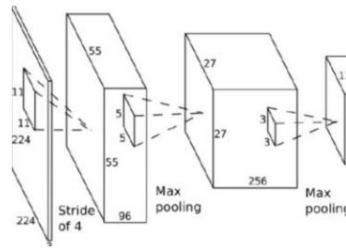


# Uczenie Maszynowe

Wprowadzenie do Reinforcement Learning 3, 4/26/2017

# Imitation Learning / Behavioral Cloning

 $\mathbf{o}_t$ 

$$\pi_{\theta}(\mathbf{u}_t | \mathbf{o}_t)$$

 $\mathbf{u}_t$  $\mathbf{o}_t$   
 $\mathbf{u}_t$ supervised  
learning

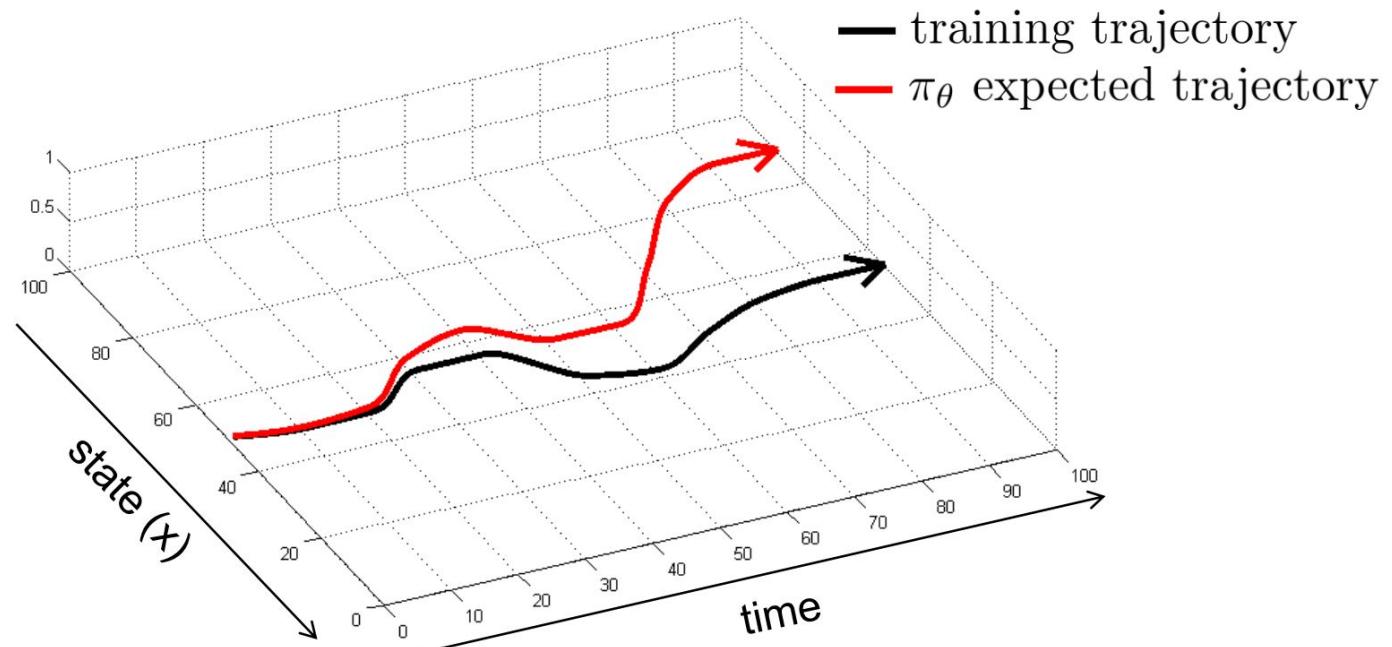
$$\pi_{\theta}(\mathbf{u}_t | \mathbf{o}_t)$$

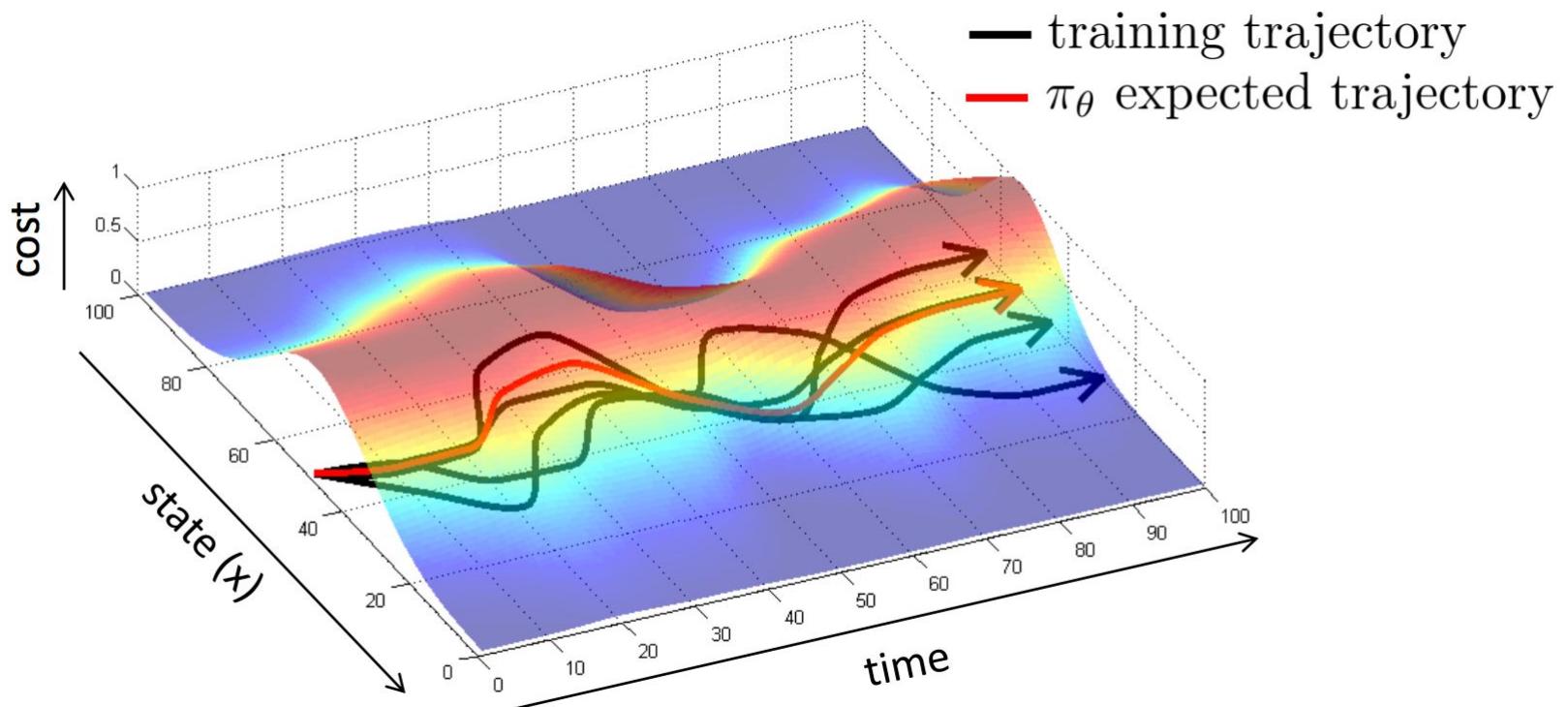
- Pojedyncza sieć konwolucyjna jest w stanie się nauczyć całkiem dobrze sterować samochodem mając dostęp do <100h ludzkich demonstracji
- Samochód jest w stanie się poruszać w różnych warunkach (autostrady/lokalne drogi/deszcze) otrzymując jedynie sygnał o skręcie kierownicy na danych treningowych

# Często supervised learning działa



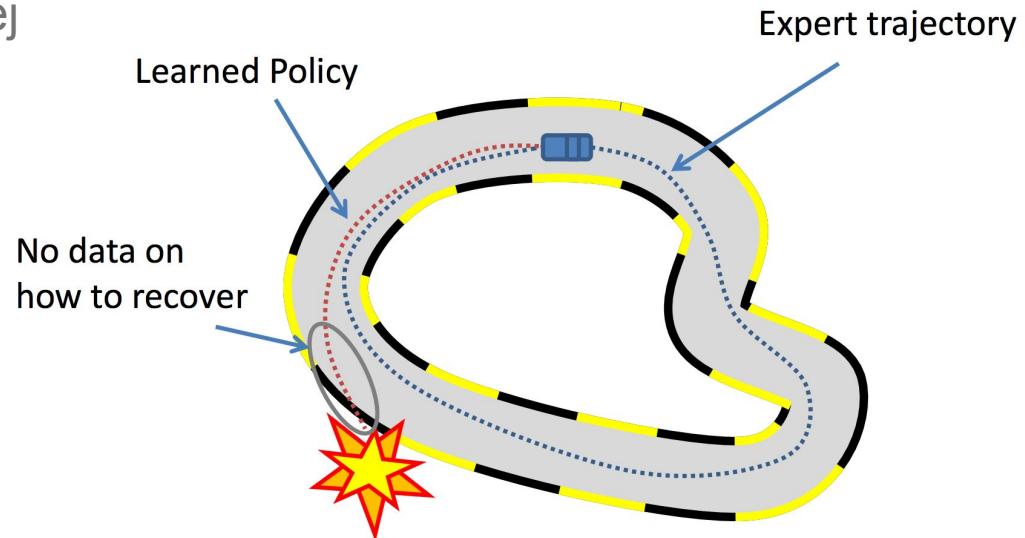
Niestety bardzo często modele się psują gdy chcemy ich użyć “na produkcji”



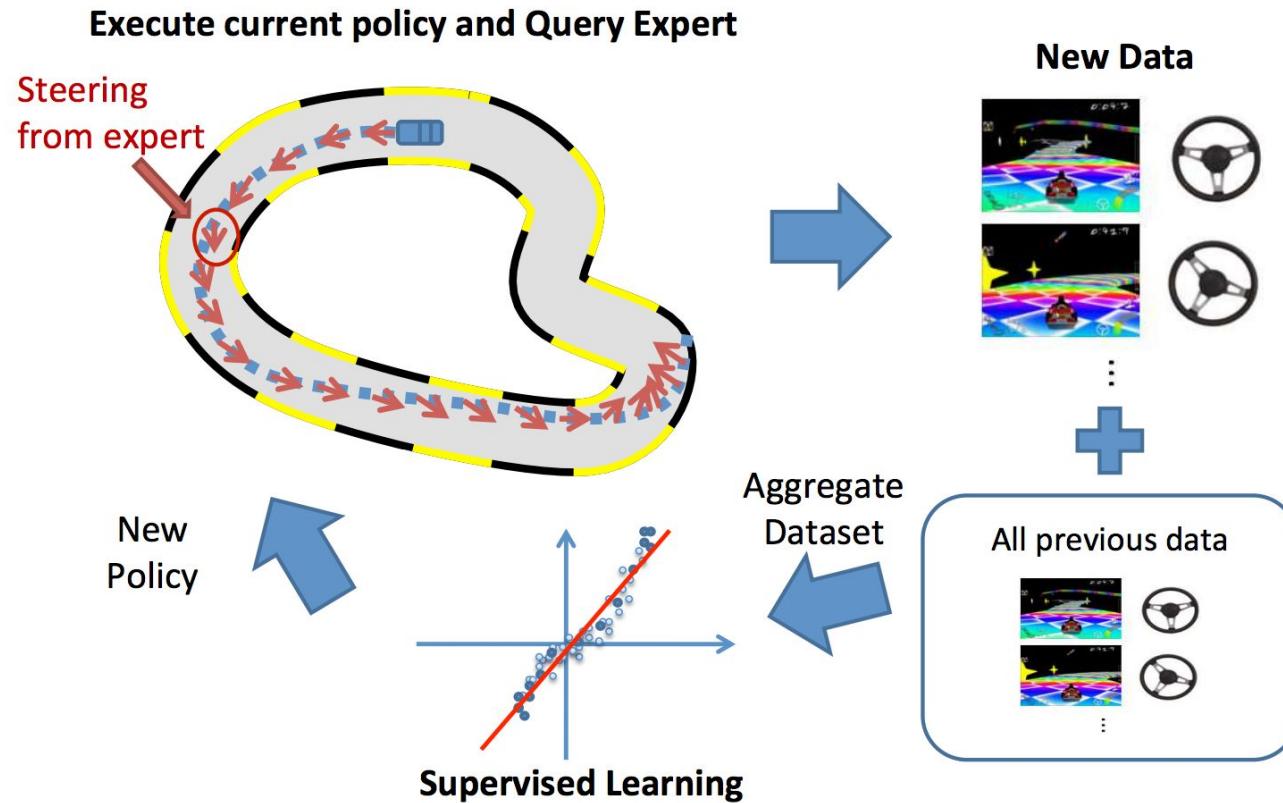


# Jak sprawić, żeby te techniki działały lepiej?

- Problemem jest to, że rozkład danych nie jest dokładnie taki sam jak ten pochodzący z sieci neuronowej



# Często możemy mieć dostęp do eksperta



# DAGGER

Initialize  $\mathcal{D} \leftarrow \emptyset$ .

Initialize  $\hat{\pi}_1$  to any policy in  $\Pi$ .

**for**  $i = 1$  **to**  $N$  **do**

    Let  $\pi_i = \beta_i \pi^* + (1 - \beta_i) \hat{\pi}_i$ .

    Sample  $T$ -step trajectories using  $\pi_i$ .

    Get dataset  $\mathcal{D}_i = \{(s, \pi^*(s))\}$  of visited states by  $\pi_i$  and actions given by expert.

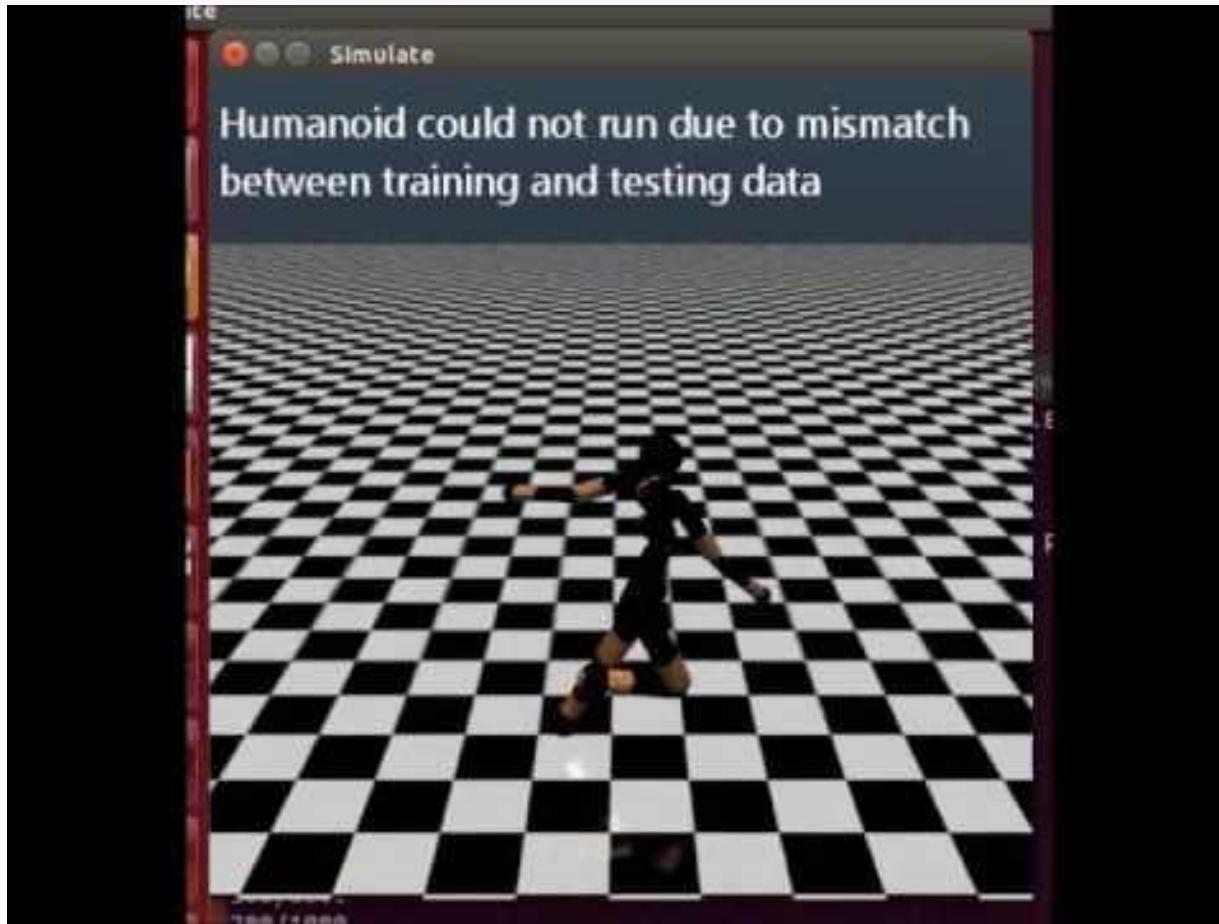
    Aggregate datasets:  $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_i$ .

    Train classifier  $\hat{\pi}_{i+1}$  on  $\mathcal{D}$  (or use online learner to get  $\hat{\pi}_{i+1}$  given new data  $\mathcal{D}_i$ ).

**end for**

**Return** best  $\hat{\pi}_i$  on validation.

## DAGGER - demo



## Uczenie się z demonstracji na Atari

Learning from Demonstrations for Real World Reinforcement Learning

<https://arxiv.org/abs/1704.03732>

- Praca przedstawia modyfikacje Deep Q-Learning, która jest w stanie wykorzystać demonstracje ludzi
- Najprostszy pomysł jest taki, żeby dorzucić wszystkie demonstracje do *Experience Replay*
- Można również zrobić pre-training modelu, aby przyspieszyć trening jeszcze bardziej. To wymaga paru dodatkowych trików

## Uczenie się z demonstracji na Atari - różnice względem DQN

- Początkowo model jest trenowany tylko na danych ekspertów z dodatkową funkcją kosztu, która będzie zwiększała wartość  $Q(s, a)$  dla akcji ekspertów.
- Używanie regularyzacji L2, aby przeciwdziałać overfitting do demonstracji
- Oddzielenie *Experience Replay* na 2 zbiory, żeby nigdy nie usuwać zachowań ekspertów

# Uczenie się z demonstracji na Atari - algorytm

$$J_{DQ}(Q) = (R(s, a) + \gamma Q(s_{t+1}, a_{t+1}^{\max}; \theta') - Q(s, a; \theta))^2$$

$$J_E(Q) = \max_{a \in A} [Q(s, a) + l(s, a_E, a)] - Q(s, a_E)$$

$$J(Q) = J_{DQ}(Q) + \lambda_1 J_E(Q) + \lambda_2 J_{L2}(Q).$$

---

## Algorithm 1 Deep Q-learning from Demonstrations.

---

```
1: Inputs:  $\mathcal{D}^{demo}$ : demonstration data set,  $\mathcal{D}^{replay}$ :  
empty,  $\theta$ : weights for initial behavior network (random),  
 $\theta'$ : weights for target network (random),  $\tau$ : frequency  
at which to update target net,  $k$ : number of pre-  
training gradient updates  
2: for steps  $t \in \{1, 2, \dots\}$  do  
3:   Sample a mini-batch of  $n$  transitions from  $\mathcal{D}^{demo}$   
4:   Calculate loss  $J(Q)$  using target network (Eq. 5)  
5:   Perform a gradient descent step to update  $\theta$   
6: end for  
7: for steps  $t \in \{1, 2, \dots\}$  do  
8:   Sample action from behavior policy  $a \sim \pi^{\epsilon Q_\theta}$   
9:   Play action  $a$  and observe  $(s', r)$ .  
10:  Store tuple  $(s, a, r, s')$  into  $\mathcal{D}^{replay}$ , overwriting old-  
est if over capacity  
11:  Sample a mini-batch of  $n$  transitions from  $\mathcal{D}^{demo} \cup$   
 $\mathcal{D}^{replay}$  with a fraction  $p$  of the samples from  $\mathcal{D}^{demo}$   
12:  Calculate loss  $J(Q)$  using target network (Eq. 5)  
13:  Perform a gradient descent step to update  $\theta$   
14:  if  $t \bmod \tau = 0$  then  $\theta' \leftarrow \theta$  end if  
15:   $s \leftarrow s'$   
16: end for
```

---

# Uczenie się z demonstracji na Atari - wyniki

- Ta metoda daje lepsze wyniki od Double DQN (bez demonstracji) na 27/42 gier Atari
- Jest też lepsza od zwykłego klonowania strategii (bez interakcji ze środowiskami) na 31/42 grach

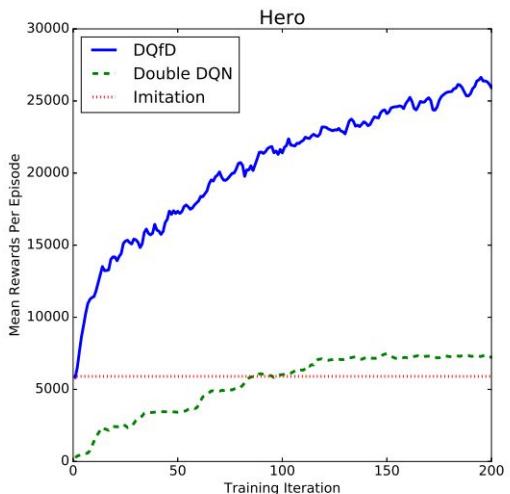


Figure 2. On-line rewards of the three algorithms on the game of Hero. Many of the games had similar results to this one, where DQfD starts out with performance near the imitation policy and improves from there.

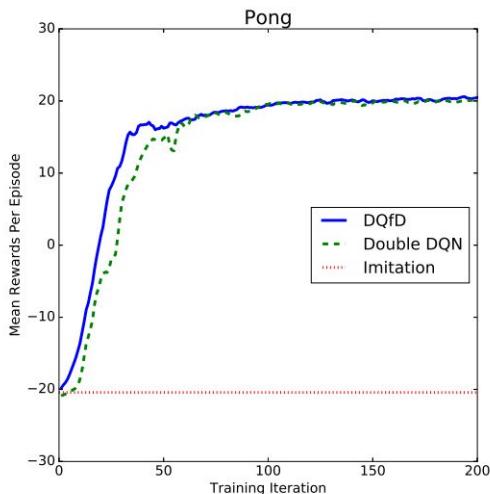


Figure 3. On-line rewards of the three algorithms on the game of Pong. Although the human demonstrator did not win a game in their demonstrations, DQfD still out-performs DQN for the first 58 iterations.

Game	DQfD	Double DQN	Imitation
Alien	<b>577.1</b>	280.1	473.9
Amidar	<b>250.4</b>	76.3	175.0
Assault	1017.4	<b>1384.9</b>	634.4
Asterix	2353.3	<b>4715.4</b>	279.9
Asteroids	<b>2507.8</b>	914.6	1267.3
Atlantis	<b>17647.0</b>	13494.8	12736.6
Bank Heist	<b>106.2</b>	8.7	95.2
Battle Zone	12486.1	3456.4	<b>14402.4</b>
Beam Rider	464.3	<b>748.8</b>	365.9
Bowling	46.5	28.5	<b>92.6</b>
Boxing	89.1	85.2	7.5
Breakout	<b>95.8</b>	5.3	3.5
Chopper Command	<b>2989.3</b>	2582.3	2485.7
Crazy Climber	103980.9	<b>108450.5</b>	14051.0
Defender	<b>7607.6</b>	3505.7	3819.1
Demon Attack	186.0	<b>405.1</b>	147.5
Double Dunk	-16.9	-20.2	-21.4
Enduro	624.8	<b>736.8</b>	134.8
Fishing Derby	-18.0	-13.5	-74.4
Freeway	<b>30.9</b>	28.5	22.7
Gopher	<b>9079.5</b>	4909.8	1142.6
Gravitar	245.1	35.6	<b>248.0</b>
Hero	<b>20428.2</b>	5373.0	5903.3
Ice Hockey	-9.8	-4.7	-13.5
James Bond	145.5	6.5	<b>262.1</b>
Kangaroo	1311.2	<b>1779.2</b>	917.3
Krull	1054.8	1880.2	<b>2216.6</b>
Kung Fu Master	<b>12328.6</b>	6677.8	556.7
Montezuma's Revenge	<b>780.9</b>	0.0	576.3
Ms Pacman	680.6	308.8	<b>692.4</b>
Name This Game	<b>4376.5</b>	4171.5	3745.3
Pitfall	-124.6	-26.3	<b>182.8</b>
Pong	<b>15.2</b>	13.6	-20.4
Private Eye	38280.5	-111.3	<b>42749.6</b>
Q-bert	2211.6	245.9	<b>5133.8</b>
River Raid	2368.0	<b>3202.6</b>	2148.5
Road Runner	38041.5	<b>39988.2</b>	8794.9
Seaquest	181.2	<b>1113.9</b>	195.6
Solaris	3107.9	221.8	<b>3589.6</b>
Up N Down	<b>10265.1</b>	8522.9	1816.7
Video Pinball	<b>10926.2</b>	7135.5	10655.5
Yars' Revenge	4764.3	<b>5731.8</b>	4225.8

Table 2. Average On-line Rewards of each algorithm over 200 iterations of 1 million Atari frames each on all 42 Atari games.

# Kilka zastosowań algorytmów do RL

AlphaGo -

<http://airesearch.com/wp-content/uploads/2016/01/deepmind-mastering-go.pdf>



# AlphaGo



# Dlaczego tak długo zajęło pokonanie ludzi?

- Przestrzeń stanów jest duża (361 na pustej planszy) przez co przeszukiwanie drzewa gier jest bardzo kosztowne
- Decyzje podejmowane na początku gry mogą wpływać na decyzje ponad 100 ruchów do przodu
- Jest bardzo ciężko ocenić kto wygrywa w trakcie rozgrywki

# AlphaGo - architektura

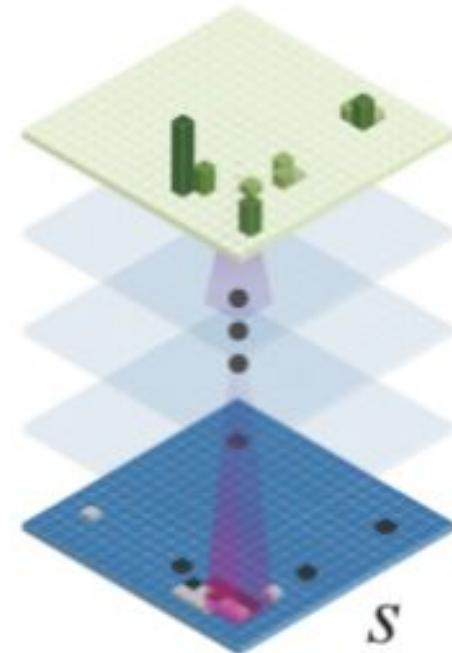
- NN z 13 warstwami, głównie konwolucyjnymi
- Na wyjściu sieć zwraca rozkład prawdopodobieństwa po wszystkich poprawnych ruchach w grze

Feature	# of planes	Description
Stone colour	3	Player stone / opponent stone / empty
Ones	1	A constant plane filled with 1
Turns since	8	How many turns since a move was played
Liberties	8	Number of liberties (empty adjacent points)
Capture size	8	How many opponent stones would be captured
Self-atari size	8	How many of own stones would be captured
Liberties after move	8	Number of liberties after this move is played
Ladder capture	1	Whether a move at this point is a successful ladder capture
Ladder escape	1	Whether a move at this point is a successful ladder escape
Sensibleness	1	Whether a move is legal and does not fill its own eyes
Zeros	1	A constant plane filled with 0
Player color	1	Whether current player is black

Extended Data Table 2: **Input features for neural networks.** Feature planes used by the policy network (all but last feature) and value network (all features).

Policy network

$$P_{\sigma/\rho}(a|s)$$



# AlphaGo - supervised pre-training

- W pierwszym kroku, model został wytrenowany na 30mln pozycji z bazy danych gier dobrych graczy
- Najlepszy model osiągnął 57% “poprawnych” predykcji na zbiorze testowym; 55.7% używając tylko pozycji kamieni jako wejście. Model ten potrzebuje 3ms na ewaluację pozycji
- Została wytrenowana też dużo mniejsza sieć, która osiąga 24.2%, ale jest za to 1500x szybsza

# AlphaGo - reinforcement learning

- Użyta została metoda policy gradients (REINFORCE) zaczynając od wag sieci wytrenowanych na historycznych grach
- Nagrody są równe 0 w trakcie rozgrywki i +1 lub -1 w ostatnim kroku
- Gry się rozgrywały przeciwko losowo wybranej poprzedniej wersji modelu
- Model wytrenowany z RL wygrywał ponad 80% gier przeciwko SL

$$\Delta \rho \propto \frac{\partial \log p_\rho(a_t | s_t)}{\partial \rho} z_t$$

# AlphaGo - value function

- W ostatnim etapie treningu uczona była nowa sieć, służąca do oceny aktualnej pozycji
- Chcielibyśmy znać szansę na zwycięstwo przeciwko optymalnemu zawodnikowi i aproksymujemy to najmocniejszym modelem
- Używanie istniejącej bazy gier prowadziło do mocnego over-fittingu. Zamiast tego, stworzyli nową bazę z 30 mln pozycjami z rozgrywek granych przeciwko sobie. Z każdej z nich była losowana jedna pozycja.

# AlphaGo - MCTS

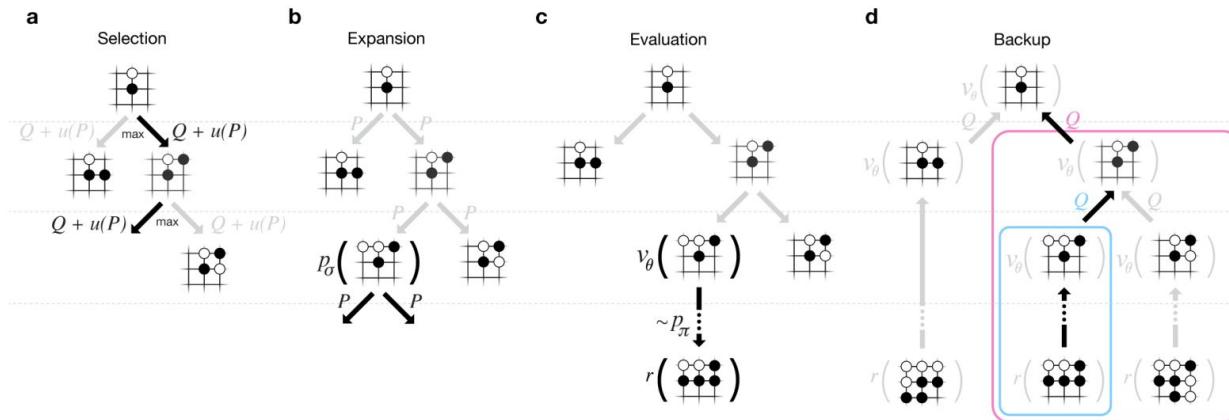
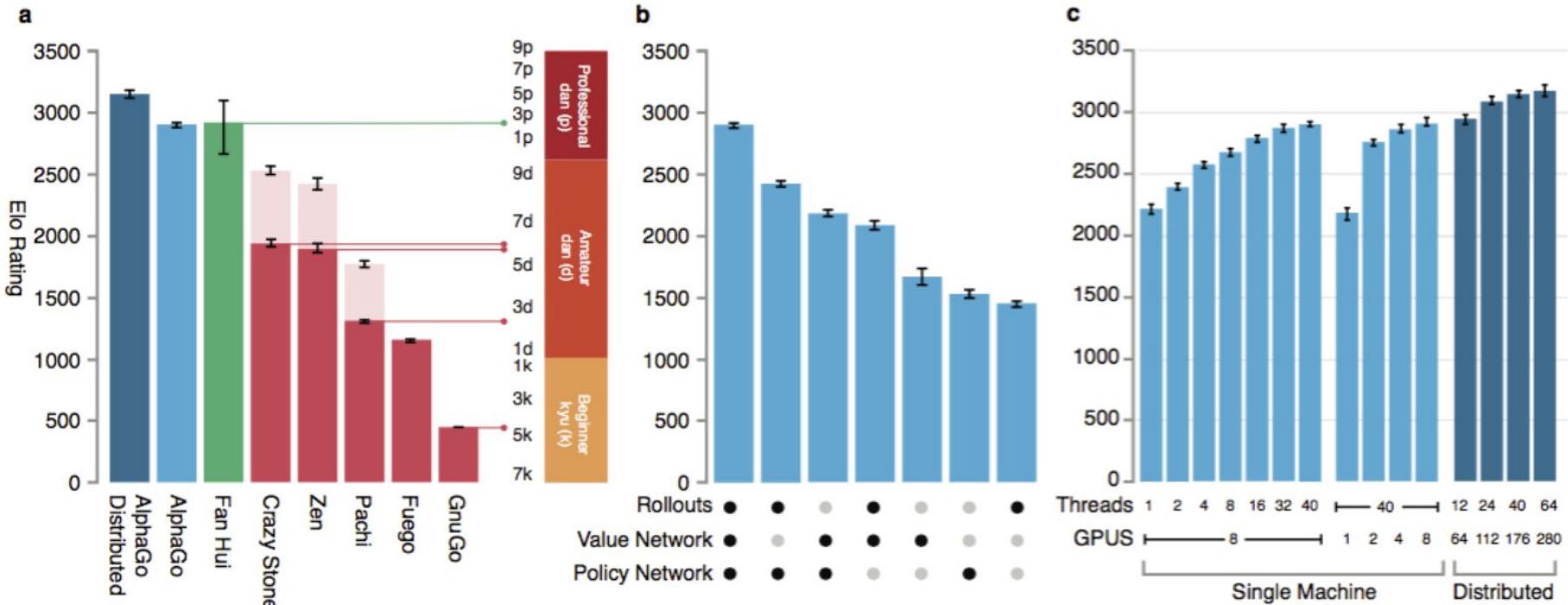


Figure 3: **Monte-Carlo tree search in AlphaGo.** **a** Each simulation traverses the tree by selecting the edge with maximum action-value  $Q$ , plus a bonus  $u(P)$  that depends on a stored prior probability  $P$  for that edge. **b** The leaf node may be expanded; the new node is processed once by the policy network  $p_\sigma$  and the output probabilities are stored as prior probabilities  $P$  for each action. **c** At the end of a simulation, the leaf node is evaluated in two ways: using the value network  $v_\theta$ ; and by running a rollout to the end of the game with the fast rollout policy  $p_\pi$ , then computing the winner with function  $r$ . **d** Action-values  $Q$  are updated to track the mean value of all evaluations  $r(\cdot)$  and  $v_\theta(\cdot)$  in the subtree below that action.

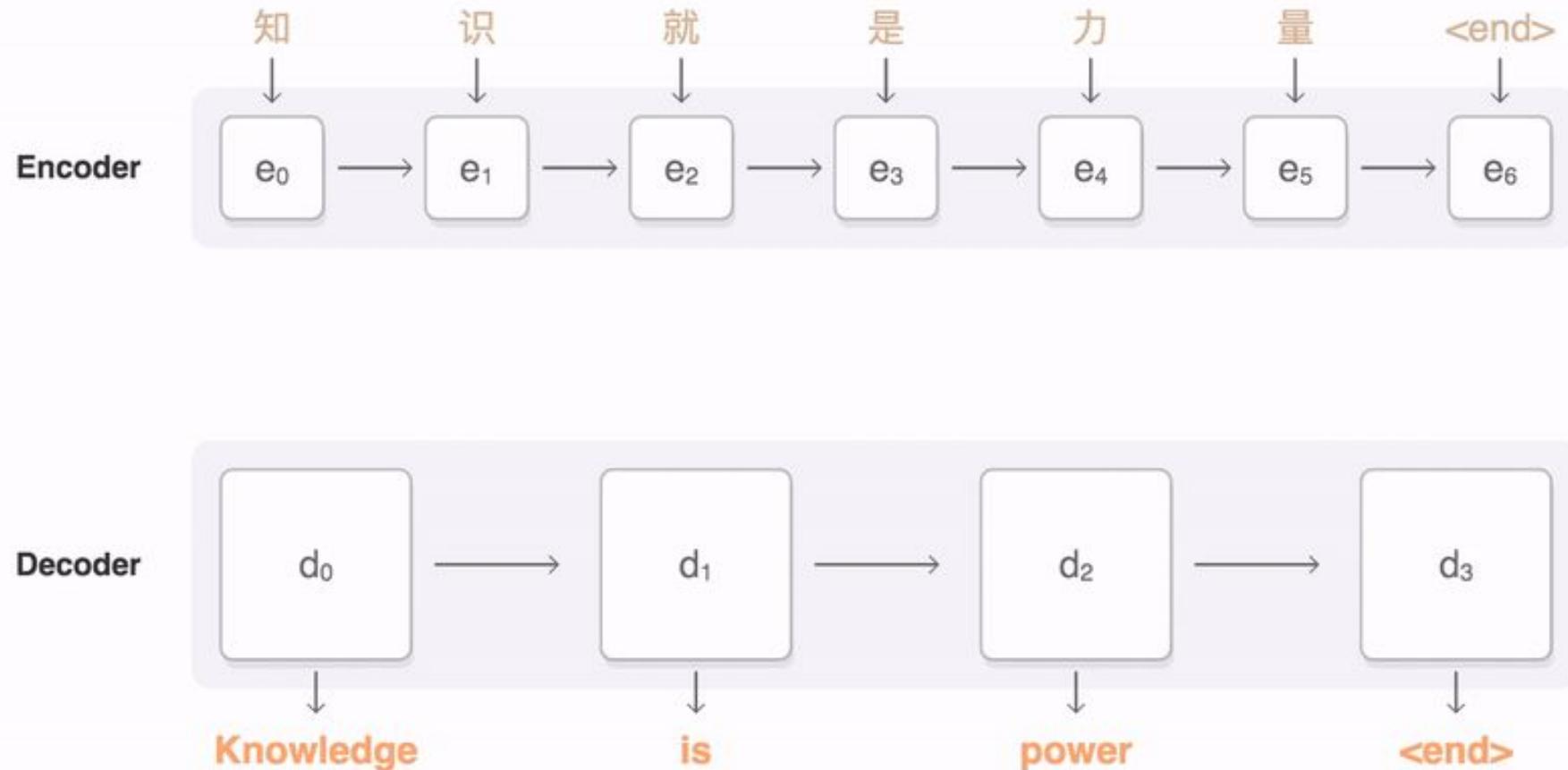
# AlphaGo - MCTS

- Ruchy w MCTS są wybierane używając  $Q(s, a) + u(s, a)$ 
  - $u(s, a)$  jest dodatkowym bonusem proporcjonalnym do prawdopodobieństwa zwróconego przez policy network podzielone przez liczbę odwiedzin danego stanu w trakcie MCTS
- Ostatecznie stany są ewaluowane korzystając z value function oraz szybkiego modelu, który symuluje rozgrywkę z danego stanu do końca

# AlphaGo - Statystyki



# Machine Translation



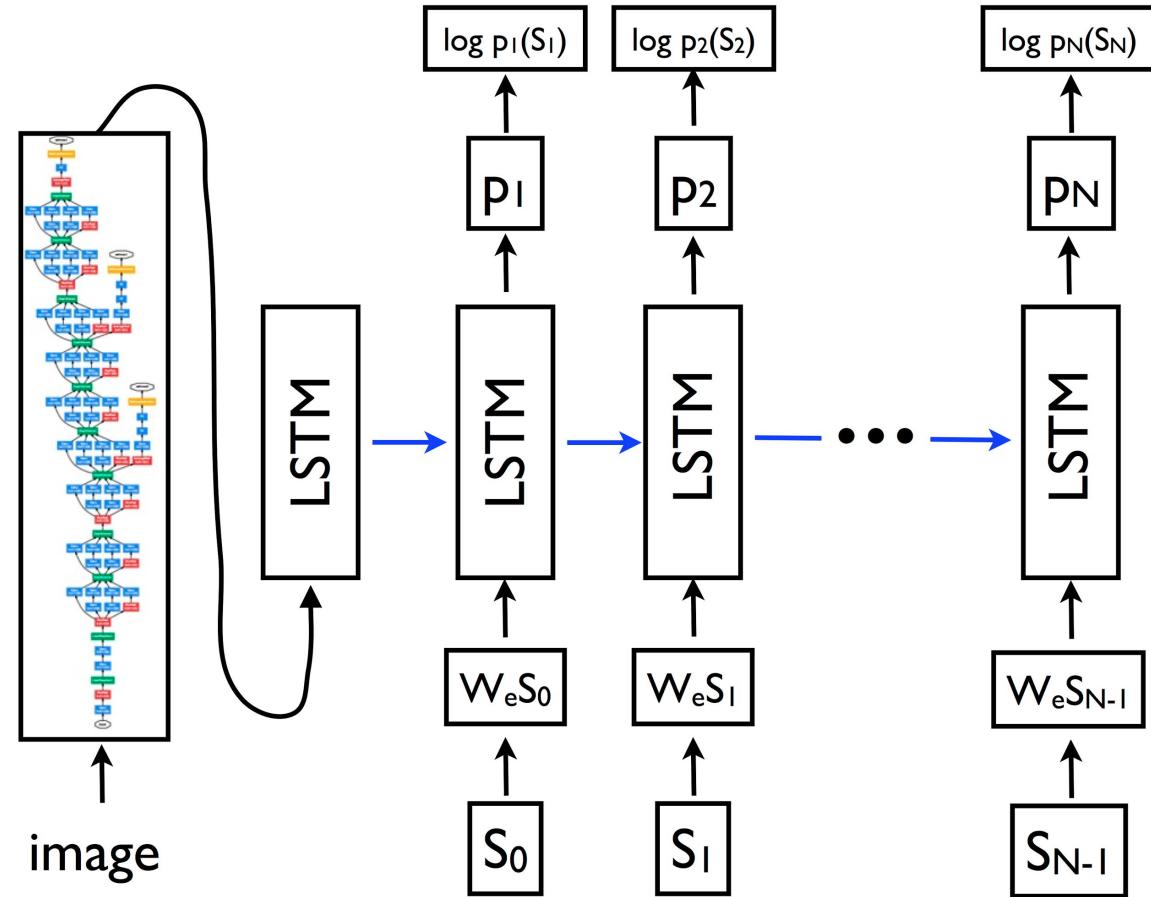
# Machine Translation

- Aktualnie, rozwiązania do tłumaczeń między językami używają modeli sequence-to-sequence (seq2seq)
- Zbiory danych zawierają pary sekwencji słów z różnych języków
  - Te metody działają najlepiej gdy mamy  $O(10^6)$  par zdań
- Trening polega na maksymalizacji prawdopodobieństw słów w zdaniu z docelowego języka przy założeniu, że wszystkie poprzednie słowa zostały wybrane poprawnie
- Ewaluacja tych modeli używa zupełnie innych funkcji kosztu niż używanych do treningu!

# Inne zastosowania tych modeli - Image Captioning

Describes without errors	Describes with minor errors	Somewhat related to the image	Unrelated to the image
			
<p>A person riding a motorcycle on a dirt road.</p>	<p>Two dogs play in the grass.</p>	<p>A skateboarder does a trick on a ramp.</p>	<p>A dog is jumping to catch a frisbee.</p>
			
<p>A group of young people playing a game of frisbee.</p>	<p>Two hockey players are fighting over the puck.</p>	<p>A little girl in a pink hat is blowing bubbles.</p>	<p>A refrigerator filled with lots of food and drinks.</p>
			
<p>A herd of elephants walking across a dry grass field.</p>	<p>A close up of a cat laying on a couch.</p>	<p>A red motorcycle parked on the side of the road.</p>	<p>A yellow school bus parked in a parking lot.</p>

<https://github.com/tensorflow/models/tree/master/im2txt>



## Inne zastosowania tych modeli - ChatBots

**Human:** what is the purpose of life ?

**Machine:** to serve the greater good .

**Human:** what is the purpose of living ?

**Machine:** to live forever .

**Human:** what is the purpose of existence ?

**Machine:** to find out what happens when we get to the planet earth .

**Human:** where are you now ?

**Machine:** i 'm in the middle of nowhere .

**Human:** what is the purpose of dying ?

**Machine:** to have a life .

**Human:** what is the purpose of being intelligent ?

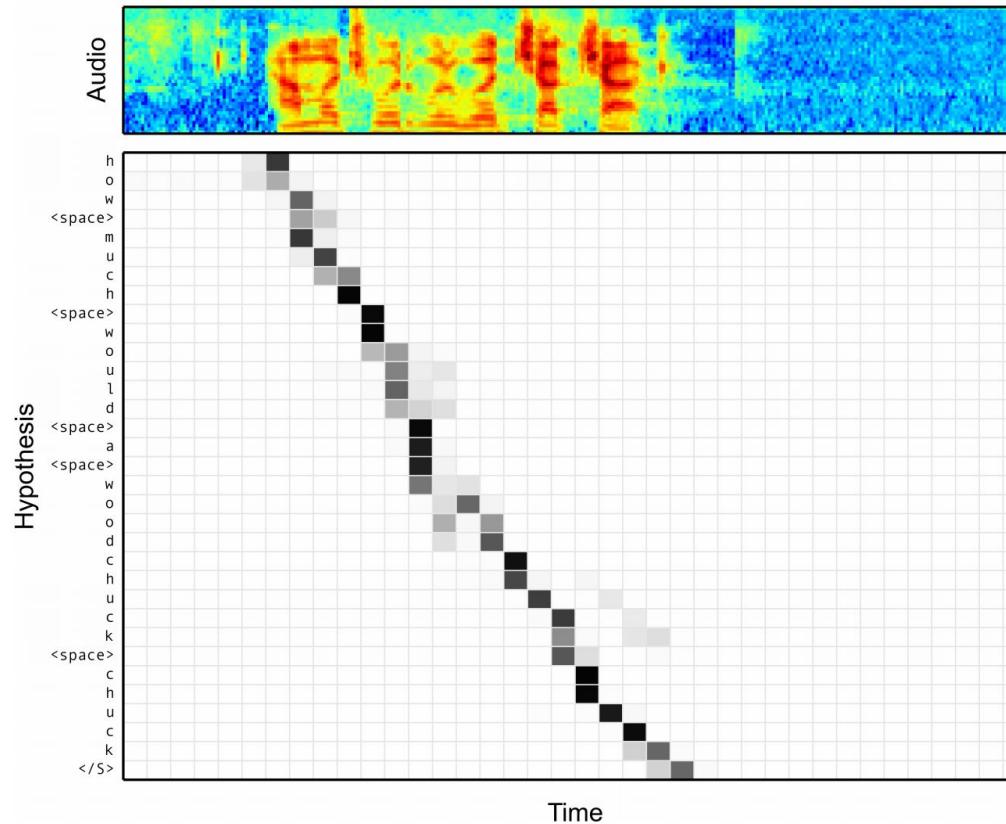
**Machine:** to find out what it is .

**Human:** what is the purpose of emotions ?

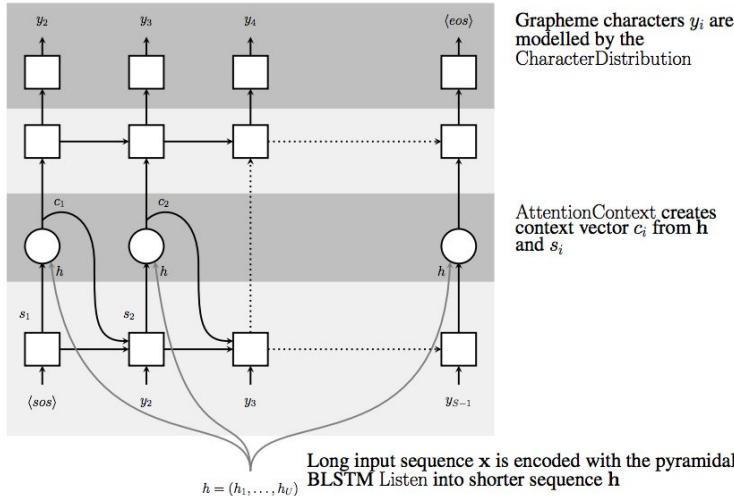
**Machine:** i don 't know .

# Inne zastosowania tych modeli - Speech Recognition

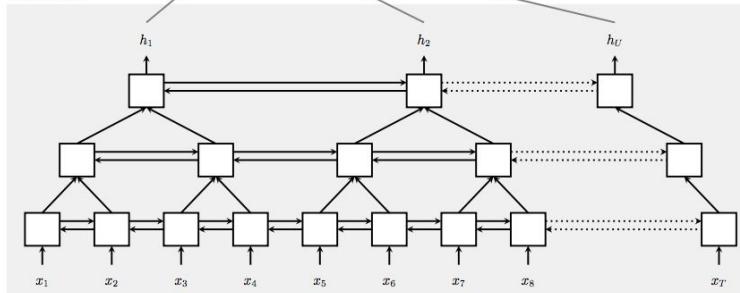
Alignment between the Characters and Audio



Speller



Listener



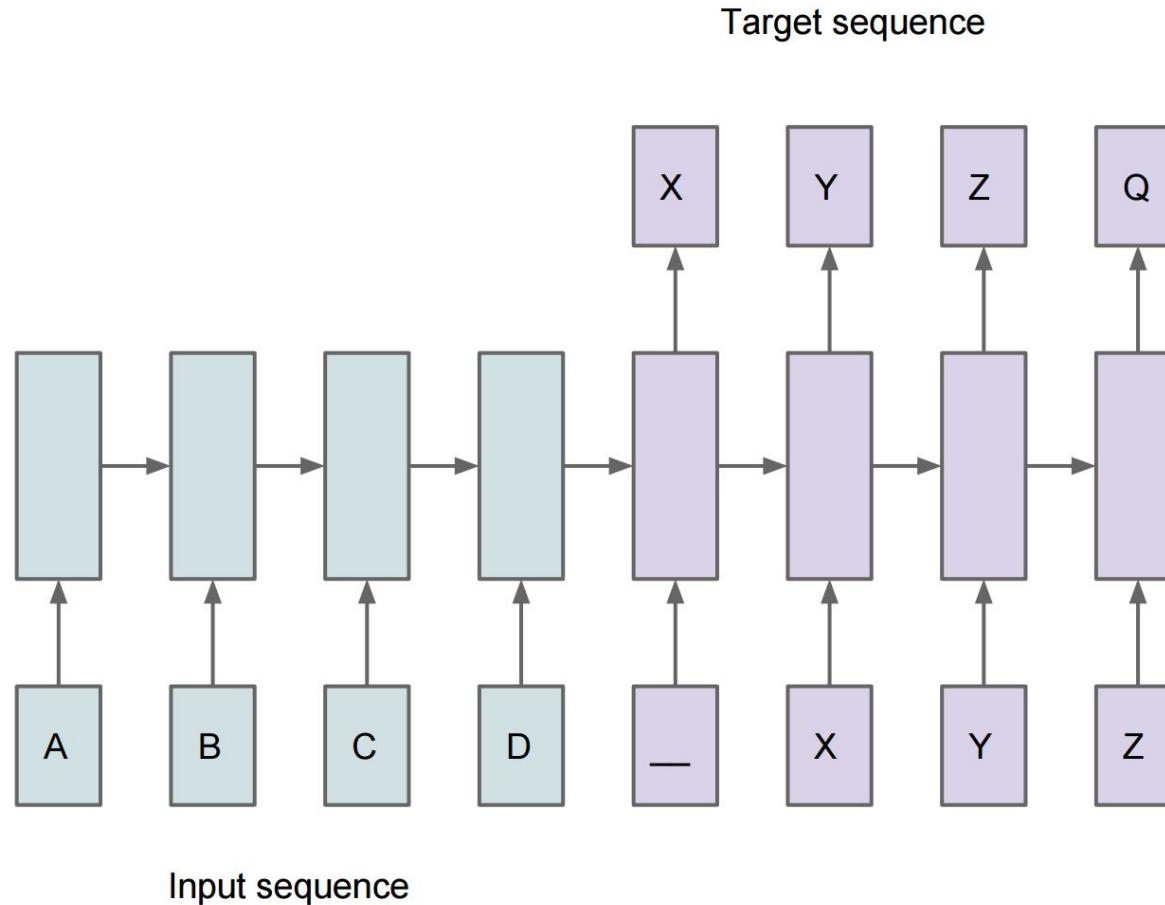
# Sequence to Sequence Models

Używa się ich również do:

- Syntezy głosu
- Rozpoznawania części mowy
- Streszczanie tekstu
- ...

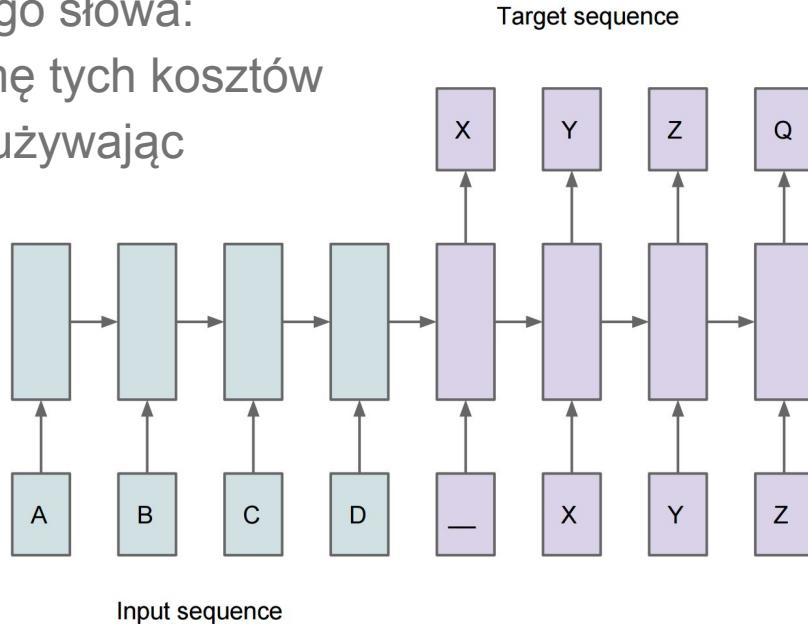
Wszystkie z tych zastosowań używają tej samej funkcji kosztu przy treningu

# Sequence to Sequence Models - Wprowadzenie



# Sequence to Sequence Models - Wprowadzenie

- LSTM wczytuje zdanie wejściowe, słowo po słowie
- Drugi LSTM używa stanu początkowego zainicjalizowanego pierwszym LSTMem i przewiduje kolejne słowa z języka docelowego
  - Podczas treningu na wejściu dostaje na słowa z poprawnego tłumaczenia
- Funkcja kosztu jest niezależna dla każdego słowa:  
 $\log p(y_t | x_{1..N}, y_{1..t-1})$  i optymalizujemy sumę tych kosztów
- Oba LSTMy są trenowane jednocześnie używając backpropagation!
- Można generować zdania poprzez losowanie akcji z dekodera



# Sequence to Sequence Models - Attention

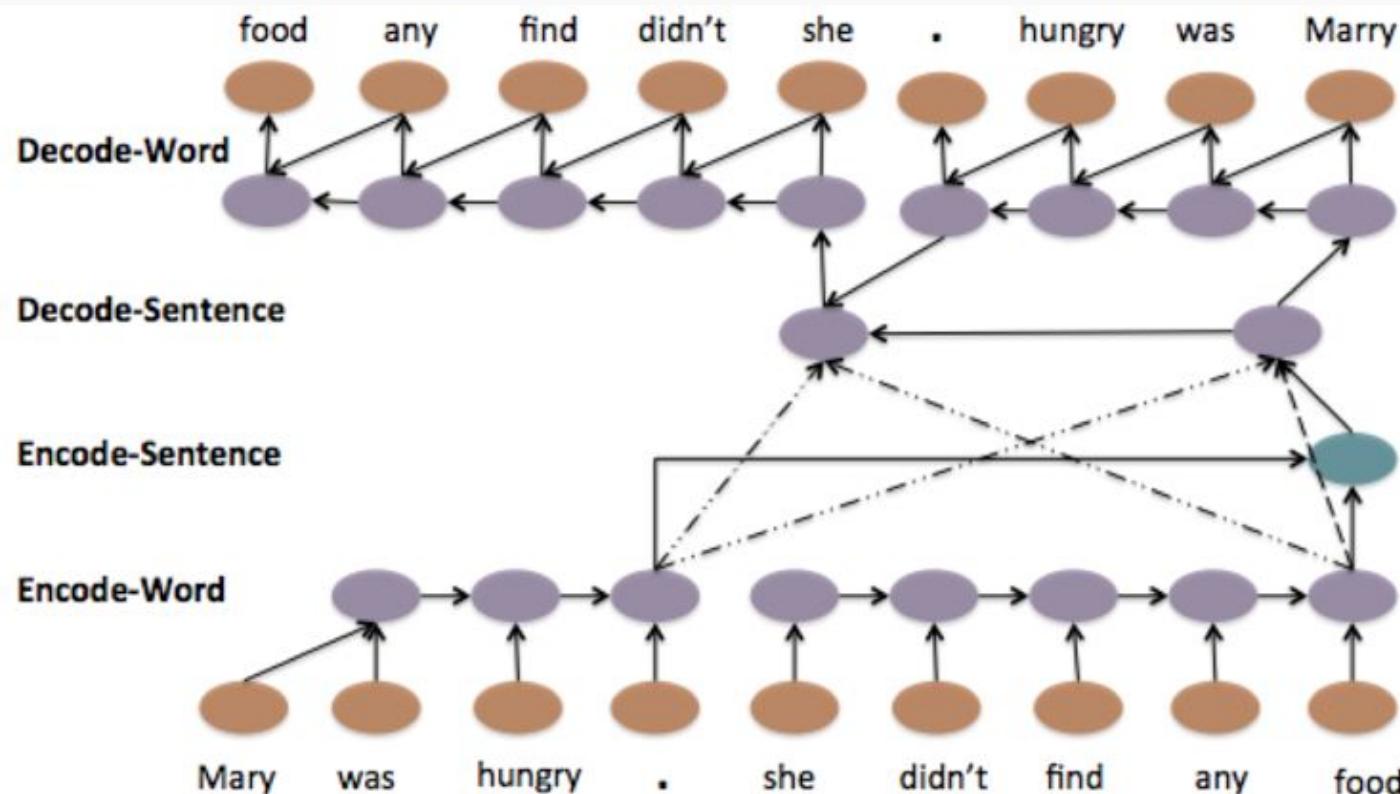
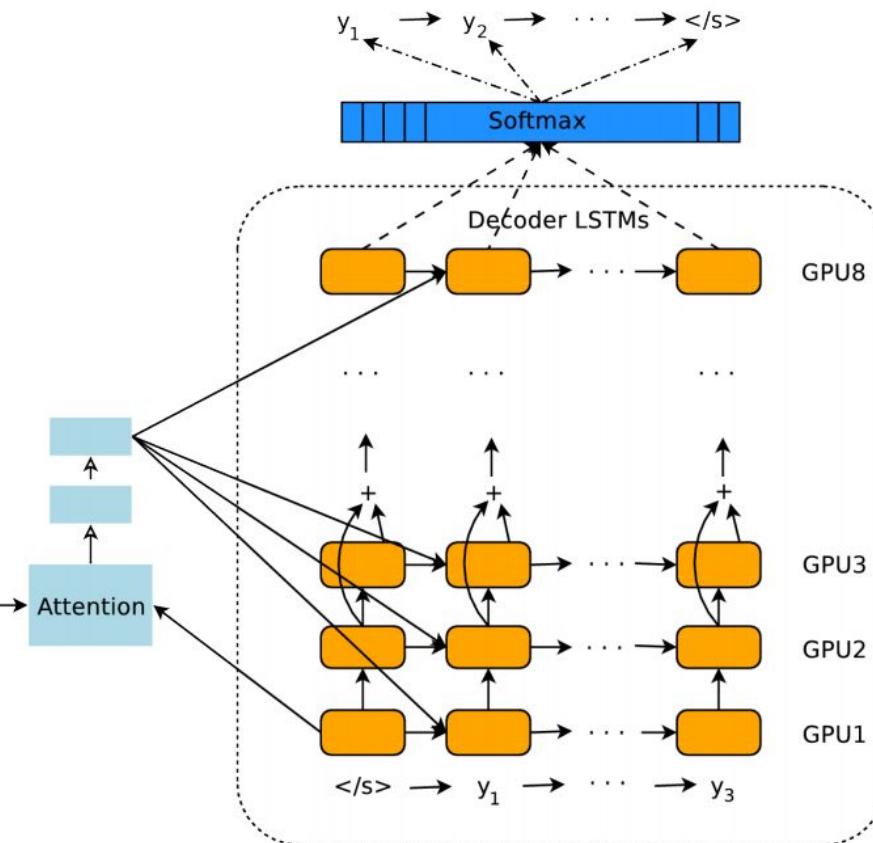
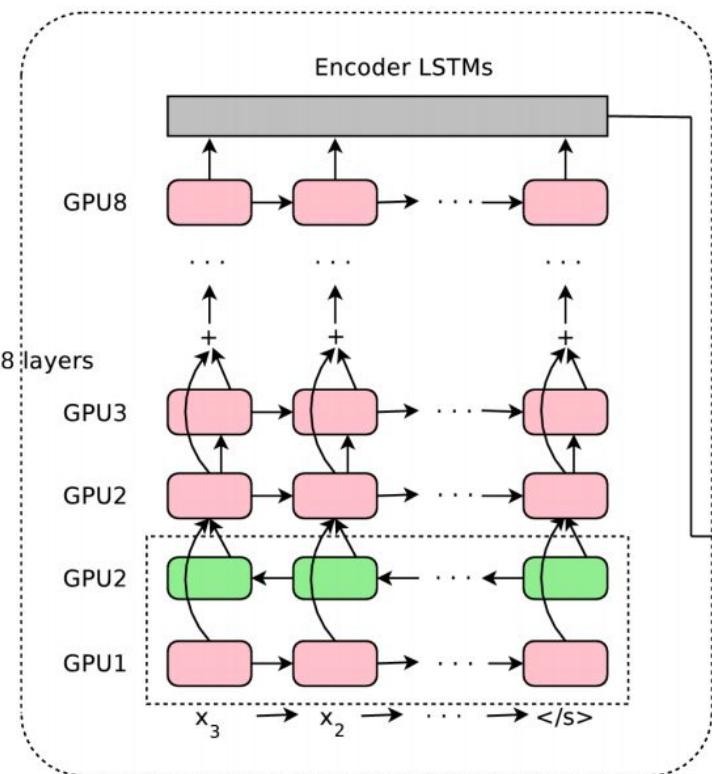


Figure 3: Hierarchical Sequence to Sequence Model with Attention.



# BLEU - metryka używana w MT

- Patrzymy na n-gramy o długości max 4 i porównujemy ile procent pasuje między naszym a poprawnym tłumaczeniem
- To nie jest idealna metryka, ale jest ona dość dobrze skorelowana z jakością tłumaczeń
- W szczególności, patrzy się na całe zdania w przeciwieństwie do logarytmów prawdopodobieństw kolejnych słów

$$\text{BLEU} = \min \left( 1, \frac{\text{output-length}}{\text{reference-length}} \right) \left( \prod_{i=1}^4 \text{precision}_i \right)^{\frac{1}{4}}$$

# Sequence to Sequence Models - Problemy?

- Podczas ewaluacji nie znamy prawdziwych etykiet, więc zachowanie modelu będzie inne niż podczas treningu
  - Zwykle na wejściu do dekodera daje się słowo wylosowane w poprzednim kroku
- Optymalizujemy metrykę, która nas tak naprawdę niezbyt interesuje
  - Każda domena ma swoje własne metryki używane do ewaluacji

# Policy Gradients do tłumaczeń

- Można wprost zastosować PG do naszego problemu, tzn. przemnożyć pochodne przez  $\text{BLEU}(y_{\text{predicted}}, y_{\text{true}})$
- W praktyce trenujemy najpierw model optymalizując Maximum Likelihood Estimation (MLE)

Table 6: Single model test BLEU scores, averaged over 8 runs, on WMT En→Fr and En→De

Dataset	Trained with log-likelihood	Refined with RL
En→Fr	38.95	39.92
En→De	24.67	24.60

# Wyniki

Table 7: Model ensemble results on WMT En→Fr (newstest2014)

Model	BLEU
WPM-32K (8 models)	40.35
RL-refined WPM-32K (8 models)	41.16
LSTM (6 layers) [31]	35.6
LSTM (6 layers + PosUnk) [31]	37.5
Deep-Att + PosUnk (8 models) [45]	40.4

Table 8: Model ensemble results on WMT En→De (newstest2014). See Table 5 for a comparison against non-ensemble models.

Model	BLEU
WPM-32K (8 models)	26.20
RL-refined WPM-32K (8 models)	26.30

## Wyniki - human evaluation

- Ludzie oceniali każde zdanie ze zbioru testowego oceną 0..6
- Modele trenowane z RL nie poprawiły się na najważniejszej metryce mimo, że są dużo lepsze przy ewaluacji z BLEU
- Całkiem prawdopodobne, że potrzebujemy lepszych automatycznych metryk, aby poprawić wyniki

Table 9: Human side-by-side evaluation scores of WMT En→Fr models.

Model	BLEU	Side-by-side averaged score
PBMT [15]	37.0	3.87
NMT before RL	40.35	4.46
NMT after RL	41.16	4.44
Human		4.82

## Alternatywne metody treningu

Minimum Risk Training for NMT - <https://arxiv.org/abs/1512.02433>

- Tak jak wcześniej, wytrenuj model optymalizujący metrykę MLE
- Po zakończeniu treningu, zmień metodę optymalizacji na następującą:
  - Dla danego zdania wejściowego  $x$ , wygeneruj z modelu 20 kandydatów do tłumaczenia
  - Oblicz prawdopodobieństwa dla każdego z nich + dla prawdziwej etykiety  $y$
  - Znormalizuj je tak, aby się sumowały do 1
  - Wykonaj krok SGD, aby zminimalizować  $\sum p(y_i | x) * -\text{BLEU}(y_i, y)$
- Główna różnica jest taka, że używamy wielu zdań podczas jednego kroku optymalizacji oraz prawdziwa etykieta znajduje się w grupie kandydatów

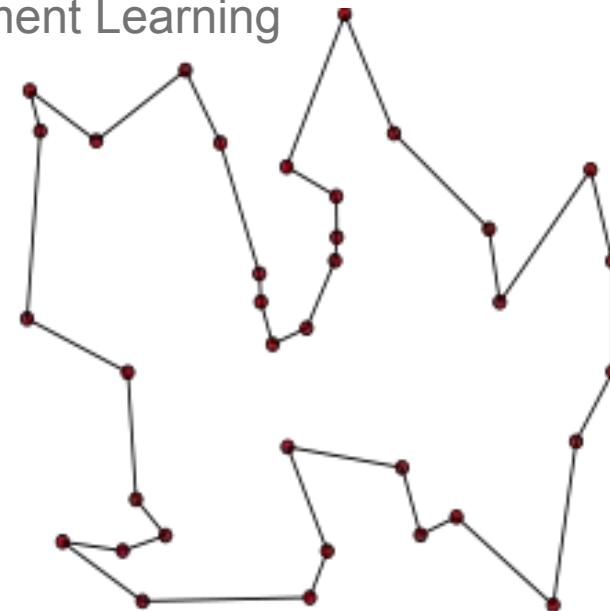
System	Training	MT06	MT02	MT03	MT04	MT05	MT08
MOSES	MERT	32.74	32.49	32.40	33.38	30.20	25.28
RNNSEARCH	MLE	30.70	35.13	33.73	34.58	31.76	23.57
	MRT	37.34	40.36	40.93	41.37	38.81	29.23

Table 3: Case-insensitive BLEU scores on Chinese-English translation.

# Problem planarnego komiwojażera

Neural Combinatorial Optimization with Reinforcement Learning

<https://arxiv.org/abs/1611.09940>



# Metoda

Do rozwiązania problemu zastosujemy ....

# Metoda

Do rozwiązania problemu zastosujemy .... Policy Gradients!

$$L(\pi \mid s) = \|\mathbf{x}_{\pi(n)} - \mathbf{x}_{\pi(1)}\|_2 + \sum_{i=1}^{n-1} \|\mathbf{x}_{\pi(i)} - \mathbf{x}_{\pi(i+1)}\|_2,$$

$$\nabla_{\theta} J(\theta \mid s) = \mathbb{E}_{\pi \sim p_{\theta}(\cdot \mid s)} \left[ (L(\pi \mid s) - b(s)) \nabla_{\theta} \log p_{\theta}(\pi \mid s) \right]$$

## Architektura używana do rozwiązania zadania

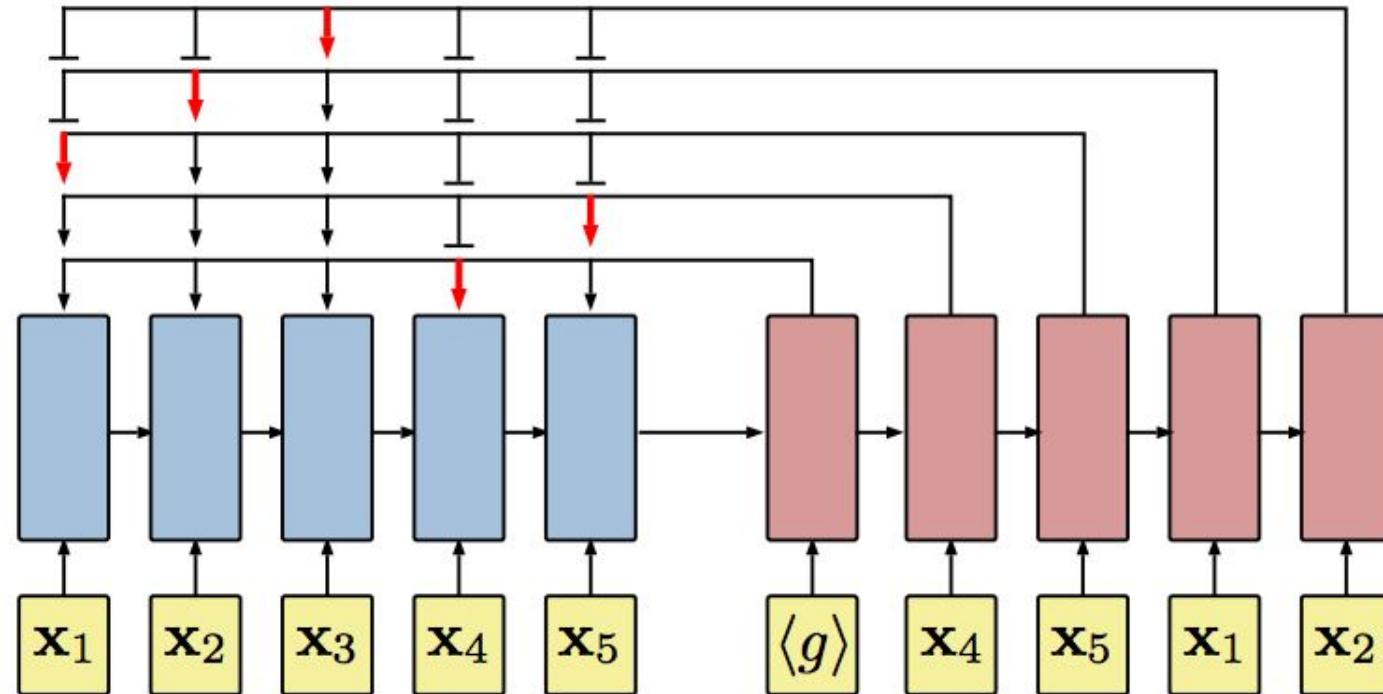


Figure 1: A pointer network architecture introduced by (Vinyals et al., 2015b).

# Sposoby nauki

- Supervised Learning dla danych z  $N \leq 20$  i mieć nadzieję, że będzie działać dla większych grafów
- Reinforcement Learning dla dużego zbioru grafów
- Reinforcement Learning i trenowanie sieci neuronowej pod konkretny graf

---

## Algorithm 1 Actor-critic training

---

```
1: procedure TRAIN(training set  $S$ , number of training steps  $T$ , batch size  $B$ )
2:   Initialize pointer network params  $\theta$ 
3:   Initialize critic network params  $\theta_v$ 
4:   for  $t = 1$  to  $T$  do
5:      $s_i \sim \text{SAMPLEINPUT}(S)$  for  $i \in \{1, \dots, B\}$ 
6:      $\pi_i \sim \text{SAMPLESOLUTION}(p_\theta(\cdot|s_i))$  for  $i \in \{1, \dots, B\}$ 
7:      $b_i \leftarrow b_{\theta_v}(s_i)$  for  $i \in \{1, \dots, B\}$ 
8:      $g_\theta \leftarrow \frac{1}{B} \sum_{i=1}^B (L(\pi_i|s_i) - b_i) \nabla_\theta \log p_\theta(\pi_i|s_i)$ 
9:      $\mathcal{L}_v \leftarrow \frac{1}{B} \sum_{i=1}^B \|b_i - L(\pi_i)\|_2^2$ 
10:     $\theta \leftarrow \text{ADAM}(\theta, g_\theta)$ 
11:     $\theta_v \leftarrow \text{ADAM}(\theta_v, \nabla_{\theta_v} \mathcal{L}_v)$ 
12:   end for
13:   return  $\theta$ 
14: end procedure
```

---

# Sposoby ewaluacji

- Greedy decoding
- Beam search
- Sampling
- **Active Search**

Configuration	Learn on training data	Sampling on test set	Refining on test set
RL pretraining-Greedy	Yes	No	No
Active Search (AS)	No	Yes	Yes
RL pretraining-Sampling	Yes	Yes	No
RL pretraining-Active Search	Yes	Yes	Yes

# Active Search

---

## Algorithm 2 Active Search

---

```
1: procedure ACTIVESEARCH(input  $s, \theta$ , number of candidates  $K, B, \alpha$ )
2:    $\pi \leftarrow \text{RANDOMSOLUTION}()$ 
3:    $L_\pi \leftarrow L(\pi \mid s)$ 
4:    $n \leftarrow \lceil \frac{K}{B} \rceil$ 
5:   for  $t = 1 \dots n$  do
6:      $\pi_i \sim \text{SAMPLESOLUTION}(p_\theta(\cdot \mid s))$  for  $i \in \{1, \dots, B\}$ 
7:      $j \leftarrow \text{ARGMIN}(L(\pi_1 \mid s) \dots L(\pi_B \mid s))$ 
8:      $L_j \leftarrow L(\pi_j \mid s)$ 
9:     if  $L_j < L_\pi$  then
10:       $\pi \leftarrow \pi_j$ 
11:       $L_\pi \leftarrow L_j$ 
12:    end if
13:     $g_\theta \leftarrow \frac{1}{B} \sum_{i=1}^B (L(\pi_i \mid s) - b) \nabla_\theta \log p_\theta(\pi_i \mid s)$ 
14:     $\theta \leftarrow \text{ADAM}(\theta, g_\theta)$ 
15:     $b \leftarrow \alpha \times b + (1 - \alpha) \times (\frac{1}{B} \sum_{i=1}^B b_i)$ 
16:  end for
17:  return  $\pi$ 
18: end procedure
```

---

# Wyniki

Table 2: Average tour lengths (lower is better). Results marked <sup>(†)</sup> are from (Vinyals et al., 2015b).

Task	Supervised Learning	RL pretraining				AS	Christo-fides	OR Tools' local search	Optimal
		greedy	greedy@16	sampling	AS				
TSP20	3.88 <sup>(†)</sup>	3.89	—	3.82	3.82	3.96	4.30	3.85	3.82
TSP50	6.09 <sup>(†)</sup>	5.95	5.80	5.70	5.70	5.87	6.62	5.80	5.68
TSP100	10.81	8.30	7.97	7.88	7.83	8.19	9.18	7.99	7.77

Table 3: Running times in seconds (s) of greedy methods compared to OR Tool's local search and solvers that find the optimal solutions. Time is measured over the entire test set and averaged.

Task	RL pretraining		OR-Tools' local search	Optimal	
	greedy	greedy@16		Concorde	LK-H
TSP50	0.003s	0.04s	0.02s	0.05s	0.14s
TSP100	0.01s	0.15s	0.10s	0.22s	0.88s

## Czas działania Active Search

Table 2: Average tour lengths (lower is better). Results marked <sup>(†)</sup> are from (Vinyals et al., 2015b).

Task	Supervised Learning	RL pretraining				AS	Christo-fides	OR Tools' local search	Optimal
		greedy	greedy@16	sampling	AS				
TSP20	3.88 <sup>(†)</sup>	3.89	—	3.82	3.82	3.96	4.30	3.85	3.82
TSP50	6.09 <sup>(†)</sup>	5.95	5.80	5.70	5.70	5.87	6.62	5.80	5.68
TSP100	10.81	8.30	7.97	7.88	7.83	8.19	9.18	7.99	7.77

Table 3: Running times in seconds (s) of greedy methods compared to OR Tool's local search and solvers that find the optimal solutions. Time is measured over the entire test set and averaged.

Task	RL pretraining		OR-Tools' local search	Optimal	
	greedy	greedy@16		Concorde	LK-H
TSP50	0.003s	0.04s	0.02s	0.05s	0.14s
TSP100	0.01s	0.15s	0.10s	0.22s	0.88s

## Problem Plecakowy używając tych samych metod

$$\max_{S \subseteq \{1, 2, \dots, n\}} \sum_{i \in S} v_i$$

subject to  $\sum_{i \in S} w_i \leq W$

Table 5: Results of RL pretraining-Greedy and Active Search on KnapSack (higher is better).

Task	RL pretraining greedy	Active Search	Random Search	Greedy	Optimal
KNAP50	19.86	<b>20.07</b>	17.91	19.24	<b>20.07</b>
KNAP100	40.27	<b>40.50</b>	33.23	38.53	<b>40.50</b>
KNAP200	57.10	<b>57.45</b>	35.95	55.42	<b>57.45</b>

# Neural Architecture Search

[http://rll.berkeley.edu/deeprlcourse/docs/quoc\\_barret.pdf](http://rll.berkeley.edu/deeprlcourse/docs/quoc_barret.pdf)