Przykłady hiperuczenia sieci neuronowych

Jan Karwowski

Wydział Matematyki i Nauk Informacyjnych PW

15 I 2020

Definicje

(W tym wystąpieniu)

- Metauczenie – proces uczenia jest stosowany nie bezpośrednio do parametrów modelu, ale do pewnej przestrzeni, która jest potem rzutowana do przestrzeni parametrów.

$$f : Z \to \Theta$$

$$\min_{z \in Z} \mathcal{L}(M(f(z); \ldots))$$

- Hiperuczenie – funkcja f sama jest tego samego typu co model M.

Hipersieć

David Ha, Andrew M. Dai, and Quoc V. Le. "HyperNetworks". In: *CoRR* abs/1609.09106 (2016). arXiv: 1609.09106. URL: http://arxiv.org/abs/1609.09106

- Sieć, której wyjściem są wagi dla większej sieci
- Uczone parametry: wagi mniejszej sieci, wejścia mniejszej sieci
- Sugestia autorów
  - wielowarstwowa konwolucja – nadmiar parametrów, duża swoboda
  - RNN – wspólne parametry we wszystkich krokach czasowych – duże ograniczenie uczenia

Static Hypernetwork

- Sieć konwolucyjna
- D warstw, każdy filtr $f_{size} \times f_{size}$
- $K^j \in \mathbb{R}^{f_{size} \times N_{out} f_{size}}$

Architektura hipersieci

$$
\begin{aligned}
a_i^j &= W_i z^j + B_i, && \forall i = 1, .., N_{in}, \forall j = 1, ..., D \\
K_i^j &= \langle W_{out}, a_i^j \rangle + B_{out}, && \forall i = 1, .., N_{in}, \forall j = 1, ..., D \\
K^j &= \begin{pmatrix} K_1^j & K_2^j & ... & K_i^j & ... & K_{N_{in}}^j \end{pmatrix}, && \forall j = 1, ..., D
\end{aligned} \tag{2}
$$

- Trenujemy $W_i, B_i, W_{out}, B_{out}, z^j$
- Aktywacja liniowa

Większe kernele

$$K_{32\times 64} = \begin{pmatrix} K_1 & K_2 & K_3 & K_4 \\ K_5 & K_6 & K_7 & K_8 \end{pmatrix} \tag{3}$$

Wyniki I

MNIST

- 99.25%, normalnie uczona sieć konwolucyjna 99.28%
- warstwa $7 \times 7 \times 16 \times 16$
- wejście (embedding) $N_z = 4$

CIFAR-10

group name	output size	block type
conv1	32 × 32	[3× 3, 16]
conv2	32× 32	3× 3, 16× k 3× 3, 16× k × N
conv3	16× 16	3× 3, 32× k 3× 3, 32× k × N
conv4	8× 8	3× 3, 64× k 3× 3, 64× k × N
avg-pool	1 × 1	[8 × 8]

Table 1: Structure of Wide Residual Networks in Zagoruyko & Komodakis (2016). N determines the number of residual blocks in each group. Network width is determined by factor k

$N = 6, K = 1, 2 \qquad N_z = 64$

Wyniki II

CIFAR-10 Wyniki

Model	Test Error	Param Count
Network in Network (Lin et al., 2014)	8.81%	
FitNet (Romero et al., 2014)	8.39%	
Deeply Supervised Nets (Lee et al., 2015)	8.22%	
Highway Networks (Srivastava et al., 2015)	7.72%	
ELU (Clevert et al., 2015)	6.55%	
Original Resnet-110 (He et al., 2016a)	6.43%	1.7 M
Stochastic Depth Resnet-110 (Huang et al., 2016b)	5.23%	1.7 M
Wide Residual Network 40-1 (Zagoruyko & Komodakis, 2016)	6.85%	0.6 M
Wide Residual Network 40-2 (Zagoruyko & Komodakis, 2016)	5.33%	2.2 M
Wide Residual Network 28-10 (Zagoruyko & Komodakis, 2016)	4.17%	36.5 M
ResNet of ResNet 58-4 (Zhang et al., 2016)	3.77%	13.3 M
DenseNet (Huang et al., 2016a)	3.74%	27.2 M
Wide Residual Network 40-1[2]	6.73%	0.563 M
Hyper Residual Network 40-1 (ours)	8.02%	0.097 M
Wide Residual Network 40-2[2]	5.66%	2.236 M
Hyper Residual Network 40-2 (ours)	7.23%	0.148 M

Table 2: CIFAR-10 Classification with hypernetwork generated weights.

HyperRNN (Dynamic Hypernetwork)

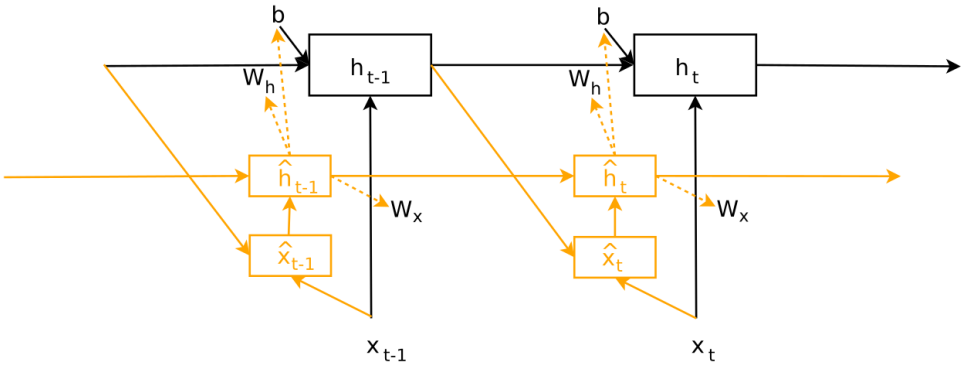


Figure 3: An overview of HyperRNNs. Black connections and parameters are associated basic RNNs. Orange connections and parameters are introduced in this work and associated with HyperRNNs. Dotted arrows are for parameter generation.

$$
\begin{aligned}
\hat{x}_t &= \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} \\
\hat{h}_t &= \phi(W_{\hat{h}} \hat{h}_{t-1} + W_{\hat{x}} \hat{x}_t + \hat{b}) \\
z_h &= W_{\hat{h}h} \hat{h}_{t-1} + b_{\hat{h}h} \\
z_x &= W_{\hat{h}x} \hat{h}_{t-1} + b_{\hat{h}x} \\
z_b &= W_{\hat{h}b} \hat{h}_{t-1}
\end{aligned}
\qquad
W(z) = W\big(d(z)\big) = \begin{pmatrix} d_0(z) W_0 \\ d_1(z) W_1 \\ \cdots \\ d_{N_h}(z) W_{N_h} \end{pmatrix} \tag{7}
$$

$$
\begin{aligned}
h_t &= \phi\big(W_h(z_h) h_{t-1} + W_x(z_x) + b(z_b)\big), \text{ where} \\
W_h(z_h) &= \langle W_{hz}, z_h \rangle \\
W_x(z_x) &= \langle W_{xz}, z_x \rangle \\
b(z_b) &= W_{bz} z_b + b_0
\end{aligned}
\tag{8}
$$

HyperLSTM I

Podstawowy LSTM

$$
\begin{aligned}
i_t &= W_h^i h_{t-1} + W_x^i x_t + b^i \\
g_t &= W_h^g h_{t-1} + W_x^g x_t + b^g \\
f_t &= W_h^f h_{t-1} + W_x^f x_t + b^f \\
o_t &= W_h^o h_{t-1} + W_x^o x_t + b^o \\
c_t &= \sigma(f_t) \odot c_{t-1} + \sigma(i_t) \odot \phi(g_t) \\
h_t &= \sigma(o_t) \odot \phi(c_t)
\end{aligned}
\tag{9}
$$

HyperLSTM II

HyperLSTM

$$
\begin{aligned}
\hat{x}_t &= \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} \\
\hat{i}_t &= LN(W_{\hat{h}}^{\hat{i}} \hat{h}_{t-1} + W_{\hat{x}}^{\hat{i}} \hat{x}_t + \hat{b}^{\hat{i}}) \\
\hat{g}_t &= LN(W_{\hat{h}}^{\hat{g}} \hat{h}_{t-1} + W_{\hat{x}}^{\hat{g}} \hat{x}_t + \hat{b}^{\hat{g}}) \\
\hat{f}_t &= LN(W_{\hat{h}}^{\hat{f}} \hat{h}_{t-1} + W_{\hat{x}}^{\hat{f}} \hat{x}_t + \hat{b}^{\hat{f}}) \\
\hat{o}_t &= LN(W_{\hat{h}}^{\hat{o}} \hat{h}_{t-1} + W_{\hat{x}}^{\hat{o}} \hat{x}_t + \hat{b}^{\hat{o}}) \\
\hat{c}_t &= \sigma(\hat{f}_t) \odot \hat{c}_{t-1} + \sigma(\hat{i}_t) \odot \phi(\hat{g}_t) \\
\hat{h}_t &= \sigma(\hat{o}_t) \odot \phi(LN(\hat{c}_t))
\end{aligned}
\tag{10}
$$

$y \in \{i, g, f, o\}$

$$
\begin{aligned}
z_h^y &= W_{\hat{h}h}^y \hat{h}_{t-1} + b_{\hat{h}h}^y \\
z_x^y &= W_{\hat{h}x}^y \hat{h}_{t-1} + b_{\hat{h}x}^y \\
z_b^y &= W_{\hat{h}b}^y \hat{h}_{t-1}
\end{aligned}
\tag{11}
$$

$$
\begin{aligned}
y_t &= LN\big(d_h^y \odot W_h^y h_{t-1} + d_x^y \odot W_x^y x_t + b^y(z_b^y)\big), \text{ where} \\
d_h^y(z_h) &= W_{hz}^y z_h \\
d_x^y(z_x) &= W_{xz}^y z_x \\
b^y(z_b^y) &= W_{bz}^y z_b^y + b_0^y
\end{aligned}
\tag{12}
$$

HyperLSTM – wyniki I

Model[1]	Test	Validation	Param Count
ME n-gram (Mikolov et al., 2012)	1.37		
Batch Norm LSTM (Cooijmans et al., 2016)	1.32		
Recurrent Dropout LSTM (Semeniuta et al., 2016)	1.301	1.338	
Zoneout RNN (Krueger et al., 2016)	1.27		
HM-LSTM[3] (Chung et al., 2016)	1.27		
LSTM, 1000 units [2]	1.312	1.347	4.25 M
LSTM, 1250 units[2]	1.306	1.340	6.57 M
2-Layer LSTM, 1000 units[2]	1.281	1.312	12.26 M
Layer Norm LSTM, 1000 units[2]	1.267	1.300	4.26 M
HyperLSTM (ours), 1000 units	1.265	1.296	4.91 M
Layer Norm HyperLSTM, 1000 units (ours)	1.250	1.281	4.92 M
Layer Norm HyperLSTM, 1000 units, Large Embedding (ours)	1.233	1.263	5.06 M
2-Layer Norm HyperLSTM, 1000 units	1.219	1.245	14.41 M

Table 3: Bits-per-character on the Penn Treebank test set.

HyperLSTM – wyniki II

Model[1]	enwik8	Param Count
Stacked LSTM (Graves, 2013)	1.67	27.0 M
MRNN (Sutskever et al., 2011)	1.60	
GF-RNN (Chung et al., 2015)	1.58	20.0 M
Grid-LSTM (Kalchbrenner et al., 2016)	1.47	16.8 M
LSTM (Rocki, 2016b)	1.45	
MI-LSTM (Wu et al., 2016)	1.44	
Recurrent Highway Networks (Zilly et al., 2016)	1.42	8.0 M
Recurrent Memory Array Structures (Rocki, 2016a)	1.40	
HM-LSTM[3] (Chung et al., 2016)	1.40	
Surprisal Feedback LSTM[4] (Rocki, 2016b)	1.37	
LSTM, 1800 units, no recurrent dropout[2]	1.470	14.81 M
LSTM, 2000 units, no recurrent dropout[2]	1.461	18.06 M
Layer Norm LSTM, 1800 units[2]	1.402	14.82 M
HyperLSTM (ours), 1800 units	1.391	18.71 M
Layer Norm HyperLSTM, 1800 units (ours)	1.353	18.78 M
Layer Norm HyperLSTM, 2048 units (ours)	1.340	26.54 M

Table 4: Bits-per-character on the `enwik8` test set.

HyperLSTM – wyniki III

Model	Log-Loss	Param Count
LSTM, 900 units (Graves, 2013)	-1,026	
3-Layer LSTM, 400 units (Graves, 2013)	-1,041	
3-Layer LSTM, 400 units, adaptive weight noise (Graves, 2013)	-1,058	
LSTM, 900 units, no dropout, no data augmentation.[1]	-1,026	3.36 M
3-Layer LSTM, 400 units, no dropout, no data augmentation.[1]	-1,039	3.26 M
LSTM, 900 units[2]	-1,055	3.36 M
LSTM, 1000 units[2]	-1,048	4.14 M
3-Layer LSTM, 400 units[2]	-1,068	3.26 M
2-Layer LSTM, 650 units[2]	-1,135	5.16 M
Layer Norm LSTM, 900 units[2]	-1,096	3.37 M
Layer Norm LSTM, 1000 units[2]	-1,106	4.14 M
Layer Norm HyperLSTM, 900 units (ours)	-1,067	3.95 M
HyperLSTM (ours), 900 units	-1,162	3.94 M

Table 5: Log-Loss of IAM Online DB validation set.

HyperLSTM – wyniki IV

Model	Test BLEU	Log Perplexity
Deep-Att + PosUnk (Zhou et al., 2016)	39.2	
GNMT WPM-32K, LSTM (Wu et al., 2016)	38.95	1.027
GNMT WPM-32K, ensemble of 8 LSTMs (Wu et al., 2016)	40.35	
GNMT WPM-32K, HyperLSTM (ours)	40.03	0.993

Table 6: Single model results on WMT En→Fr (newstest2014)

HyperNEAT – idea

Kenneth O. Stanley, David B. D'Ambrosio, and Jason Gauci. "A Hypercube-Based Encoding for Evolving Large-Scale Neural Networks". In: *Artificial Life* 15.2 (2009), pp. 185–212. DOI: 10.1162/artl.2009.15.2.15202. URL: https://doi.org/10.1162/artl.2009.15.2.15202

- Za pomocą metod ewolucyjnych (NEAT) zbudować sieć, która będzie generować parametry do sieci konwolucyjnej.
- Użycie specyficznych funkcji aktywacji, związanych z typową charakterystyka filtrów konwolucyjnyuch.
- Spostrzeżenie: wagi w sieciach często dają się kompresować: Jan Koutník, Faustino J. Gomez, and Jürgen Schmidhuber. "Evolving neural networks in compressed weight space". In: *Genetic and Evolutionary Computation Conference, GECCO 2010, Proceedings, Portland, Oregon, USA, July 7-11, 2010*. Ed. by Martin Pelikan and Jürgen Branke. ACM, 2010, pp. 619–626. ISBN: 978-1-4503-0072-8. DOI: 10.1145/1830483.1830596. URL: https://doi.org/10.1145/1830483.1830596.

NEAT

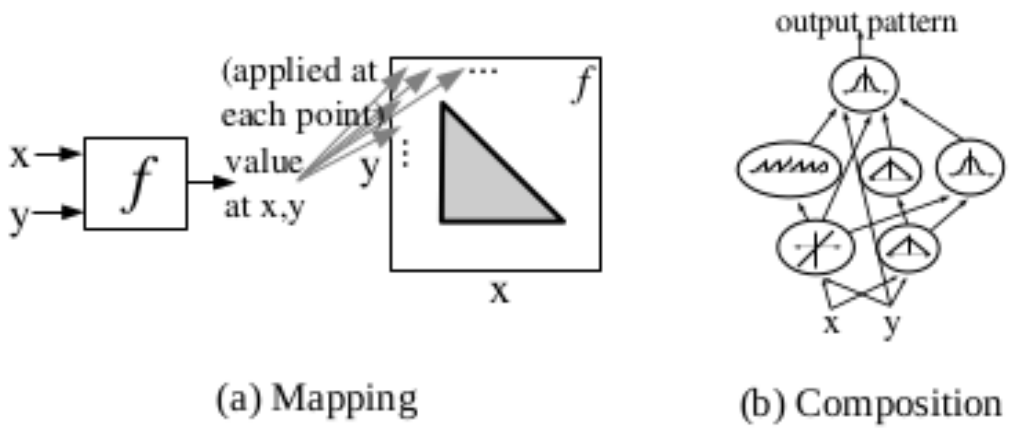
Kenneth O Stanley, Bobby D Bryant, and Risto Miikkulainen. "Evolving adaptive neural networks with and without adaptive synapses". In: *Evolutionary Computation, 2003. CEC'03. The 2003 Congress on*. Vol. 4. IEEE. 2003, pp. 2557–2564

- Ewolucja struktury sieci neuronowej (zazwyczaj) feedforward
- W początkowej populacji jeden neuron ukryty – ma zapewnić możliwie małą strukturę na koniec
- Mutacje dodają nowe połączenia i neurony, każda mutacja ma unikatowy identyfikator w całej populacji
- Krzyżowanie uwzględnia te same mutacje w obu osobnikach
- Niszowanie populacji

HyperNEAT – szczegóły I

Compositional Pattern Producing Network

- Sieć feedforward. Wejście: kilkuwymiarowy wektor $[0,1]^n$ opisujący współrzędne.
- Funkcje aktywacji: gausowska, sigmoida, sinus (lub inne okresowe), moduł, liniowa. Modelują typowe filtry wykrywające cechy w obrazach.

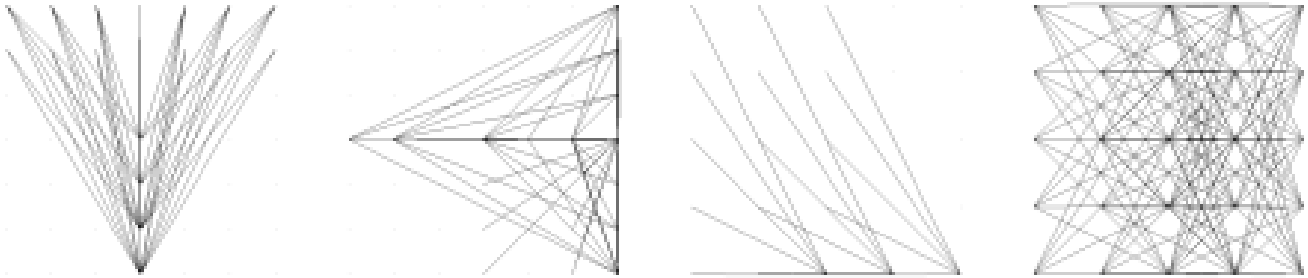


(a) Mapping (b) Composition

Figure 2: **CPPN Encoding.** (a) The function f takes arguments x and y, which are coordinates in a two-dimensional space. When all the coordinates are drawn with an intensity corresponding to the output of f, the result is a spatial pattern, which can be viewed as a phenotype whose genotype is f. (b) The CPPN is a graph that determines which functions are connected. The connections are weighted such that the output of a function is multiplied by the weight of its outgoing connection. Note that the topology is unconstrained and can represent any relationships.

HyperNEAT – szczegóły II

Compositional Pattern Producing Network – Przykład



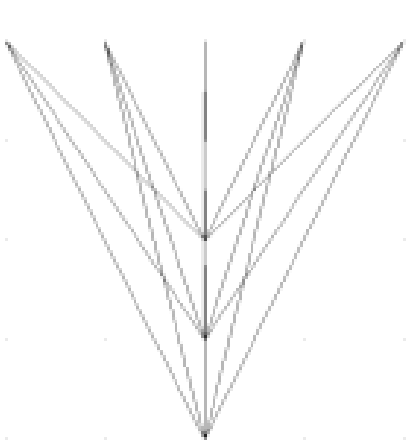
(a) Symmetry (b) Imperfect Sym. (c) Repetition (d) Rep. with Var.

Figure 5: **Connectivity Patterns Produced by Connective CPPNs.** These patterns, produced through interactive evolution, exhibit several important connectivity motifs: (a) bilateral symmetry, (b) imperfect symmetry, (c) repetition, and (d) repetition with variation. That these fundamental motifs are compactly represented and easily produced suggests the power of this encoding.

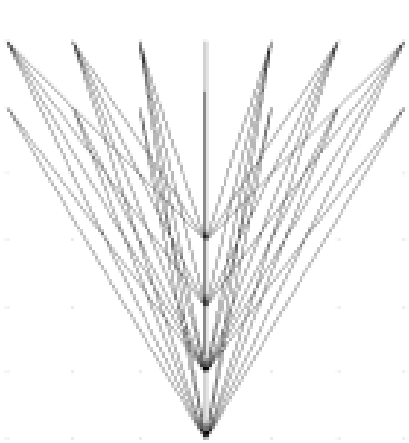
NEAT aplikowany co CPPN. Dodatkowo – wybór funkcji aktywacji.

HyperNEAT – szczegóły III

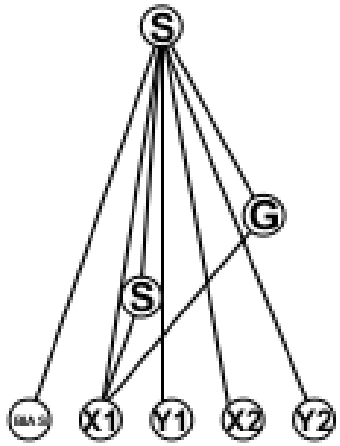
Zmiana rozdzielczości



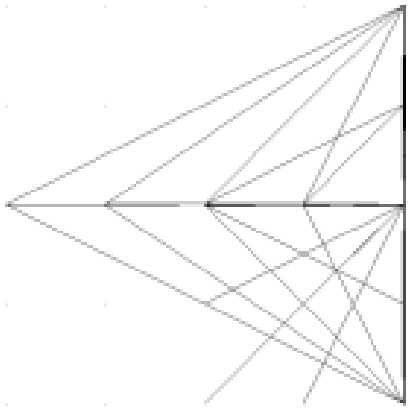
(a) Concept 1 at 5×5



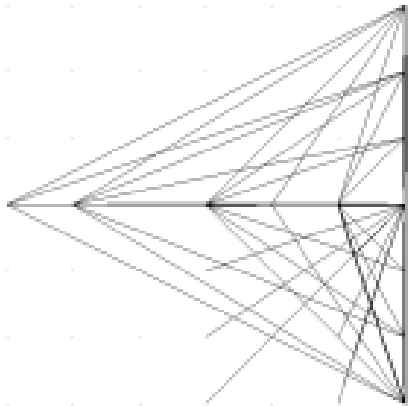
(b) Concept 1 at 7×7



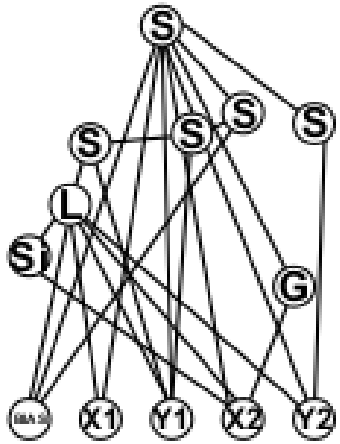
(c) Concept 1 CPPN



(d) Concept 2 at 5×5



(e) Concept 2 at 7×7



(f) Concept 2 CPPN

Wyniki HyperNEAT I

Eksperyment 1

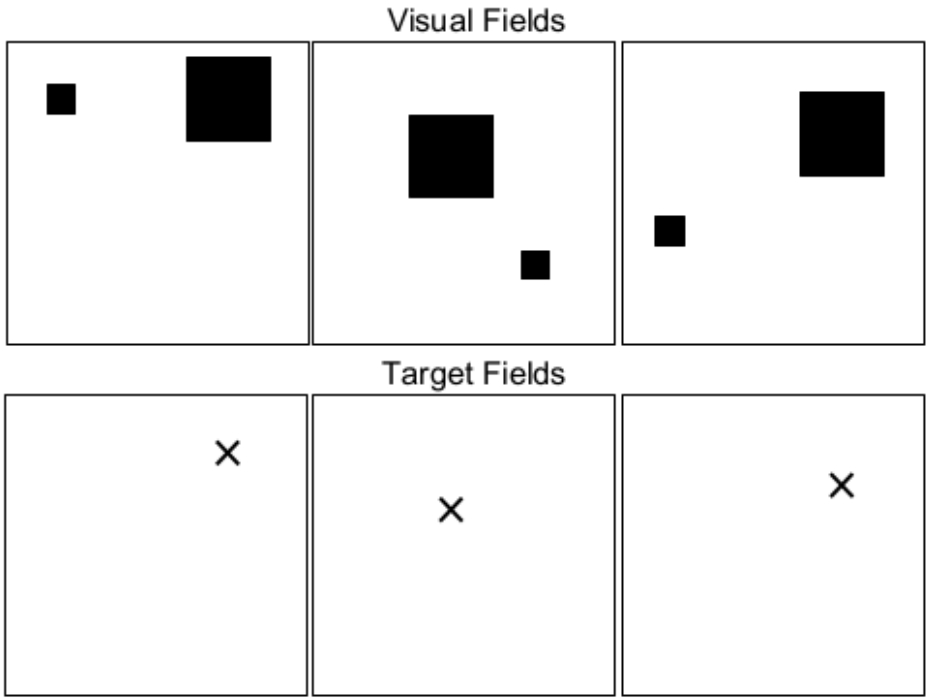
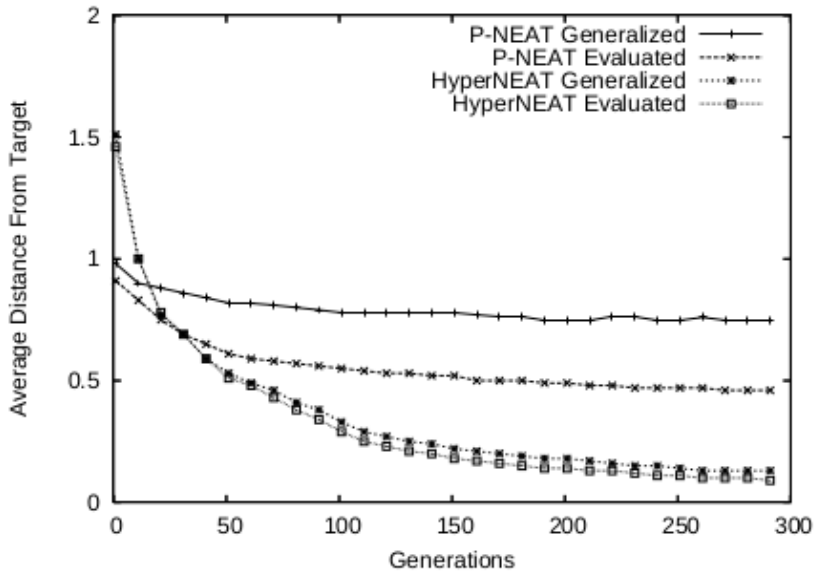


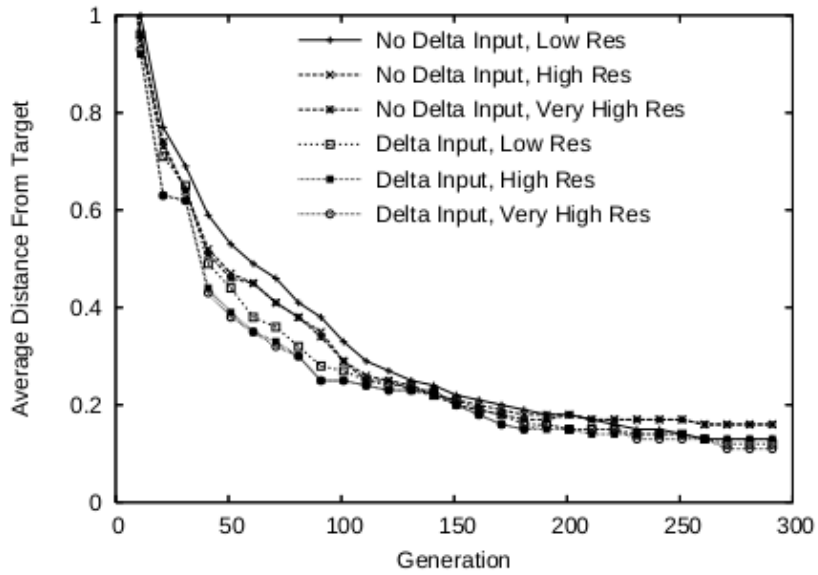
Figure 9: **The Visual Discrimination Task.** The task is to identify the center of the larger box. Example visual

Wyniki HyperNEAT II

Wyniki 1



(a) P-NEAT and HyperNEAT Generalization

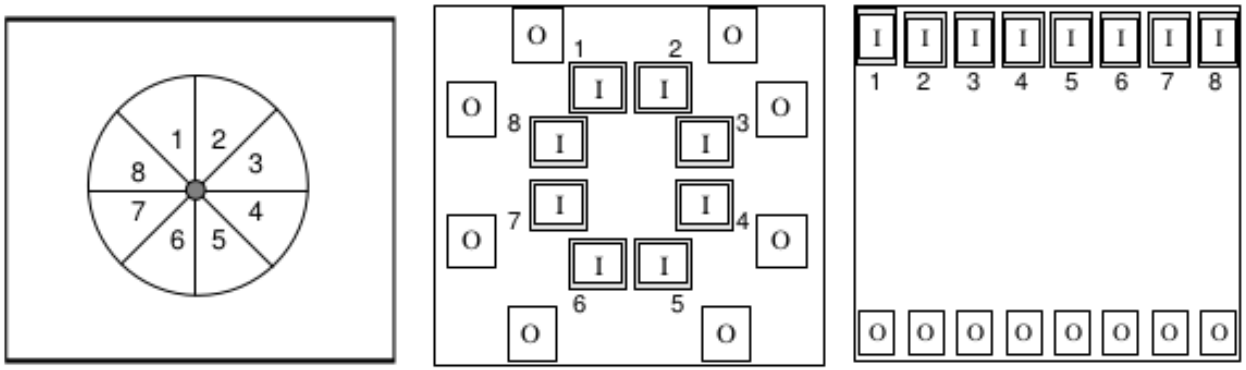


(b) HyperNEAT Generation Champion Scaling

Figure 10: **Generalization and Scaling.** The graphs show performance curves over 300 generations averaged over 20 runs each. (a) P-NEAT is compared to HyperNEAT on both evaluation and generalization. (b) HyperNEAT generation champions with and without delta inputs are evaluated for their performance on 11×11, 33×33, and 55×55 substrate resolutions. The results show that HyperNEAT generalizes significantly better than P-NEAT ($p < 0.01$) and scales almost perfectly.

Wyniki HyperNEAT III

Eksperyment 2 – Food Gathering

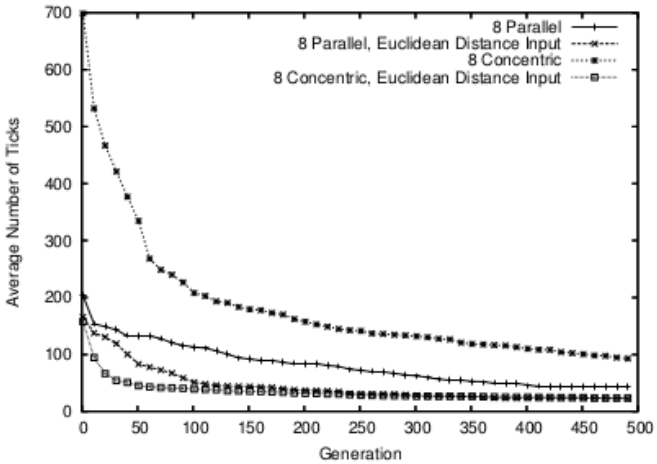


(a) Robot (b) Concentric (c) Parallel

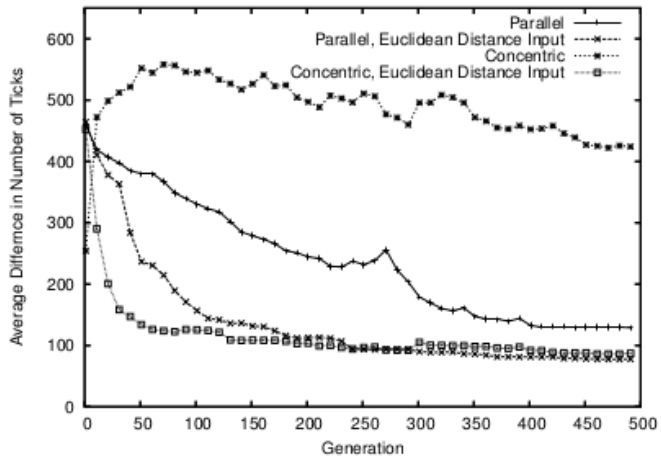
Figure 7: **Placing Inputs and Outputs.** A robot (a) is depicted with eight radar sensors along its circumference and eight motion effectors set at the same angle. In (b), the inputs (labeled *I*) and outputs (labeled *O*) are laid out literally according to the eight directions in space. In (c), the inputs are placed such at their location along x determines whether they represent a corresponding direction. Both arrangements create a geometric relationship between each input and its corresponding output. In this way, it is possible the give evolution a significant advantage from the start.

Wyniki HyperNEAT IV

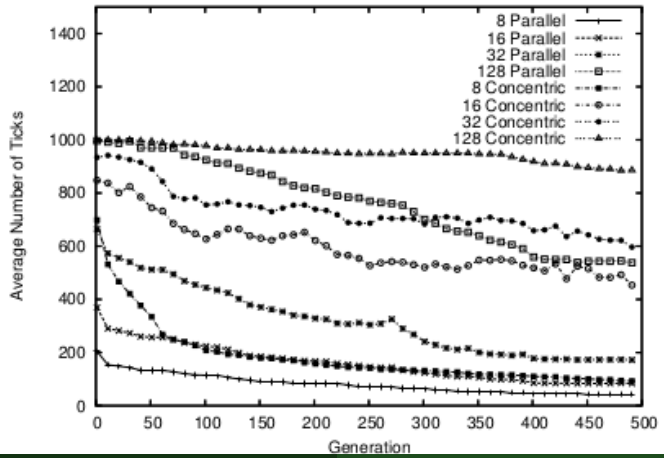
Wyniki 2 – Food Gathering

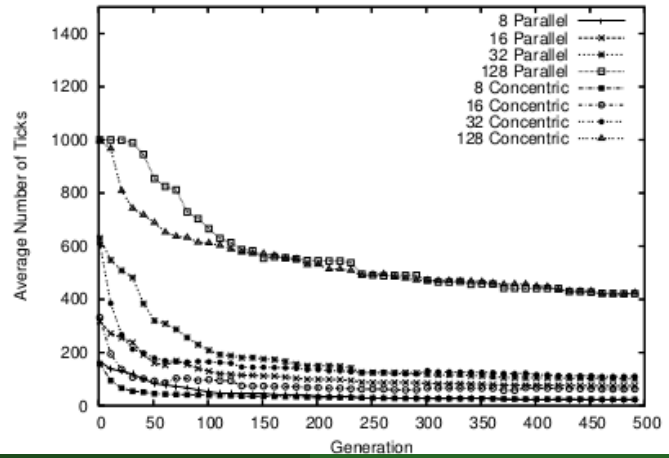


(a) Initial Performance



(b) Scalability





Enhanced HyperNEAT

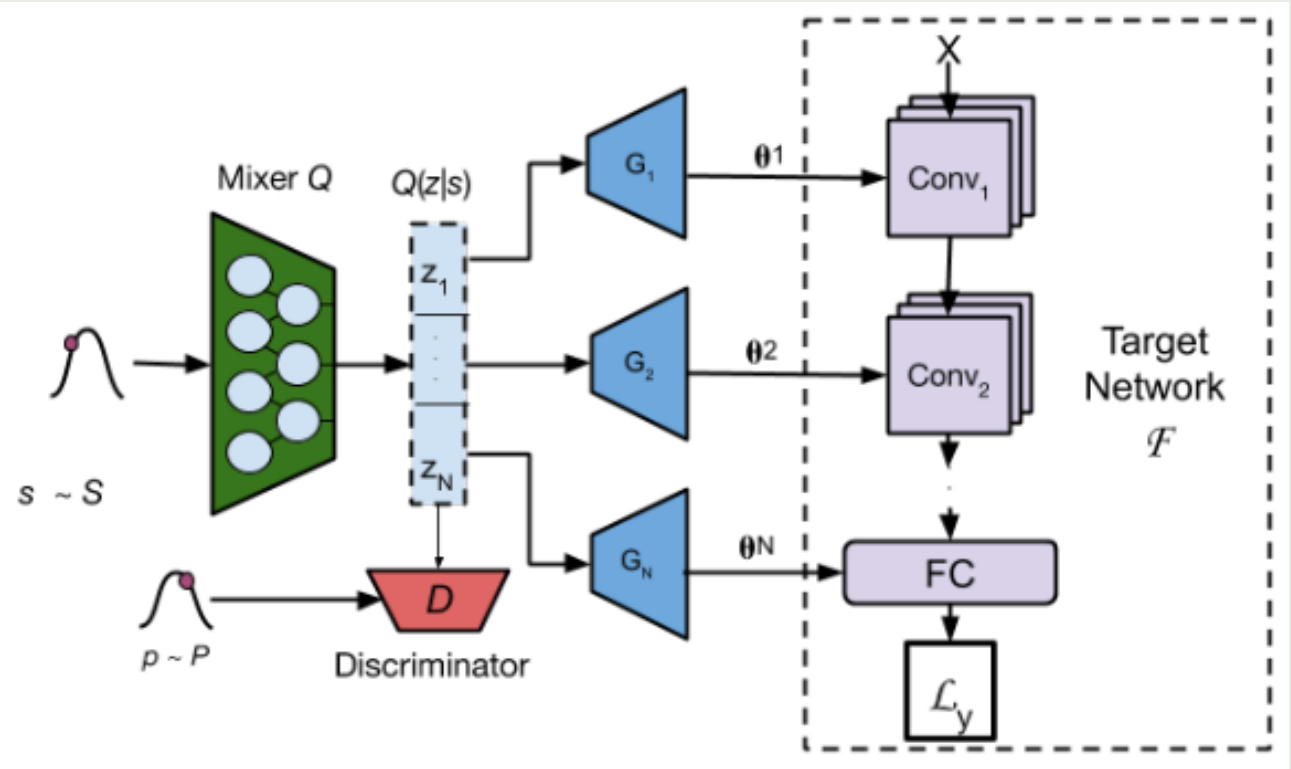
Sebastian Risi and Kenneth O. Stanley. "An Enhanced Hypercube-Based Encoding for Evolving the Placement, Density, and Connectivity of Neurons". In: *Artificial Life* 18.4 (2012), pp. 331–363. DOI: 10.1162/ARTL_a_00071. URL: https://doi.org/10.1162/ARTL_a_00071

HyperGAN I

Neale Ratzlaff and Fuxin Li. "HyperGAN: A Generative Model for Diverse, Performant Neural Networks". In: *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, 2019, pp. 5361–5369. URL: http://proceedings.mlr.press/v97/ratzlaff19a.html

HyperGAN II

Architektura



$$\inf_{G,Q} \mathbb{E}_{s\sim\mathcal{S}} \mathbb{E}_{(x,y)\sim(X,Y)} \left[\mathcal{L}(\mathcal{F}(x; G(Q(s))), y)\right] - \beta\mathcal{D}(Q(s), \mathcal{P}) \quad (2)$$

Eksperymenty I

Parametry

Both of our models take samples $s \sim S \in \mathbb{R}^{256}$ as input, but have different sized mixed latent spaces. For MNIST experiments, HyperGAN has weight generators, each taking a latent vector $z \sim Q(z|s) \in \mathbb{R}^{128}$ as input. The target network for the MNIST experiments is a small two layer convolutional network followed by 1 fully-connected layer, using leaky ReLU activations and 2x2 max pooling after each convolutional layer. For CIFAR-10, we use 5 weight generators with latent codes $z \sim Q(z|s) \in \mathbb{R}^{256}$. The target architecture for CIFAR-10 consists of three convolutional layers, each followed by leaky ReLU and 2x2 max pooling, followed by 2 fully connected layers.

The mixer, generators, and discriminator are each 2 layer MLPs with 512 units in each layer and ReLU nonlinearity. We found that larger generators offered little performance benefit, and ultimately hurt scalability. We trained our HyperGAN on MNIST using less than 1.5GB of memory on a single GPU, while CIFAR-10 used just 4GB, making HyperGAN surprisingly scalable.

Eksperymenty II

Wyniki – skuteczność

Method	MNIST	MNIST 5000	CIFAR-5	CIFAR-10
1 network	98.64 ±.3	96.69 ±.3	84.50 ±.6	76.32 ±.3
5 networks	98.75 ±.3	97.24 ±.14	85.51 ±.2	76.84 ±.1
10 networks	99.22 ±.09	97.33 ±.1	85.54 ±.2	77.52 ±.09
100 networks	**99.31 ±.02**	**97.71 ±.05**	**85.81 ±.02**	**77.71 ±.03**
APD	98.61	96.35	83.21	75.62
MNF	99.30	97.52	84.00	76.71
MC Dropout	98.73	95.58	84.00	72.75
Random Start	99.14	97.09	83.84	74.79

Table 2. Classification performance of each method on MNIST and CIFAR-10. CIFAR-5 refers to a dataset with only the first 5 classes of CIFAR-10. MNIST 5000 refers to training on only 5000 examples of MNIST, which has been used in prior work (e.g. (Krueger et al., 2017)). When held to the same architecture, the ensemble from HyperGAN performs better than ensembles using other approaches (100 model ensembles are used for APD, MNF and MC Dropout)

Eksperymenty III

Wyniki – różnorodność

	HyperGAN			Standard Training			APD		
	Conv1	Conv2	Linear	Conv1	Conv2	Linear	Conv1	Conv2	Linear
Mean	7.49	51.10	22.01	27.05	160.51	5.97	2.63	5.01	17.4
σ	1.59	10.62	6.01	0.31	0.51	0.06	0.22	0.41	1.43

Table 1. L_2-norm statistics on the layers of ensembles sampled from HyperGAN, compared to standard networks trained from different random initializations as well as samples from the posterior learned by APD. All models were trained on MNIST to 98% accuracy. Its easy to see that HyperGAN generates far more diverse networks

Eksperymenty IV

Wyniki – ataki

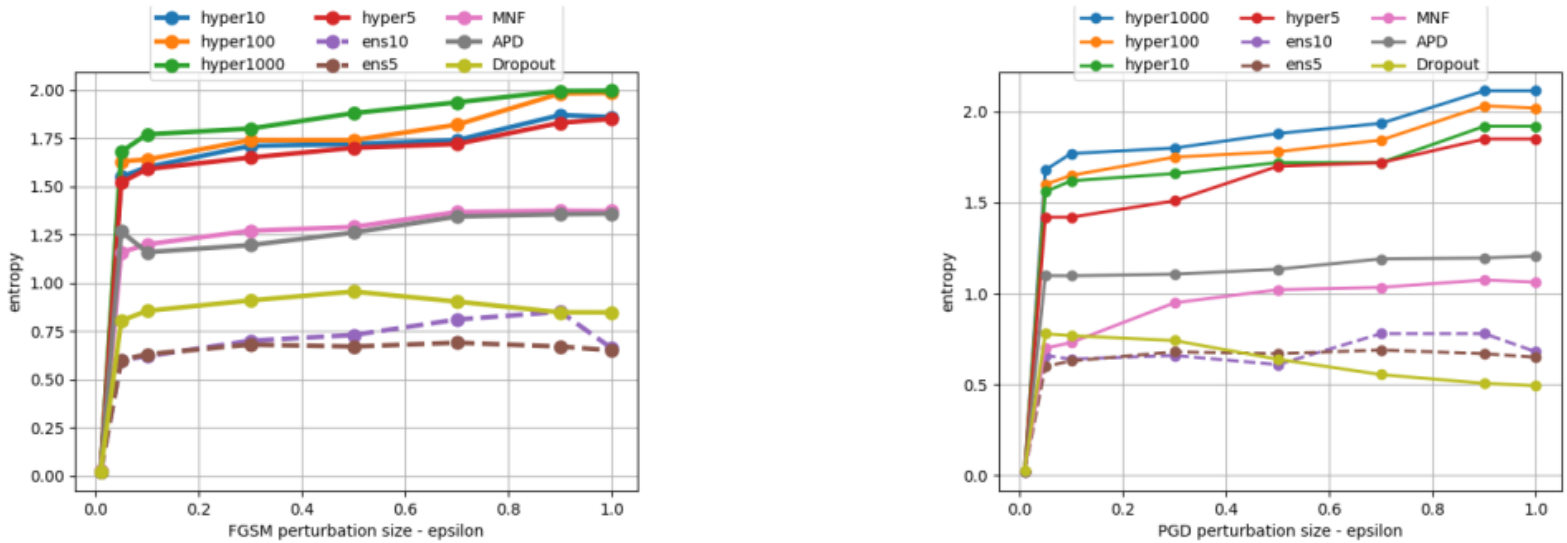


Figure 5. Entropy of predictions on FGSM and PGD adversarial examples. HyperGAN generates ensembles that are far more effective than standard ensembles even with equal population size. Note that for large ensembles, it is hard to find adversarial examples with small norms e.g. $\epsilon = 0.01$

Interpolacja w przestrzeni ukrytej I

Oscar Chang et al. “Agent Embeddings: A Latent Representation for Pole-Balancing Networks”. In: *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS '19, Montreal, QC, Canada, May 13-17, 2019*. Ed. by Edith Elkind et al. International Foundation for Autonomous Agents and Multiagent Systems, 2019, pp. 656–664. ISBN: 978-1-4503-6309-9. URL: http://dl.acm.org/citation.cfm?id=3331753

- Pole-balancing problem
- Generowanie sieci

Interpolacja w przestrzeni ukrytej II

Architektura sieci docelowej

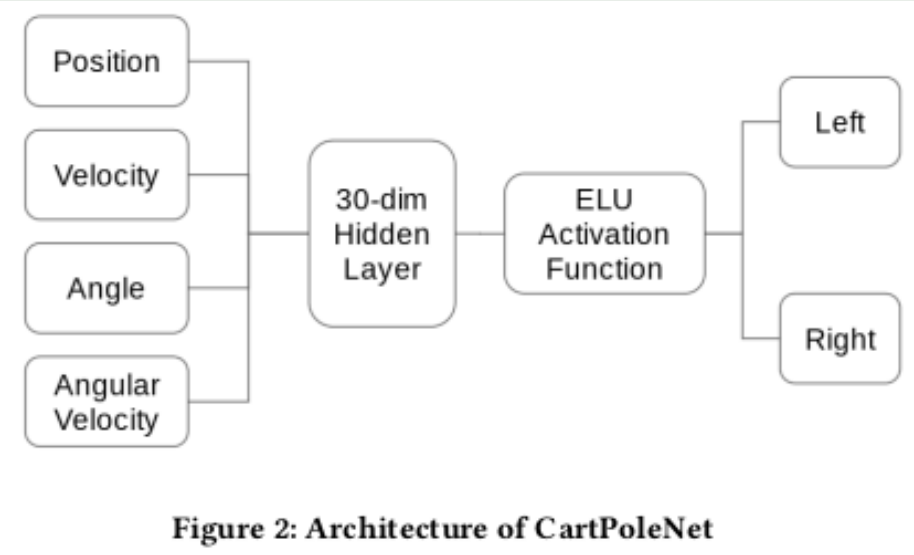


Figure 2: Architecture of CartPoleNet

- Funkcja straty wg. dyskontowanej wypłaty z 200 iteracji
- 74 000 nauczonych sieci
- podział na 4 zestawy wg. długości $[1, 50]$; $[51 - 100]$; $[101 - 150]$; $[151 - 200]$

Interpolacja w przestrzeni ukrytej III

Generator – VAE

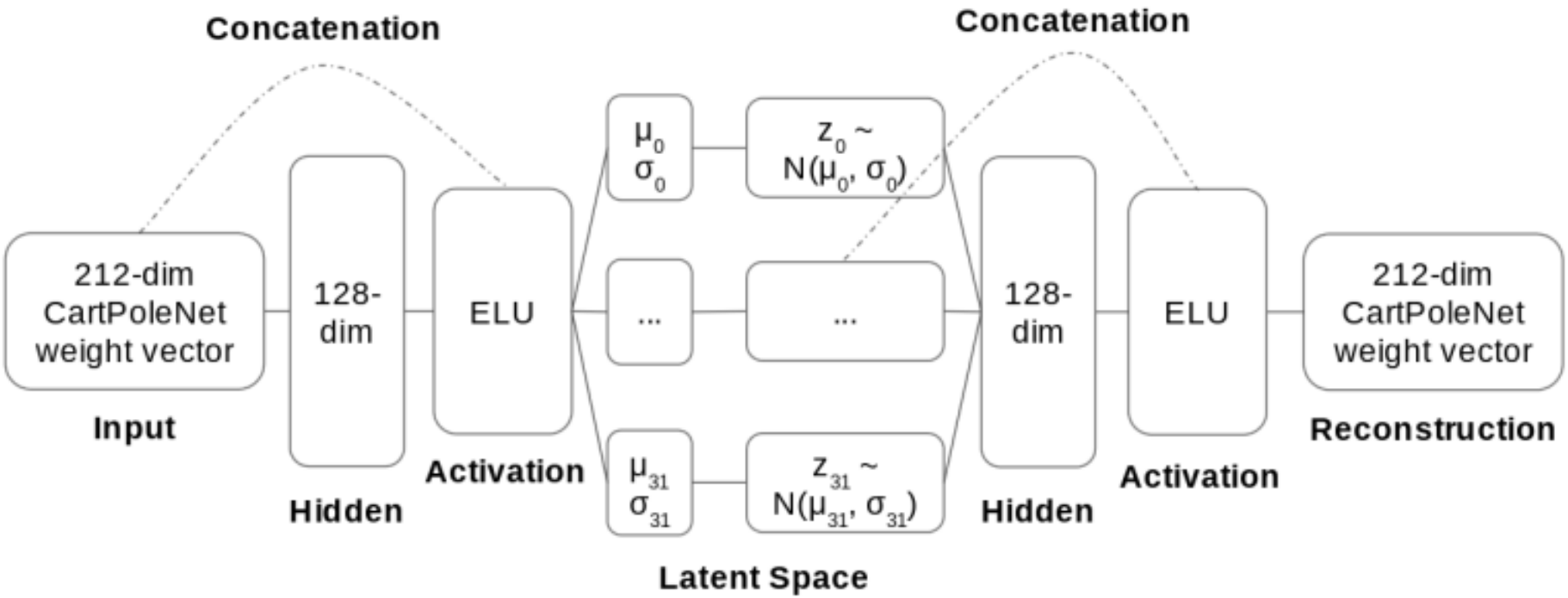


Figure 3: Architecture of CartPoleGen

Interpolacja w przestrzeni ukrytej IV

Wyniki

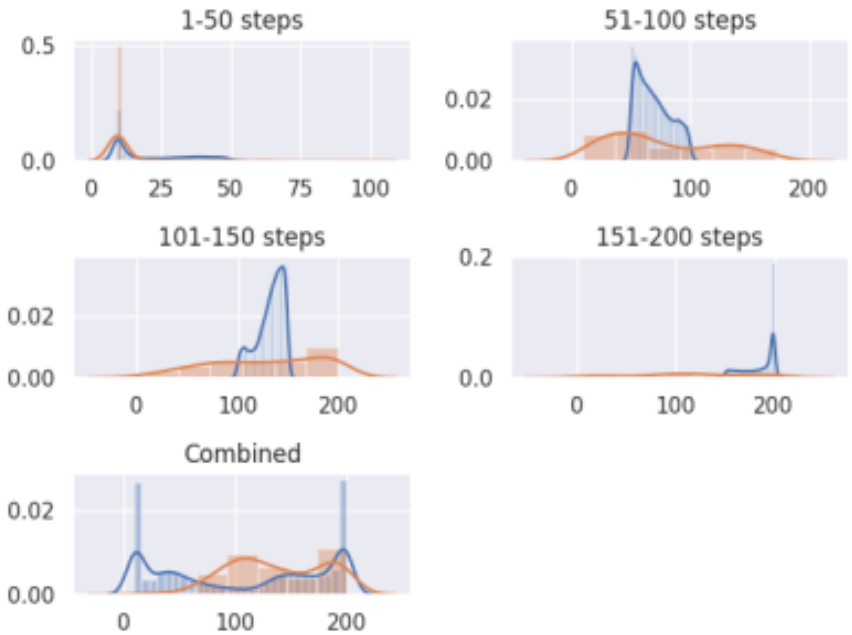


Figure 4: The figures are plotted as histograms, with KDE curves fitted on them. The x-axis denotes the survival time, and the y-axis denotes the percentage of networks with that survival time. The figures in blue represent the networks from the trainset, while the figures in orange represent the sampled networks.

Table 1: Sampling new instances of CartPoleNet

Group	Trainset Size	(Mean, Std) of Survival Time in Trainset	(Mean, Std) of Survival Time in Generated Samples
1 – 50 steps	25608	21.8, 11.5	11.0, 9.7
51 – 100 steps	9400	69.7, 14.2	77.3, 46.5
101 – 150 steps	10103	132.6, 13.1	127.0, 55.3
151 – 200 steps	28889	184.9, 16.3	116.4, 58.6
Combined	74000	106.7, 73.3	136.7, 42.8

Interpolacja w przestrzeni ukrytej V

Skalowanie

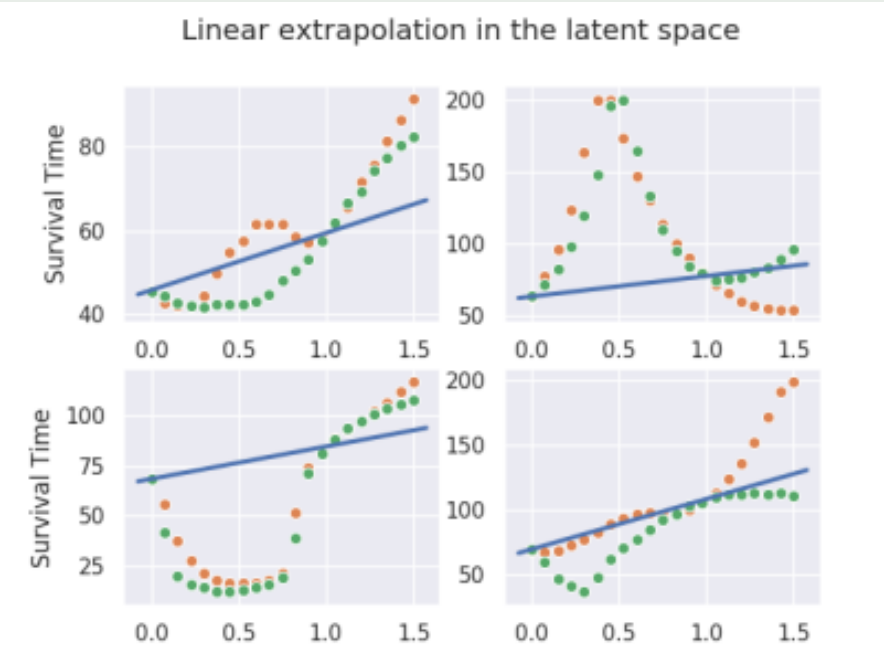


Figure 7: The x axis represents the coefficient of interpolation α, while the y axis represents the survival time of the sampled networks. The orange dots represent networks sampled from interpolating within the latent space, while the green dots represent networks interpolated within the weight space with the same coefficient of interpolation. The blue line is a straight line drawn from the survival time of the network sampled from the first agent embedding to the survival time of the network sampled from the second agent embedding.