Metody transferu wiedzy wykorzystujące sieci neuronowe

Transfer learning with neural networks

Mikołaj Małkiński
mikolaj.malkinski.dokt@pw.edu.pl

14 kwietnia 2021

Politechnika Warszawska
Szkoła Doktorska nr 3

O mnie

- Praca magisterska: AI systems for solving IQ tests
- Temat doktoratu: Metody przenoszenia wiedzy między problemami w zagadnieniach klasyfikacji
- Promotor: prof. dr hab. inż. Jacek Mańdziuk

Cel doktoratu

- Samodzielne systemy rozwiązujące wiele zadań danego typu
- Jednolite zasoby oraz jednolita meta-metoda nauki
- Różnorodne zagadnienia klasyfikacji

Możliwe ścieżki

- Odpowiednia architektura systemu
- Współdzielona pamięć
- Metody dekompozycji problemów uczących

Motywacja

- Koszty
- Ograniczenia sprzętowe
- Zrozumienie inteligencji ludzkiej

Spis treści

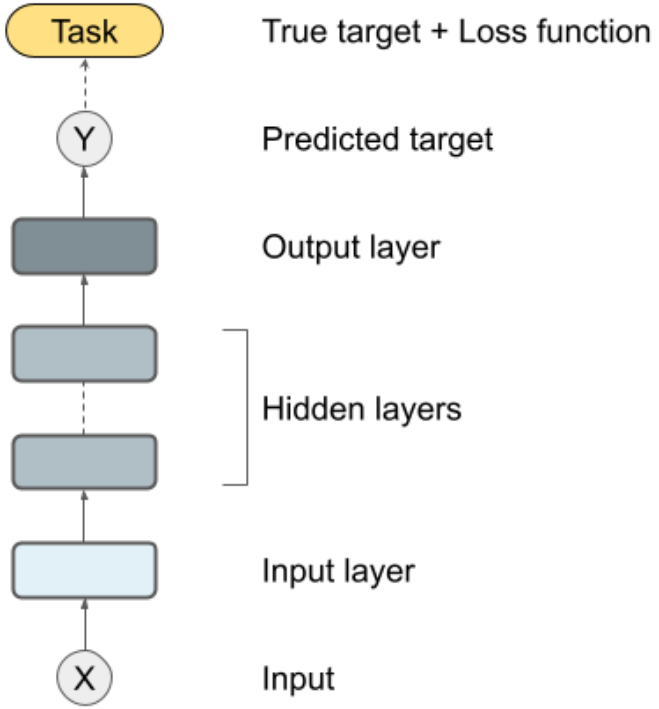
1. Single-task learning (STL)
2. Inductive transfer Learning
3. Self-supervised Learning (SSL)
4. Multi-task Learning (MTL)
5. Zadania pomocnicze
6. Pseudo MTL
7. Podsumowanie

Single-task learning (STL)

Single-task learning

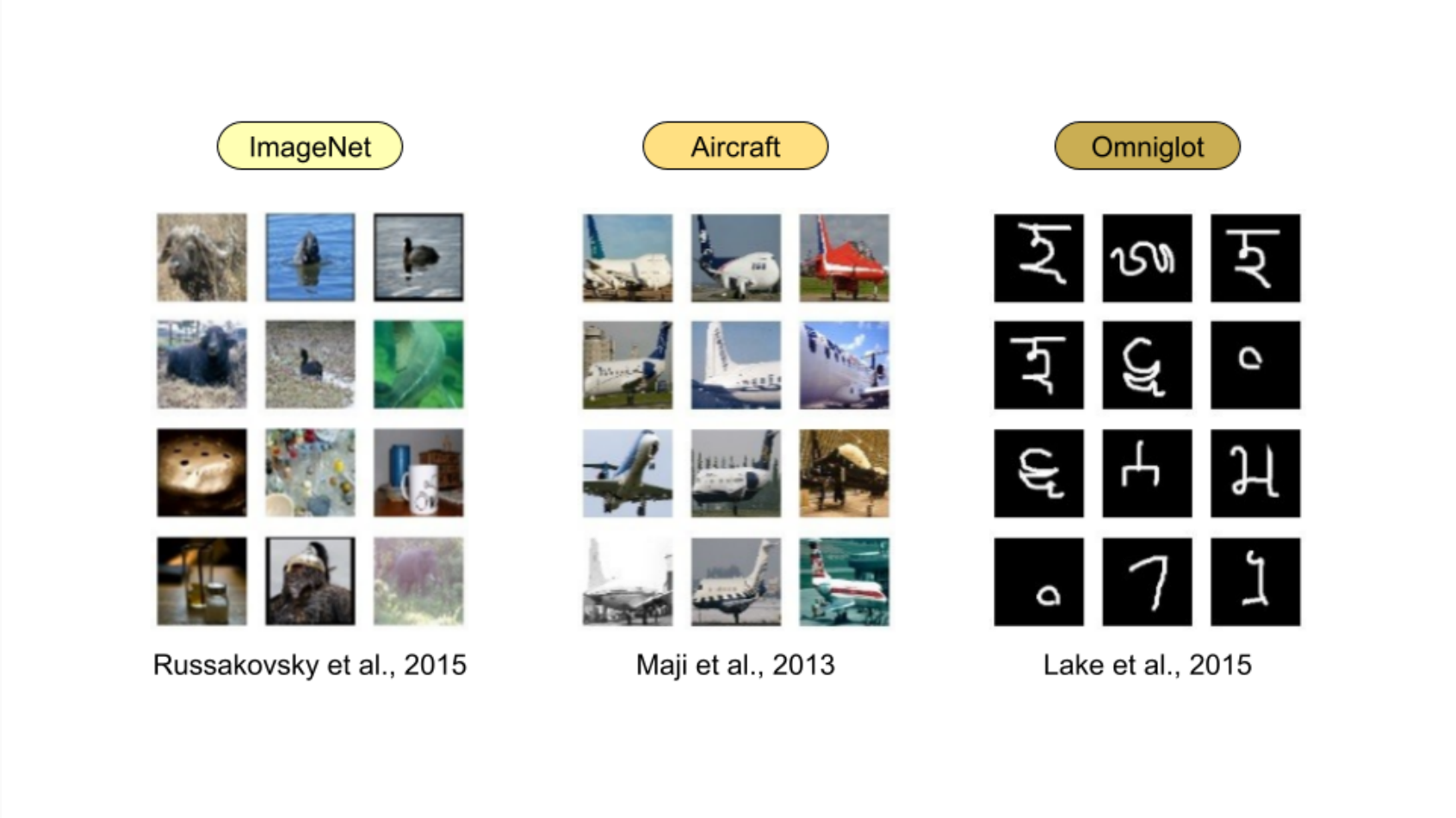
- $\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^{N}$ - zadanie
- N - liczba obserwacji
- $\mathcal{F}$ - trenowany model
- $\mathcal{L}$ - funkcja kosztu

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 [11].

STL do wielu zadań

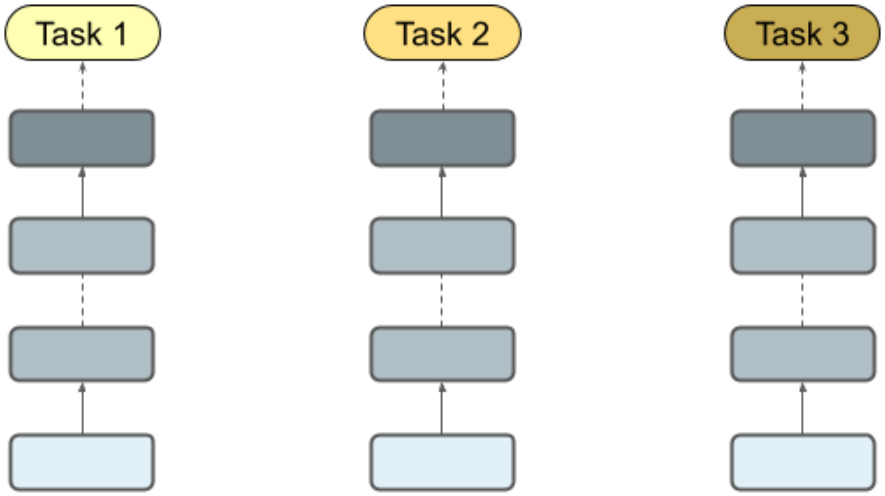
- $(\{\mathbf{x}_{ti}, \mathbf{y}_{ti}\}_{i=1}^{N_t}; \mathcal{F}_t; \mathcal{L}_t)_{t=1}^{T}$ - zadania
- T - liczba zadań
- N_t - liczba obserwacji dla zadania t
- $\mathcal{F}_t$ - model do zadania t
- $\mathcal{L}_t$ - funkcja kosztu zadania t

STL do wielu zadań

Dla zadań t_i i t_j takich, że $i \neq j$ może zachodzić:

- $\dim(\mathbf{x}_{ti}) \neq \dim(\mathbf{x}_{tj})$
- $\dim(\mathbf{y}_{ti}) \neq \dim(\mathbf{y}_{tj})$
- $N_{ti} \neq N_{tj}$
- $\mathcal{F}_i \neq \mathcal{F}_j$
- $\mathcal{L}_i \neq \mathcal{L}_j$

STL do wielu zadań



Rysunek 3: 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
- Multi-task learning
- Zadania pomocnicze (auxiliary tasks)

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

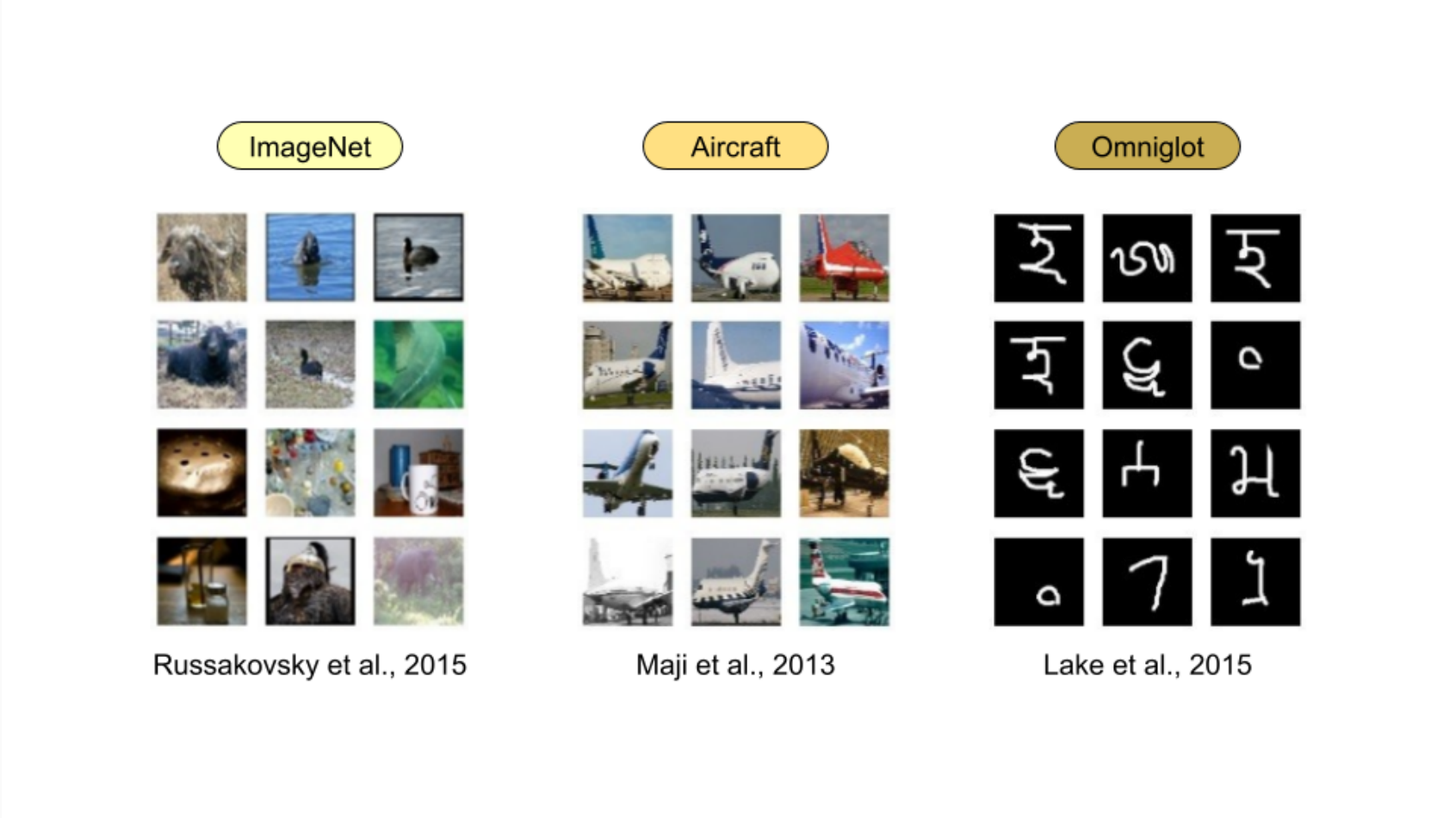
Inductive transfer Learning

Inductive transfer Learning

Założenie: wiedza z zadania wstępnego jest przydatna do rozwiązania zadania docelowego

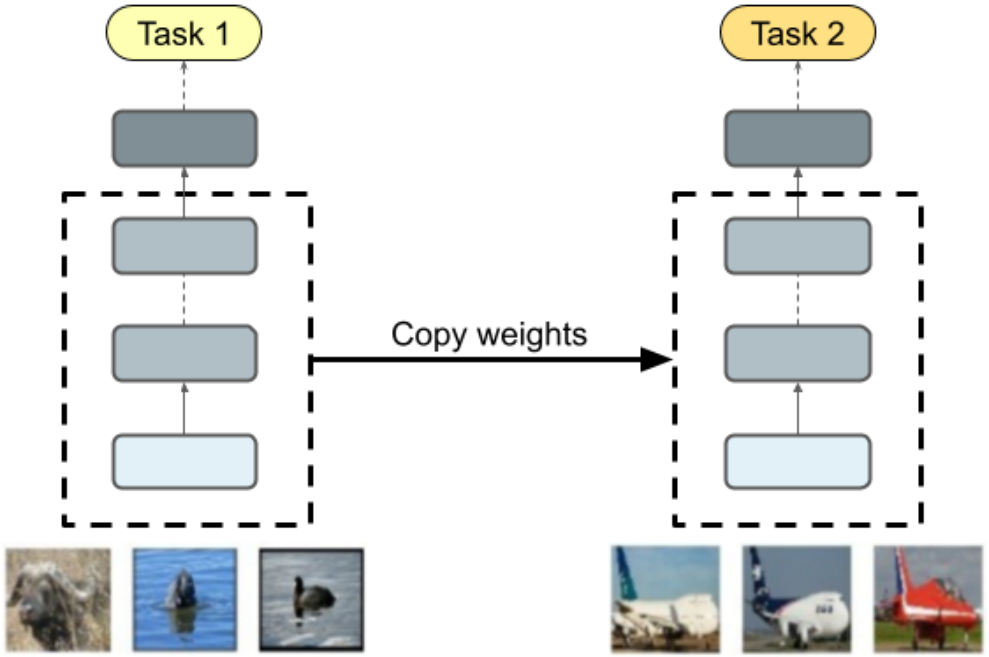
- $(\{\mathbf{x}_{1i}, \mathbf{y}_{1i}\}_{i=1}^{N_1}; \mathcal{F}_1; \mathcal{L}_1)$ - zadanie wstępne
- $(\{\mathbf{x}_{2i}, \mathbf{y}_{2i}\}_{i=1}^{N_2}; \mathcal{F}_2; \mathcal{L}_2)$ - zadanie docelowe
- $\theta_{\mathcal{F}_1} \cap \theta_{\mathcal{F}_2} \neq \emptyset$

Powiązane zadania



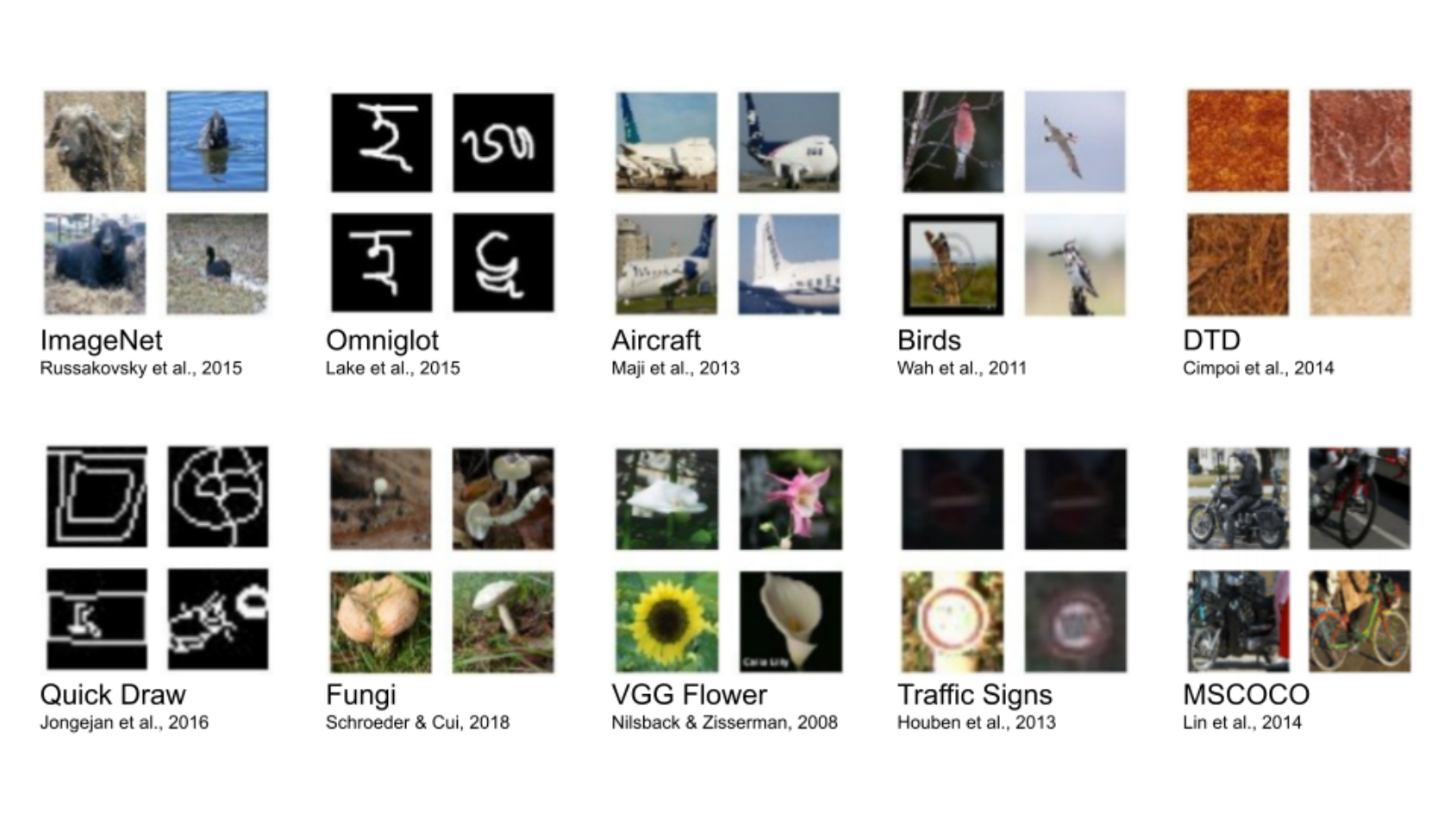
Rysunek 4: Powiązane zadania mogą być rozwiązane podobnymi sposobami, np. wykrywając krawędzie obiektów na obrazku.

Inductive transfer learning



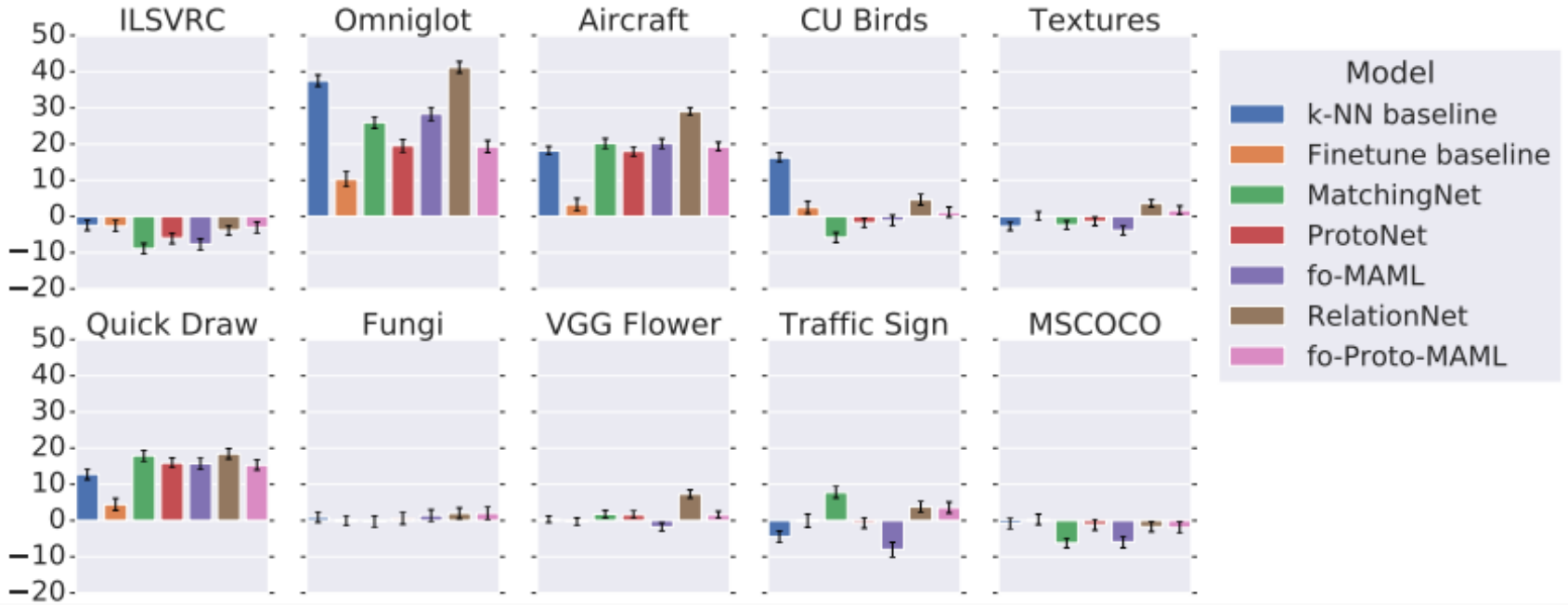
Rysunek 5: Zdobyta wiedza może być użyta do rozwiązania nowego zadania.

Meta-Dataset



Rysunek 6: Przykłady obrazków z Meta-Dataset [11].

Meta-Dataset



Rysunek 7: Różnica w skuteczności przenoszenia wiedzy ze wszystkich zbiorów vs tylko ze zbioru ImageNet. Wyniki pochodzą z [11].

Obserwacja 1: Obecne metody nie potrafią w pełni wykorzystać heterogenicznych zbiorów danych.

Jak zbudować wstępny model bez dostępu do etykiet?

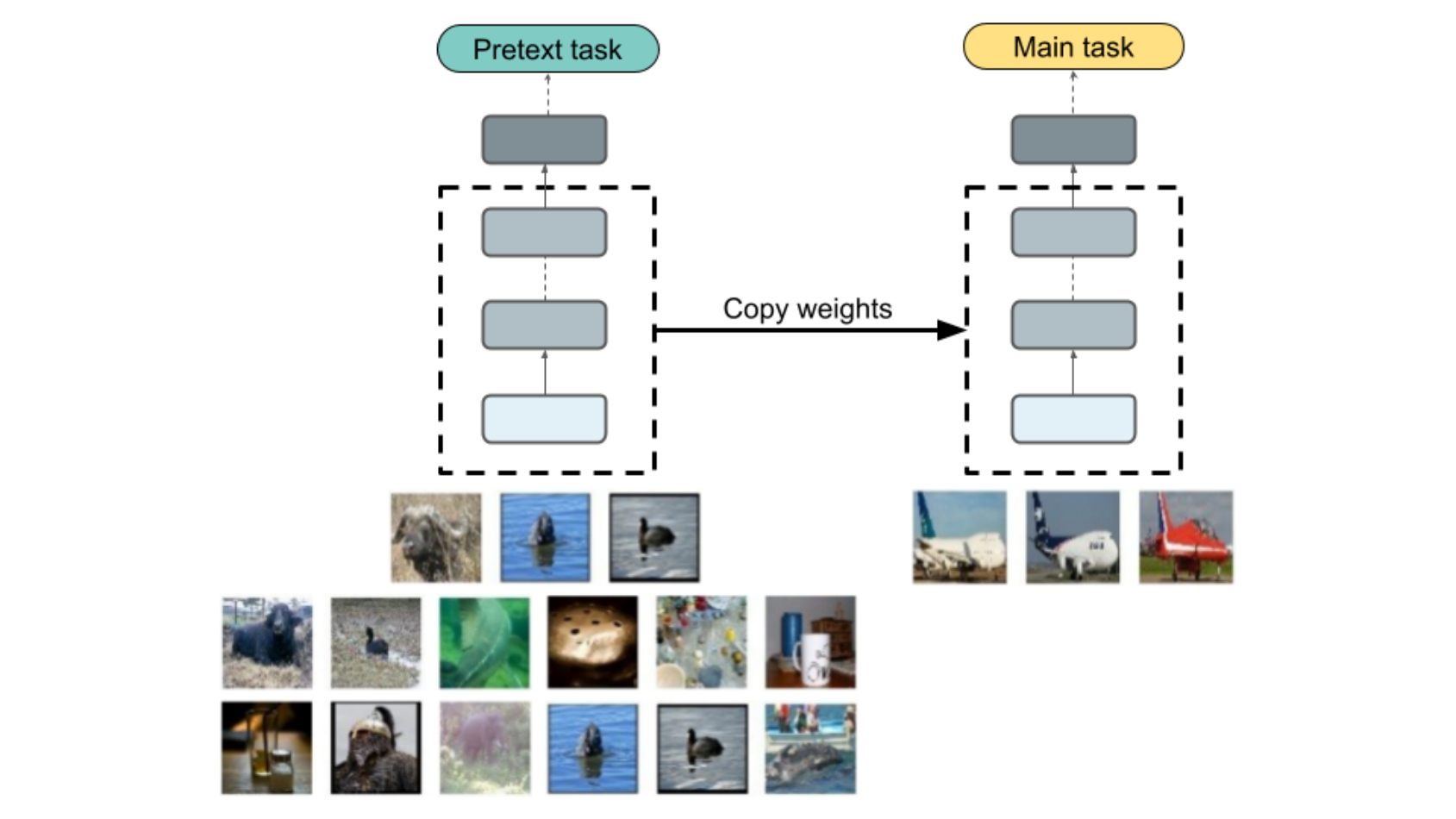
Self-supervised Learning (SSL)

Self-supervised Learning

Założenie: wiedza z zadania wstępnego jest przydatna do rozwiązania zadania docelowego

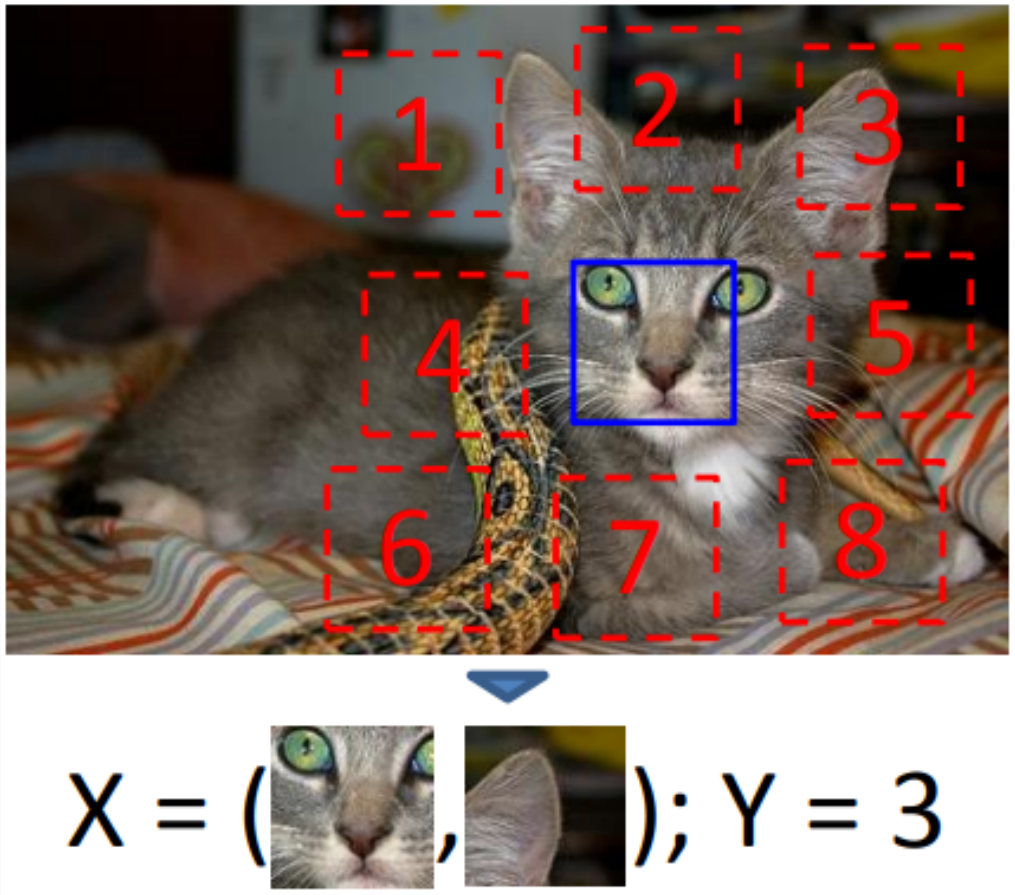
- $(\{\mathbf{x}_{1i}\}_{i=1}^{N_1}; \mathcal{F}_1; \mathcal{L}_1)$ - zadanie wstępne
- $(\{\mathbf{x}_{2i}, \mathbf{y}_{2i}\}_{i=1}^{N_2}; \mathcal{F}_2; \mathcal{L}_2)$ - zadanie docelowe
- $\theta_{\mathcal{F}_1} \cap \theta_{\mathcal{F}_2} \neq \emptyset$
- $N_1 \gg N_2$

Self-supervised learning



Rysunek 8: Transfer wiedzy metodą Self-supervised learning.

Self-supervised learning

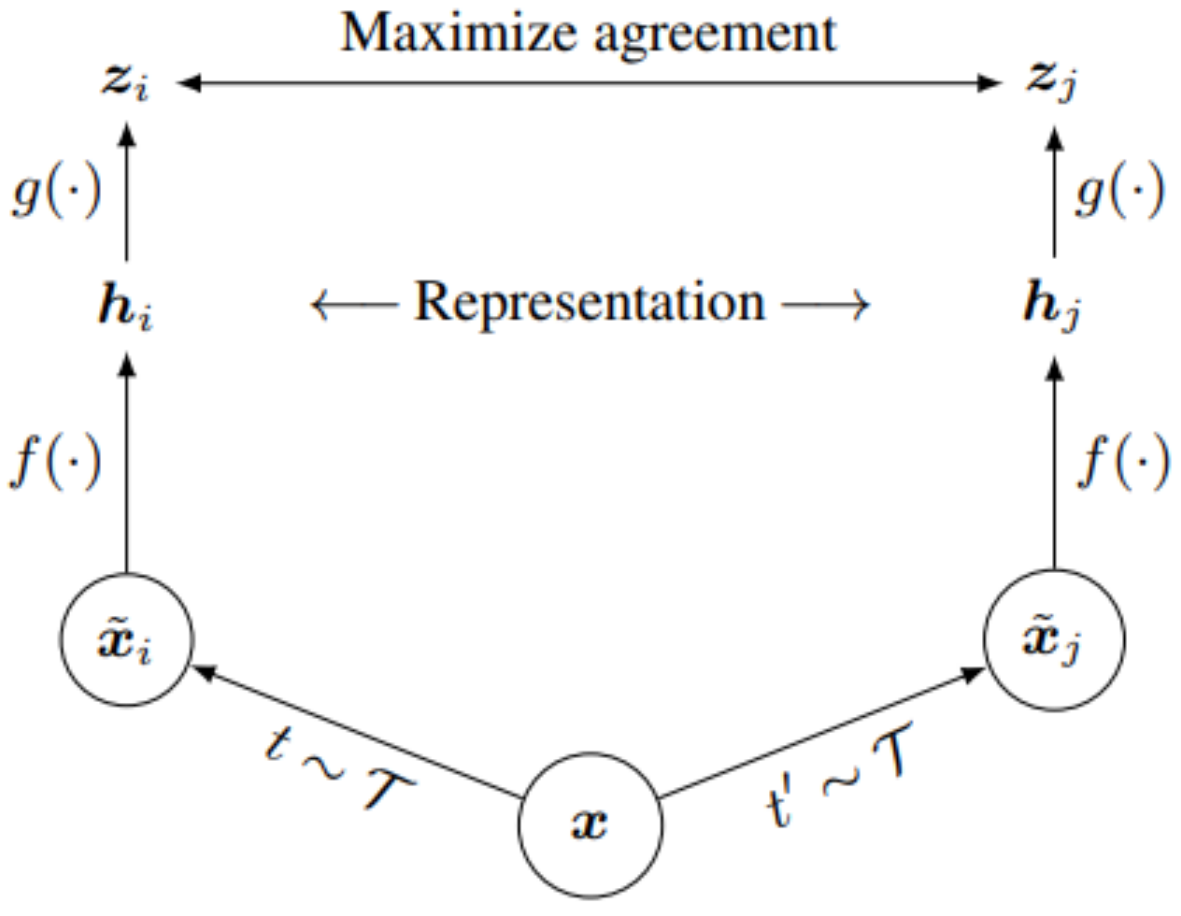


Rysunek 9: Przykład zadania SSL polegającego na przewidywaniu względnych pozycji wycinków obrazka [4].

Self-supervised learning

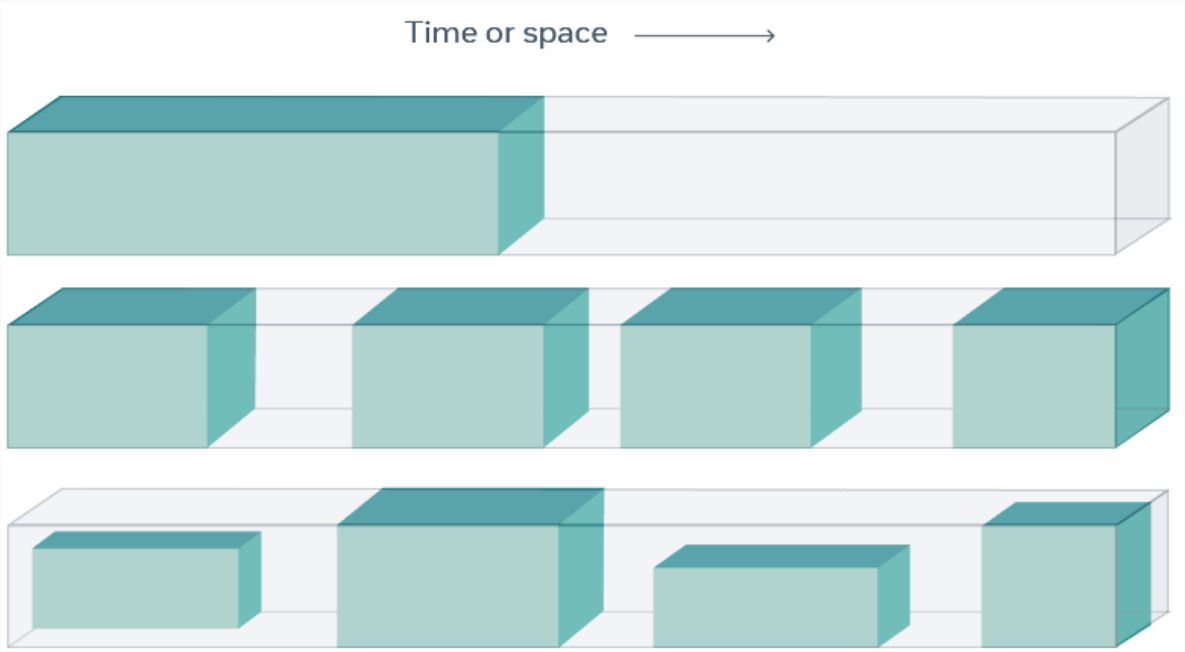
Rysunek 10: Przykład zadania SSL polegającego na przewidywaniu obrotu obrazka [5].

Self-supervised learning



Rysunek 11: Przykład zadania SSL polegającego na kontrastowaniu [2].

Self-supervised learning



Rysunek 12: Przykład zadania SSL bazującego na filmach [7].

Self-supervised learning

Rysunek 13: Przykład zadania SSL polegającego na predykcji ukrytych słów [1, 3].

Multi-task Learning (MTL)

Multi-task learning

- $(\{\mathbf{x}_{ti}, \mathbf{y}_{ti}\}_{i=1}^{N_t}; \mathcal{F}; \mathcal{D}_t; \mathcal{L}_t)_{t=1}^{T}$ - zadania
- $\mathcal{F}$ - model bazowy (backbone) z parametrami $\theta_{\mathcal{F}}$
- $\mathcal{F}$ jest używany we wszystkich zadaniach
- $\mathcal{D}_t$ - głowa modelu (dekoder) wyłącznie do zadania t z parametrami $\theta_{\mathcal{D}_t}$

Model do zadania t jest zdefiniowany jako

$$\hat{\mathbf{y}}_{ti} = \mathcal{D}_t(\mathcal{F}(\mathbf{x}_{ti}; \theta_{\mathcal{F}}); \theta_{\mathcal{D}_t}) \tag{1}$$

Multi-task learning

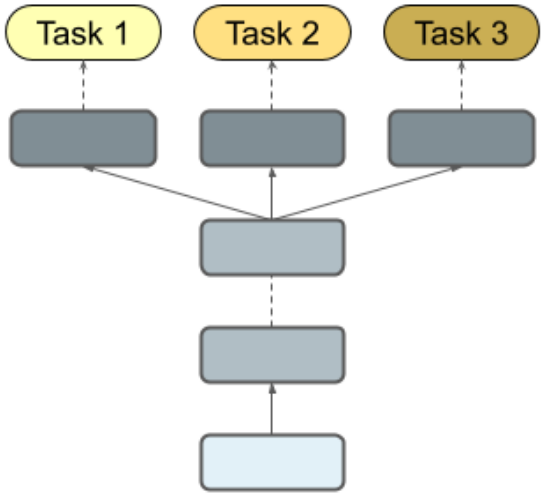
Wspólny model jest określony parametrami

$$\theta = (\{\theta_{\mathcal{D}_t}\}_{t=1}^{T}, \theta_{\mathcal{F}}) \tag{2}$$

MTL polega na znalezieniu parametrów θ^* takich, że

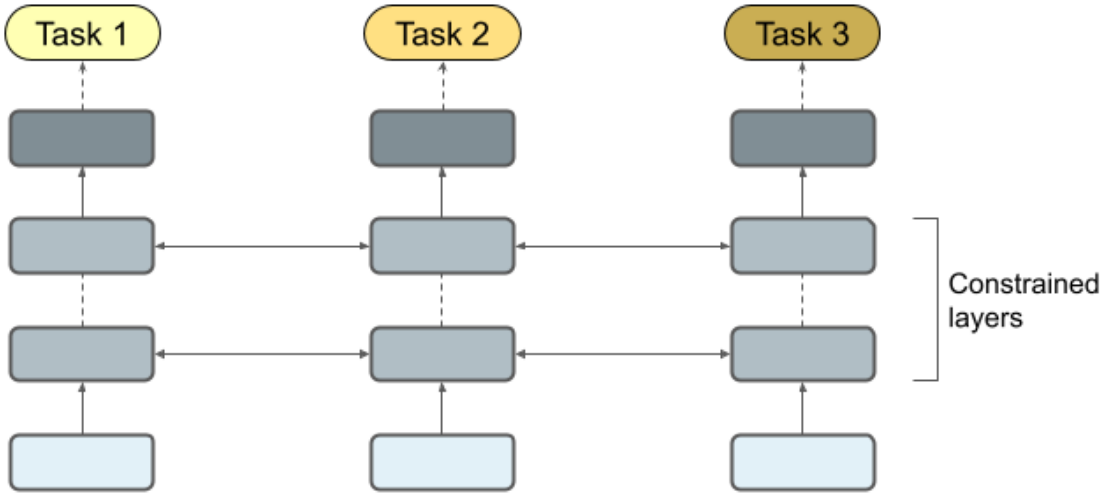
$$\theta^* = \underset{\theta}{\operatorname{argmin}} \frac{1}{T} \sum_{t=1}^{T} \frac{1}{N_t} \sum_{i=1}^{N_t} \mathcal{L}_t(\mathbf{y}_{ti}, \hat{\mathbf{y}}_{ti}) \tag{3}$$

Multi-task learning



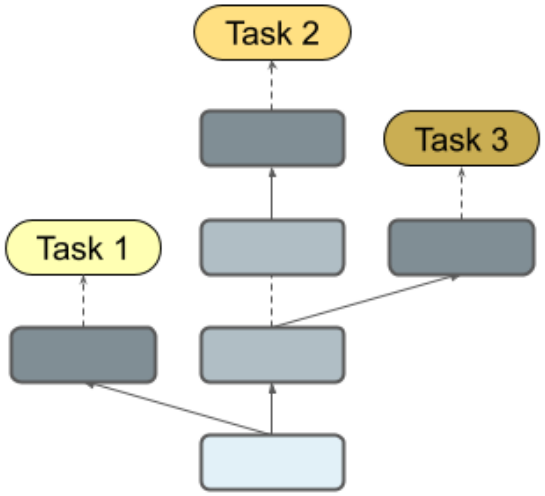
Rysunek 14: W podejściu multi-task, ten sam model służy do rozwiązania wielu zadań (hard parameter sharing).

Multi-task learning



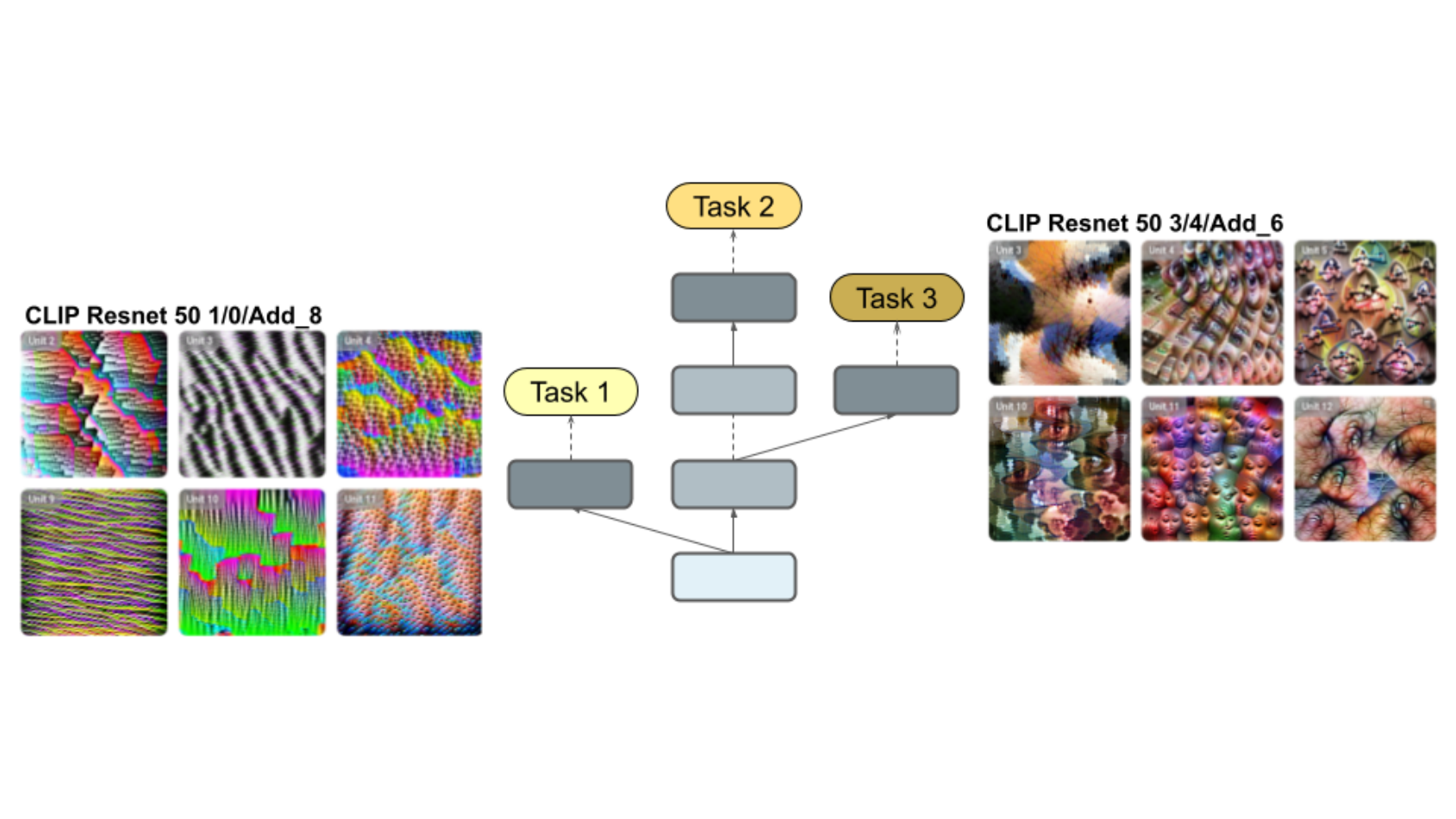
Rysunek 15: Podejście multi-task z łagodnym współdzieleniem parametrów (soft parameter sharing).

Multi-task learning



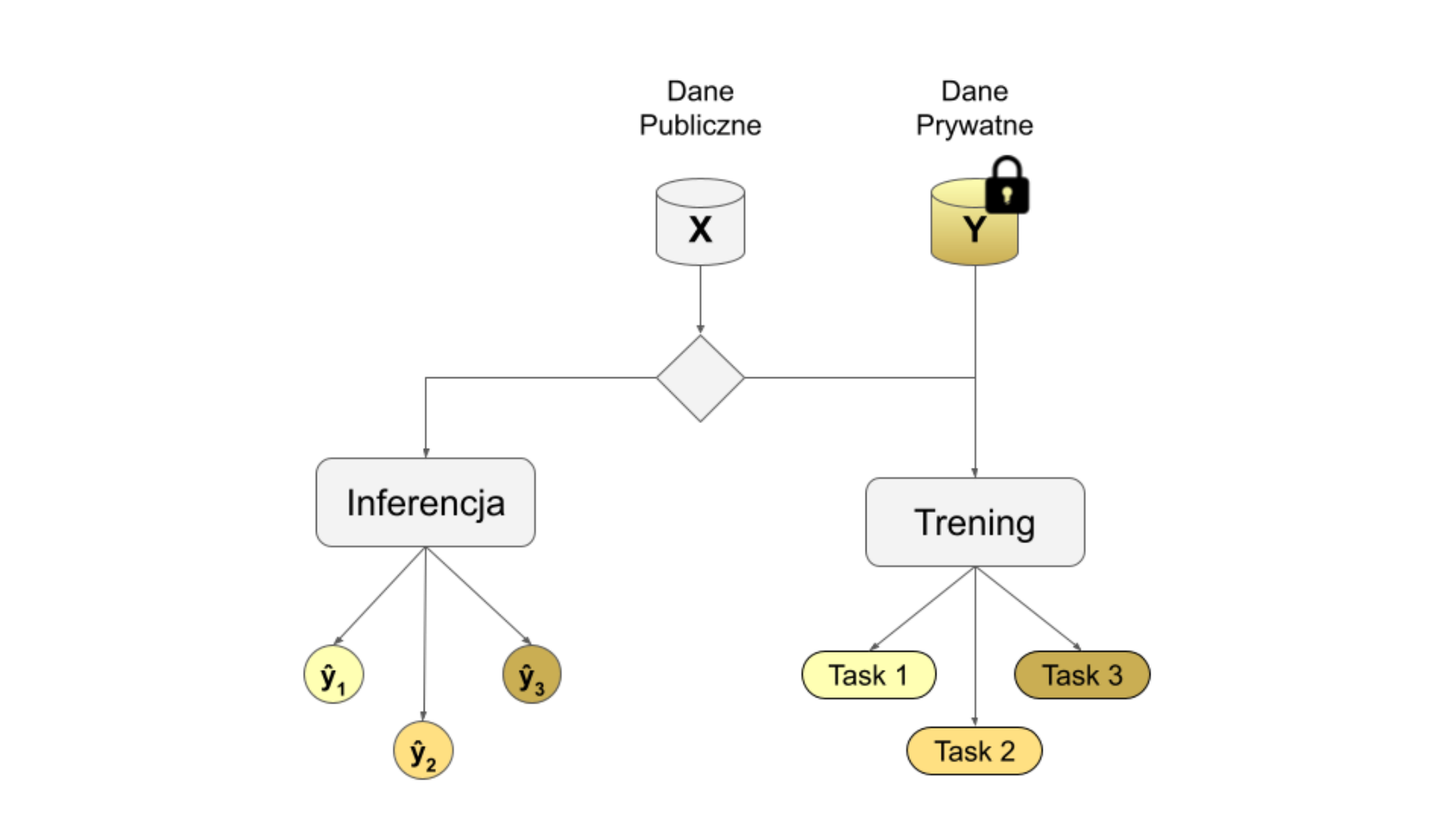
Rysunek 16: Podejście multi-task z dekoderami w różnych głębokościach.

Multi-task learning



Rysunek 17: Wizualizacje filtrów konwolucyjnych w różnych głębokościach sieci [10].

Multi-task learning



Rysunek 18: MTL do problemu z chronionymi zmiennymi celu.

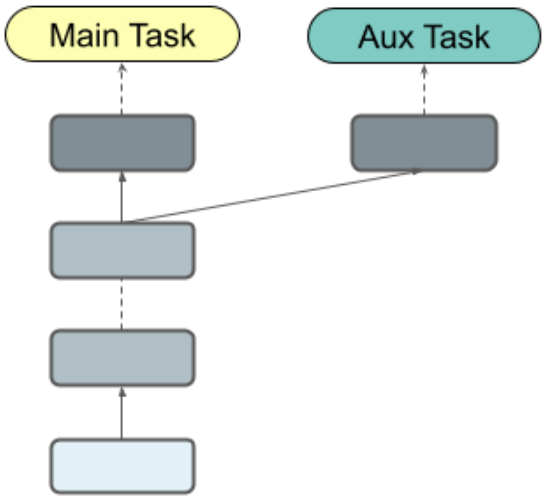
Multi-task learning

Zalety MTL:

- pośrednia metoda augmentacji danych
- skupienie uwagi na sednie problemu
- zapożyczanie dobrych cech z innych zadań
- budowa reprezentacji przydatnych do wielu (więc także do nowych) zadań

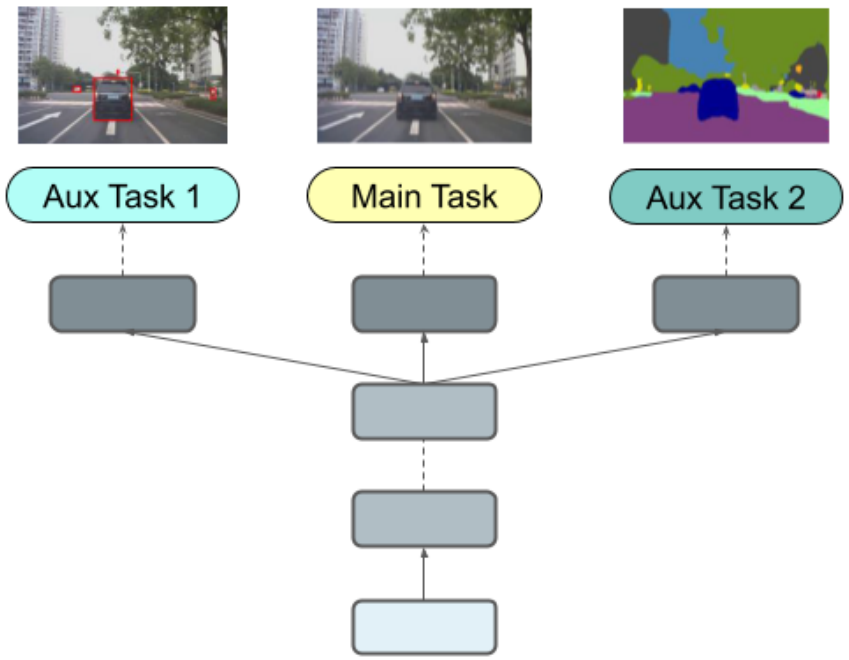
Zadania pomocnicze

Zadania pomocnicze



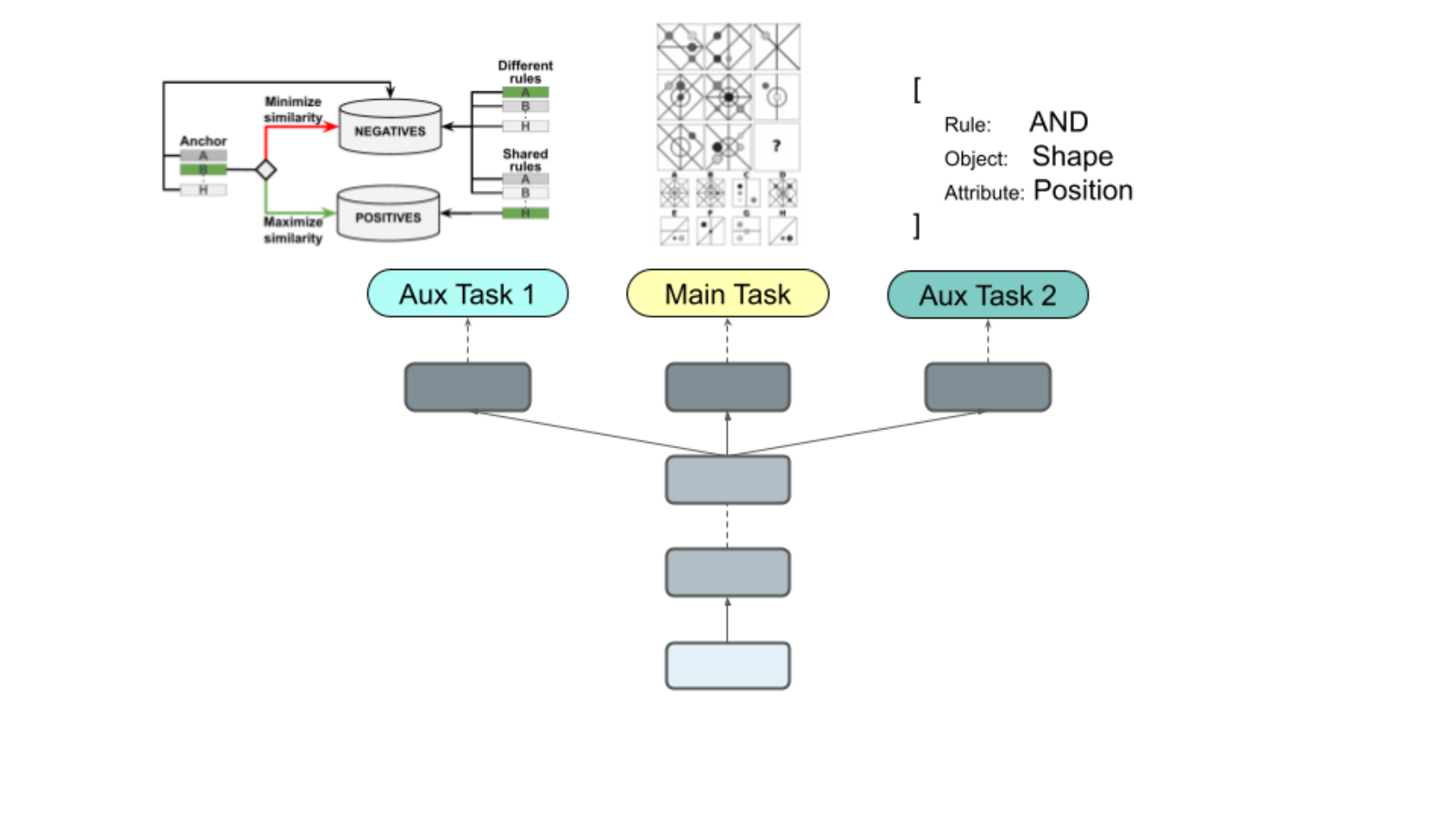
Rysunek 19: W nauce rozwiązywania zadania głównego może być przydatne zadanie pomocnicze.

Zadania pomocnicze w autonomicznej jeździe



Rysunek 20: Przykładowe zadania pomocnicze w nauce autonomicznej jazdy mogą polegać na detekcji obiektów i segmentacji obrazu.

Zadania pomocnicze w matrycach Ravena



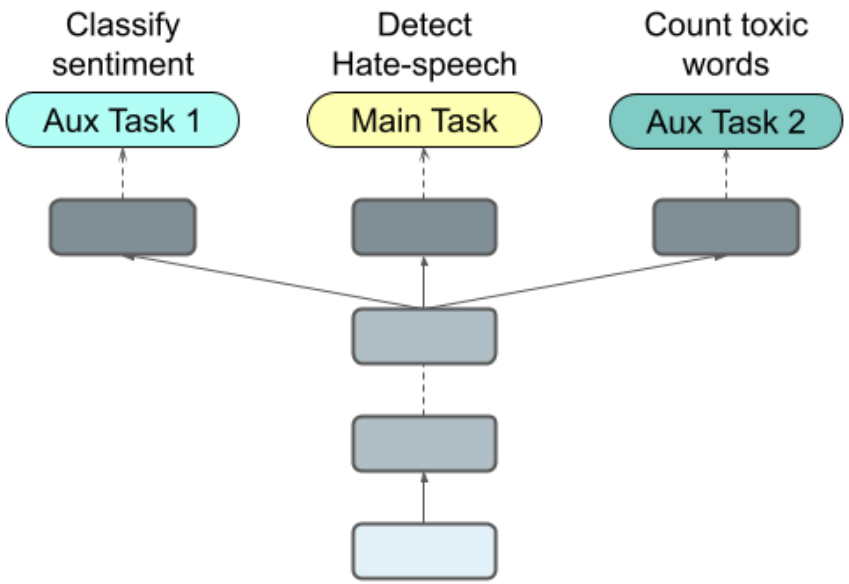
Rysunek 21: Przykładowe zadania pomocnicze w nauce rozwiązywania matryc Ravena polegające na kontrastowaniu embeddingów i predykcji reguł [8].

Zadania pomocnicze w matrycach Ravena

Tablica 1: Wpływ zadań pomocniczych na wyniki na zbiorze I-RAVEN.

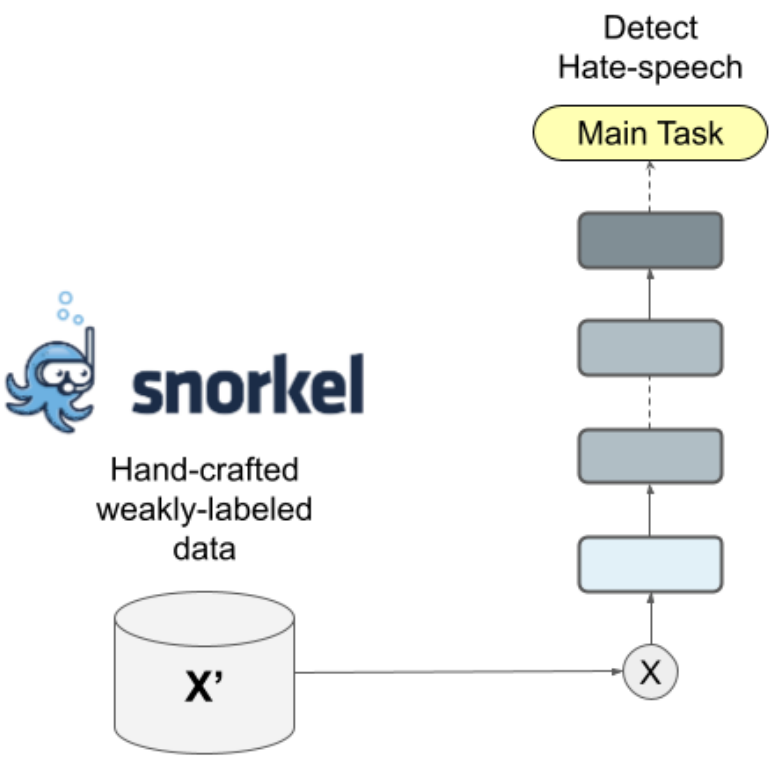
Zadanie pomocnicze			Skuteczność (%)		
dense	sparse	contrastive	SCL	SRAN	CoPINet
×	×	×	82.8	56.0	44.8
✓	×	×	86.9	59.4	32.2
×	✓	×	95.6	59.9	50.3
×	✓	✓	**95.7**	**66.1**	**52.3**

Zadania pomocnicze w wykrywaniu mowy nienawiści



Rysunek 22: Zadania pomocnicze w nauce wykrywania mowy nienawiści.

Weakly-supervised learning



Rysunek 23: Trening słabo-nadzorowany w nauce wykrywania mowy nienawiści.

Pseudo MTL

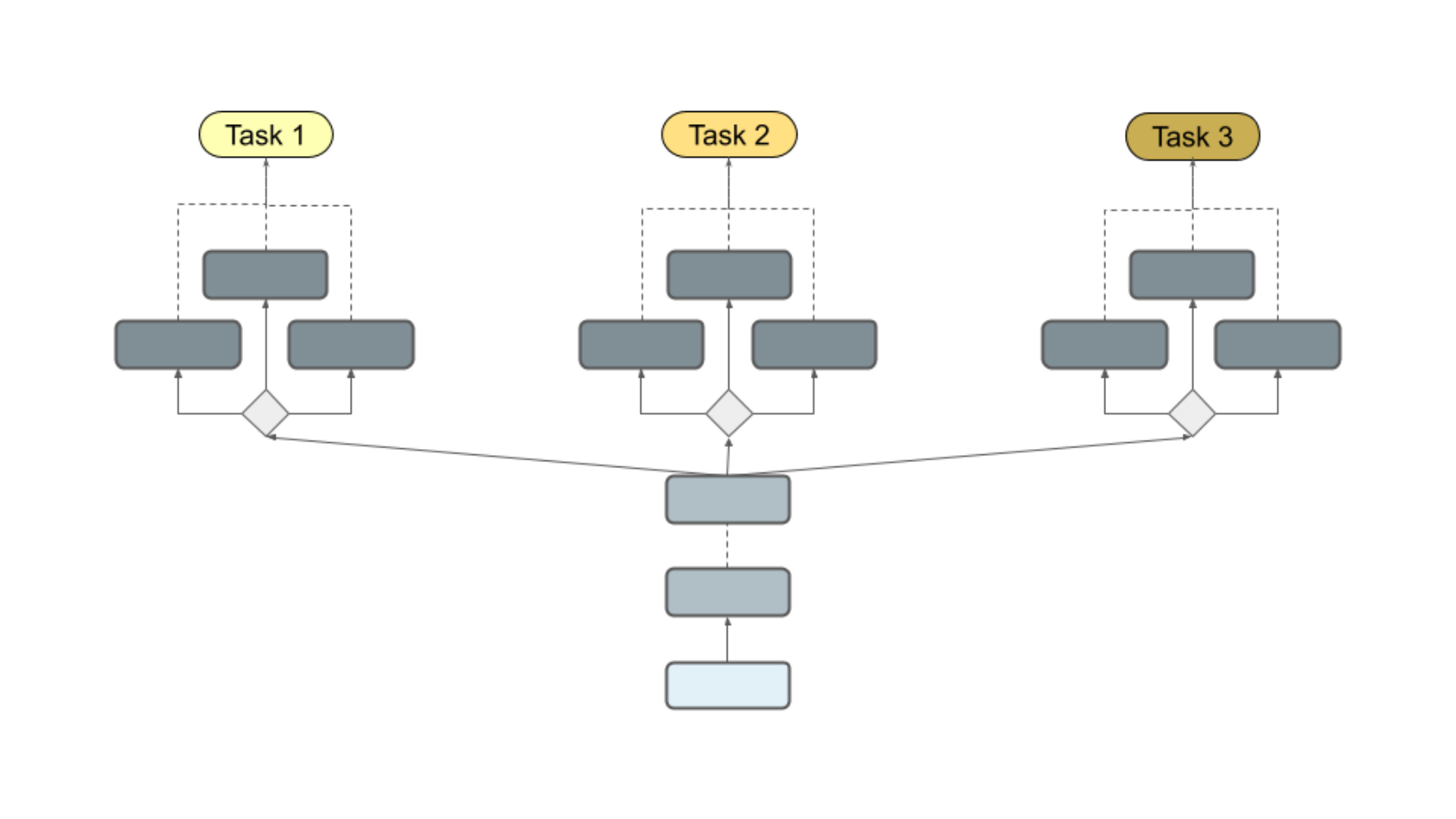
Czy można nauczyć model rozwiązywać to samo zadanie różnymi sposobami?

Pseudo-task Augmentation

Pseudo-task Augmentation (PTA) [9]:

- $((\{\mathbf{x}_{ti}, \mathbf{y}_{ti}\}_{i=1}^{N_t}; \mathcal{F}; \mathcal{D}_{td}; \mathcal{L}_t)_{d=1}^{D})_{t=1}^{T}$ - zadania
- każde zadanie t ma D różnych dekoderów $\{\mathcal{D}_{td}\}_{d=1}^{D}$

Pseudo-task Augmentation



Rysunek 24: W metodzie PTA każde zadanie jest rozwiązywane na D różnych sposobów. W przykładzie na ilustracji $D = 3$.

Pseudo-task Augmentation

Model wykorzystujący dekoder d do zadania t jest zdefiniowany jako

$$\hat{\mathbf{y}}_{tdi} = \mathcal{D}_{td}(\mathcal{F}(\mathbf{x}_{ti}; \theta_{\mathcal{F}}); \theta_{\mathcal{D}_{td}}) \tag{4}$$

PTA - funkcja kosztu

W metodzie PTA, trening polega na znalezieniu parametrów θ^* takich, że

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \frac{1}{TD} \sum_{t=1}^{T} \frac{1}{N_t} \sum_{i=1}^{N_t} \sum_{d=1}^{D} \mathcal{L}_t(\mathbf{y}_{ti}, \hat{\mathbf{y}}_{tdi}) \tag{5}$$

gdzie

$$\theta = (\{\{\theta_{\mathcal{D}_{td}}\}_{d=1}^{D}\}_{t=1}^{T}, \theta_{\mathcal{F}}) \tag{6}$$

PTA - walidacja

Podczas walidacji używany jest najlepszy dekoder dla danego zadania, to znaczy

$$\theta^*_{\text{eval}} = \underset{\theta}{\operatorname{argmin}} \frac{1}{T} \sum_{t=1}^{T} \frac{1}{N_t} \underset{d \in 1 \ldots D}{\operatorname{argmin}} \sum_{i=1}^{N_t} \mathcal{L}_t(\mathbf{y}_{ti}, \hat{\mathbf{y}}_{tdi}) \tag{7}$$

PTA - algorytm treningu

Given T tasks $\{\{x_{ti}, y_{ti}\}_{i=1}^{N_t}\}_{t=1}^{T}$, and D decoders per task
$\{\{\theta_{\mathcal{D}_{td}}\}_{d=1}^{D}\}_{t=1}^{T} \leftarrow$ *DecInitialize()*
Initialize $\theta_{\mathcal{F}}$
Initialize decoder costs $c_{td} \leftarrow \infty \; \forall \; (t, d)$
while not done training **do**
    for $m = 1$ **to** M **do** ▷ M is meta-iteration length
        Update $\theta_{\mathcal{F}}$ and $\theta_{\mathcal{D}_{td}}$ via a joint gradient step.
    for $t = 1$ **to** T **do**
        for $d = 1$ **to** D **do**
            $c_{td} \leftarrow \text{evaluate}(\theta_{\mathcal{D}_{td}}, \theta_{\mathcal{F}}, t)$ ▷ e.g., get validation error
        for $d = 1$ **to** D **do**
            $\theta_{\mathcal{D}_{td}} \leftarrow$ *DecUpdate*$\left(d, \{\theta_{\mathcal{D}_{td_o}}, c_{td_o}\}_{d_o=1}^{D}\right)$
return $\left(\{\{\theta_{\mathcal{D}_{td}}\}_{d=1}^{D}\}_{t=1}^{T}, \theta_{\mathcal{F}}\right)$

PTA - inicjalizacja dekoderów

- Independent Initialization (I) - niezależna losowa inicjalizacja wszystkich $\theta_{\mathcal{D}_{td}}$
- Freeze (F) - zamrożenie wszystkich wag dekoderów oprócz $\theta_{\mathcal{D}_{t1}}$
- Independent Dropout (D) - dropout przed każdym dekoderem (opcjonalnie z różną intensywnością)

PTA - inicjalizacja dekoderów

Niech c_{td} oznacza błąd na zbiorze walidacyjnym i $c_t^{\min} = \min(c_{t1}, \ldots, c_{tD})$.

- Perturb (P) - dodanie szumu $\sim \mathcal{N}(0, \epsilon_p I)$ do parametrów $\theta_{\mathcal{D}_{td}}$ dla każdego d takiego, że $c_{td} \neq c_t^{\min}$
- Hyperperturb (H) - jak Perturb, tylko że *DecUpdate* aktualizuje hiperparametry (dropout) nienajlepszych dekoderów dodając szum $\sim \mathcal{N}(0, \epsilon_h)$
- Greedy (G) - dla każdego zadania, niech $\theta_t^{\min}$ będzie wagami najlepszego dekodera z kosztem $c_t^{\min}$. Wtedy, *DecUpdate* aktualizuje wszystkie $\theta_{td} := \theta_t^{\min}$, włączając hiperparametry.

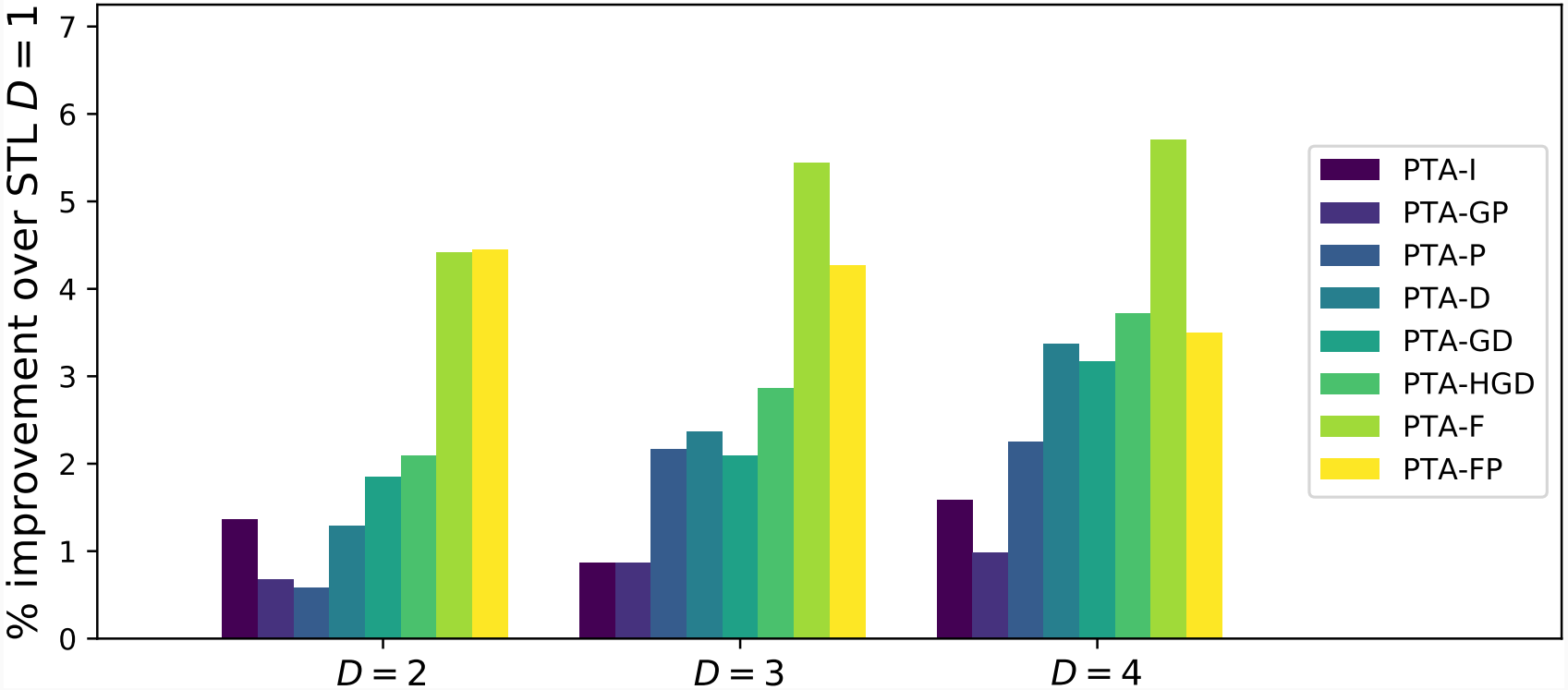
PTA - inicjalizacja dekoderów

Wymienione metody inicjalizacji mogą być ze sobą łączone tworząc klasę algorytmów PTA-*.

PTA - wyniki Omniglot

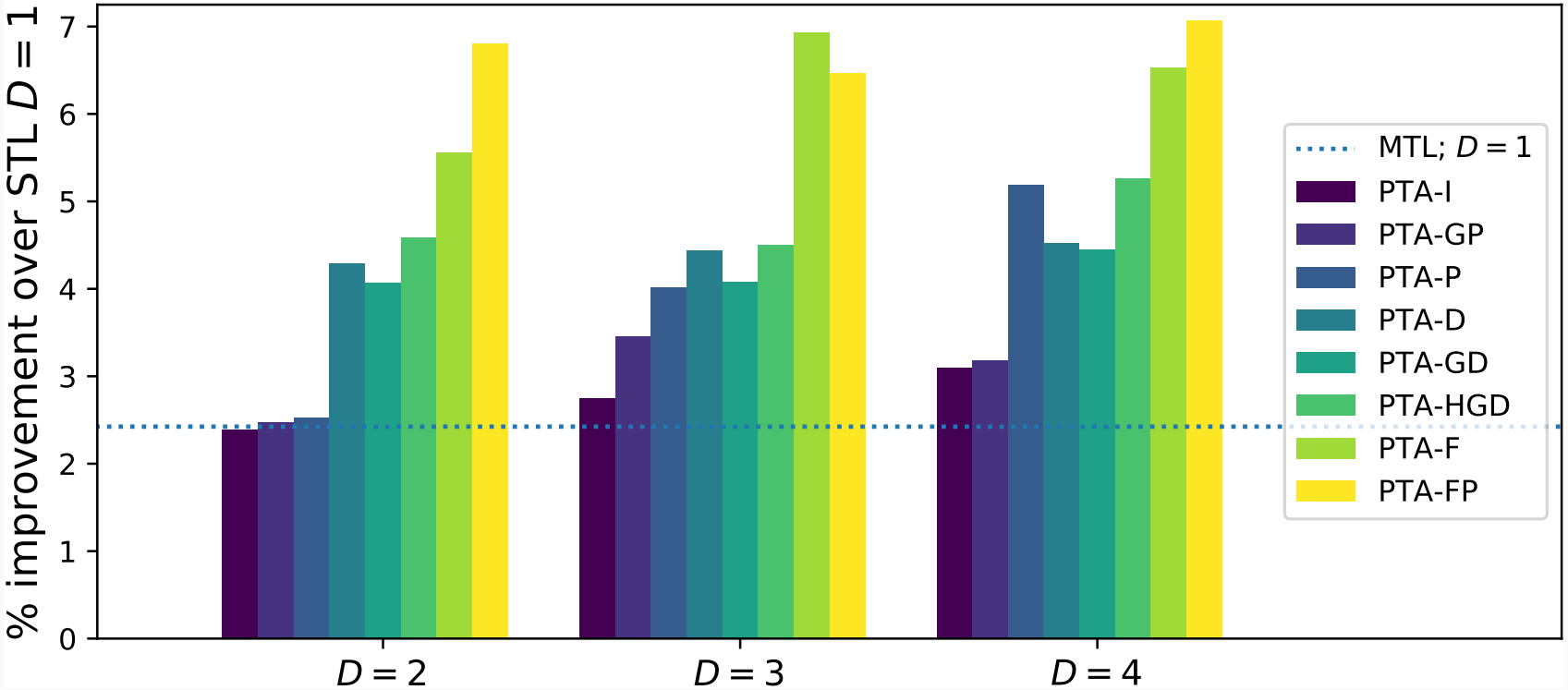
Rysunek 25: Próbka ze zbioru danych Omniglot [6], który zawiera czarno-białe zdjęcia znaków z 50 alfabetów o wymiarze 105×105.

PTA - wyniki Omniglot STL



Rysunek 26: Średnia poprawa nad $D = 1$ uśredniona dla 20 zadań ze zbioru Omniglot rośnie wraz z D: 1.86% dla $D = 2$; 2.33% dla $D = 3$ i 2.70% dla $D = 4$.

PTA - wyniki Omniglot MTL



Rysunek 27: Średnia poprawa nad $D = 1$ uśredniona dla 50 zadań ze zbioru Omniglot rośnie wraz z D: 3.63% dla $D = 2$; 4.07% dla $D = 3$ i 4.37% dla $D = 4$.

PTA - pomysły na dalsze prace

- Zamiana liniowych dekoderów na nieliniowe
- Różne architektury dekoderów, np. warstwy liniowe z 2^i neuronami dla $i \in [0, 10]$
- Automatyczny wybór optymalnej architektury dekodera (metody neuroewolucyjne)
- Inne metody zapewniające różnorodność dekoderów

Podsumowanie

Podsumowanie

- Inductive transfer learning - transfer wiedzy z problemu wstępnego do docelowego
- SSL - transfer wiedzy z problemu wstępnego bez etykiet
- MTL jest przydatny gdy zadania są powiązane
- Zadania pomocnicze pomagają odkryć sedno problemu
- MTL może być wykorzystany do STL za pomocą PTA

Czy metody transferu wiedzy są drogą do Silnej Sztucznej Inteligencji?

Zapraszam do dyskusji

Bibliografia i

Amit Chaudhary.
Self supervised representation learning in nlp.
https://amitness.com/2020/05/self-supervised-learning-nlp/, 2020.
Accessed: 2021-04-13.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton.
A simple framework for contrastive learning of visual representations.
In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

Bibliografia ii

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova.
Bert: Pre-training of deep bidirectional transformers for language understanding.
arXiv preprint arXiv:1810.04805, 2018.

Carl Doersch, Abhinav Gupta, and Alexei A Efros.
Unsupervised visual representation learning by context prediction.
In *Proceedings of the IEEE international conference on computer vision*, pages 1422–1430, 2015.

Bibliografia iii

Nikos Komodakis and Spyros Gidaris.
Unsupervised representation learning by predicting image rotations.
In *International Conference on Learning Representations (ICLR)*, 2018.

Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum.
Human-level concept learning through probabilistic program induction.
Science, 350(6266):1332–1338, 2015.

Bibliografia iv

Yann LeCun and Ishan Misra.
Self-supervised learning: The dark matter of intelligence.
https://ai.facebook.com/blog/self-supervised-learning-the-dark-matter-of-intelligence/, 2021.

Accessed: 2021-04-13.

Mikołaj Małkiński and Jacek Mańdziuk.
Multi-label contrastive learning for abstract visual reasoning.
arXiv preprint arXiv:2012.01944, 2020.

Bibliografia v

Elliot Meyerson and Risto Miikkulainen.
Pseudo-task augmentation: From deep multitask learning to intratask sharing—and back.
In *International Conference on Machine Learning*, pages 3511–3520. PMLR, 2018.

Ludwig Schubert, Michael Petrov, Shan Carter, Nick Cammarata, Gabriel Goh, and Chris Olah.
Openai microscope.
https://openai.com/blog/microscope/, 2020.
Accessed: 2021-04-13.

Bibliografia vi

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.