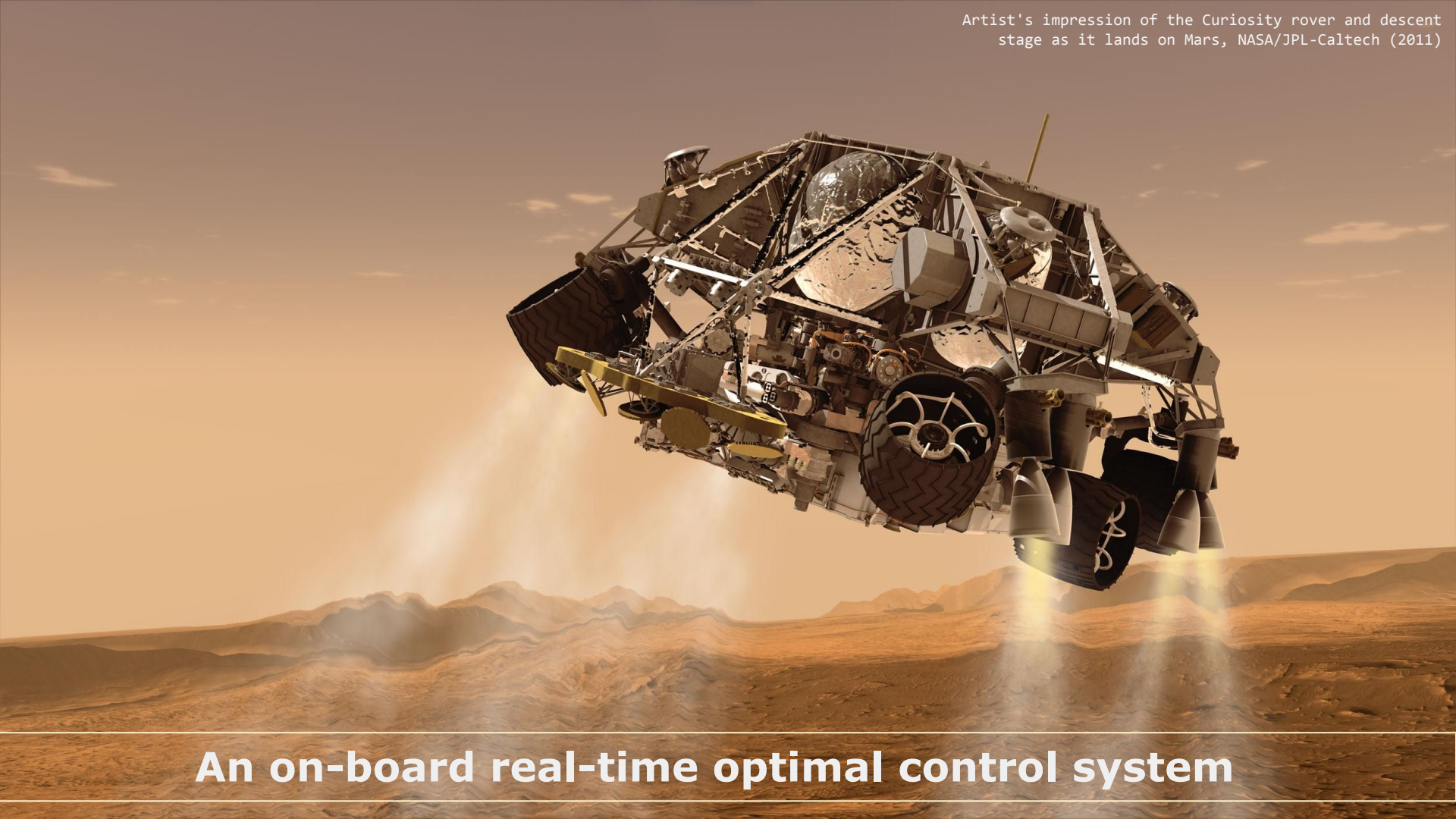
Artist's impression of the Phoenix spacecraft as it lands on Mars, NASA/JPL/Corby Waste (2007)

Optimal real-time landing using deep networks

Carlos Sánchez and Dario Izzo
Advanced Concepts Team (ESA)

Daniel Hennes
Robotics Innovation Center (DFKI)

6th International Conference on
Astrodynamics Tools and Techniques
(ICATT)



Artist's impression of the Curiosity rover and descent stage as it lands on Mars, NASA/JPL-Caltech (2011)

An on-board real-time optimal control system

Optimal control problem

Goal: Solve the deterministic continuous-time optimal control problem, that is:

$$\mathbf{u}^*(\mathbf{x}) = \text{argmin}_{\mathbf{u}} \{\mathcal{L}(\mathbf{x}, \mathbf{u}) + f(\mathbf{x}, \mathbf{u}) \cdot \nabla v(\mathbf{x})\}$$

Hamilton-Jacobi-Bellman equation

Current methods (direct or indirect) are not suitable for real-time on-board implementation, an alternative is to correct the deviations from a precomputed profile

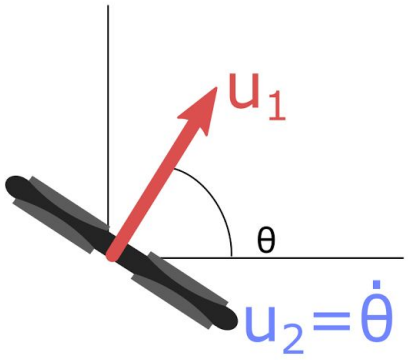
Our approach:

1. Pre-compute many optimal trajectories
2. Train an artificial neural network to approximate the optimal behaviour (supervised learning)
3. Use the network to drive the spacecraft

Landing models

Multicopter
(Earth pinpoint landing)

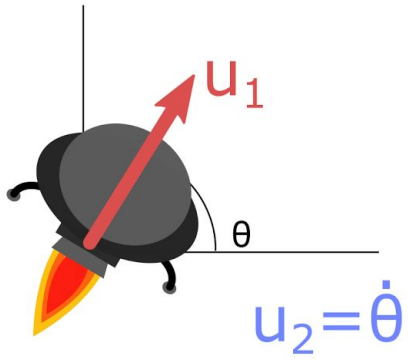
in $[x, v_x, z, z_x, \theta]$
out $[u_1, u_2]$



- Fixed mass
- Thrust + Torque
- Optimize power & time

Spacecraft I
(Moon free landing)

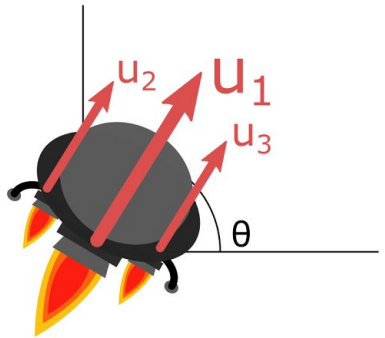
in $[m, x, v_x, z, z_x, \theta]$
out $[u_1, u_2]$



- Variable mass
- Thrust + exchange momentum wheel
- Optimize mass

Spacecraft II
(Moon free landing)

in $[m, x, v_x, z, v_z, \theta, v_\theta]$
out $[u_1, u_2, u_3]$

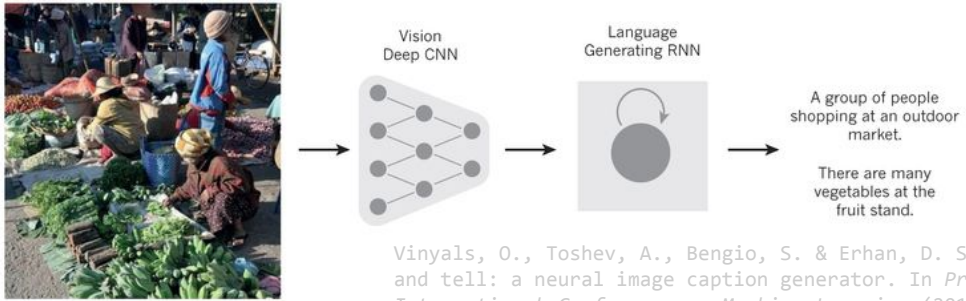


- Variable mass
- Main thrust + two lateral engines
- Optimize mass

Recent success of Deep Neural Networks

Generate captions for images

(Convolutional and recurrent neural networks)



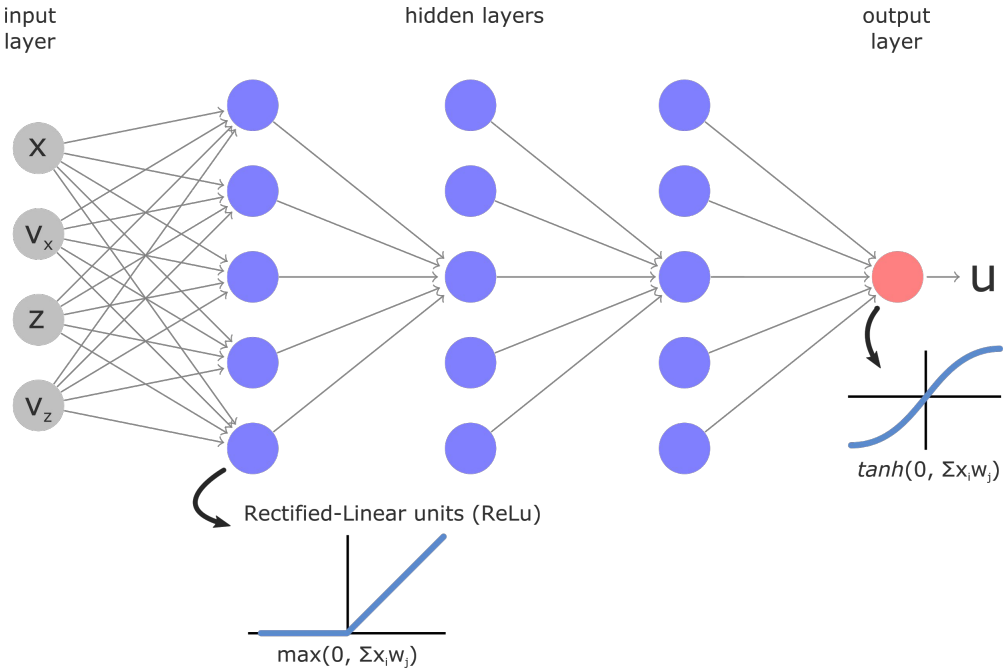
Vinyals, O., Toshev, A., Bengio, S. & Erhan, D. Show and tell: a neural image caption generator. In *Proc. International Conference on Machine Learning* (2015)

Beat the world's best Go player

(Convolutional neural networks and Monte Carlo tree search)

Silver, David, et al. Mastering the game of Go with deep neural networks and tree search. In *Nature* (2016)

Methodology



- Networks with 1 - 4 hidden layers
- **Stochastic Gradient Descent** (and momentum)

$$v_i \to v_i' = \mu v_i - \eta \frac{\partial C}{\partial w_i}$$

$$w_i \to w_i' = w_i + v_i'$$

- Minimize the squared loss error (C)
- We integrate over time the dynamics to get the full DNN-driven trajectory

Training data generation

- The training data is generated using the **Hermite-Simpson transcription** and a **non-linear programming** (NLP) solver

- Chattering effects in the training data have a huge negative impact on the results. Regularization techniques are used to remove them, although the use of a direct method would alleviate this problem resulting in cleaner data.

- **150,000 trajectories** are generated for each one of the problems (9,000,000 - 15,000,000 data points)

Results

- High success rates (landing without crash)
  - 100% from the training area (except Sp II)
  - High values from outside

- The results of the value function are always close to the optimal ones

Multicopter - Power

Training area	Outside (5m)	Outside (10m)	Time
0.47% (100%)	0.73% (100%)	2.28% / (64%)	9.75%

Multicopter - Time

Training area	Outside (5m)	Outside (10m)	Power
0.41% (100%)	2.12% (70%)	-	14.92%

Spacecraft model 1 - Mass

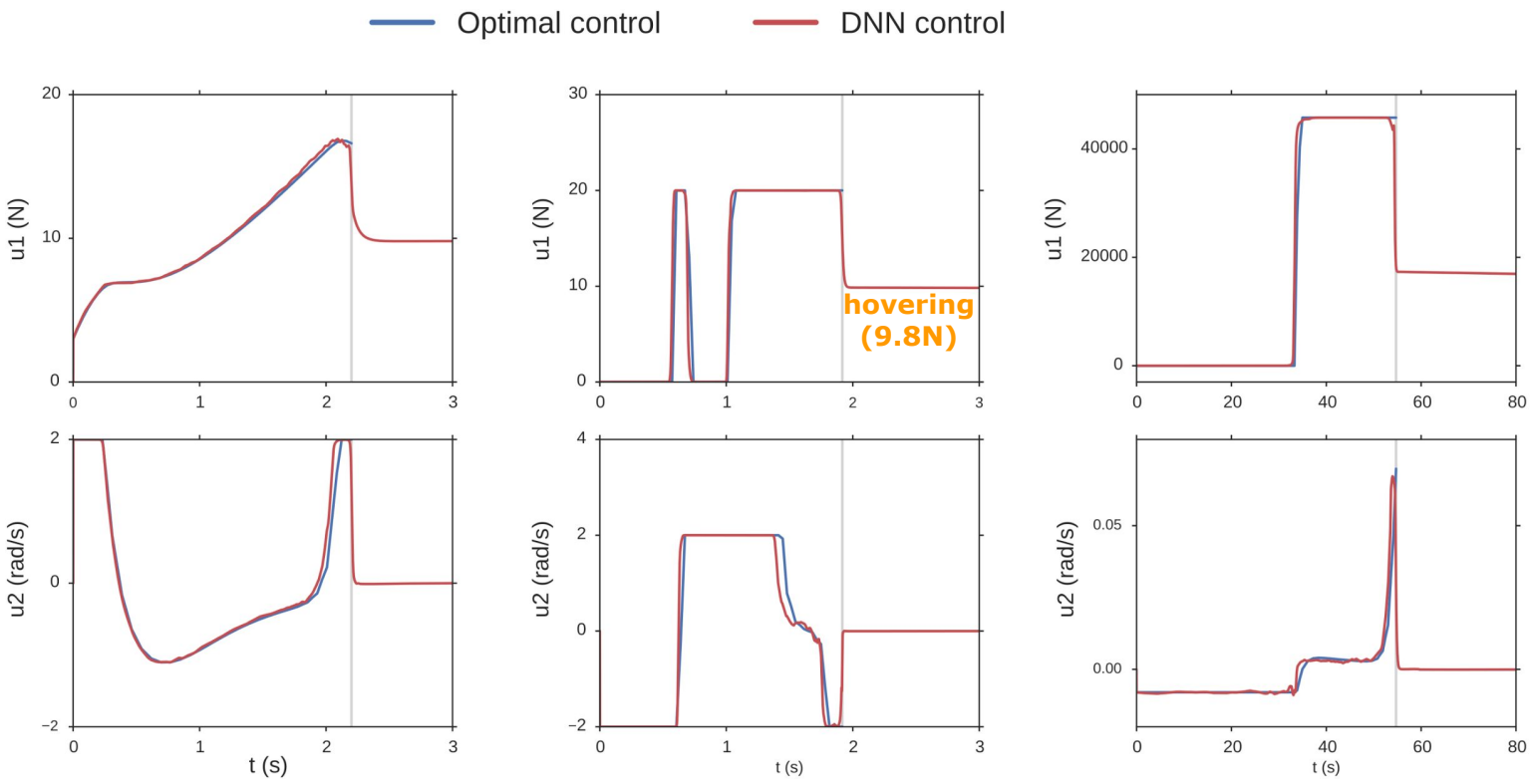
Training area	Outside (1km)	Outside (2km)	Time
1.50% (100%)	1.07% (100%)	1.83% (53%)	4.46%

Spacecraft model 2 - Mass

Training area	Outside (1km)	Outside (2km)
1.44% (70%/25%)	1.33% (54%/24%)	- (25%/1%)%

Error with respect to optimal value and success rate for 100 randomly initialized trajectories

Results

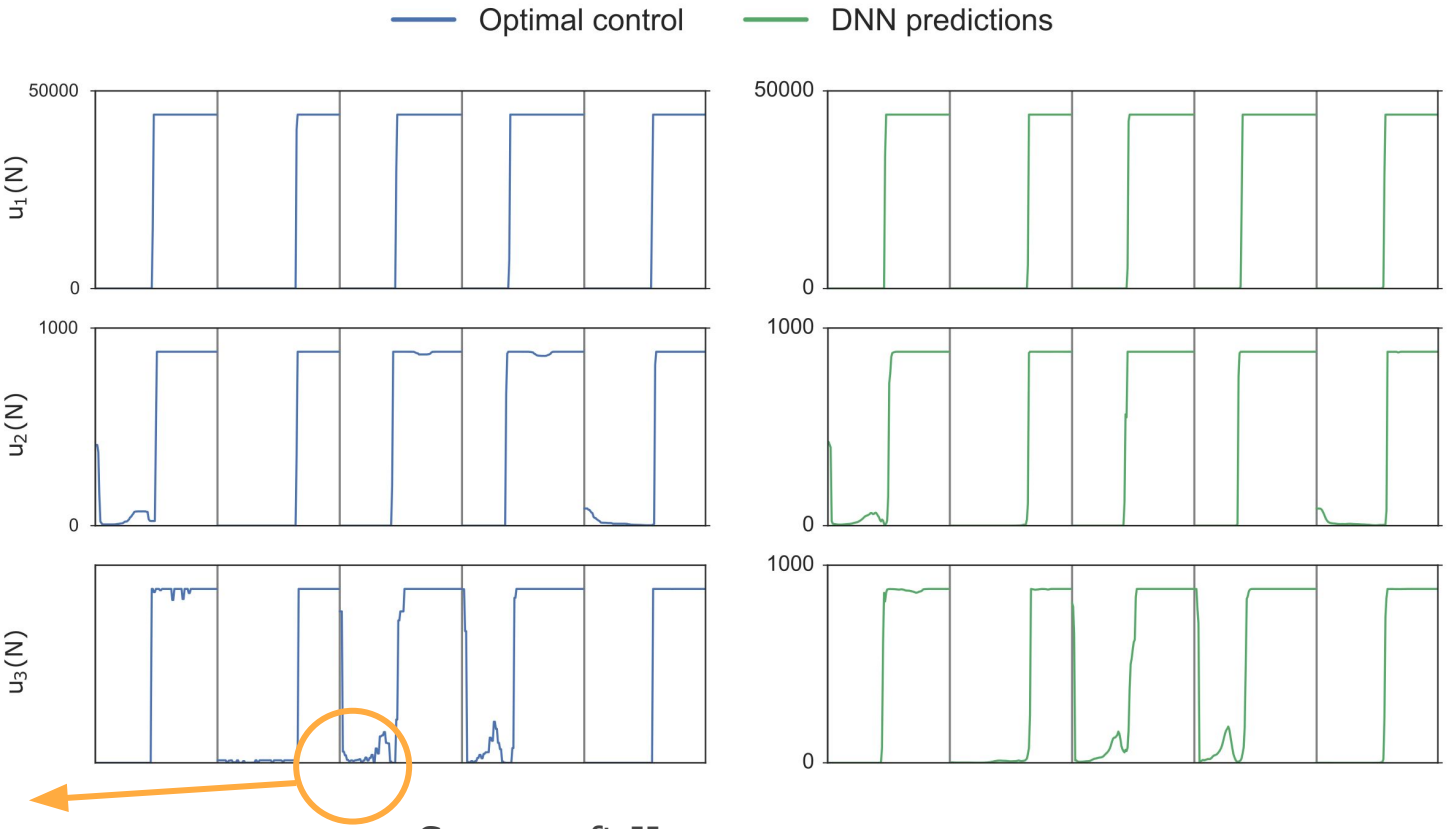


Quadrotor (power)

Quadrotor (time)

Spacecraft I

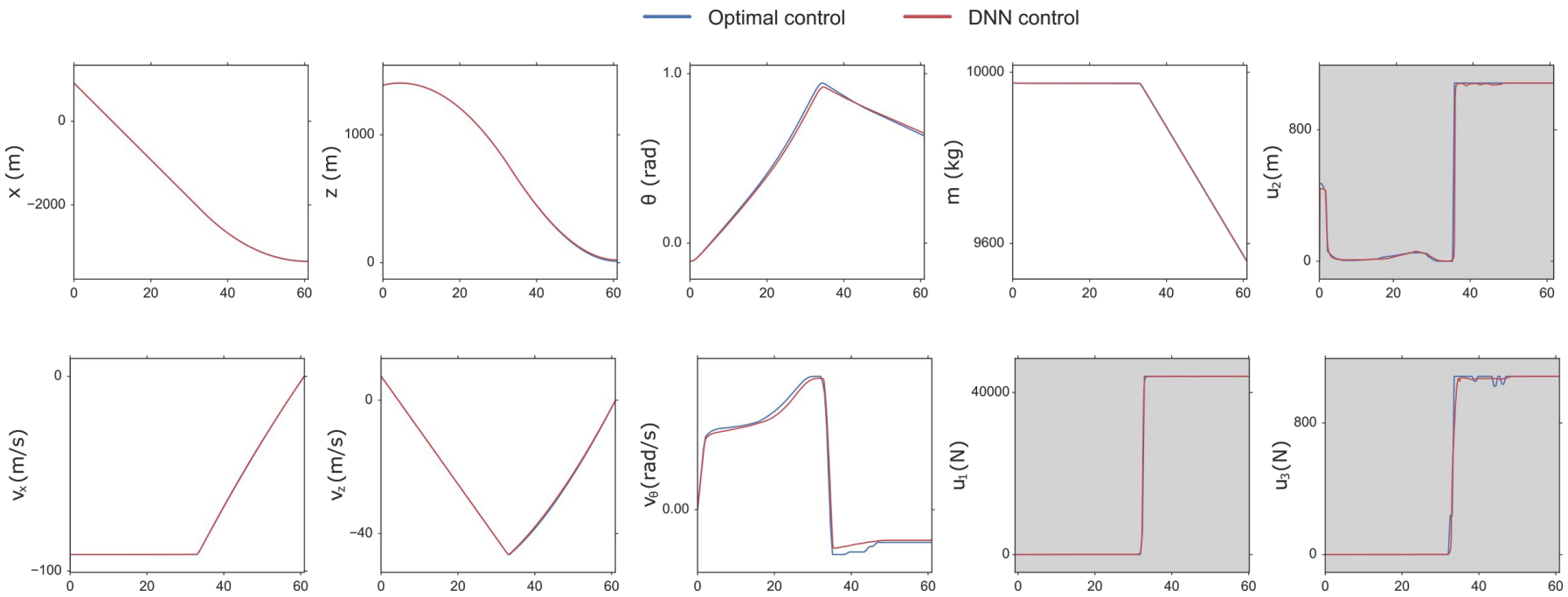
Results



Noisy training data due to the indirect method. We expect a direct method to considerably improve the results.

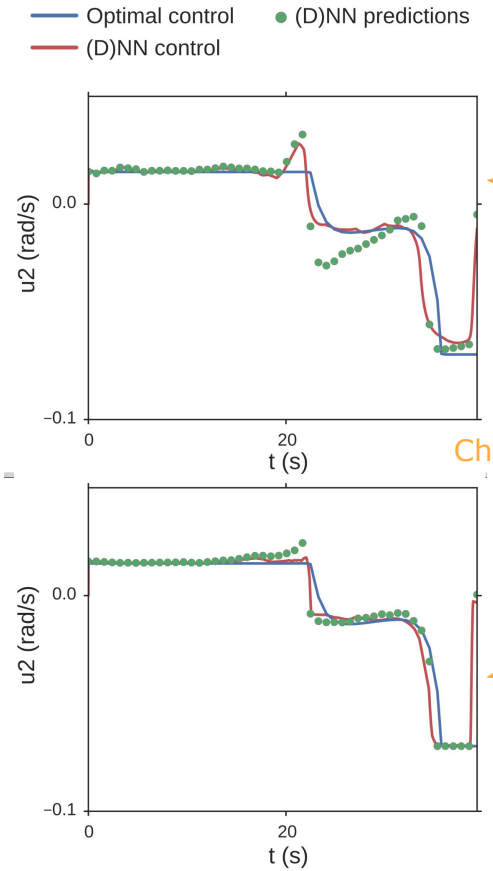
Spacecraft II

Results



Spacecraft II

Shallow vs Deep Networks



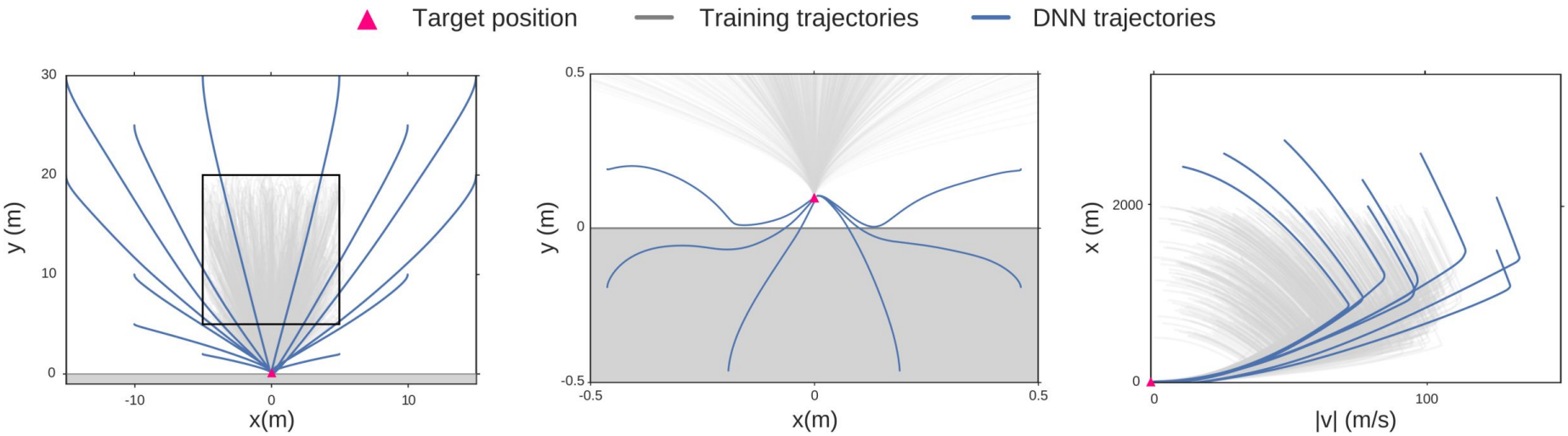
Challenging landing example

Spacecraft model 1 - Mass

	Layers	Units	#params	Train	Val.
u_2	2	256	2,048	0,00462	0,00460
	2	1,104	8,834	0,00379	0,00379
	2	2,048	16,384	0,00370	0,00371
	3	64	4,672	0,00307	0,00307
	3	128	17,536	0,00270	0,00272
	3	256	67,840	0,00263	0,00264
	4	64	8,832	0,00250	0,00252
	4	128	34,048	0,00241	0,00242
	5	64	12,992	0,00236	0,00237

- Deep networks are always better than shallow networks with the same number of parameters.

Generalization



Quadrotor (power) | Quadrotor (power) | Spacecraft I

- Successful landings from states outside of the training initial conditions
- This suggest that an approximation to the solution of the HJB equations is learned

NLP vs DNN

Non linear programming (NLP) solver

	5 nodes	20 nodes	50 nodes
Qt	38.86 ms	168.71 ms	594.87 ms
Qp	32.60 ms	222.58 ms	1245.98 ms
S1	82.82 ms	280.70 ms	1830.57 ms
S2	72.95 ms	386.44 ms	3488.82 ms

DNN (network for each variable)

1 - 1140	4 - 64	4 - 128	4 - 256
0.048 ms	0.056 ms	0.068 ms	0.11 ms

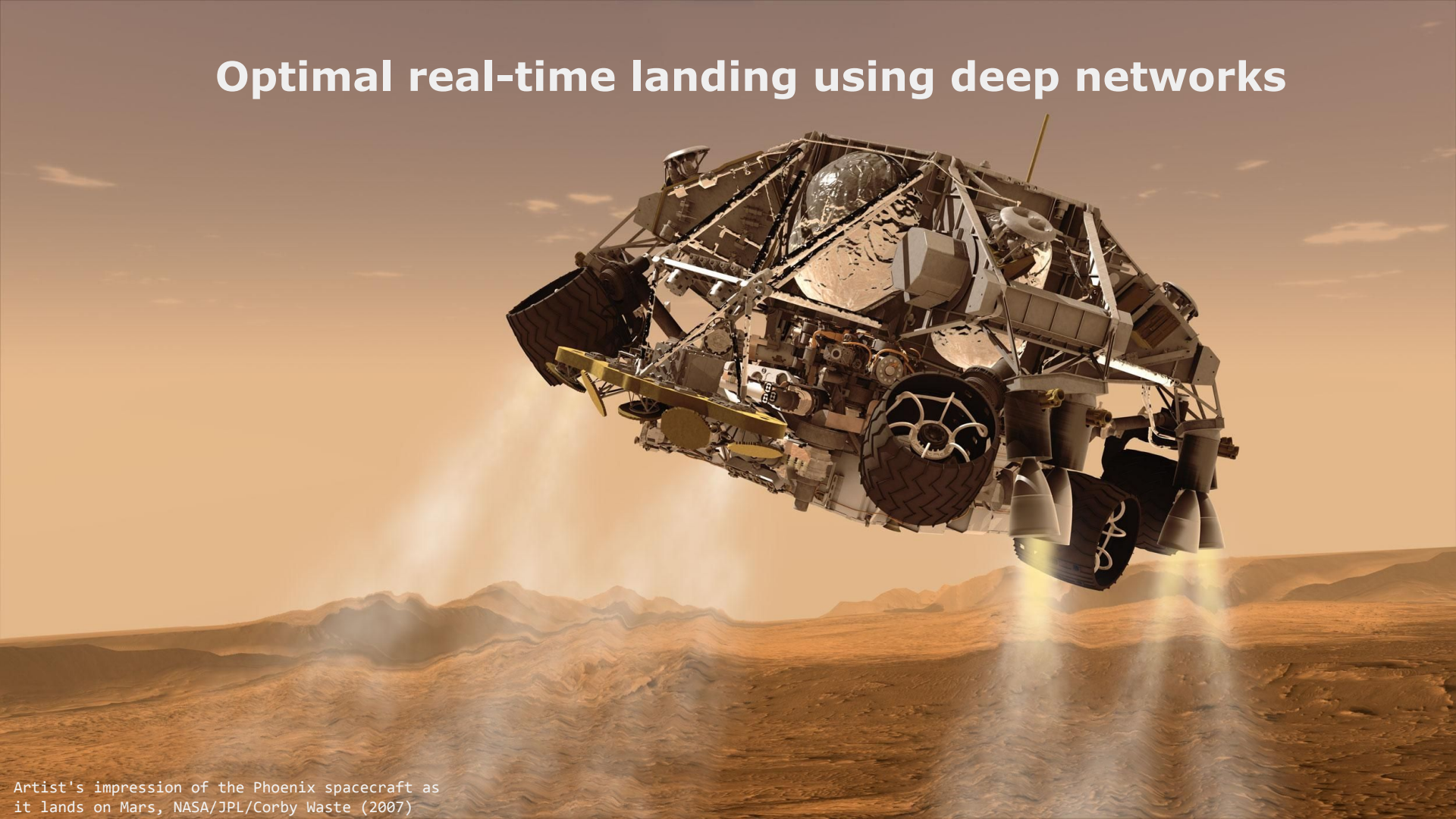
Even w.r.t. an NLP solver with a low number of nodes (sub-optimal), **the DNNs are up to 500 times faster.**

Conclusions

- **Deep networks trained with large datasets** succesfully approximate the optimal control even for regions out of the training data

- DNNs can be used as an **on-board reactive control system**, while current methods fail to compute optimal trajectories in an efficient way

Optimal real-time landing using deep networks



Artist's impression of the Phoenix spacecraft as it lands on Mars, NASA/JPL/Corby Waste (2007)