Evolutionary-Neural Hybrid Agents for Architecture Search

Krzysztof Maziarz

4 Czerwca 2020

Uniwersytet Jagielloński

Praca

Uniwersytet Jagielloński

Wydział Matematyki i Informatyki

Instytut Informatyki Analitycznej

Nr indeksu: 1115077

Krzysztof Maziarz

Evolutionary-Neural Hybrid Agents for Architecture Search

Opiekun pracy magisterskiej:
dr Jacek Tabor

Praca

Evolutionary-Neural Hybrid Agents for Architecture Search

Krzysztof Maziarz*
Jagiellonian University
krzysztof.s.maziarz@gmail.com

Mingxing Tan
Google AI
tanmingxing@google.com

Andrey Khorlin
Google AI
akhorlin@google.com

Marin Georgiev
Google AI
maringeorgiev@google.com

Andrea Gesmundo
Google AI
agesmundo@google.com

Poza promotorem (prof. dr hab. Jacek Tabor), do powstania pracy przyczyniła się grupa ludzi z Google Zurich (powyżej).

arxiv.org/abs/1811.09828

Wstęp do Auto-ML

Auto-ML

Uzyskiwanie modelu uczenia maszynowego dla danych zazwyczaj składa się z kilku kroków:

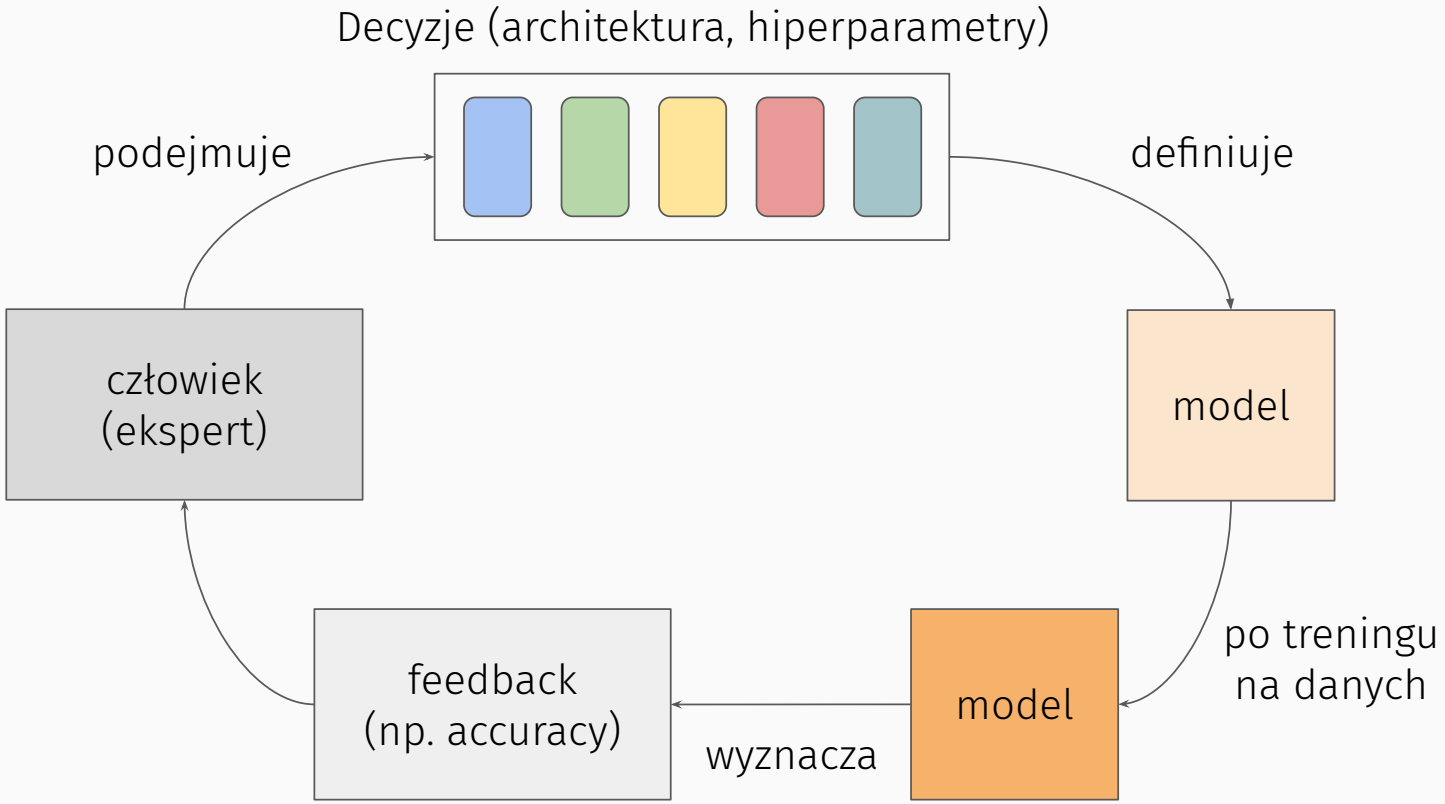
- definiujemy *architekturę*, czyli ”schemat” modelu
- wybieramy wartości *hiperparametrów* (takich jak learning rate czy użyte metody regularyzacji)
- optymalizujemy parametry tak aby dopasować je do danych (trening)

Auto-ML

O ile parametry (zazwyczaj) jesteśmy w stanie optymalizować przy użyciu prostych algorytmów (np. *gradient descent*), o tyle architektury i hiperparametrów już nie.

Typowo, architektura i hiperparametry są dobierane przez człowieka (eksperta) na bazie wiedzy i doświadczenia.

Auto-ML



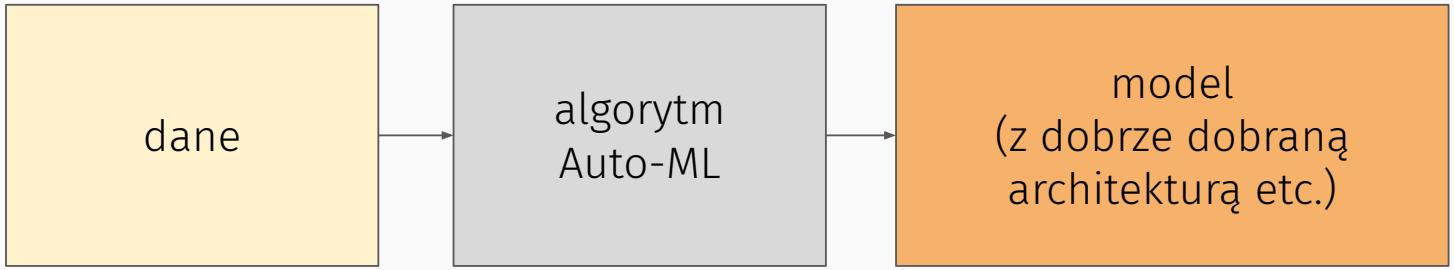
Projektowanie modelu uczenia maszynowego

Auto-ML

Warto byłoby mieć metodę, która automatycznie dopasuje do danych nie tylko parametry modelu, ale też architekturę i wszelkie hiperparametry...

Auto-ML

Auto-ML to właśnie metody pozwalające na optymalizację *wszystkich* części modelu - łącznie z architekturą i hiperparametrami.



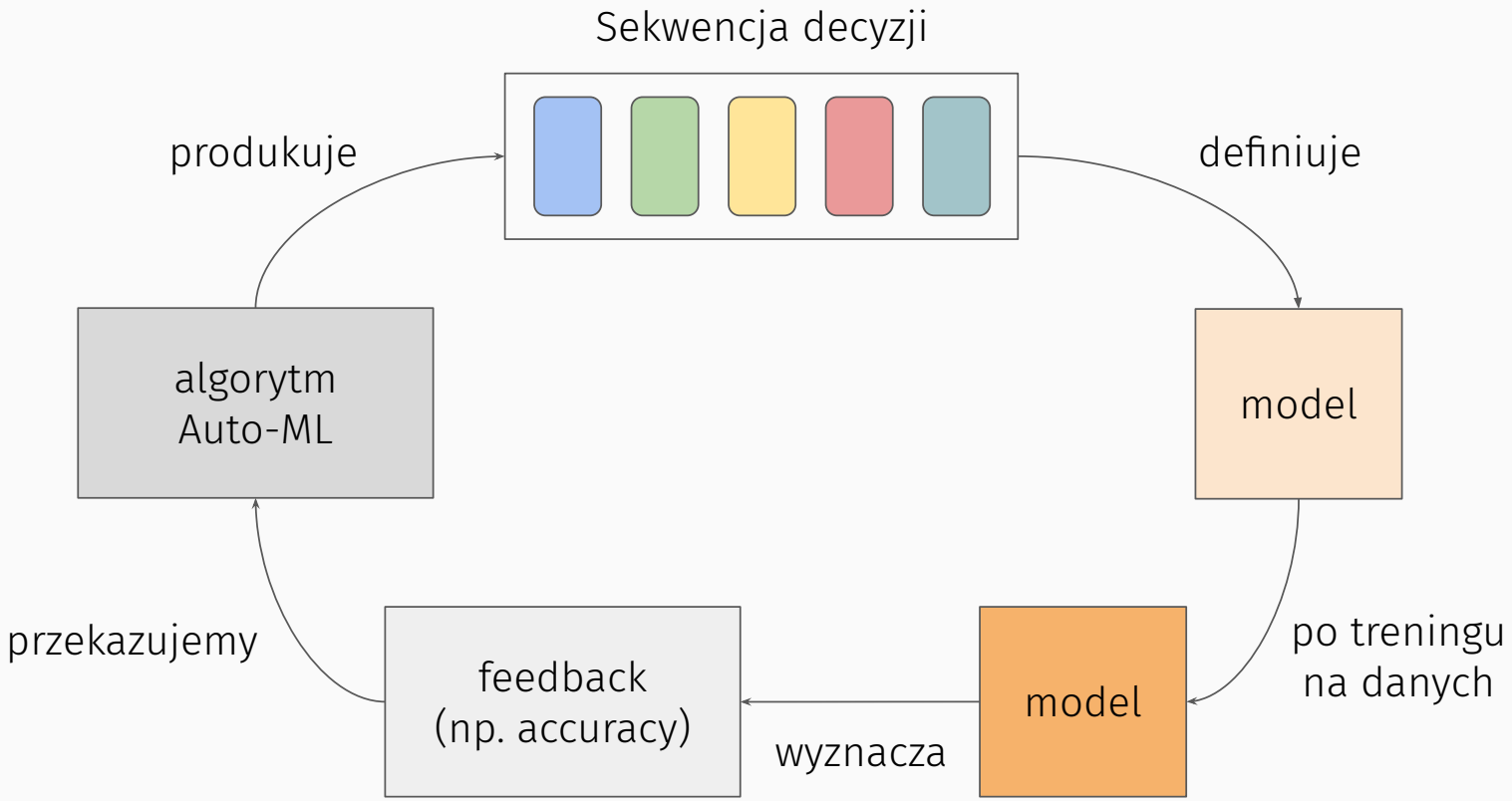
Auto-ML

By sformalizować problem definiuje się przestrzeń przeszukiwań, która opisuje wszystkie możliwe konfiguracje modeli jakie rozważamy.

Hidden layer size	Activation function	Use batch-norm	Optimizer	Initialize with pretrained weights	Training epochs
64 **256** 1024	tanh sigmoid **ReLU**	**Yes** No	**SGD** Adam AdaGrad	**Yes** No	25 50 **100**

Prosta przestrzeń przeszukiwań

Auto-ML



Procedura Auto-ML

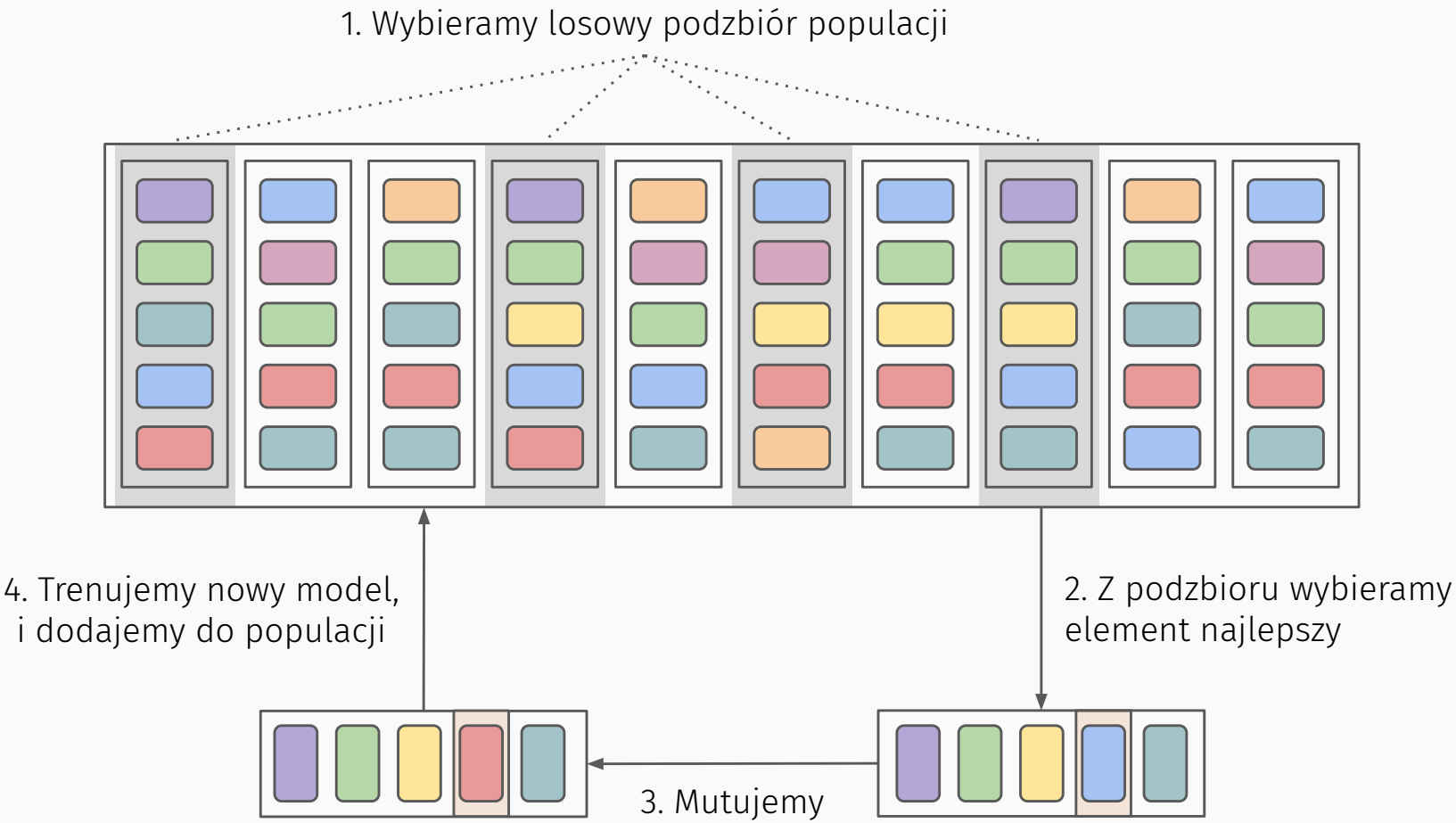
Podejścia do Auto-ML

Algorytm ewolucyjny [1]

Utrzymujemy *populację* modeli (odpowiadających elementom przestrzeni przeszukiwań, które już przetestowaliśmy).

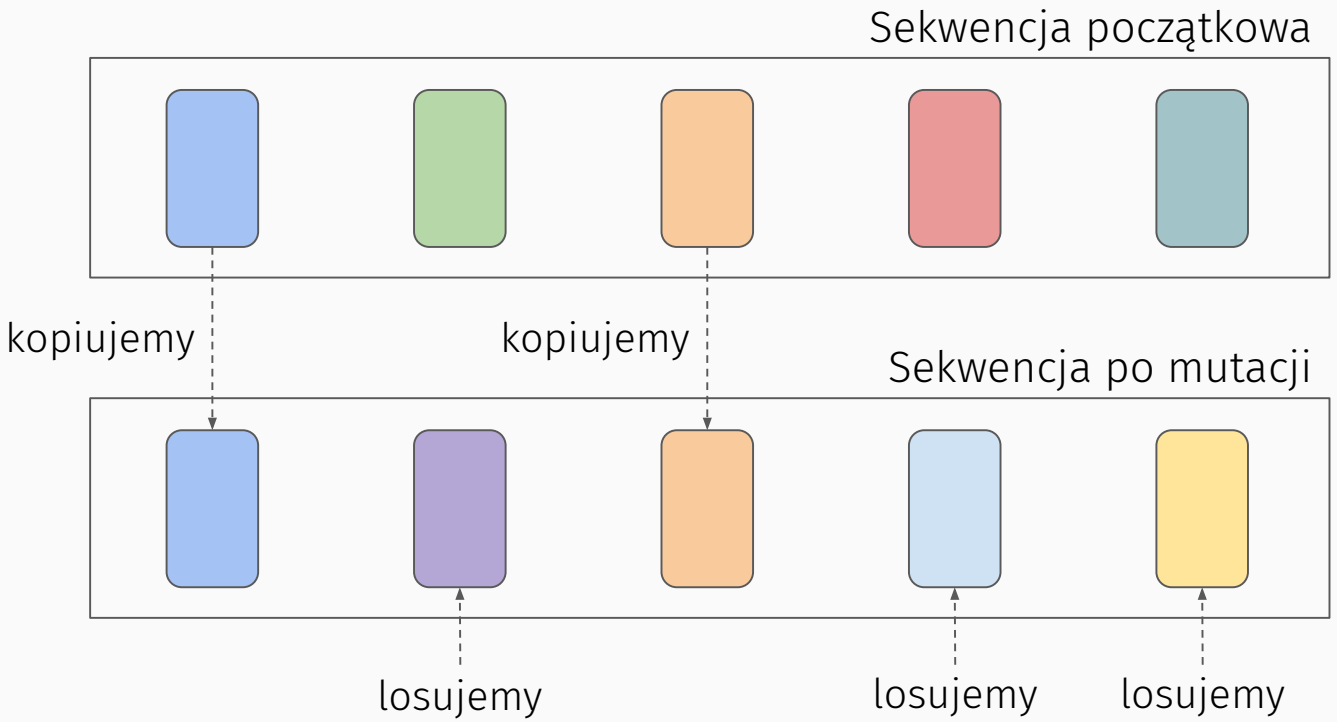
W każdym kroku wybieramy jeden model z populacji, *mutujemy* go, i dodajemy do populacji.

Algorytm ewolucyjny [1]



Algorytm ewolucyjny [1]

Aby wykonać mutację, wybieramy niektóre z decyzji i zmieniamy ich wartości losowo.



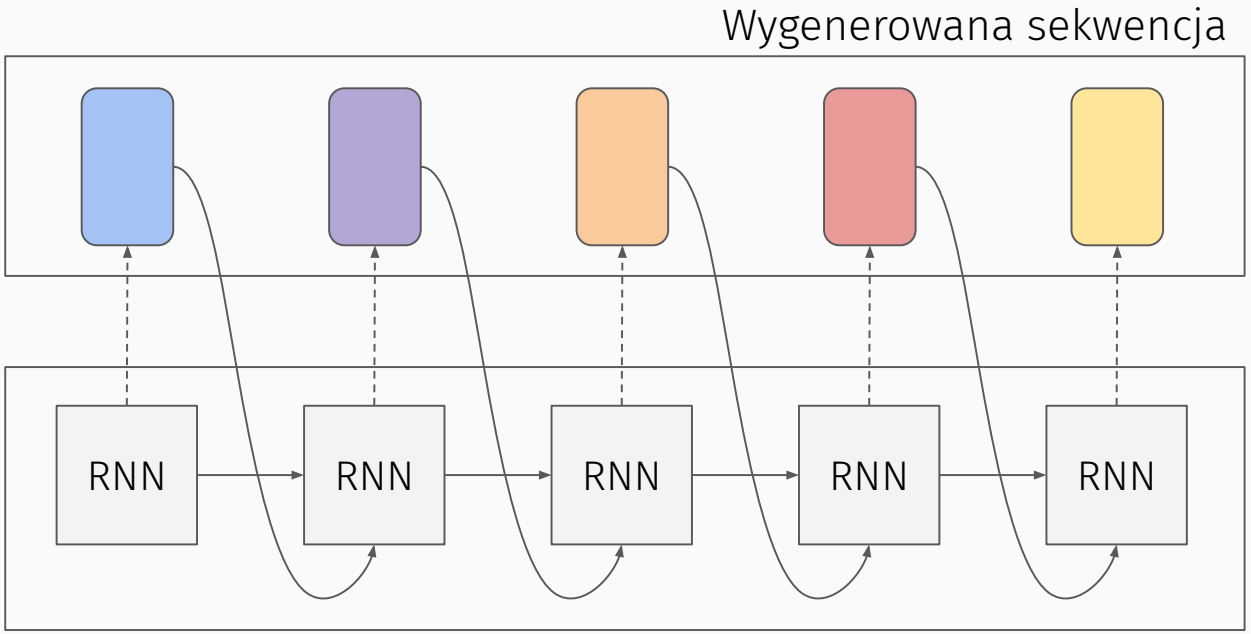
Algorytm z użyciem meta-sieci [2]

Inne z podejść opiera się na *meta-sieci neuronowej*.

Definiujemy w nim meta-sieć, której zadaniem będzie proponować elementy przestrzeni przeszukiwań do sprawdzenia.

Użyjemy *sieci rekurencyjnej* (RNN), która będzie podejmować decyzje jedna po drugiej.

Algorytm z użyciem meta-sieci [2]



Algorytm z użyciem meta-sieci [2]

Nie znamy ”poprawnych” decyzji których moglibyśmy użyć do trenowania meta-sieci.

Możemy użyć feedbacku który dostajemy (tj. jak dobry był model odpowiadający sekwencji decyzji) aby trenować meta-sieć za pomocą *reinforcement learning* (RL).

Porównanie

Podejście ewolucyjne

+ prostsze w zastosowaniu
+ w praktyce poprawia wynik stosunkowo szybko już od samego początku procesu
- niezależnie jak długo będzie działać, część sprawdzanych sekwencji będzie słaba (niefortunne losowe mutacje)

Podejście z użyciem meta-sieci

+ uczy się które decyzje są dobre / złe, i z czasem zawęża się już tylko do testowania samych dobrych sekwencji
- trudniejsze w zastosowaniu, wymaga wyboru architektury i hiperparametrów dla meta-sieci

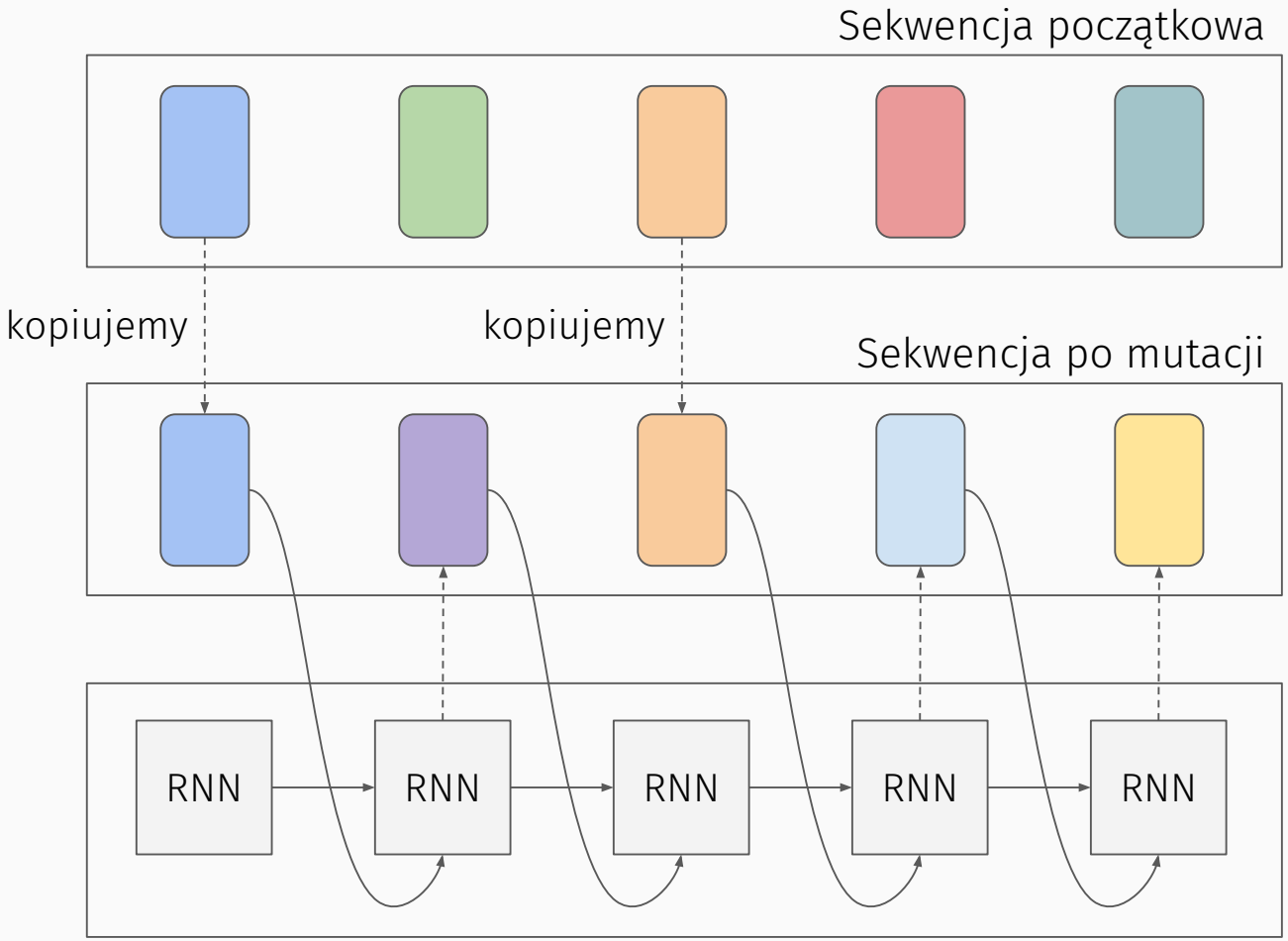
Podejście hybrydowe: Evo-NAS

Evolutionary-Neural Architecture Search (Evo-NAS)

Hybryda ewolucji i meta-sieci:

- wykonujemy ewolucję, gdzie mutacje podpowiadane są przez meta-sieć rekurencyjną
- każda decyzja w mutowanej sekwencji jest albo kopiowana, albo mutowana zgodnie z tym co radzi meta-sieć
- na początku mutacje są podpowiadane losowo, czyli hybryda zachowuje się tak jak czyste podejście ewolucyjne
- z czasem meta-sieć uczy się jakie mutacje są dobre

Evolutionary-Neural Architecture Search (Evo-NAS)



Eksperymenty

Problem syntetyczny

Zacznijmy od problemu syntetycznego, gdzie eksperymenty są bardzo tanie.

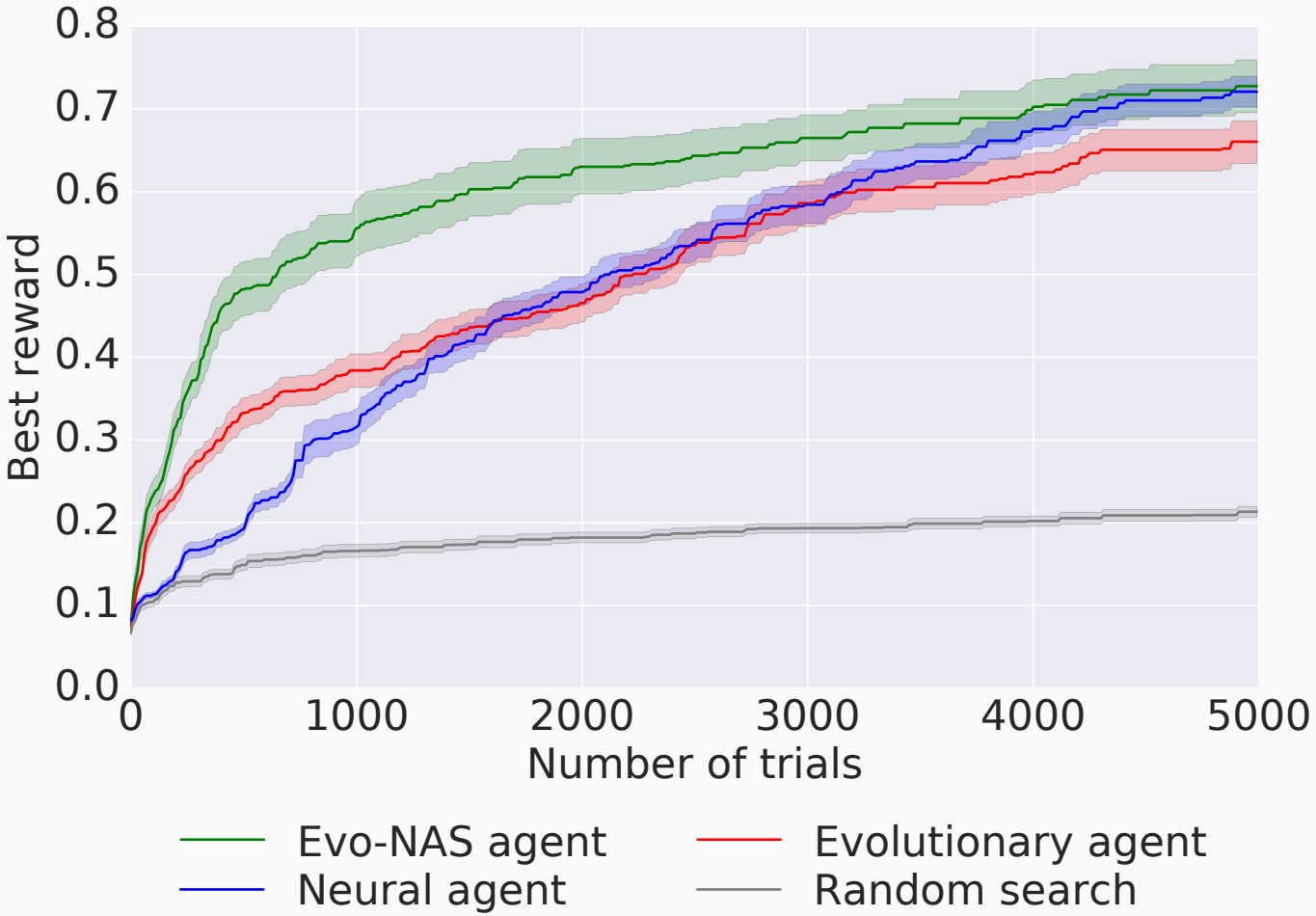
Problem syntetyczny

Wybieramy sewkencję n decyzji $a_i \in \{1, \cdots, n\}$. Wynik dla sekwencji $\mathsf{a} = (a_1, \cdots, a_n)$ to

$$r(\mathsf{a}) = \frac{n+1}{a_1^2 + \sum_{k=1}^{n-1} (a_{k+1} - a_k)^2 + (a_n - (n+1))^2}$$

Chcemy osiągnąć jak nawiększe $r(\mathsf{a})$.

Problem syntetyczny

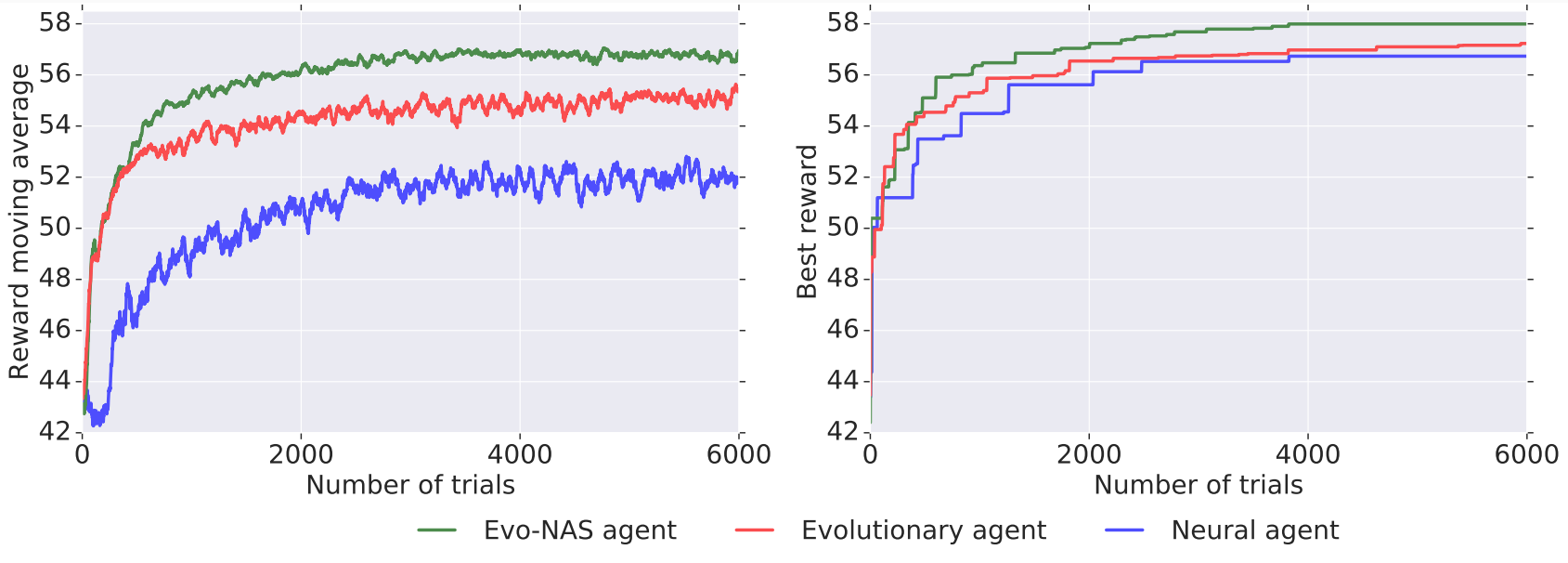


ImageNet

Problem

Wybieramy architekturę sieci konwolucyjnej dla zbioru ImageNet.

ImageNet



Po lewej: średnia krocząca wyniku poszczególnych prób.

Po prawej: najlepszy uzyskany dotychczas wynik.

Podsumowanie

Podsumowanie

Model hybrydowy (Evo-NAS) zawsze radzi sobie co najmniej tak dobrze, jak lepsze z wchodzących w jego skład podejść.

Często zaś radzi sobie istotnie lepiej, szczególnie jeśli chodzi o wyniki osiągane we wczesnej fazie procesu.

Więcej informacji

O tym samym temacie (ale przez 2h) opowiadam tutaj:

youtube.com/watch?v=VbGMYp8Wlzk

Zastosowania

Model jest używany przez niektóre zespoły w Google do optymalizacji architektur i hiperparametrów.

Temporal coding in spiking neural networks with alpha synaptic function

Iulia M. Comșa*[1], Krzysztof Potempa[1], Luca Versari[1], Thomas Fischbacher[1], Andrea Gesmundo[1], and Jyrki Alakuijala[1]

[1] *Google Research Zürich, Switzerland*

to solve temporally-encoded Boolean logic and other benchmark problems. We perform a search for the best set of hyperparameters for this model using evolutionary-neural hybrid agents [44] in order to solve temporally-encoded MNIST. The result improves the state-of-the-art accuracy on non-convolutional spiking

Bibliografia

[1] Real *et al.* (2018)
Regularized evolution for image classifier architecture search

[2] Zoph and Le (2016)
Neural architecture search with reinforcement learning

Dziękuję za uwagę!