Sieci neuronowe rozwiązujące wiele zadań jednocześnie

Mikołaj Małkiński
m.malkinski@mini.pw.edu.pl

15 grudnia 2021

Politechnika Warszawska
Szkoła Doktorska nr 3

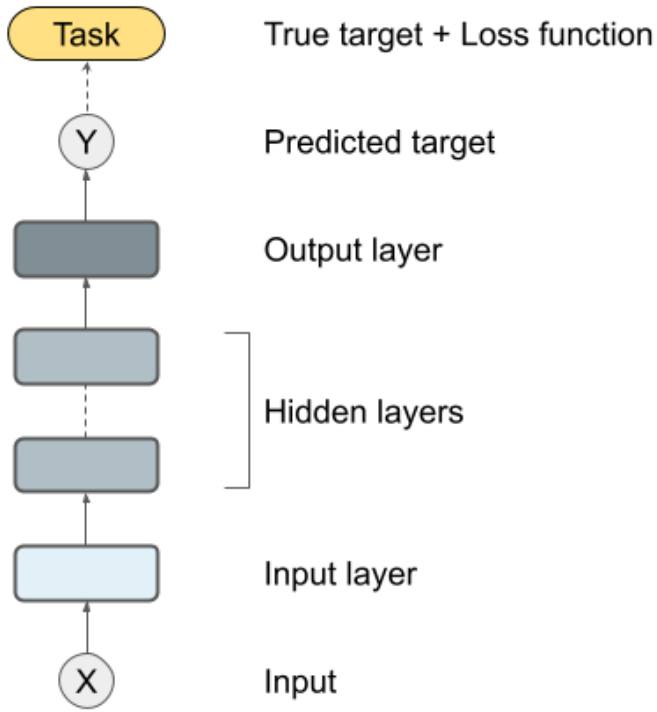
Spis treści

Single-task learning (STL)

Single-task learning

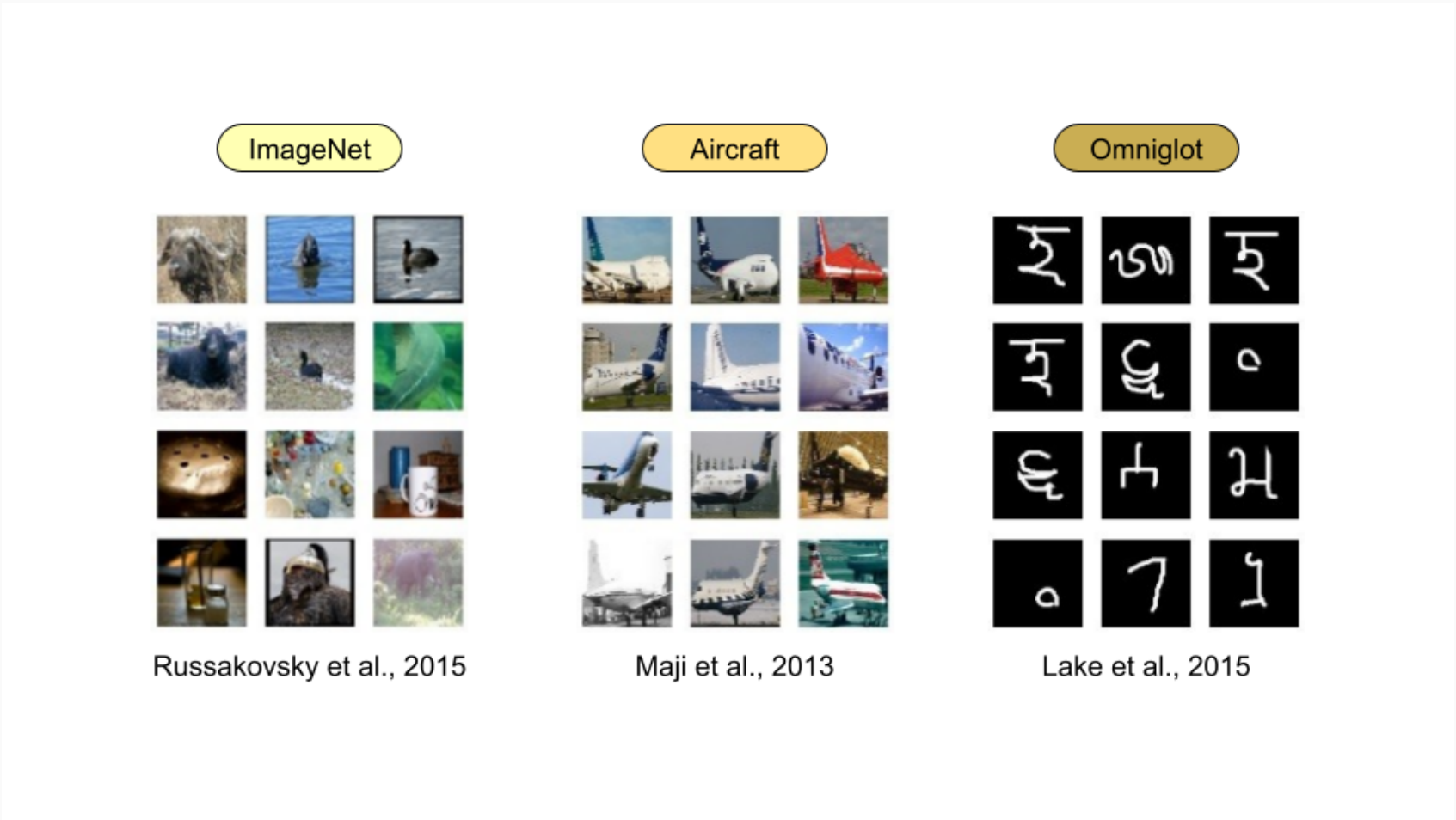
- (x, y) - zadanie
- $x \in \mathbb{R}^n$ - zmienne wejściowe
- $y \in \mathbb{R}^m$ - zmienne wyjściowe
- $\mathcal{F}$ - trenowany model
- $\hat{y} = \mathcal{F}(x)$

Single-task learning



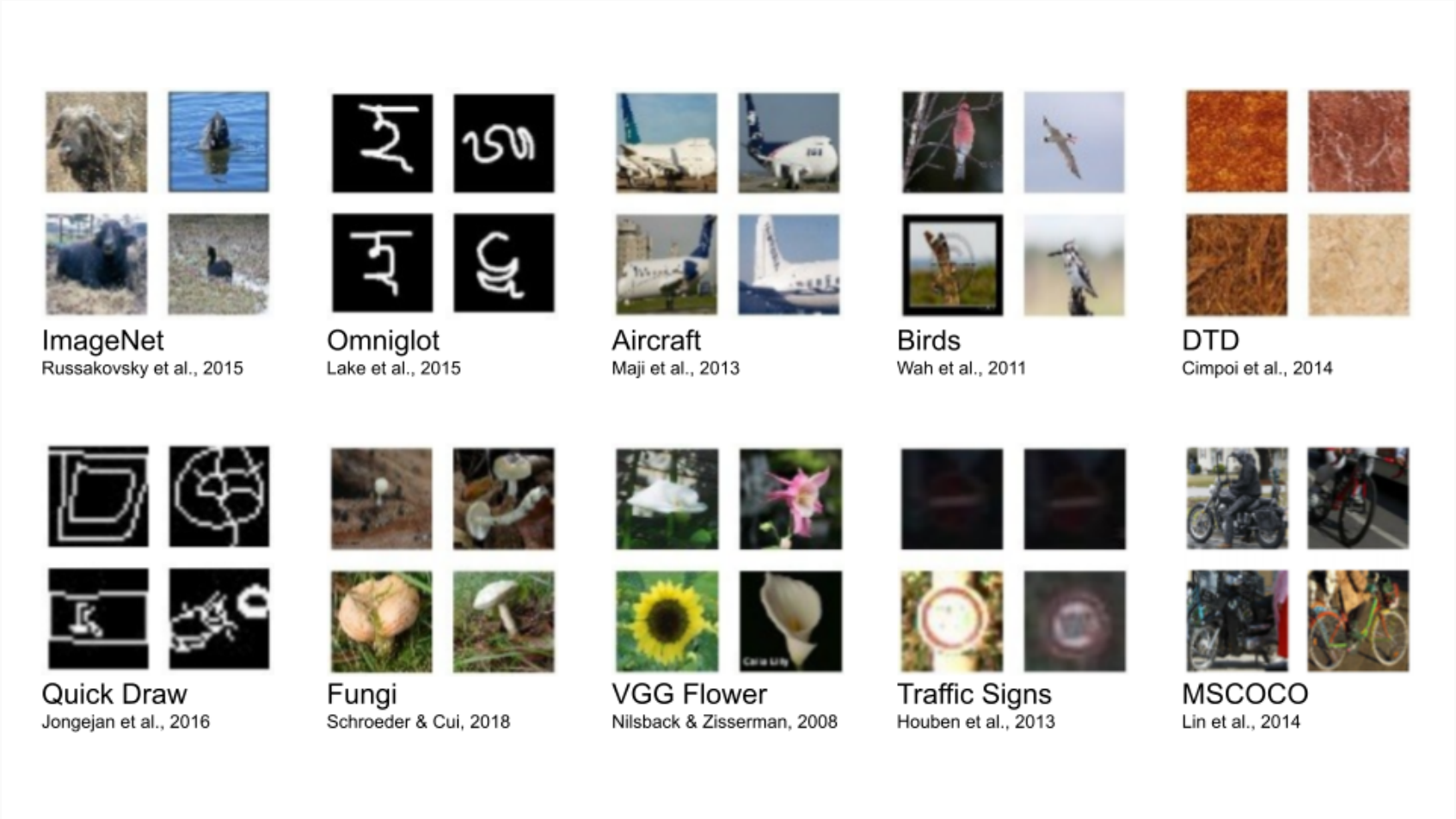
Rysunek 1: W klasycznym podejściu, do zadania trenowany jest nowy model.

Przykładowe zadania



Rysunek 2: Przykładowe zadania. Obrazki pochodzą z Meta-Dataset [10].

Meta-Dataset



Rysunek 3: Przykłady obrazków z Meta-Dataset [10].

STL do wielu zadań

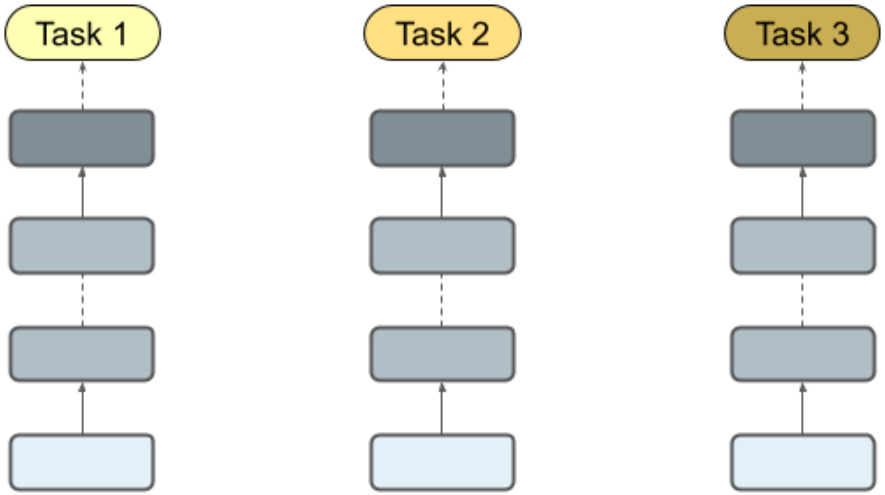
- $\{(x_t, y_t)\}_{t=1}^{T}$ - zadania (potencjalnie z różnych dziedzin)
- $x_t \in \mathbb{R}^{n_t}$ - zmienne wejściowe
- $y_t \in \mathbb{R}^{m_t}$ - zmienne wyjściowe
- T - liczba zadań
- $\mathcal{F}_t$ - odrębny model do każdego zadania
- $\hat{y}_t = \mathcal{F}_t(x_t)$

STL do wielu zadań

Dla zadań t i t' takich, że $t \neq t'$, może zachodzić:

- $\dim(\mathrm{x}_t) \neq \dim(\mathrm{x}_{t'})$
- $\dim(\mathrm{y}_t) \neq \dim(\mathrm{y}_{t'})$
- $\mathcal{F}_t \neq \mathcal{F}_{t'}$
- $\theta_{\mathcal{F}_t} \neq \theta_{\mathcal{F}_{t'}}$, gdzie $\theta_{\mathcal{F}_t}$ to parametry modelu $\mathcal{F}$ dopasowane do zadania t

STL do wielu zadań



Rysunek 4: W klasycznym podejściu, do każdego zadania trenowany jest oddzielny model.

Wyzwania

- Rozmiar zbioru danych
- Niewystarczająca wiedza zawarta w etykietach
- Nabyta wiedza nie jest wykorzystywana do innych zadań

Jak zbudować dobrze generalizujący model?

Możliwe podejścia

- Inductive transfer learning
- Self-supervised learning
- Zadania pomocnicze (auxiliary tasks)

- Multi-task learning (MTL)
  - Encoder-decoder MTL

- Transductive transfer learning
- Few-shot / One-shot / Zero-shot learning
- Domain adaptation
- ...

Multi-task Learning (MTL)

Multi-task learning

Wspólny model jest określony parametrami

$$\theta = \{\theta_{\mathcal{F}_t}\}_{t=1}^{T} \tag{1}$$

MTL polega na znalezieniu parametrów θ^* takich, że

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \frac{1}{T} \sum_{t=1}^{T} \frac{1}{N_t} \sum_{i=1}^{N_t} \mathcal{L}_t(\mathrm{y}_{t_i}, \hat{\mathrm{y}}_{t_i}) \tag{2}$$

gdzie

- N_t - liczba obserwacji w zadaniu t
- $\mathcal{L}_t$ - funkcja kosztu w zadaniu t

Podejścia enkoder-dekoder

- $\mathcal{F} = f \circ g$ - trenowany model
- f - enkoder
- g - dekoder

Podejścia enkoder-dekoder

Intra-domain	$\hat{y}_t = g_t(f(x_t))$
Semantyczne deskryptory zadań	$\hat{y}_t = g(f(x_t, z_t))$
Cross-domain	$\hat{y}_t = g_t(f_t(x_t))$
Variable Embeddings (TOM)	$\hat{y}_t = g(\Sigma_i f(x_i, z_i), z_j)$

Intra-domain

$\hat{y}_t = g_t(f(x_t))$

- Wspólny enkoder f
- Oddzielne dekodery g_t

Semantyczne deskryptory zadań

$\hat{y}_t = g(f(x_t, z_t))$

- Wspólny enkoder f
- Wspólny dekoder g
- z_t - task embeddings / zanurzenia zadań / semantyczne deskryptory zadań

Cross-domain

$\hat{y}_t = g_t(f_t(x_t))$

- Oddzielny enkoder f
- Oddzielny dekoder g
- Wymagane jest użycie mechanizmu do współdzielenia parametrów między zadaniami

Variable Embeddings (TOM)

$\hat{y}_t = g(\Sigma_i f(x_i, z_i), z_j)$

- Wspólny enkoder f
- Wspólny dekoder g
- z_* - variable embeddings / zanurzenia zmiennych; pozwalają zmapować zmienne wejściowe i wyjściowe do wspólnej przestrzeni

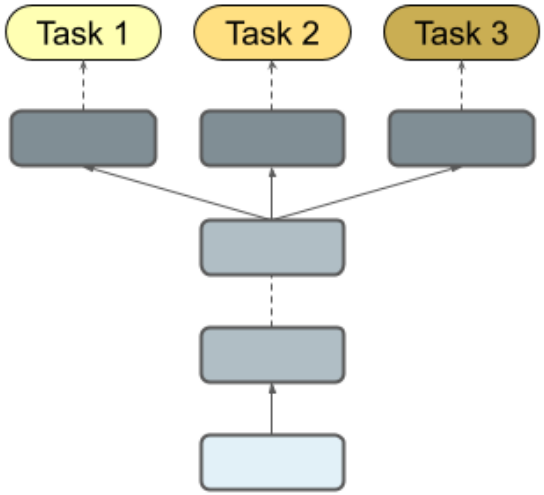
Intra-domain MTL

Intra-domain MTL

$\hat{y}_t = g_t(f(x_t))$

- Wspólny enkoder f
- Oddzielne dekodery g_t - zazwyczaj warstwy liniowe
- Ograniczenie: zbyt duży nacisk może być nałożony na dekodery
- Ograniczenie: zadania pochodzą z tej samej dziedziny (ten sam format danych wejściowych)

Intra-domain MTL



Rysunek 5: W podejściu intra-domain MTL, ten sam enkoder jest używany do rozwiązywania wielu zadań (hard parameter sharing).

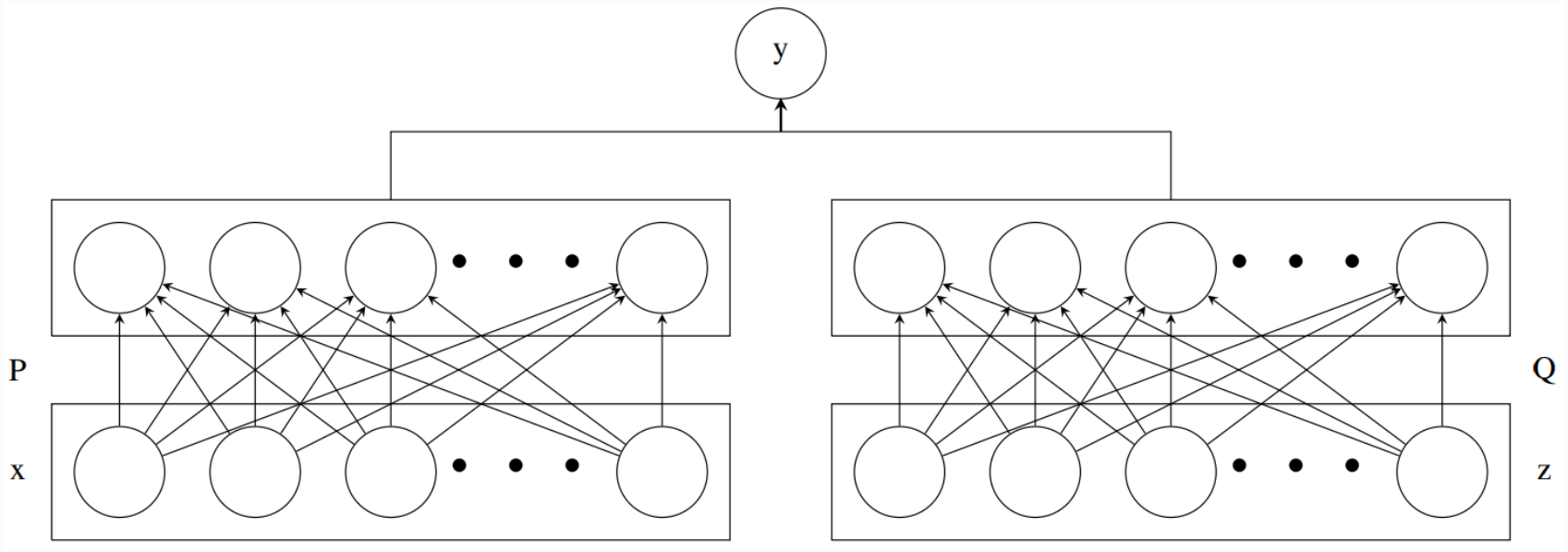
Semantyczne deskryptory zadań

Semantyczne deskryptory zadań

$\hat{y}_t = g(f(x_t, z_t))$

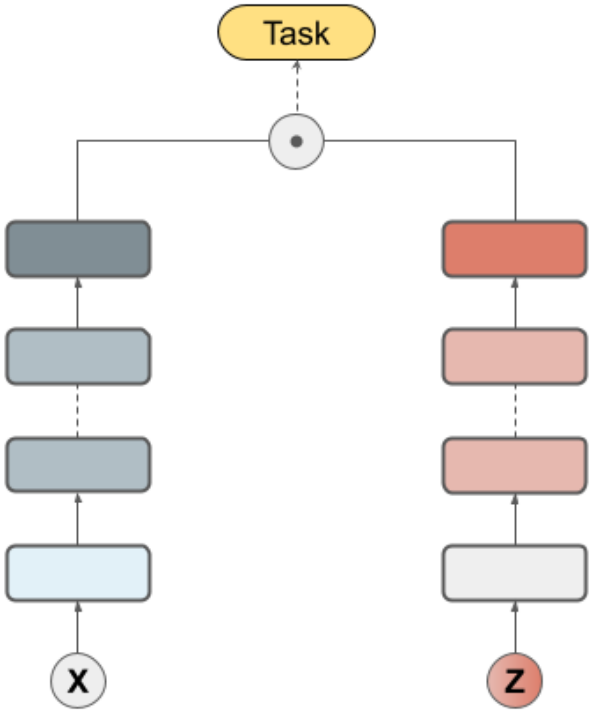
- Wspólny enkoder f oraz dekoder g do wszystkich zadań
- Wektory z_t opisują parametry charakterystyczne dla konkretnych zadań (dodatkowe wejście do modelu)
- Możliwość porównania zadań za pomocą wektorów z_t
- Ograniczenie: ta sama przestrzeń danych wejściowych i wyjściowych

Dwustronna sieć neuronowa



Rysunek 6: Dwustronna sieć neuronowa do MTL [11].

Dwustronna sieć neuronowa



Rysunek 7: Dwustronna sieć neuronowa do MTL.

Dwustronna sieć neuronowa

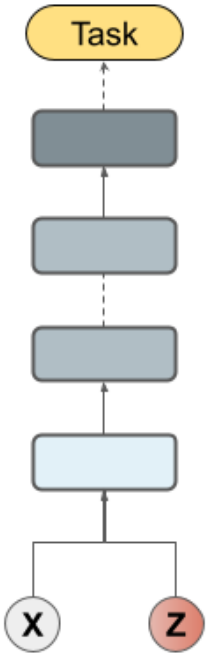
Optymalizacja dwustronnej sieci neuronowej [11] polega na znalezieniu parametrów θ^* takich, że

$$\theta^* = \underset{\theta_P, \theta_Q}{\operatorname{argmin}} \frac{1}{T} \sum_{t=1}^{T} \frac{1}{N_t} \sum_{i=1}^{N_t} \mathcal{L}_t(\mathrm{y}_{t_i}, \hat{\mathrm{y}}_{t_i}) \tag{3}$$

gdzie

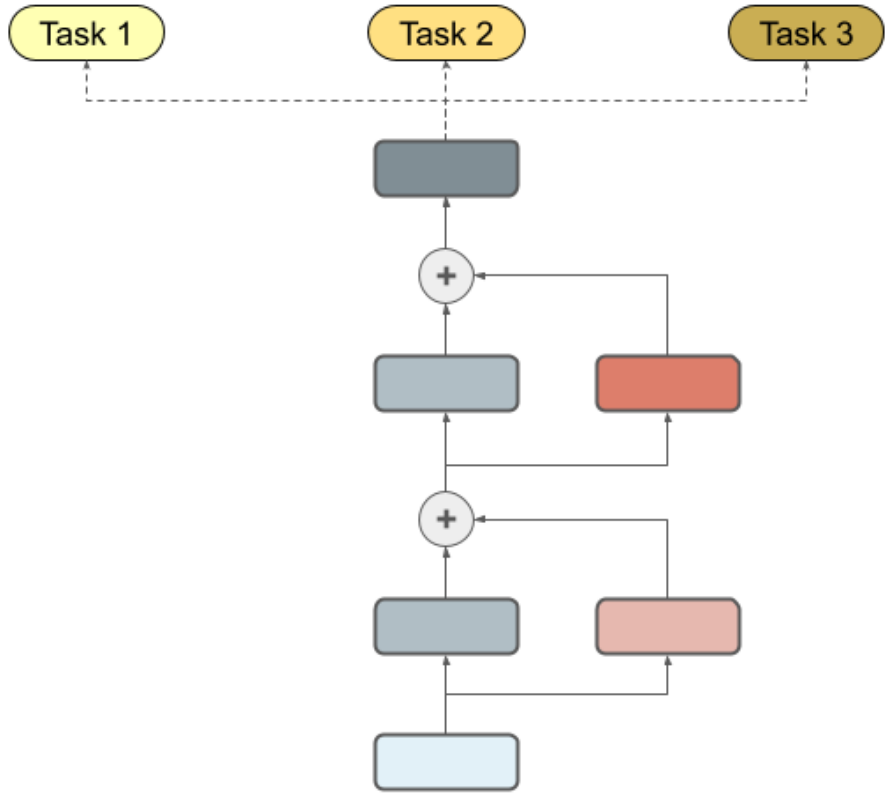
- $\hat{\mathrm{y}}_{t_i} = P(x_{t_i}) \bullet Q(z_t)$
- P i Q to sieci neuronowe
- z_t to deskryptor zadania (np. one-hot)

Zanurzenie zadania



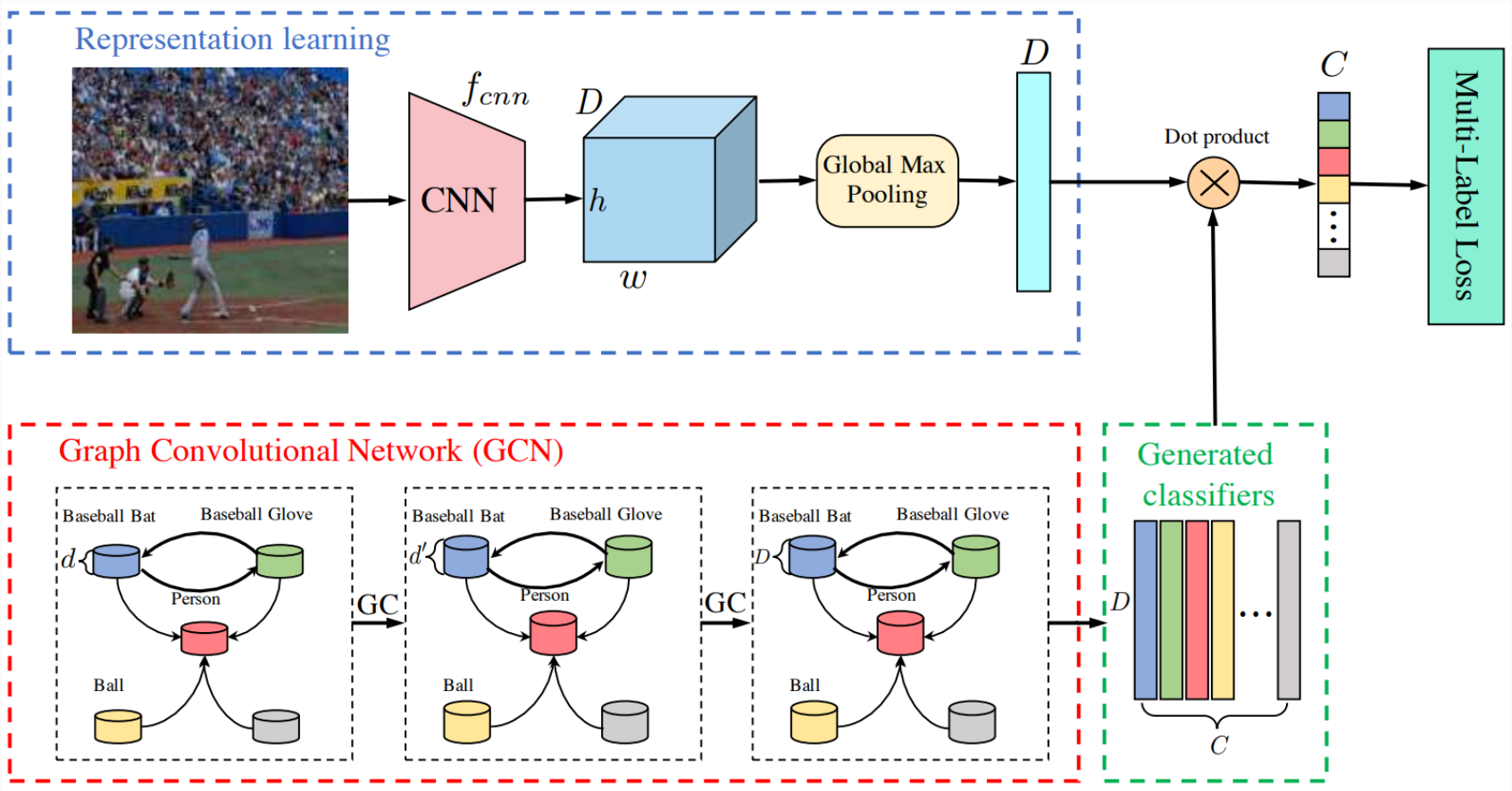
Rysunek 8: Zanurzenie zadania doklejone do danych wejściowych.

Dodatkowe parametry w modelu



Rysunek 9: Dodatkowe parametry w modelu, które są oddzielne do każdego zadania. Użyte np. w [9] do usprawnienia modelu BERT.

Relacje między deskryptorami



Rysunek 10: Relacje między deskryptorami można wykorzystać do poprawienia skuteczności modelu [1]. Problem klasyfikacji wielo-etykietowej (multi-label) można rozważać jako rozwiązywanie wielu zadań klasyfikacji binarnej jednocześnie.

Cross-domain MTL

Cross-domain

$\hat{y}_t = g_t(f_t(x_t))$

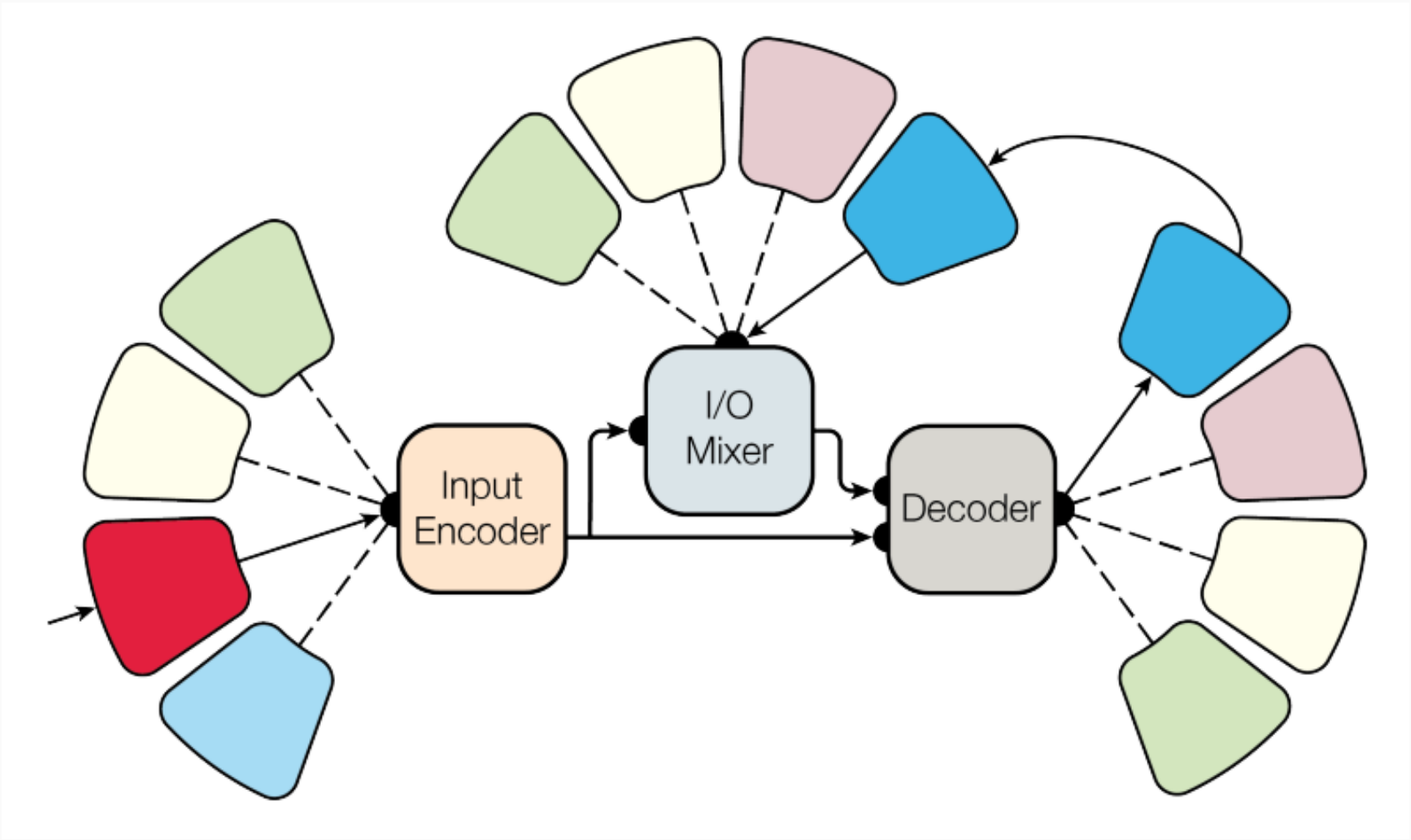
- Oddzielny enkoder f i dekoder g dla każdej dziedziny
- Pozwala na rozwiązywanie zadań o dowolnej wymiarowości danych wejściowych i wyjściowych
- Wymagane jest użycie mechanizmu do współdzielenia parametrów między zadaniami
- Ograniczenie: w wielu przypadkach, oddzielne enkodery i dekodery utrudniają skuteczne współdzielenie wiedzy

MultiModel

MultiModel [2]:

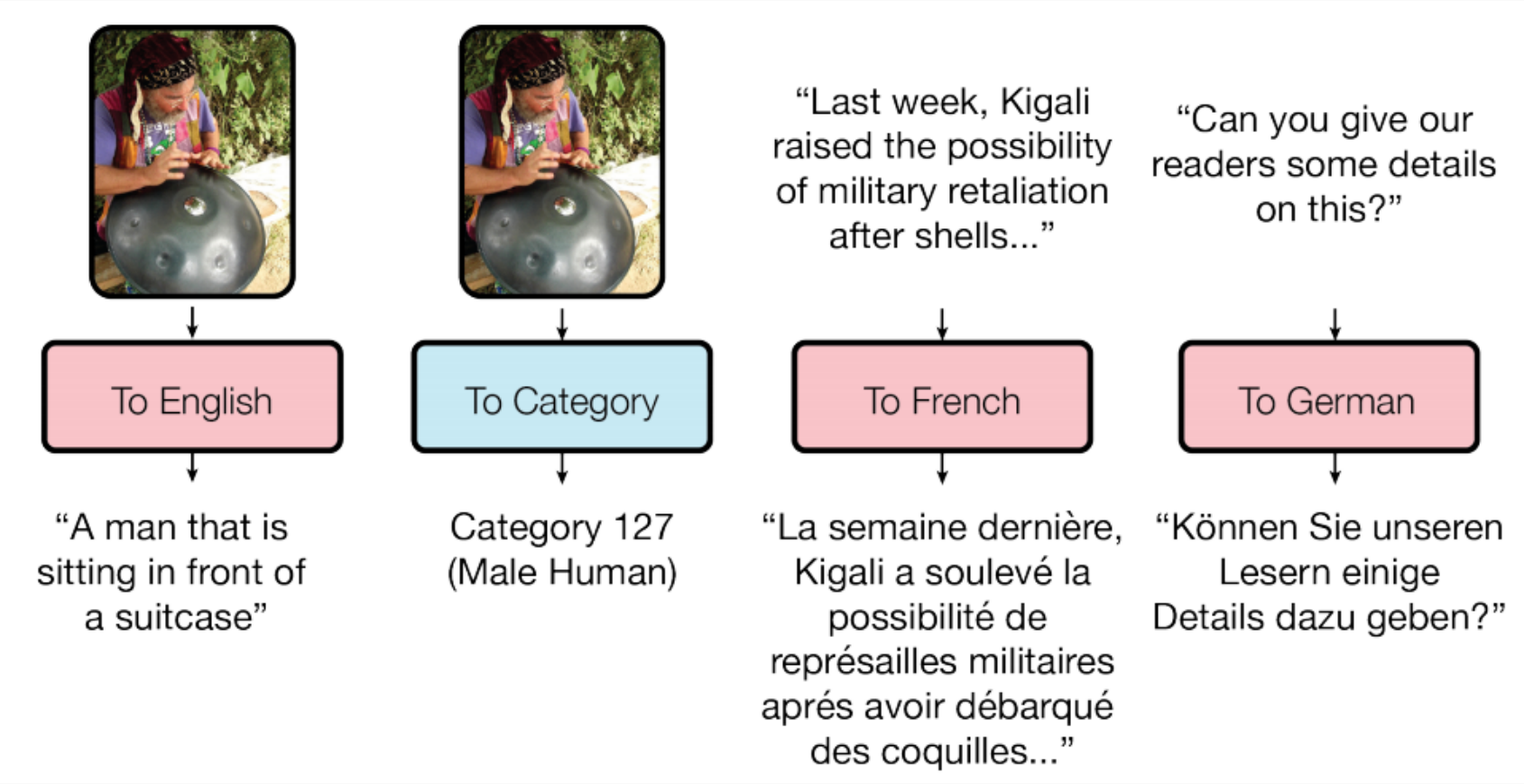
- Modalności: tekst, obraz, dźwięk, dane kategoryczne (łącznie 8 zadań)
- Oddzielny enkoder do każdej modalności
- Ukryta reprezentacja wspólna dla wszystkich modalności
- Różne moduły obliczeniowe (sieci konwolucyjne, mechanizm atencji/uwagi, mikstura ekspertów)

MultiModel



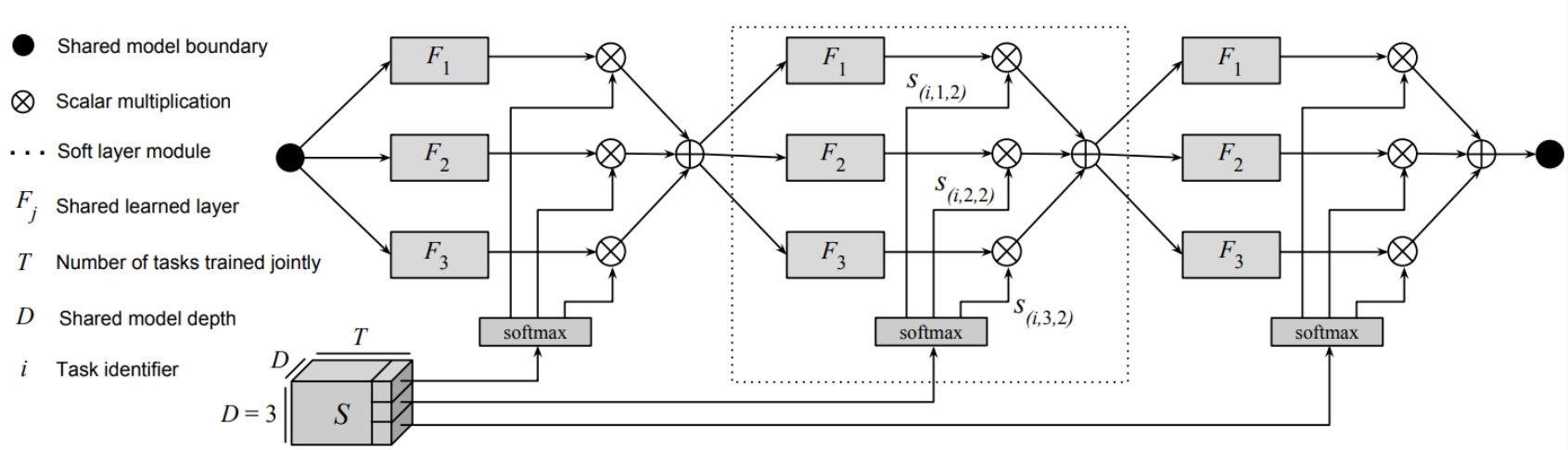
Rysunek 11: MultiModel składający się z enkodera czterech modalności, mixera oraz auto-regresywnego dekodera [2].

MultiModel



Rysunek 12: MultiModel jest w stanie dekodować dane do odpowiednich zadań dzięki tokenowi, który jest wybierany przed rozpoczęciem dekodowania [2].

Dostosowanie kolejności warstw



Rysunek 13: W rozwiązywaniu różnorodnych zadań, pomocne może być dostosowanie kolejności warstw modelu do każdego zadania oddzielnie [6].

Traveling observer model

Traveling observer model

- Ugruntowanie wszystkich obserwacji we wspólnej przestrzeni
- Rozważmy zbiór wszystkich zmiennych losowych, które mogą zostać zmierzone $\{v_1, v_2, \ldots\} = V, v_i \in \mathbb{R}$
- Każda zmienna v_i może być wejściem lub wyjściem do jakiegoś zadania predykcyjnego
- Przypiszmy do każdej zmiennej v_i wektor $z_i \in \mathbb{R}^C$, który określa jej znaczenie
- z_i to embedding/zanurzenie zmiennej (ang. *variable embedding* (VE))

Traveling observer model

- Rozważmy zadanie określone poprzez zaobserwowane zmienne (x, y)
- $x \in \mathbb{R}^n$ oraz $y \in \mathbb{R}^m$
- Skoro V zawiera wszystkie mierzalne zmienne losowe, to $\{x_i\}_{i=1}^n \subseteq V$ oraz $\{y_j\}_{j=1}^m \subseteq V$
- Celem jest znalezienie funkcji Ω, która może być zastosowana do dowolnych zadań

Traveling observer model

Niech z_i będzie VE odpowiadającym x_i, a z_j odpowiadającym y_j. Wtedy uniwersalny model predykcyjny można sformułować jako:

$$\mathbb{E}[y_j|x] = \Omega(x, \{z_i\}_{i=1}^{n}, z_j) \tag{4}$$

Dodatkowo, funkcja Ω powinna:

- móc zostać zaaplikowana do dowolnej liczby zmiennych wejściowych
- móc przewidywać wartości dowolnej liczby zadań
- być obojętna na kolejność zmiennych

Traveling observer model

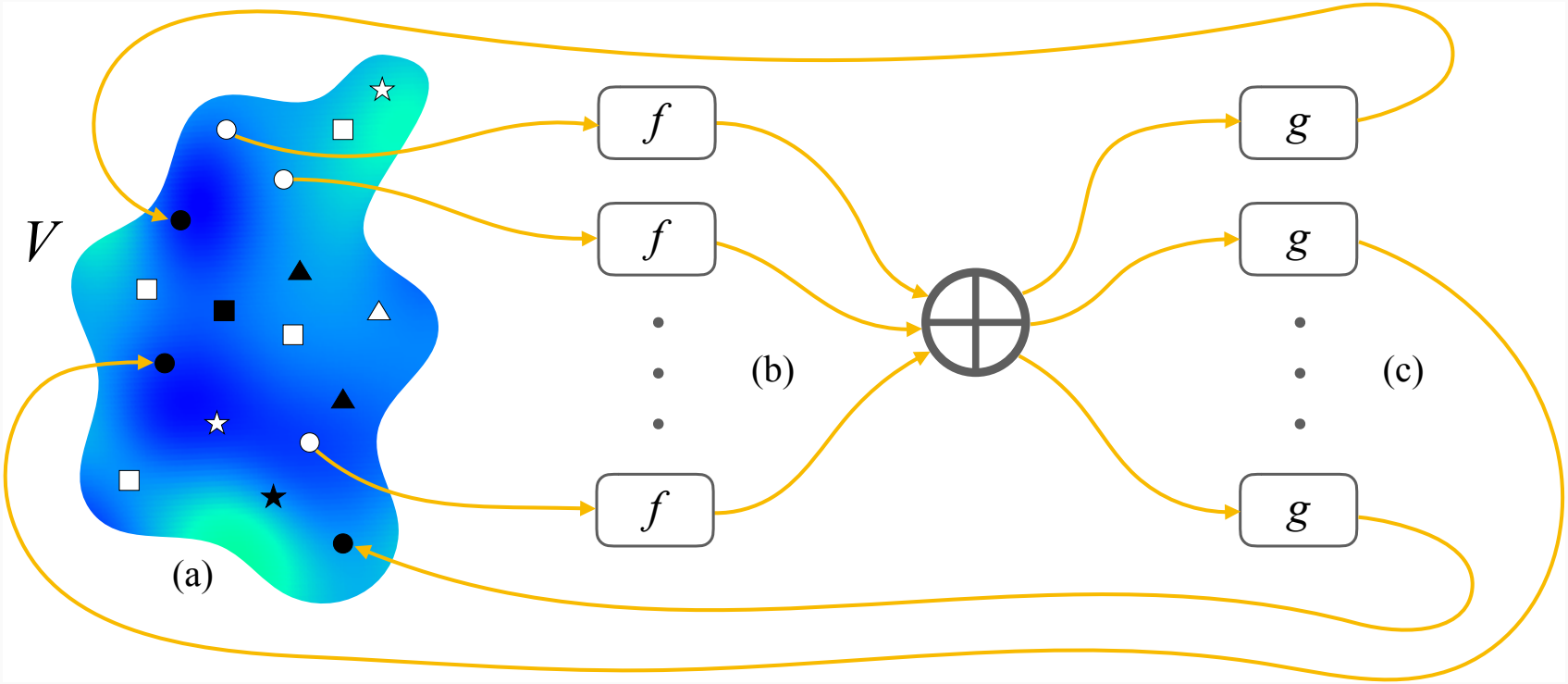
Powyższe wymagania prowadzą do:

$$\mathbb{E}[y_j|x] = \Omega(x, \{z_i\}_{i=1}^n, z_j) = g\Big(\sum_{i=1}^{n} f(x_i, z_i), z_j\Big) \tag{5}$$

gdzie:

- $f : \mathbb{R}^{C+1} \to \mathbb{R}^M$ to enkoder
- $g : \mathbb{R}^{M+C} \to \mathbb{R}$ to dekoder
- $M \in \mathbb{Z}$ to rozmiar ukrytej wymiarowości zależny od implementacji f i g

Traveling observer model



Rysunek 14: Traveling Observer Model [7].

Traveling observer model

Aby ułatwić proces uczenia, wartość oczekiwana jest liczona jako:

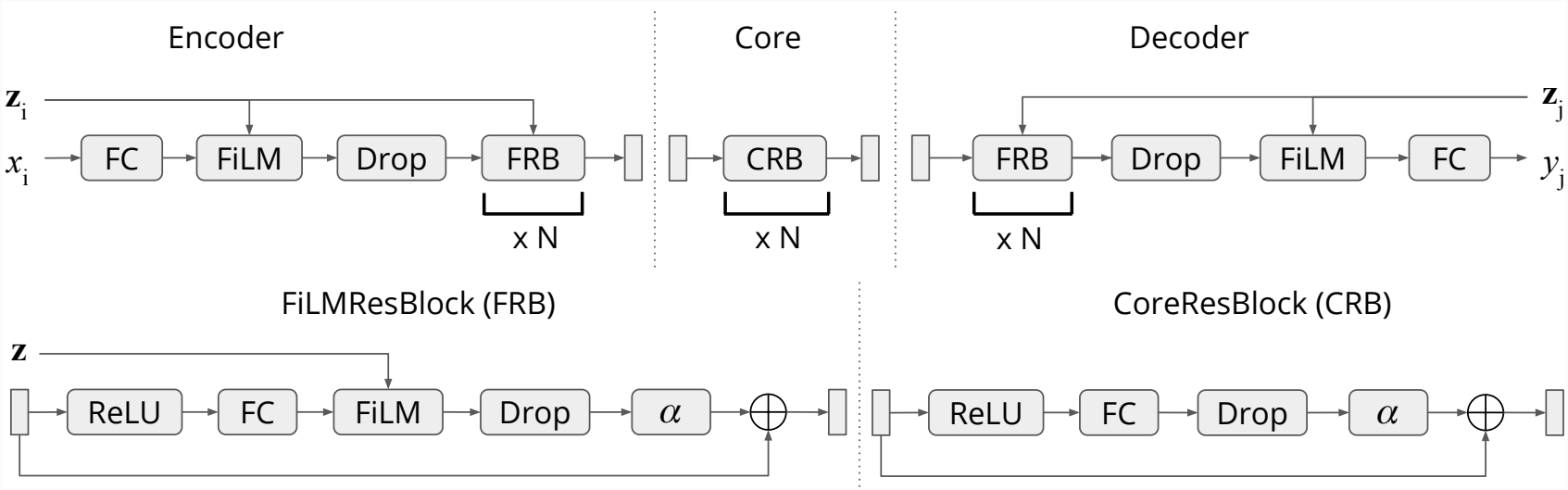
$$\mathbb{E}[y_j|x] = g_2\big(g_1\big(\sum_{i=1}^{n} f(x_i, z_i)\big), z_j\big) \tag{6}$$

Dodatkowo, model wykorzystuje afiniczne warstwy FiLM [8], które pozwalają połączyć ukryty stan h oraz VE z jako:

$$FiLM(h) = W^*(z) \odot h + W^+(z) \tag{7}$$

gdzie $\odot$ to produkt Hadamarda, a W^* i W^+ to przekształcenia liniowe z parametrami dostrajanymi podczas treningu.

Traveling observer model



Rysunek 15: Architektura modelu TOM [7]. *Encoder*, *Core*, i *Decoder* odpowiadają f, g_1 oraz g_2 z równania 6.

Czy Variable Embeddings (VEs) mogą być pomocne bez wcześniejszej znajomości znaczenia zmiennych?

Modele bazowe

Modele bazowe:

- Brak - wszystkie VE ustawione na 0 (celem jest sprawdzenie czy VE są w ogóle potrzebne)
- Losowe - losowe VE (celem jest sprawdzenie czy losowe, ale unikalne, VE wystarczą)
- Wyrocznia - ludzka intuicja wskazująca jak VE powinny być ułożone

CIFAR

- Obrazki o wymiarze 32x32
- Piksele w skali szarości $\in [0, 1]$
- W sumie 1024 zmiennych
- Cel: przewidywanie zamaskowanych fragmentów obrazka
- Funkcja kosztu: binary cross-entropy (BCE)
- Oczekiwanie: VE ($C = 2$) ułożą się w kratkę odpowiadającą siatce pikseli obrazka

Daily Temperature

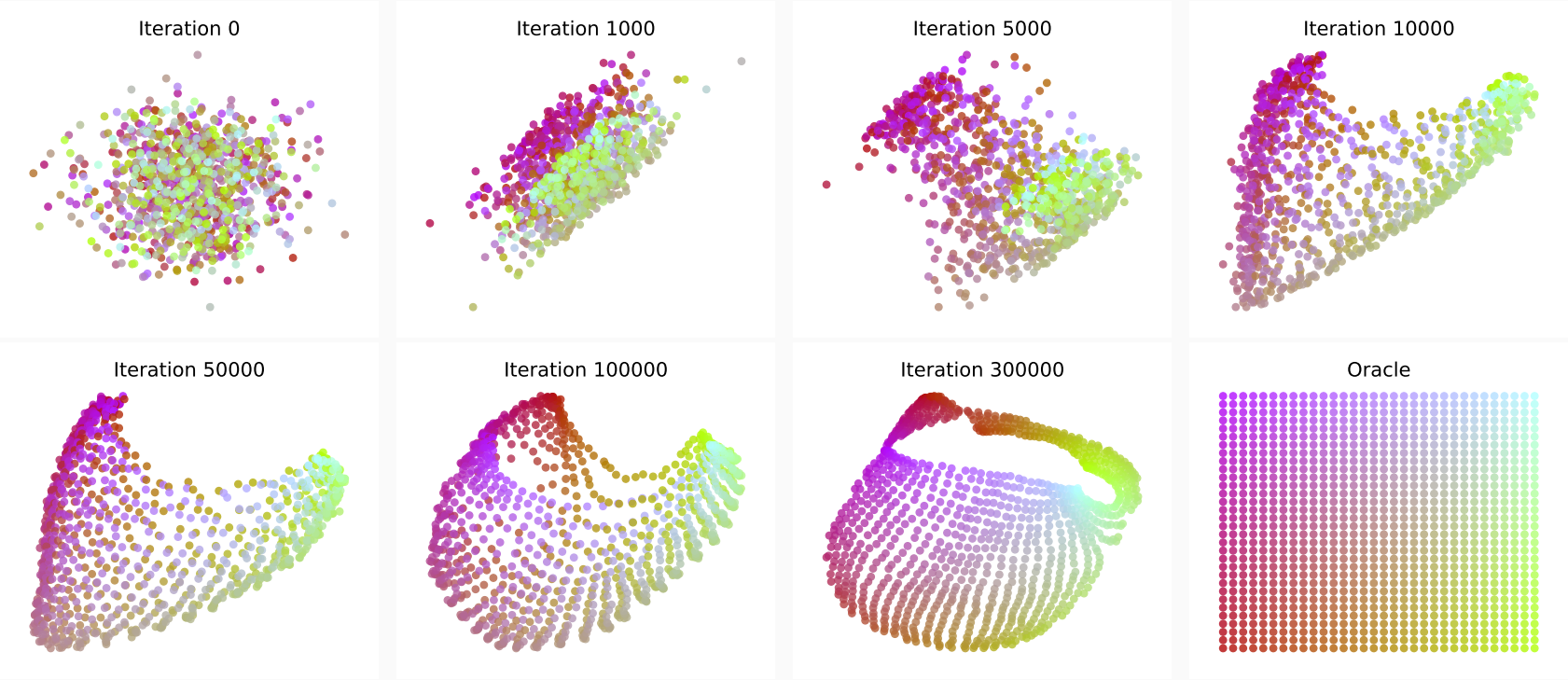
- Szereg czasowy minimalnych dziennych temperatur w Melbourne
- Cel: przewidywanie zamaskowanych temperatur z ostatnich 10 dni
- Funkcja kosztu: mean squared error (MSE)
- Oczekiwanie: VE ($C = 2$) ułożą się w szeregu reprezentując kolejne dni

Przydatność wytrenowanych VE

	VE			
	Brak	Losowe	Wytrenowane	Wyrocznia
CIFAR	0.662	0.660	**0.591**	0.590
Daily Temperature	4.29	4.27	**3.32**	3.37

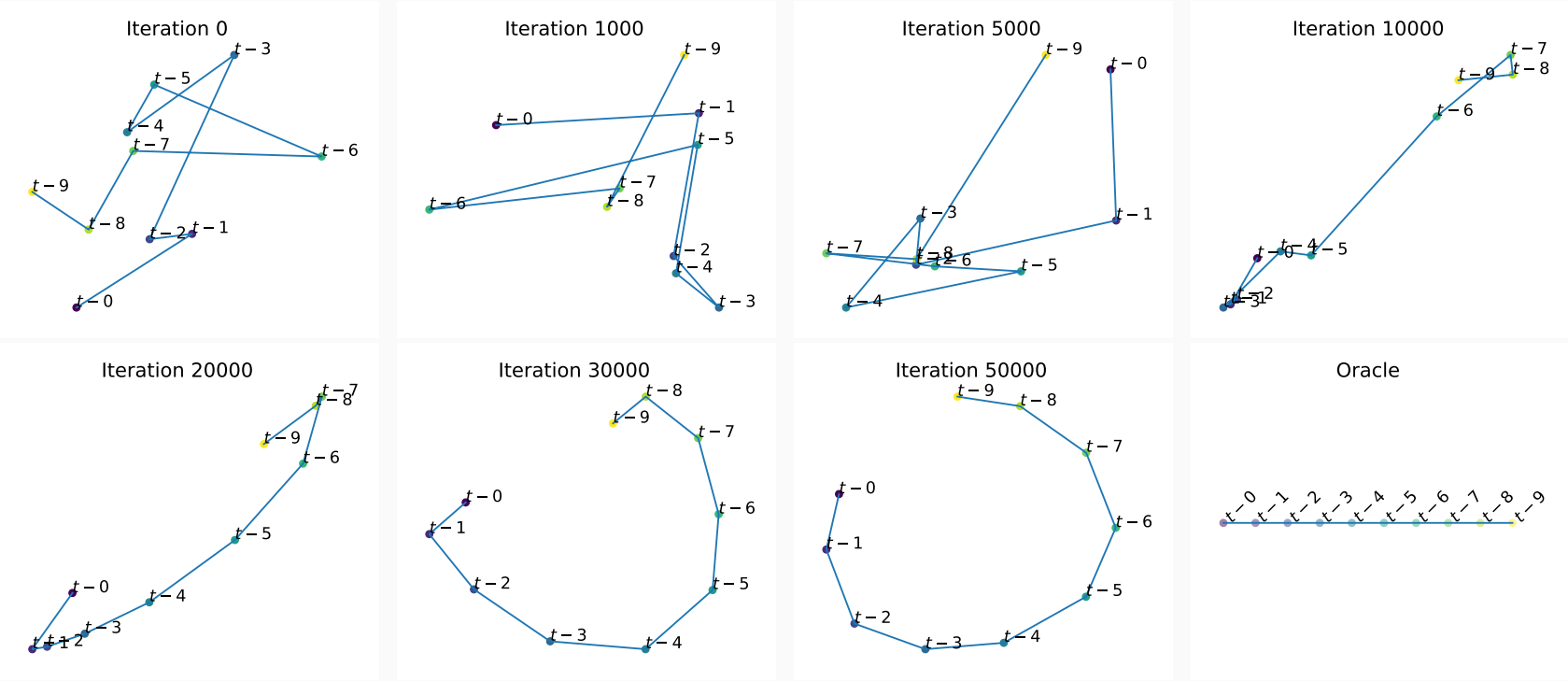
Tablica 1: Liczbowe wyniki przewidywania przestrzeni i czasu. Tablica porównuje błędy na zbiorze testowym używając BCE dla CIFAR oraz RMSE dla Daily Temperature.

Wytrenowane VE



Rysunek 16: VE wytrenowane na zbiorze CIFAR [7].

Wytrenowane VE



Rysunek 17: VE wytrenowane na zbiorze Daily Temperature [7].

Czy Variable Embeddings (VEs) mogą być pomocne w rozwiązywaniu wielu różnorodnych zadań jednocześnie?

UCI-121

- 121 zadań klasyfikacyjnych
- Dziedziny: medycyna, geologia, inżynieria, botanika, socjologia, polityka i gry
- Liczba cech od 3 do 262
- Liczba klas od 2 do 100
- Liczba obserwacji od 10 do 130 064
- $C = 128$ (wymiarowość VE)
- Model jest dostrajany do każdego zadania z co najmniej 5K obserwacji

UCI-121

Method	Win %	Best %	Mean Rank	Norm. Acc.	Mean Acc.
ResNet	3.31	12.40	3.89	50.07	79.24
MS	4.96	14.88	3.35	60.11	80.11
BN	5.79	13.22	4.20	42.15	77.01
WN	7.44	10.74	4.05	45.87	77.43
HW	8.26	15.70	3.61	53.00	78.68
LN	9.92	16.53	3.45	56.73	79.85
SNN	13.22	21.49	2.78	65.29	81.39
TOM	**28.93**	**34.71**	**2.60**	**70.72**	**81.53**

Tablica 2: Wyniki na zbiorze UCI-121. TOM jest porównany z innymi jedno-zadaniowymi modelami opisanymi w [3].

UCI-121

Method	Win %	Best %	Mean Rank	Norm. Acc.	Mean Acc.
DR-STL	10.74	19.01	2.31	54.72	76.48
TOM-STL	7.44	16.53	2.72	35.21	68.18
DR-MTL	9.09	28.10	2.02	56.47	78.40
SLO	16.53	30.06	1.62	73.88	80.31
TOM	**32.23**	**47.10**	**1.34**	**76.70**	**81.53**

Tablica 3: Wyniki na zbiorze UCI-121. TOM jest porównany z innymi wielo-zadaniowymi modelami: DR-STL - TOM bez warstw FiLM oraz z osobnym liniowym enkoderem i dekoderem do każdego zadania; DR-MTL - jak DR-STL, ale trenowany jednocześnie na wszystkich zadaniach; TOM-STL - TOM trenowany oddzielnie na każdym zadaniu; SLO - soft layer ordering [6].

Podsumowanie

Podsumowanie

- W teorii, ogólna wiedza o wykonywaniu skutecznych predykcji powinna wynikać nawet z pozornie niepowiązanych zadań [4, 5]
- W praktyce, jest to demonstrowane dopiero w nielicznych nowych pracach

Bibliografia i

Zhao-Min Chen, Xiu-Shen Wei, Peng Wang, and Yanwen Guo. **Multi-label image recognition with graph convolutional networks.** In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5177–5186, 2019.

Lukasz Kaiser, Aidan N Gomez, Noam Shazeer, Ashish Vaswani, Niki Parmar, Llion Jones, and Jakob Uszkoreit. **One model to learn them all.** *arXiv preprint arXiv:1706.05137*, 2017.

Bibliografia ii


Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter.
Self-normalizing neural networks.
In *Proceedings of the 31st international conference on neural information processing systems*, pages 972–981, 2017.


MM Mahmud and Sylvian R Ray.
Transfer learning using kolmogorov complexity: Basic theory and empirical evaluations.
Technical report, 2007.


MM Hassan Mahmud.
On universal transfer learning.
Theoretical Computer Science, 410(19):1826–1846, 2009.

Bibliografia iii


Elliot Meyerson and Risto Miikkulainen.
Beyond shared hierarchies: Deep multitask learning through soft layer ordering.
arXiv preprint arXiv:1711.00108, 2017.


Elliot Meyerson and Risto Miikkulainen.
The traveling observer model: Multi-task learning through spatial variable embeddings.
arXiv preprint arXiv:2010.02354, 2020.

Bibliografia iv


Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville.
Film: Visual reasoning with a general conditioning layer.
In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.


Asa Cooper Stickland and Iain Murray.
Bert and pals: Projected attention layers for efficient adaptation in multi-task learning.
In *International Conference on Machine Learning*, pages 5986–5995. PMLR, 2019.

Bibliografia v

Eleni Triantafillou, Tyler Zhu, Vincent Dumoulin, Pascal Lamblin, Utku Evci, Kelvin Xu, Ross Goroshin, Carles Gelada, Kevin Swersky, Pierre-Antoine Manzagol, and Hugo Larochelle.
Meta-dataset: A dataset of datasets for learning to learn from few examples.
In *International Conference on Learning Representations*, 2020.

Yongxin Yang and Timothy M Hospedales.
A unified perspective on multi-domain and multi-task learning.
arXiv preprint arXiv:1412.7489, 2014.