

# Contributions to Deep Transfer Learning

## From Supervised to Reinforcement Learning

**Matthia Sabatelli**

Montefiore Institute, Department of Electrical Engineering and  
Computer Science, Université de Liège, Belgium

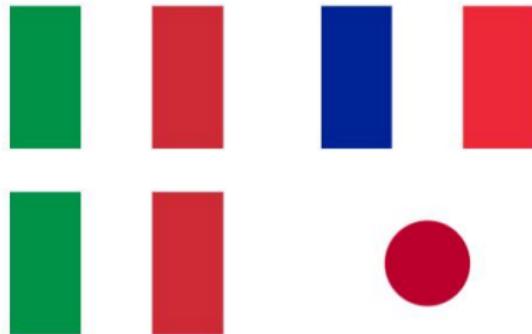
March 30th 2022

# PART I

# ① Part I: Preliminaries

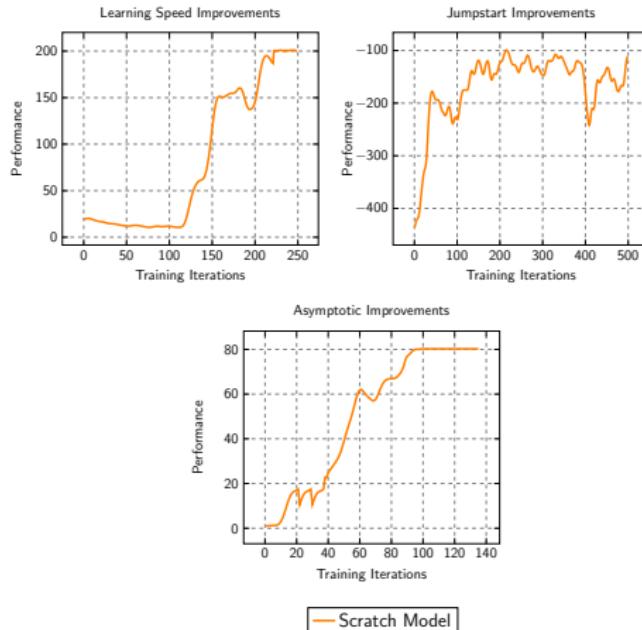
## Transfer Learning

# Transfer Learning



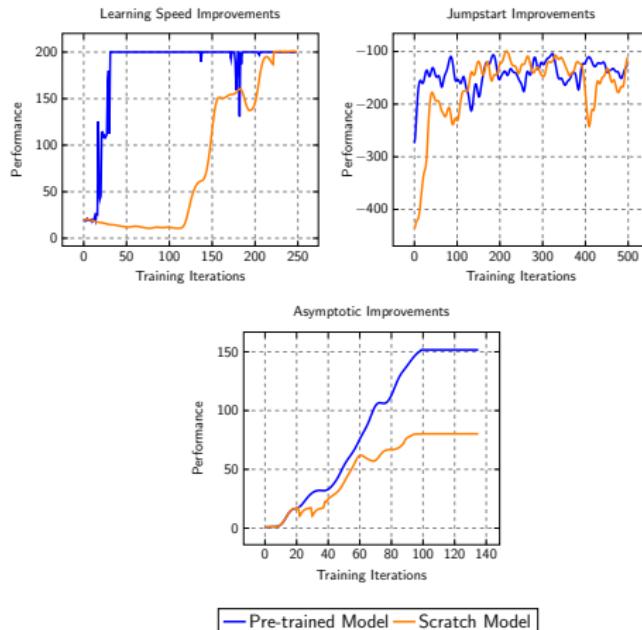
# Transfer Learning

Transfer Learning in practice ...



# Transfer Learning

Transfer Learning in practice ...



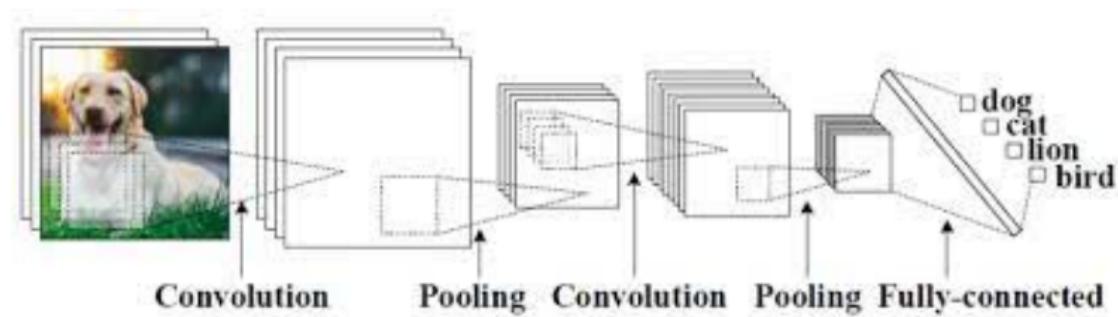
# Transfer Learning

The family of machine learning models that is studied under the lens of transfer learning is that of [Convolutional Neural Networks](#)

- One of the main architectures when it comes to high-dimensional and spatially organized inputs
- Have achieved state-of-the-art results across many different machine learning problems
- We focus our analysis on Supervised Learning and Reinforcement Learning

# Transfer Learning

When it comes to Supervised Learning ...



# Transfer Learning

When it comes to Reinforcement Learning ...

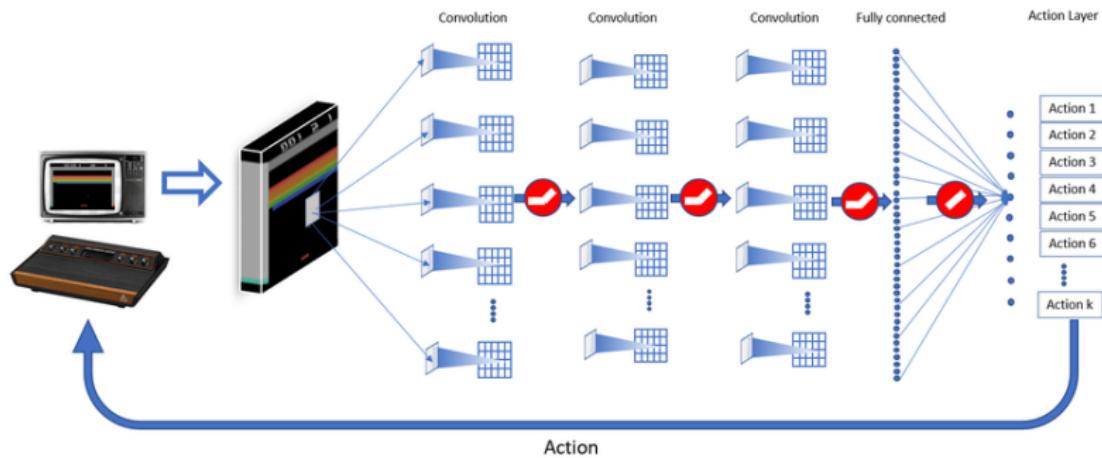


Figure: Image courtesy of Patel et al. (2019)

# Transfer Learning

Why is Transfer Learning so interesting?

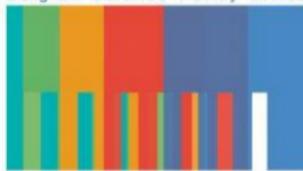
- It allows us to train deep learning models when there is a **lack** of training data
- Is useful when computational resources are **scarce**
- Results in **better performing** models
- Provides a unique tool for **better understanding** neural networks
- Having models that generalize and transfer across different domains and tasks is **key** for the development of Artificial Intelligence

## PART II

## ② Part II: Supervised Learning

- On the Transferability of Convolutional Neural Networks
- On the Transferability of Lottery Winners

Belgian Science Policy Office



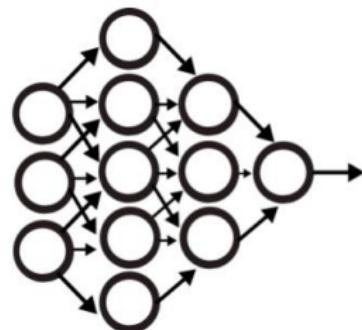
We start by defining the components of **Supervised Learning**:

- An input space  $\mathcal{X}$ ,
- An output space  $\mathcal{Y}$ ,
- A joint probability distribution  $P(X, Y)$
- A loss function  $\ell : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}$

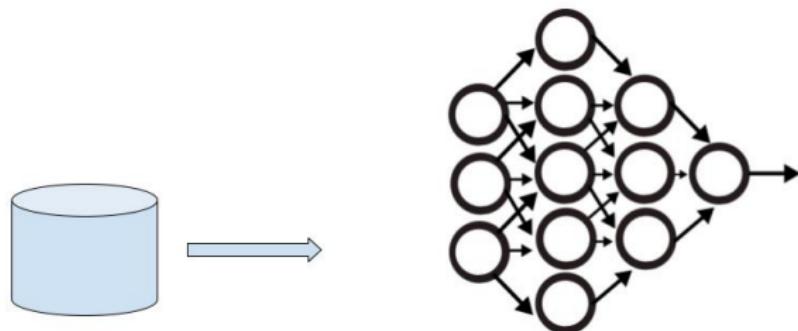
The **goal** is to find a function  $f : \mathcal{X} \rightarrow \mathcal{Y}$  that minimizes the expected risk:

$$R(f) = \mathbb{E}_{(\mathbf{x}, y) \sim P(X, Y)} [\ell(y, f(\mathbf{x}))]$$

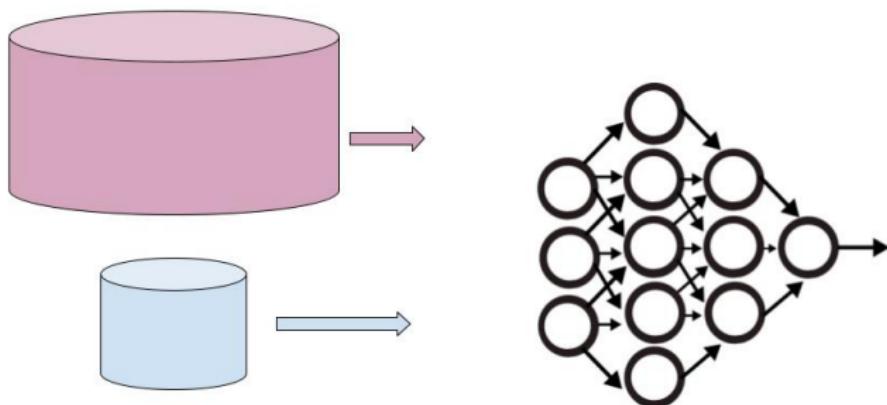
# On the Transferability of Convolutional Neural Networks



# On the Transferability of Convolutional Neural Networks



# On the Transferability of Convolutional Neural Networks

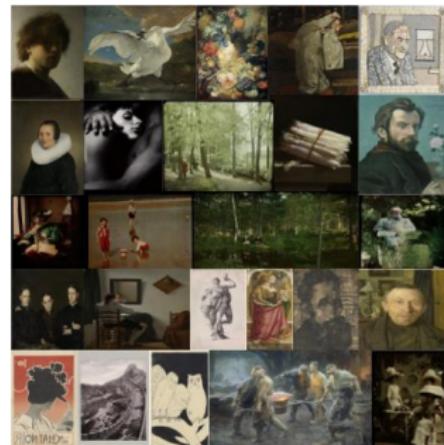


# On the Transferability of Convolutional Neural Networks

As source-task  $\mathcal{T}_S$  we have a CNN pre-trained on the ImageNet dataset



We have three target-tasks  $\mathcal{T}_T$  defined by the Rijksmuseum collection



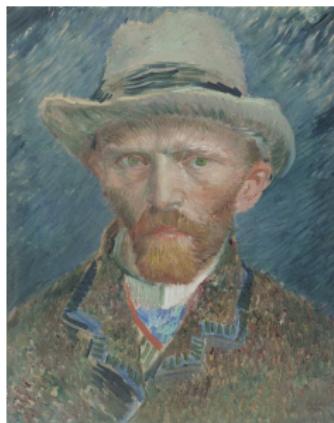
# On the Transferability of Convolutional Neural Networks

More specifically we aim to classify the artworks of the Rijksmuseum by:

Material:  $\mathcal{T}_T$  ①

Type:  $\mathcal{T}_T$  ②

Artist:  $\mathcal{T}_T$  ③



# On the Transferability of Convolutional Neural Networks

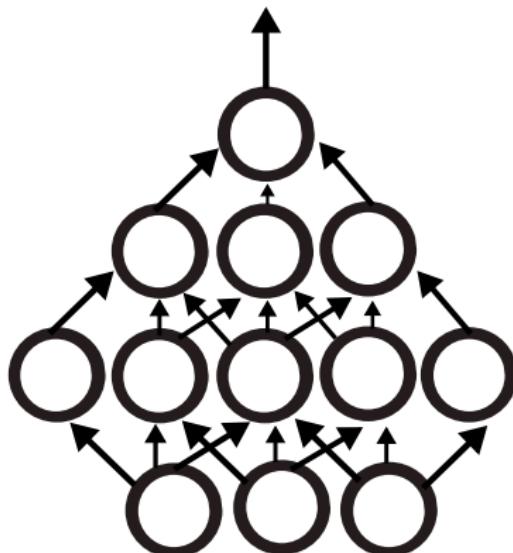
$T_T$	$N_t$	$Q_t$
Material ①	110.668	206
Type ②	112.012	1.054
Artist ③	82.018	1.196

⇒ Experimental setup ...

- We compare the transfer learning performance of four CNN architectures
- Explore three training strategies

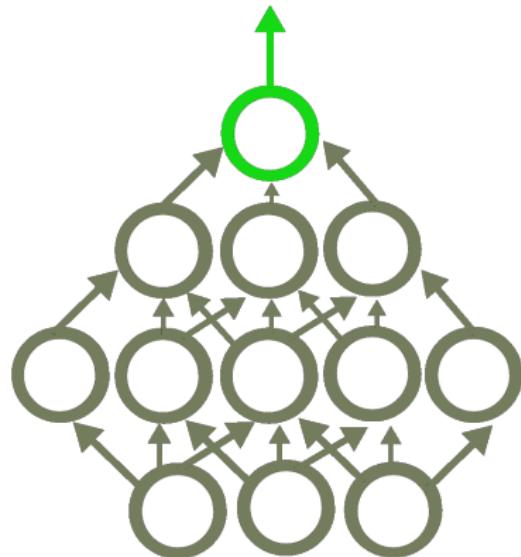
# On the Transferability of Convolutional Neural Networks

- Let us consider a simplified neural network



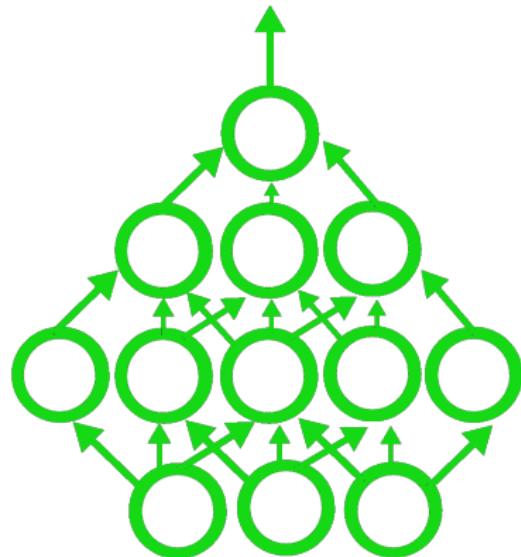
# On the Transferability of Convolutional Neural Networks

- The **off-the-shelf** approach



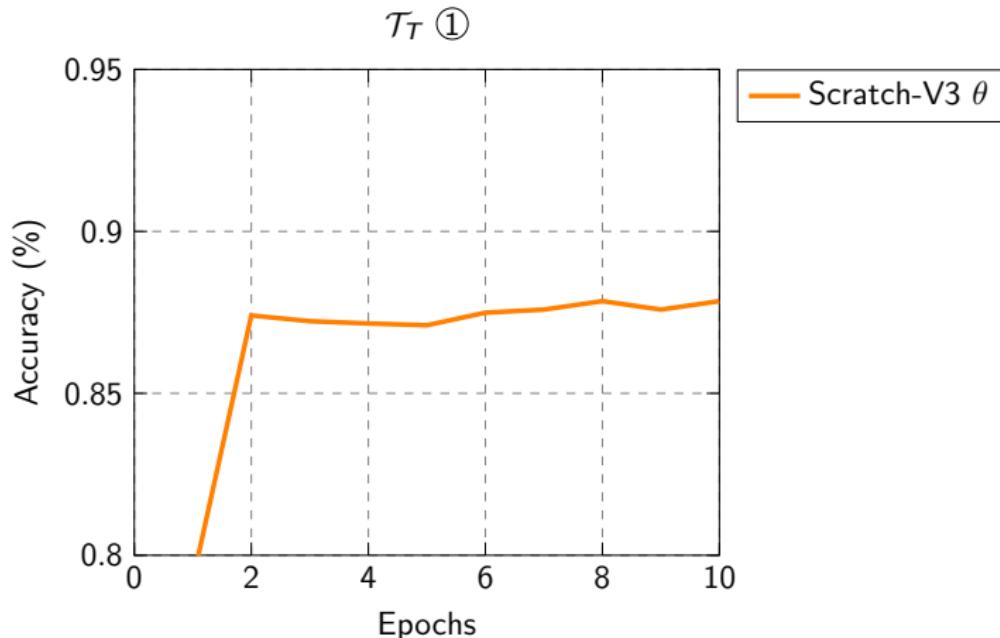
# On the Transferability of Convolutional Neural Networks

- The **fine-tuning** approach



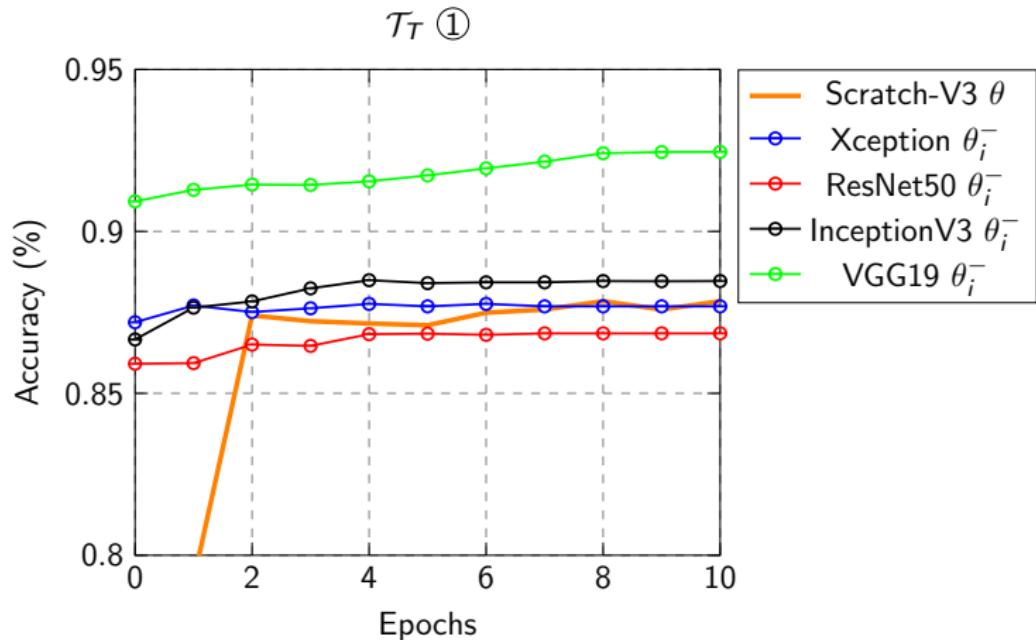
# On the Transferability of Convolutional Neural Networks

The performance on  $\mathcal{T}_T$  ①



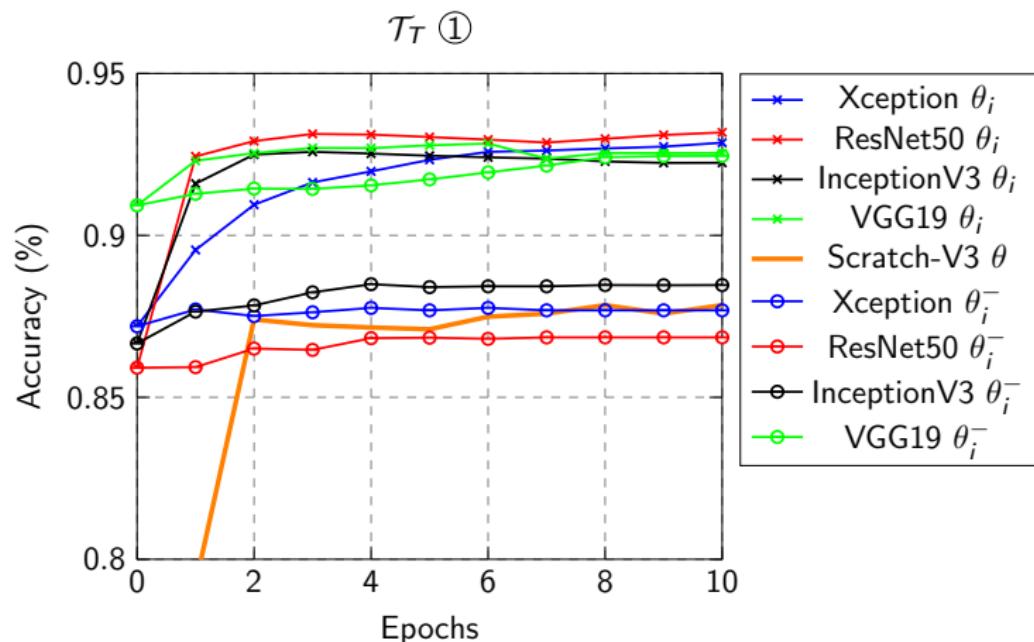
# On the Transferability of Convolutional Neural Networks

The performance on  $\mathcal{T}_T$  ①



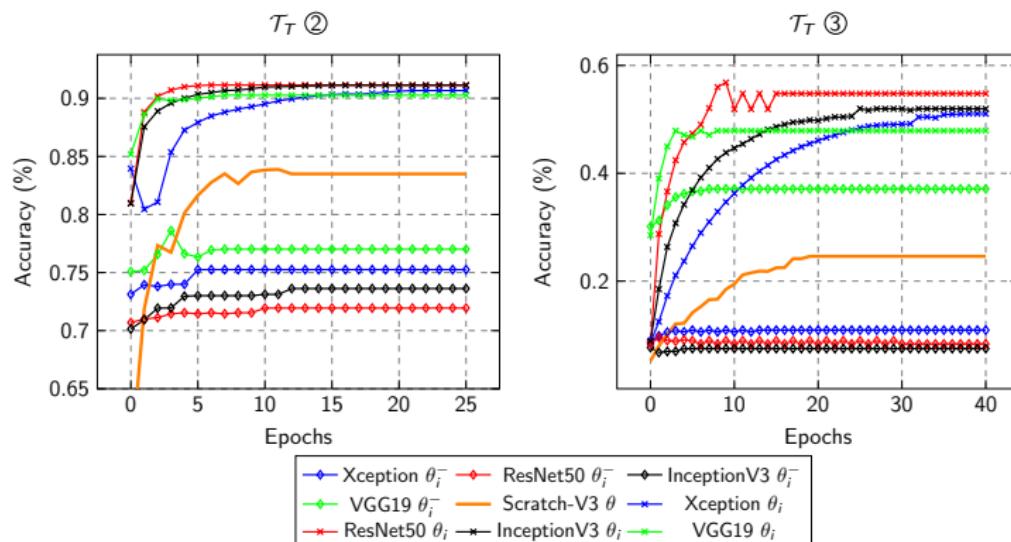
# On the Transferability of Convolutional Neural Networks

The performance on  $\mathcal{T}_T$  ①



# On the Transferability of Convolutional Neural Networks

The performance on  $\mathcal{T}_T$  ② and  $\mathcal{T}_T$  ③



# On the Transferability of Convolutional Neural Networks

We observe that:

- Both transfer learning approaches have significant **benefits**
- The off-the-shelf approach performs **well** for  $\mathcal{T}_T$  ① and  $\mathcal{T}_T$  ② but **fails** for  $\mathcal{T}_T$  ③
- Fine-tuning the networks results in **best** performance

# On the Transferability of Convolutional Neural Networks

To conclude we show that

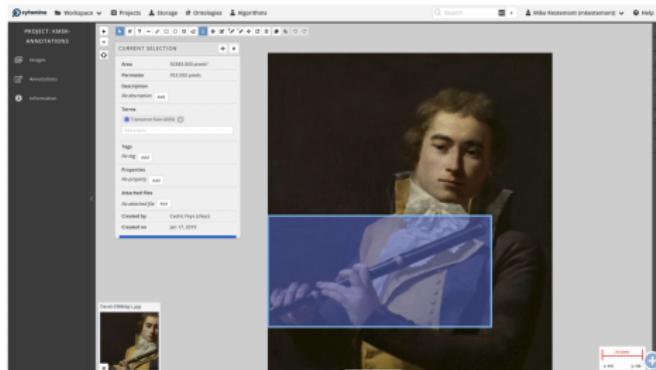
- Modern Convolutional Neural Networks exhibit **strong transfer learning** potential
- Even when the source task  $\mathcal{T}_S$  (natural images) and the target task are very different  $\mathcal{T}_T$  (art classification)
- Best performance is obtained after fine-tuning, which however comes at a computational cost

The present study only considers the computer vision task of **image classification** ...

# On the Transferability of Convolutional Neural Networks

⇒ Which is a **limitation** which we have addressed in our follow up work, where we perform a similar study for the computer vision task of **object detection**

For this purpose we have created the MINERVA dataset within the open-source Cytomine platform <sup>1</sup>

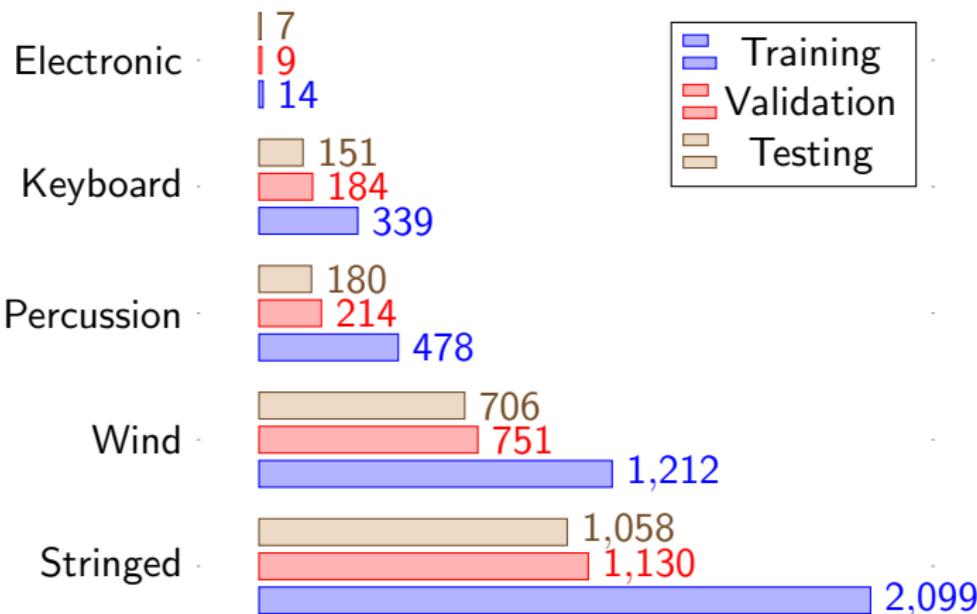


---

<sup>1</sup>Marée, Raphaël, et al. "Collaborative analysis of multi-gigapixel imaging data using Cytomine." Bioinformatics 32.9 (2016): 1395-1401.

# On the Transferability of Convolutional Neural Networks

Hypernym distribution in Minerva-0



# On the Transferability of Convolutional Neural Networks



# On the Transferability of Convolutional Neural Networks

In line with the classification experiments performed on the Rijksmuseum dataset, also when it comes to object detection we show that

- Pre-trained object detectors **can get transferred** across domains
- Fine-tuning these networks results in **best** performance
- This is the case for **different families** of object detection models ranging from YOLO, to networks that use region proposals and selective search <sup>2</sup>.

---

<sup>2</sup>Claes, Yann. "Deep Learning for the Classification and Detection of Animals in Artworks." (2021).

# On the Transferability of Lottery Winners

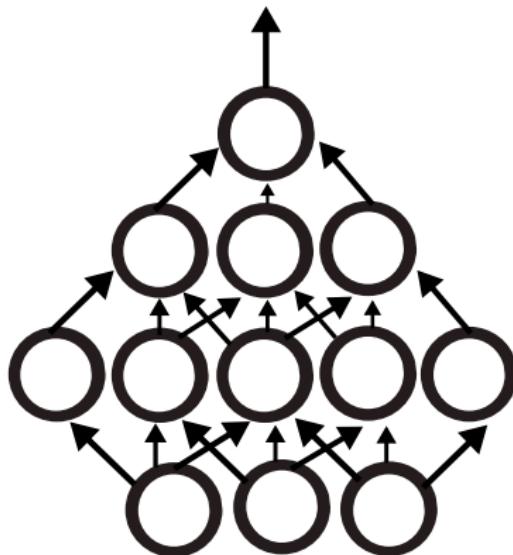
# On the Transferability of Lottery Winners

**The Lottery Ticket Hypothesis:** "*A randomly-initialized dense neural network contains a subnetwork that is initialized such that when trained in isolation it can match the test accuracy of the original network after training for at most the same number of iterations.<sup>3</sup>*"

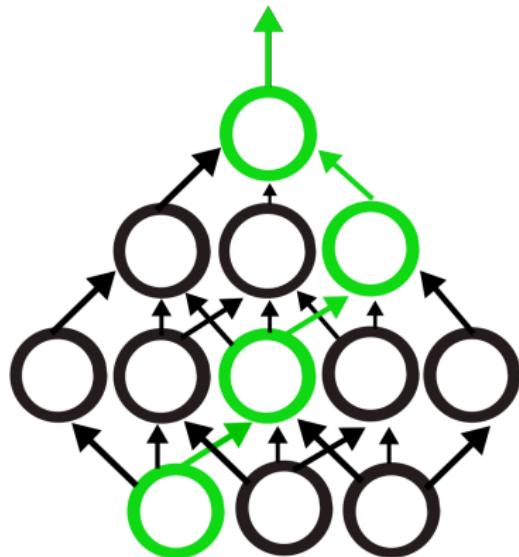
---

<sup>3</sup>Frankle, Jonathan, and Michael Carbin. "The lottery ticket hypothesis: Finding sparse, trainable neural networks." (2018)

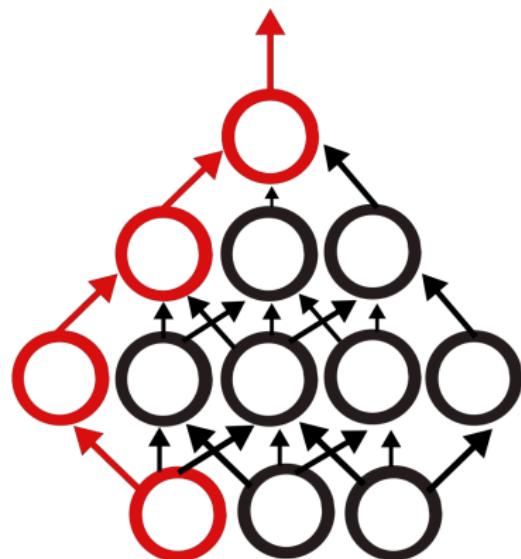
# On the Transferability of Lottery Winners



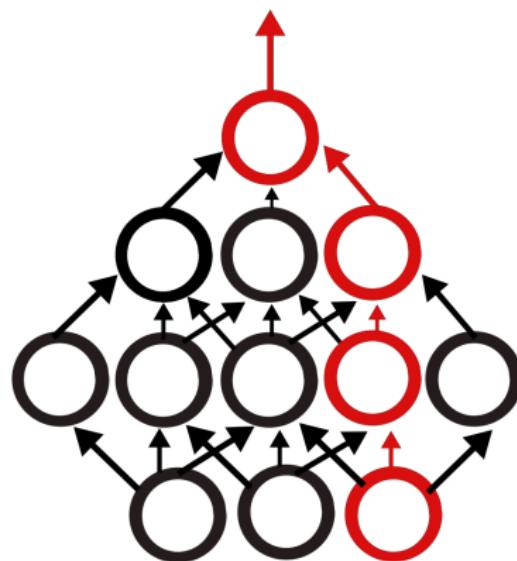
# On the Transferability of Lottery Winners



# On the Transferability of Lottery Winners



# On the Transferability of Lottery Winners



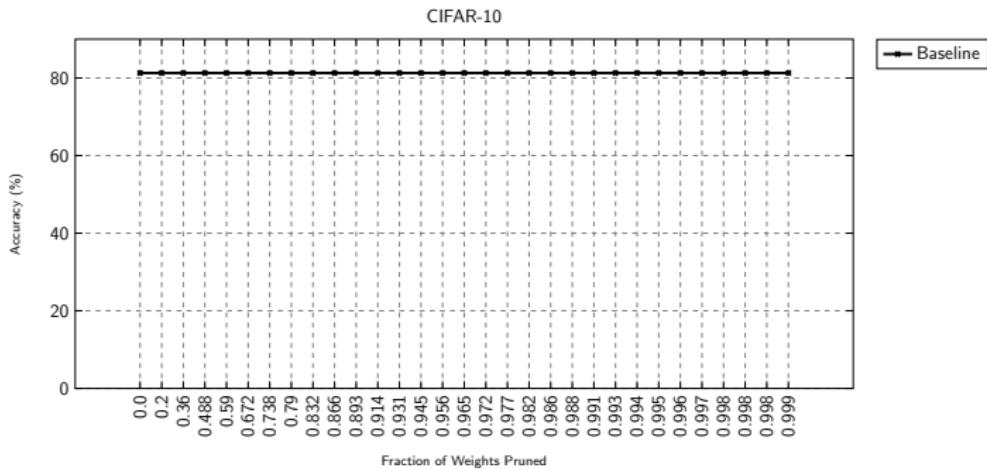
# On the Transferability of Lottery Winners

⇒ How does one **find** a winning ticket?

- Randomly initialize a network  $f(x; \theta_0)$  where  $\theta_0 \sim \mathcal{D}_\theta$
- We train the network for  $j$  iterations
- We prune  $p\%$  of the parameters in  $\theta_j$  creating a mask  $m$
- Reset the remaining parameters to their values at  $\theta_0$ ,  
creating a winning ticket  $f(x; m \odot \theta_0)$

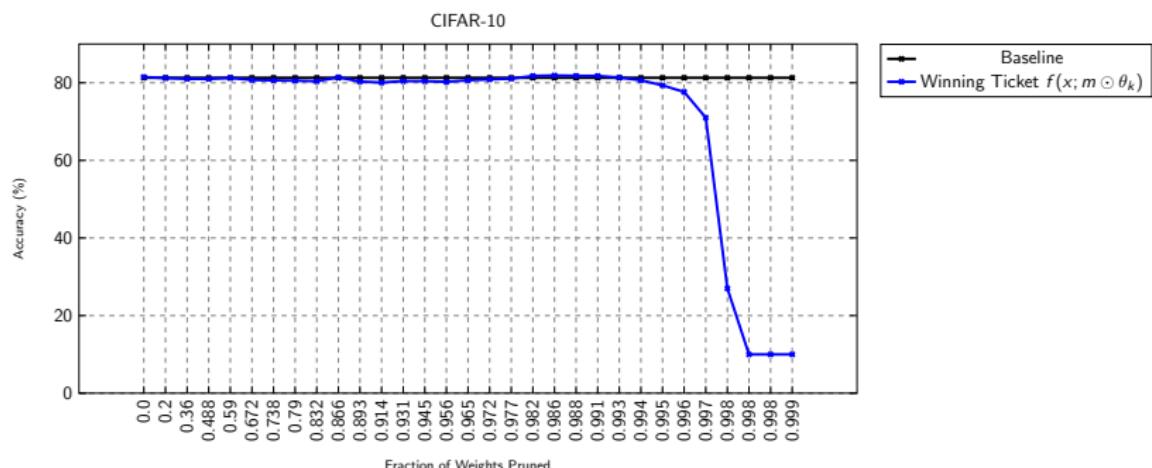
# On the Transferability of Lottery Winners

The Lottery Ticket Hypothesis in practice ...



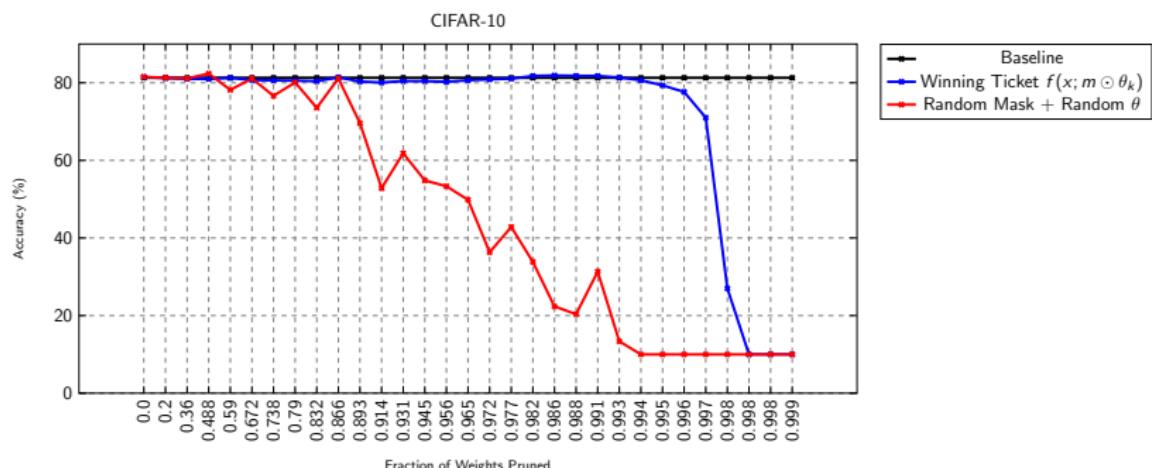
# On the Transferability of Lottery Winners

The Lottery Ticket Hypothesis in practice ...



# On the Transferability of Lottery Winners

The Lottery Ticket Hypothesis in practice ...



# On the Transferability of Lottery Winners

Why are lottery tickets  $f(x; m \odot \theta_0)$  so **special**?

- Train faster
- Faster Inference
- (Sometimes) obtain a better final performance

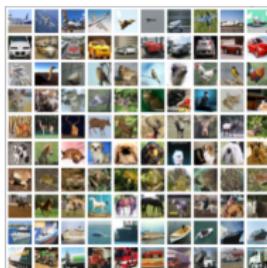
However ...

- Identifying a winning ticket is **computationally expensive**
- We **do not** know how and why lottery winners appear throughout learning

# On the Transferability of Lottery Winners

⇒ Therefore we study whether winning tickets  $f(x; m \odot \theta_0)$  found on natural image datasets can get transferred to the non-natural realm

We use three popular Computer Vision datasets as source domains  $\mathcal{D}_S$ :



# On the Transferability of Lottery Winners

⇒ Therefore we study whether winning tickets  $f(x; m \odot \theta_0)$  found on natural image datasets can get transferred to the non-natural realm

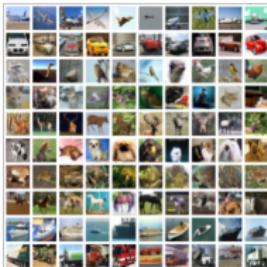
We use three popular Computer Vision datasets as source domains  $\mathcal{D}_S$ :



# On the Transferability of Lottery Winners

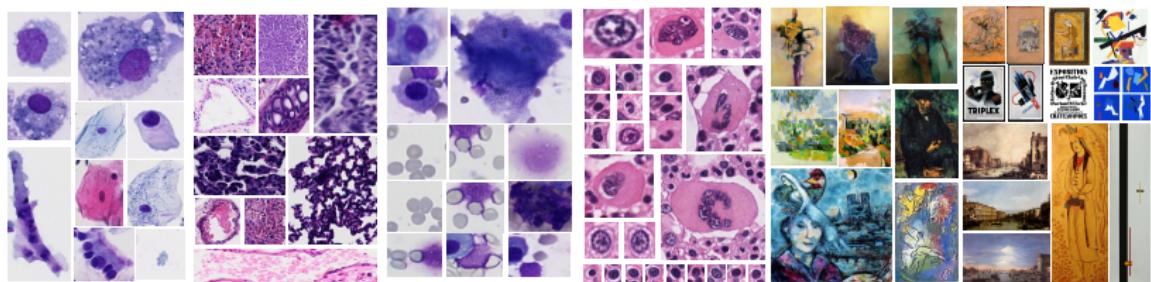
⇒ Therefore we study whether winning tickets  $f(x; m \odot \theta_0)$  found on natural image datasets can get transferred to the non-natural realm

We use three popular Computer Vision datasets as source domains  $\mathcal{D}_S$



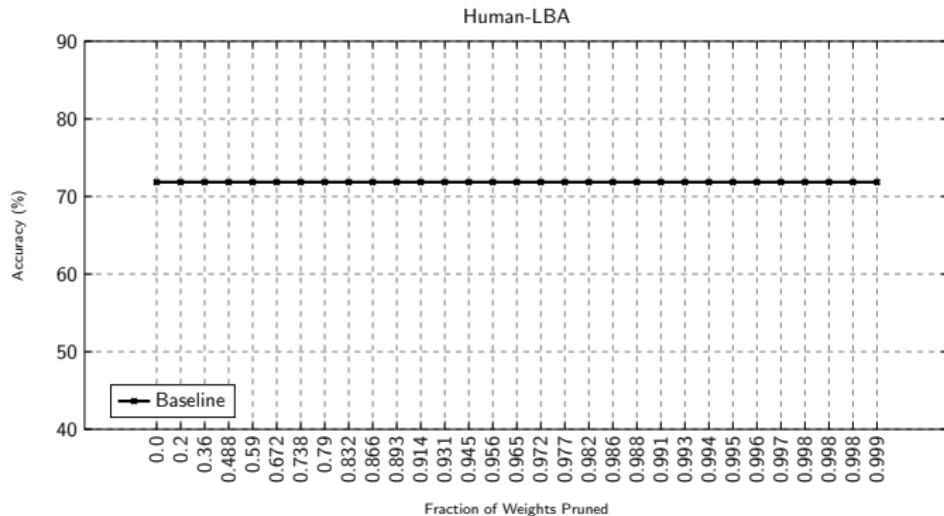
# On the Transferability of Lottery Winners

And we use **seven datasets** of non-natural images as target tasks  $\mathcal{T}_T$  coming from the fields of Digital Pathology and Digital Heritage



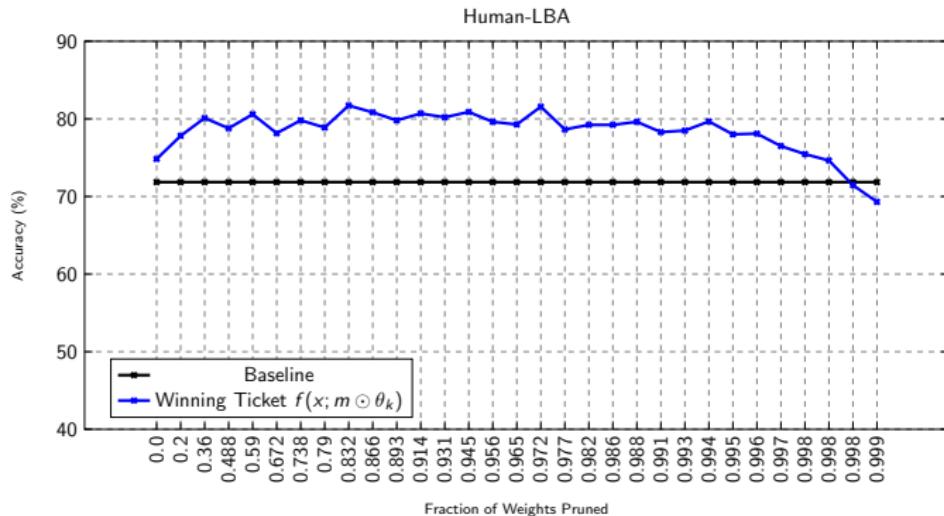
# On the Transferability of Lottery Winners

Transferring lottery winners to the [Digital Pathology](#) data ...



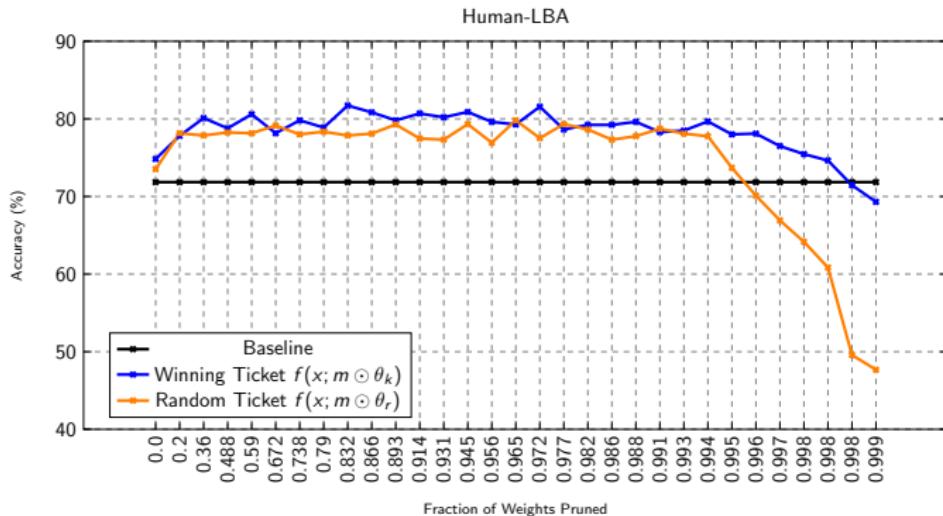
# On the Transferability of Lottery Winners

Transferring lottery winners to the [Digital Pathology](#) data ...



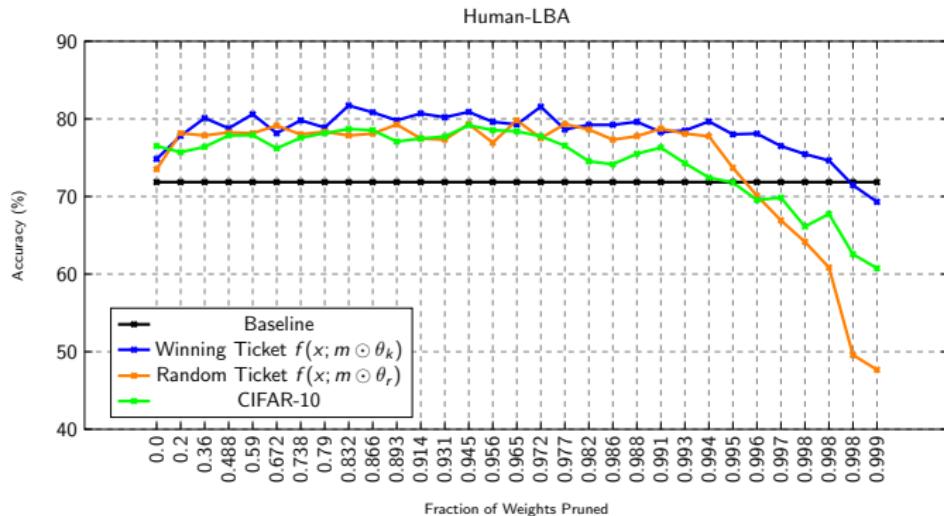
# On the Transferability of Lottery Winners

Transferring lottery winners to the [Digital Pathology](#) data ...



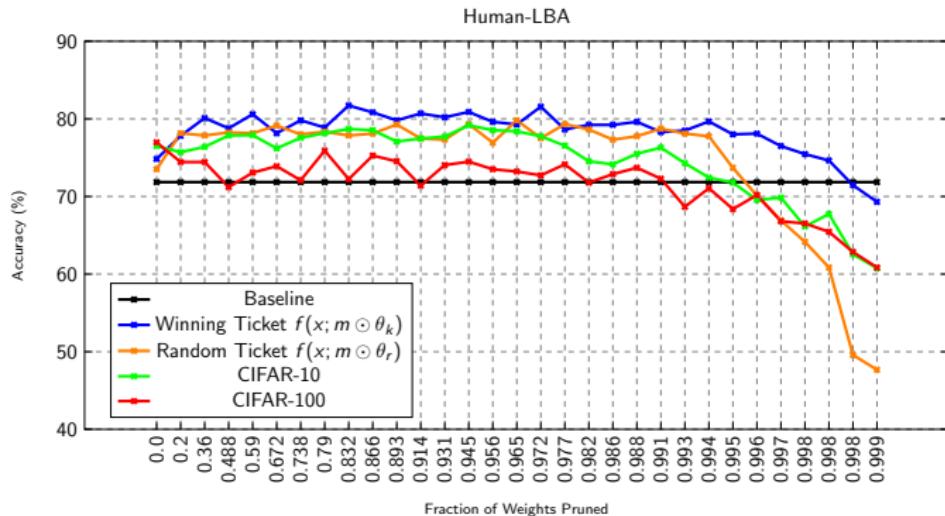
# On the Transferability of Lottery Winners

Transferring lottery winners to the [Digital Pathology](#) data ...



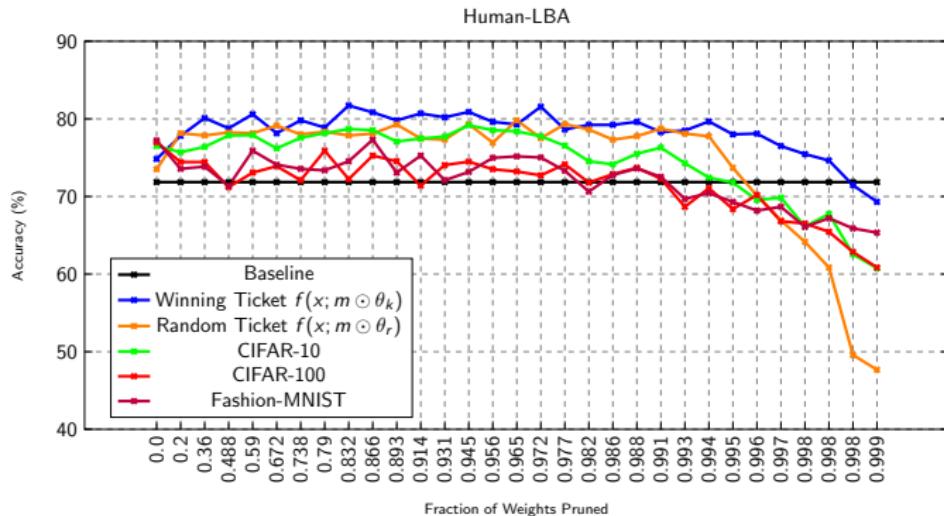
# On the Transferability of Lottery Winners

Transferring lottery winners to the [Digital Pathology](#) data ...



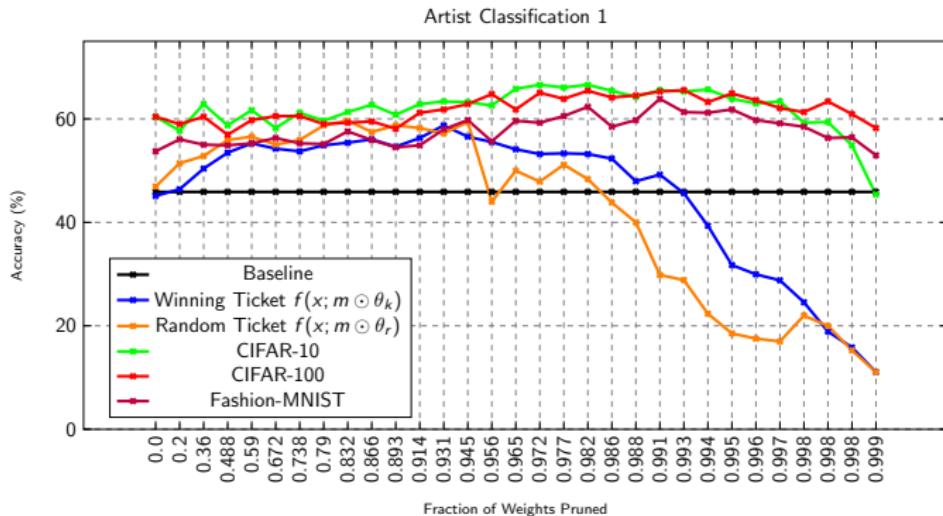
# On the Transferability of Lottery Winners

Transferring lottery winners to the [Digital Pathology](#) data ...



# On the Transferability of Lottery Winners

Transferring lottery winners to the [Digital Heritage](#) data ...



# On the Transferability of Lottery Winners

Our main findings show that on Digital Pathology data

- All lottery winners significantly outperform unpruned models
- Natural lottery winners contain a generic inductive bias (to some extent)
- Best performance is obtained by identifying a winning ticket directly on the target task  $\mathcal{T}_T$

whereas on Digital Heritage data

- Natural lottery winners transfer much better
- They can even outperform target task  $\mathcal{T}_T$  tickets

## On the Transferability of Lottery Winners

We also provide additional empirical insights into the Lottery Ticket Hypothesis:

- We show that completely pre-trained winning tickets  $f(x; m \odot \theta_i)$  **overfit** on the source task  $\mathcal{T}_S$
- The presence of winning tickets **does not depend** on the size of the training data
- The closer the source task  $\mathcal{S}_T$  and the target task  $\mathcal{T}_T$  the **better** the transferability of  $f(x; m \odot \theta_k)$

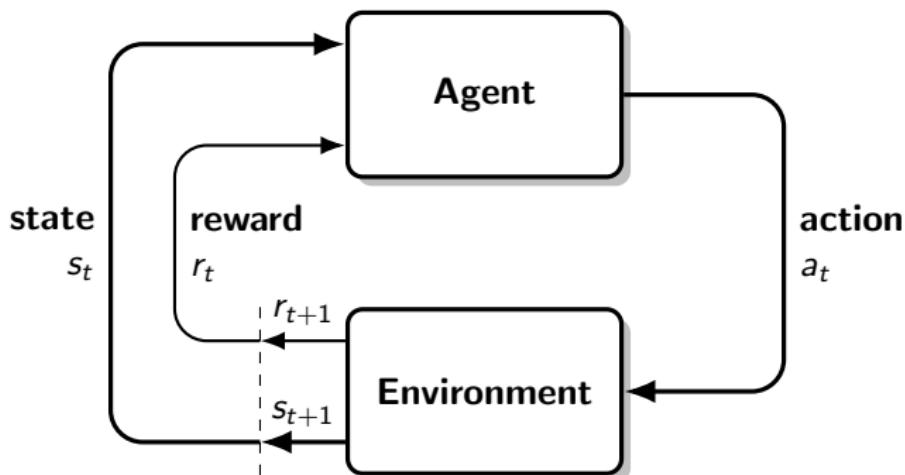
## PART III

### ③ Part III: Reinforcement Learning

A Novel Family of DRL Algorithms  
On the Transferability of Deep-Q Networks

# A Novel Family of DRL Algorithms

We now consider a different machine learning paradigm:  
**Reinforcement Learning (RL)**, where an agent needs to learn  
how to interact with its environment



## A Novel Family of DRL Algorithms

Such interaction is modeled as a [Markov Decision Process](#) (MDP) consisting of the following elements:

- A set of possible states  $\mathcal{S}$ ,
- A set of possible actions  $\mathcal{A}$ ,
- A transition function  $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ ,
- A reward function  $\mathfrak{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$ ,
- A discount factor denoted as  $\gamma \in [0, 1]$ .

Therefore we can define an MDP as  $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathfrak{R}, \gamma \rangle$ .

## A Novel Family of DRL Algorithms

The interaction between the agent and the environment is given by the agent's **policy**  $\pi$ , a probability distribution over  $a \in \mathcal{A}(s)$  for each  $s \in \mathcal{S}$ :

$$\pi(a|s) = \Pr \{a_t = a | s_t = s\}, \text{ for all } s \in \mathcal{S} \text{ and } a \in \mathcal{A}.$$

The **goal** of the agent is to maximize the expected discounted return as:

$$\begin{aligned} G_t &= r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \dots \\ &= \sum_{k=0}^{\infty} \gamma^k r_{t+k}. \end{aligned}$$

# A Novel Family of DRL Algorithms

In RL some components of the MDP are unknown

$$\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle,$$

which in the case of model-free RL can be overcome by learning either the state-value function

$$V^\pi(s) = \mathbb{E} \left[ \sum_{k=0}^{\infty} \gamma^k r_{t+k} \middle| s_t = s, \pi \right],$$

or the state-action value function

$$Q^\pi(s, a) = \mathbb{E} \left[ \sum_{k=0}^{\infty} \gamma^k r_{t+k} \middle| s_t = s, a_t = a, \pi \right].$$

# A Novel Family of DRL Algorithms

- Model-free algorithms are implemented in a **tabular** fashion, meaning that the state values, or state-action values, are stored within tables of sizes  $|\mathcal{S}|$  and  $|\mathcal{S} \times \mathcal{A}|$  respectively
- For many problems we typically seek to learn an **approximation** of the value functions

$$V^\pi(s) \approx V^\pi(s; \theta) \text{ and } Q^\pi(s, a; \theta) \approx Q(s, a; \theta)$$

This approximation can be represented by a **Convolutional Neural Network**

# A Novel Family of DRL Algorithms

Typical DRL algorithms aim at only learning an approximation of the state-action value function  $Q^\pi(s, a) \approx Q^\pi(s, a; \theta)$

- A policy  $\pi$  can only be derived from  $Q^\pi(s, a)$
- It is hard to combine RL algorithms with neural networks
- Dealing with one value function can be complicated enough

However ...

- Training can be very slow
- Algorithms are prone to diverge
- Do not correctly estimate  $Q^\pi(s, a)$

## A Novel Family of DRL Algorithms

⇒ Therefore we suggest to **jointly approximate** the state-value function  $V^\pi(s; \phi)$  alongside the state-action value function  $Q^\pi(s, a; \theta)$

We can do this by either training a **DQV** agent which learns  $V^\pi(s; \phi)$  by minimizing

$$L(\phi) = \mathbb{E}_{(s_t, a_t, r_t, s_{t+1}) \sim U(D)} \left[ (r_t + \gamma V(s_{t+1}; \phi^-) - V(s_t; \phi))^2 \right],$$

and  $Q^\pi(s, a)$  with:

$$L(\theta) = \mathbb{E}_{(s_t, a_t, r_t, s_{t+1}) \sim U(D)} \left[ (r_t + \gamma V(s_{t+1}; \phi^-) - Q(s_t, a_t; \theta))^2 \right],$$

# A Novel Family of DRL Algorithms

... or by training a **DQV-Max** agent that learns  $V^\pi(s; \phi)$  with

$$L(\phi) = \mathbb{E}_{\langle s_t, a_t, r_t, s_{t+1} \rangle \sim U(D)} \left[ (r_t + \gamma \max_{a \in \mathcal{A}} Q(s_{t+1}, a; \theta^-) - V(s_t; \phi))^2 \right],$$

and  $Q^\pi(s, a)$  with:

$$L(\theta) = \mathbb{E}_{\langle s_t, a_t, r_t, s_{t+1} \rangle \sim U(D)} \left[ (r_t + \gamma V(s_{t+1}; \phi) - Q(s_t, a_t; \theta))^2 \right].$$

# A Novel Family of DRL Algorithms

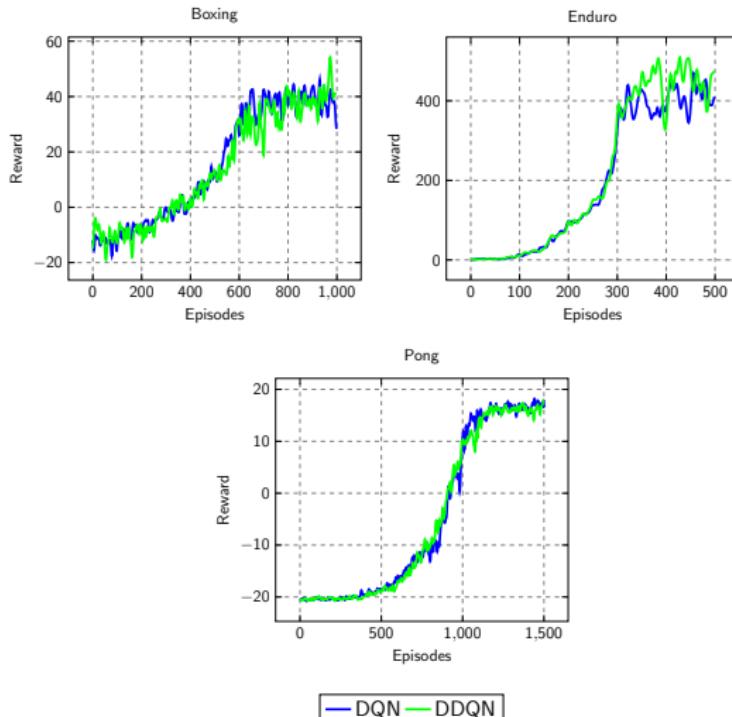
⇒ We compare their performance to algorithms which only learn an approximation of the state-action value function  $Q^\pi(s, a)$ <sup>4 5</sup>

---

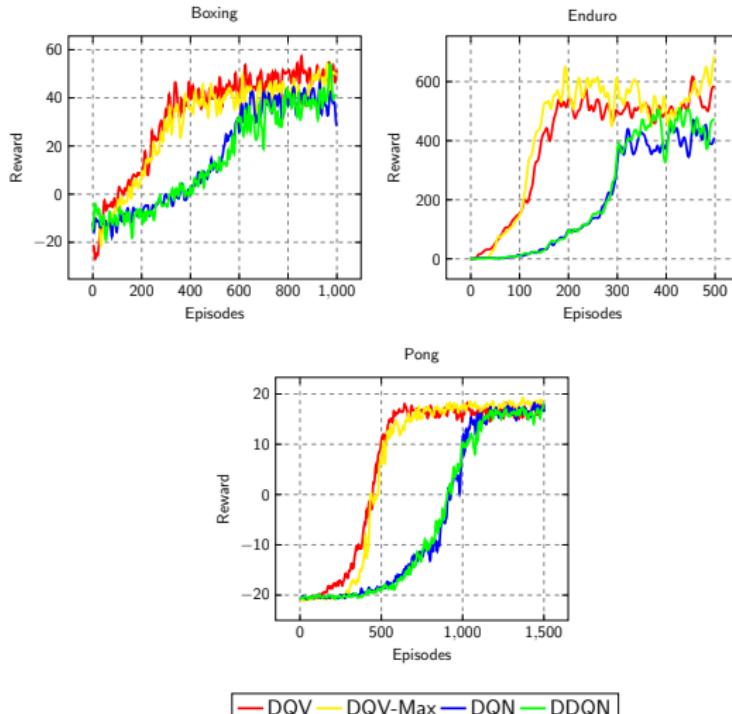
<sup>4</sup>Mnih, Volodymyr, et al. "Human-level control through deep reinforcement learning." *nature* 518.7540 (2015)

<sup>5</sup>Van Hasselt, Hado, Arthur Guez, and David Silver. "Deep reinforcement learning with double q-learning." *Proceedings of the AAAI conference on artificial intelligence* (2016).

# A Novel Family of DRL Algorithms

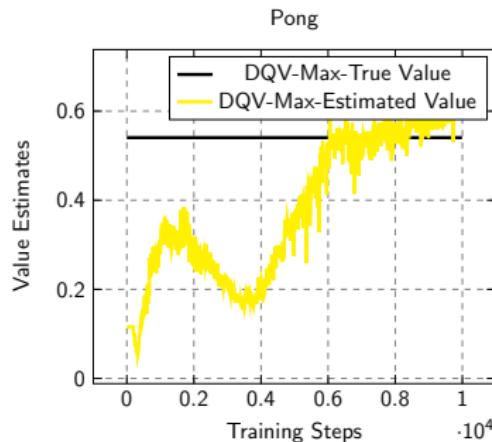
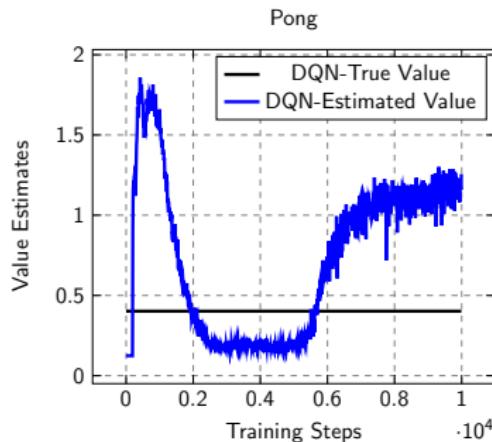


# A Novel Family of DRL Algorithms



# A Novel Family of DRL Algorithms

⇒ Jointly approximating two value functions over one results in faster convergence, but also in accurate value estimates:



# A Novel Family of DRL Algorithms

To conclude DQV-Learning and DQV-Max Learning

- Result in **faster learning**
- Are **robust** to the "Deadly Triad of DRL"
- Suffer **less** from the overestimation bias of the  $Q$  function
- **Scale well** to the multi-agent setting <sup>6</sup>

However ...

- Memory wise are **twice** as expensive as DQN and DDQN
- **Do not** always result in better policies

---

<sup>6</sup>Leroy, Pascal, et al. "QVMix and QVMix-Max: Extending the Deep Quality-Value Family of Algorithms to Cooperative Multi-Agent Reinforcement Learning." (2021).

# On the Transferability of Deep-Q Networks

# On the Transferability of Deep-Q Networks

Let's take a look at the performance of the DQV algorithms more closely ...

Environment	Random	Human	DQN	DDQN	DQV	DQV-Max
Asteroids	719.10	13156.70	1629.33	930.60	1445.40	1846.08
Bank Heist	14.20	734.40	429.67	728.30	1236.50	1118.28
Boxing	0.10	4.30	71.83	81.70	78.66	80.15
Crazy Climber	10780.50	35410.50	114103.33	101874.00	108600.00	1000131.00
Enduro	0.00	309.60	301.77	319.50	829.33	875.64
Fishing Derby	-91.70	5.50	-0.80	20.30	1.12	20.42
Frostbite	65.20	4334.70	328.33	241.50	271.86	281.36
Gopher	257.60	2321.00	8520.00	8215.40	8230.30	7940.00
Ice Hockey	-11.20	0.90	-1.60	-2.40	-1.88	-1.12
James Bond	29.00	406.70	576.67	438.00	372.41	440.80
Montezuma's Revenge	0.00	4366.70	0.00	0.00	0.00	0.00
Ms. Pacman	307.30	15693.40	2311.00	3210.00	3590.00	3390.00
Pong	-20.70	9.30	18.90	21.00	21.00	21.00
Road Runner	11.50	7845.00	18256.67	48377.00	39290.00	20700.00
Zaxxon	32.50	9173.30	4976.67	10182.00	10950.00	8487.00

# On the Transferability of Deep-Q Networks

Let's take a look at the performance of the DQV algorithms more closely ...

Environment	Random	Human	DQN	DDQN	DQV	DQV-Max
Asteroids	719.10	13156.70	1629.33	930.60	1445.40	1846.08
Bank Heist	14.20	734.40	429.67	728.30	1236.50	1118.28
Boxing	0.10	4.30	71.83	81.70	78.66	80.15
Crazy Climber	10780.50	35410.50	114103.33	101874.00	108600.00	1000131.00
Enduro	0.00	309.60	301.77	319.50	829.33	875.64
Fishing Derby	-91.70	5.50	-0.80	20.30	1.12	20.42
<b>Frostbite</b>	<b>65.20</b>	<b>4334.70</b>	<b>328.33</b>	<b>241.50</b>	<b>271.86</b>	<b>281.36</b>
Gopher	257.60	2321.00	8520.00	8215.40	8230.30	7940.00
Ice Hockey	-11.20	0.90	-1.60	-2.40	-1.88	-1.12
James Bond	29.00	406.70	576.67	438.00	372.41	440.80
Montezuma's Revenge	0.00	4366.70	0.00	0.00	0.00	0.00
Ms. Pacman	307.30	15693.40	2311.00	3210.00	3590.00	3390.00
Pong	-20.70	9.30	18.90	21.00	21.00	21.00
Road Runner	11.50	7845.00	18256.67	48377.00	39290.00	20700.00
Zaxxon	32.50	9173.30	4976.67	10182.00	10950.00	8487.00

# On the Transferability of Deep-Q Networks

Based on these results we have formulated the following research questions:

- Can we **improve** the performance of DQV and DQV-Max Learning through Transfer Learning?
- Is parametric transfer **as effective** in the model-free Reinforcement Learning setting as it is in the Supervised Learning setting?

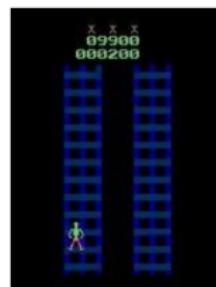
# On the Transferability of Deep-Q Networks

⇒ We study the Transfer Learning potential of several types of Deep-Q Networks

- We start by considering **two variants** of Deep-Q Networks: the DDQN and DQV-Learning algorithms
- The popular Atari games are used as benchmark
- We explore what happens when models that are pre-trained on  $\mathcal{M}_S$  get **fine-tuned** on  $\mathcal{M}_T$

# On the Transferability of Deep-Q Networks

Some examples of source and target tasks ...



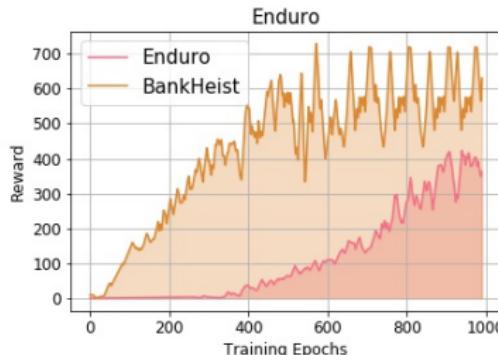
**Figure:** The visually similar Ms Pacman and Bank Heist games.

**Figure:** The highly different Crazy Climber and Pong games.

# On the Transferability of Deep-Q Networks

We quantitatively measure the benefits of transferring and fine-tuning a pre-trained model by computing the **area ratio metric**<sup>7</sup>  $\mathcal{R}$

$$\mathcal{R} = \frac{\text{area of } \mathcal{M}_S - \text{area of } \mathcal{M}_T}{\text{area of } \mathcal{M}_T}$$



<sup>7</sup>Lazaric, Alessandro. "Transfer in reinforcement learning: a framework and a survey." Reinforcement Learning. Springer (2012).

# On the Transferability of Deep-Q Networks

However, we noticed that adopting Transfer Learning strategies performed **surprisingly bad**

DQV	BankHeist	Boxing	CrazyClimber	Enduro	FishingDerby	Frostbite	JamesBond	MsPacman	Pong	Zaxxon
BankHeist	-	-0.019	-1	-0.317	0.5	0.729	0.973	-0.089	-1.238	-0.998
Boxing	-0.494	-	-0.278	-0.852	0.552	-0.01	0.247	-0.184	-0.841	-0.999
CrazyClimber	-0.569	-0.261	-	-0.593	0.19	0.277	0.621	-0.111	-1.206	-0.178
Enduro	-0.571	-0.018	-0.25	-	0.726	-0.017	-0.41	-0.08	-0.466	-0.164
FishingDerby	-1	-0.893	-0.093	-0.45	-	0.068	0.197	-0.136	-3.083	-0.999
Frostbite	-0.933	0.024	-1	-0.348	0.222	-	0.569	0.009	-0.663	-0.076
JamesBond	-0.123	-0.106	-0.131	-0.033	0.519	0.262	-	0.218	-1.329	-1
MsPacman	-0.985	-0.219	-0.012	-0.494	0.6	0.346	0.398	-	-1.646	-0.997
Pong	-1	-0.083	-0.428	-0.476	0.725	-0.024	0.896	0.123	-	-0.729
Zaxxon	-0.76	-0.028	0.037	-0.116	0.385	0.16	-0.253	0.06	-1.602	-

DDQN	BankHeist	Boxing	CrazyClimber	Enduro	FishingDerby	Gopher	IceHockey	Jamesbond	MsPacman	Pong
BankHeist	-	0.121	-0.378	-0.006	-0.107	0.042	-0.006	-0.058	0.001	-3.013
Boxing	-0.316	-	-0.104	-0	0.038	0.06	0.015	-0.225	-0.027	0.936
CrazyClimber	-0.192	-0.487	-	-0.012	-0.084	0.016	0.015	0.016	-0.015	-2.64
Enduro	-0.296	0.193	-0.167	-	0.039	0.03	0.019	-0.235	-0.039	0.248
FishingDerby	-0.212	-0.545	-1	-0.085	-	0.016	0.001	-0.055	-0.026	-0.935
Gopher	-0.466	0.044	-0.108	-0.005	0.007	-	-0.005	-0.094	-0.02	-1.816
IceHockey	-0.046	0.245	-0.067	0.014	-0.178	0.072	-	0.037	-0.015	0.112
Jamesbond	-0.145	0.232	-0.064	0.005	-0.267	0.031	-0.092	-	-0.006	-1.578
MsPacman	-0.173	-1.179	-0.129	-0.06	0.003	-0.019	0.007	0.071	-	-2.774
Pong	-0.127	0.028	-0.12	0.01	0.037	0.042	0.002	-0.174	-0.006	-

# On the Transferability of Deep-Q Networks

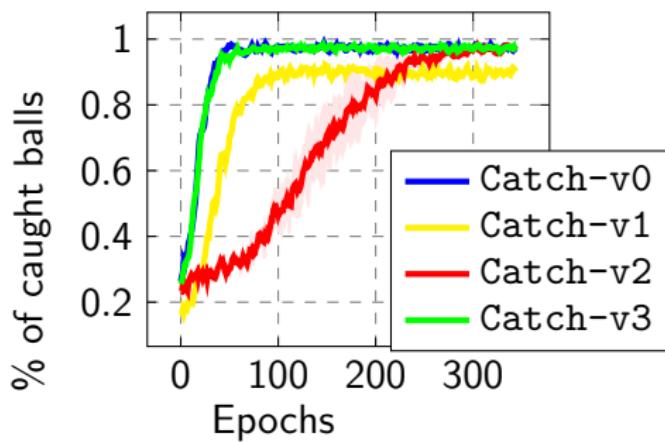
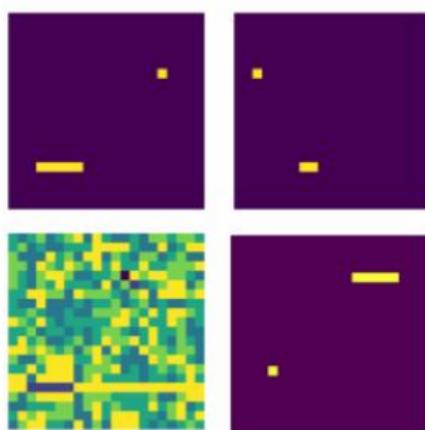
Specifically, on the Atari benchmark we have noticed that:

- Transferring pre-trained Deep-Q Networks mostly results in **negative transfer**, even across similar games Bank Heist → Ms. Pacman
- Transfer across environments is **not symmetric**: Bank Heist ↔ Fishing Derby
- Different algorithms result in **different** transfer learning performance

⇒ Can we **explain** these results any further?

# On the Transferability of Deep-Q Networks

We have designed a set of novel **control tasks** that could help us better understanding the transfer learning potential of Deep-Q Networks



# On the Transferability of Deep-Q Networks

⇒ Even on our newly designed problems, we keep observing **very poor** transfer learning performance

	Catch-v0	Catch-v2	Catch-v3	Catch-v4
Catch-v0	-	-0.026	-0.486	-0.479
Catch-v2	-0.16	-	-0.121	-0.248
Catch-v3	-0.406	-0.313	-	-0.465
Catch-v4	-0.016	-0.24	-0.179	-

# On the Transferability of Deep-Q Networks

We therefore asked ourselves whether Deep-Q Networks are at least able to **self-transfer** ...

- What happens when the source task  $\mathcal{M}_S$  and the target task  $\mathcal{M}_T$  are identical?
- Are there any differences between using an off-the-shelf approach compared to a fine-tuning strategy?
- Let's recall that when transferring a pre-trained model only the last layer responsible for estimating  $Q(s, a)$  is randomly initialized

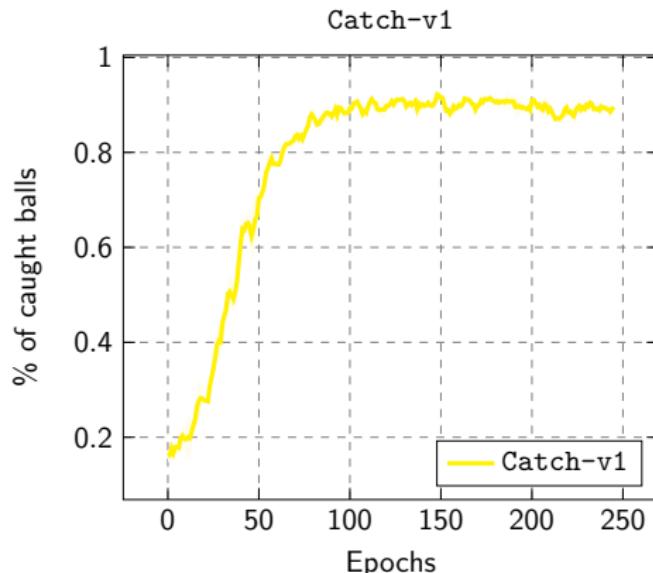
# On the Transferability of Deep-Q Networks

We notice that positive self-transfer can get obtained as long as the networks **do not get fine-tuned** but are used as feature extractors instead

	Catch-v0	Catch-v1	Catch-v2	Catch-v3
Only-Head	0.05	0.141	0.674	0.059
Fine-Tuning	0.017	-0.218	0.393	-0.236

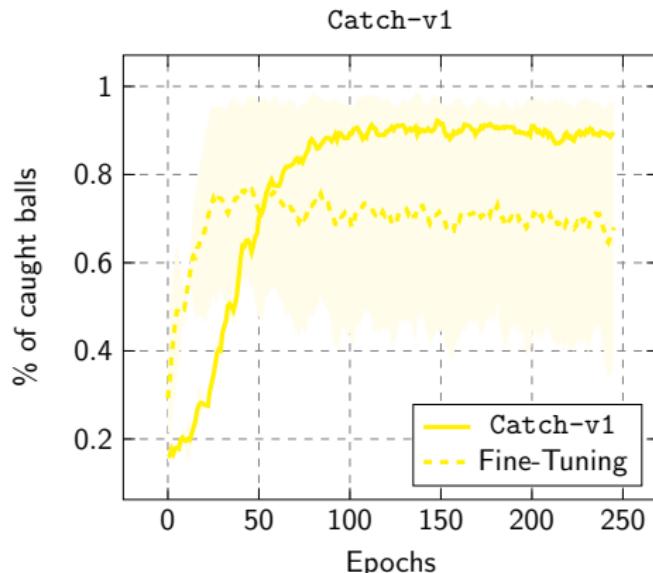
# On the Transferability of Deep-Q Networks

Our **conjecture** is that a Deep-Q Network can only solve a problem if it has carefully found a balance between its feature extractor components and the linear layer that estimates  $Q(s, a)$



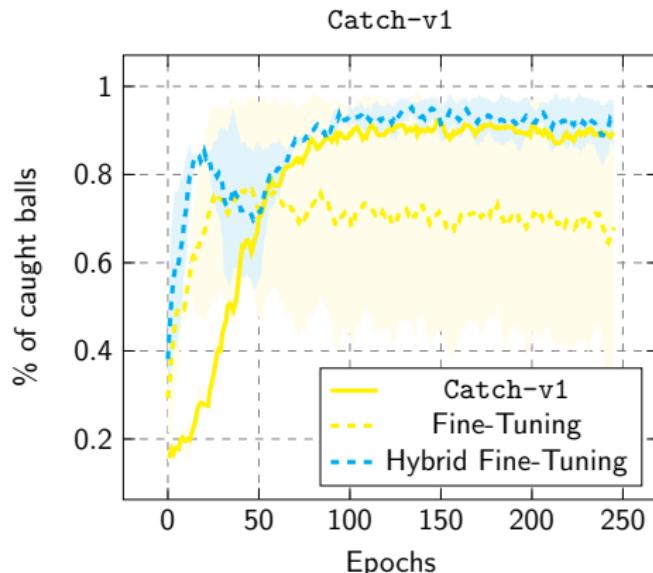
# On the Transferability of Deep-Q Networks

Our **conjecture** is that a Deep-Q Network can only solve a problem if it has carefully found a balance between its feature extractor components and the linear layer that estimates  $Q(s, a)$



# On the Transferability of Deep-Q Networks

Our **conjecture** is that a Deep-Q Network can only solve a problem if it has carefully found a balance between its feature extractor components and the linear layer that estimates  $Q(s, a)$



# On the Transferability of Deep-Q Networks

Takeaway of PART III ...

- We show that fine-tuning pre-trained models in a model-free Deep Reinforcement Learning context is **particularly challenging**
- There is a clear **difference** in terms of performance with the results presented in PART II
- We provide some **initial intuition** about why negative transfer seems to be happening
- This remains an **open problem** that we believe deserves attention

# Conclusion

## Future Work

- A Transfer Learning study of Vision Transformers (ViTs)<sup>8</sup>
- Consider alternative training approaches such as meta-learning and multi-task learning
- Focus on deep model-based Reinforcement Learning

---

<sup>8</sup>Dosovitskiy, Alexey, et al. "An image is worth 16x16 words: Transformers for image recognition at scale." arXiv preprint arXiv:2010.11929 (2020).

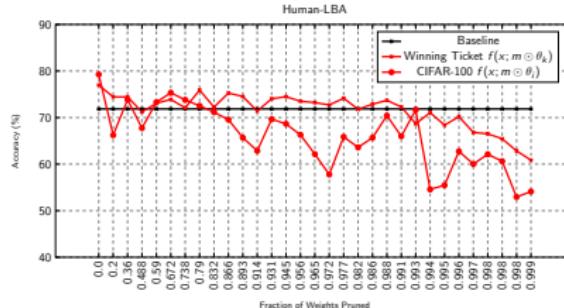
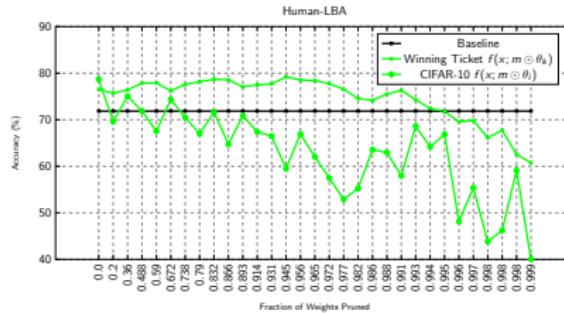
**The End.**

# Final References

- Sabatelli, Matthia, et al. "Deep transfer learning for art classification problems." Proceedings of the European Conference on Computer Vision (ECCV) Workshops. 2018.
- Sabatelli, Matthia, et al. "Advances in Digital Music Iconography: Benchmarking the detection of musical instruments in unrestricted, non-photorealistic images from the artistic domain." Digital Humanities Quarterly 15.1 (2021).
- Sabatelli, Matthia, Mike Kestemont, and Pierre Geurts. "On the Transferability of Winning Tickets in Non-natural Image Datasets." VISIGRAPP (5: VISAPP). 2021.
- Sabatelli, Matthia, et al. "Deep Quality Value (DQV) Learning." Deep Reinforcement Learning Workshop of the 32nd Conference on Neural Information Processing Systems (2018).
- Sabatelli, Matthia, et al. "The deep quality-value family of deep reinforcement learning algorithms." 2020 International Joint Conference on Neural Networks (IJCNN). IEEE, 2020.
- Sabatelli, Matthia, and Pierre Geurts. "On The Transferability of Deep-Q Networks." Deep Reinforcement Learning Workshop of the 35th Conference on Neural Information Processing Systems. 2021.

# Backup

*"Why not transfer fully pre-trained pruned models  $f(x, m \odot \theta_i)$  instead of lottery tickets  $f(x, m \odot \theta_k)$ ?"*



## Backup

*"What is the difference between the DQV-Learning algorithms and the Dueling Architecture?"<sup>9</sup>*

There's two main **differences** among e.g. DQV-Learning and the Dueling Architecture:

1. The Dueling Architecture does not explicitly learn the state-value function  $V^\pi(s)$
2. DQV-Learning requires two independent sets of parameters to perform well  $\phi$  and  $\theta$

---

<sup>9</sup>Wang, Ziyu, et al. "Dueling network architectures for deep reinforcement learning." International conference on machine learning. PMLR, 2016.

## Backup

- 1) In the Dueling Architecture the state-value estimates and the advantage estimates are computed

$$Q(s, a; \theta^{(1)}, \theta^{(2)}, \theta^{(3)}) = V(s; \theta^{(1)}, \theta^{(3)}) + \\ (A(s, a; \theta^{(1)}, \theta^{(2)}) - \max_{a_{t+1} \in \mathcal{A}} A(s, a_{t+1}; \theta^{(1)}, \theta^{(2)})).$$

such that it is easier to train the model with DDQN's TD-target

$$y_t^{DDQN} = r_t + \gamma \max_{a \in \mathcal{A}} Q(s_{t+1}, a; \theta); \theta^-.$$

## Backup

- 1) But the true state-value function  $V^\pi(s)$  is never explicitly learned, which **is the case** for DQV-Learning and DQV-Max Learning as there's a dedicated objective function for learning  $\approx V^\pi(s; \phi)$  where we either have

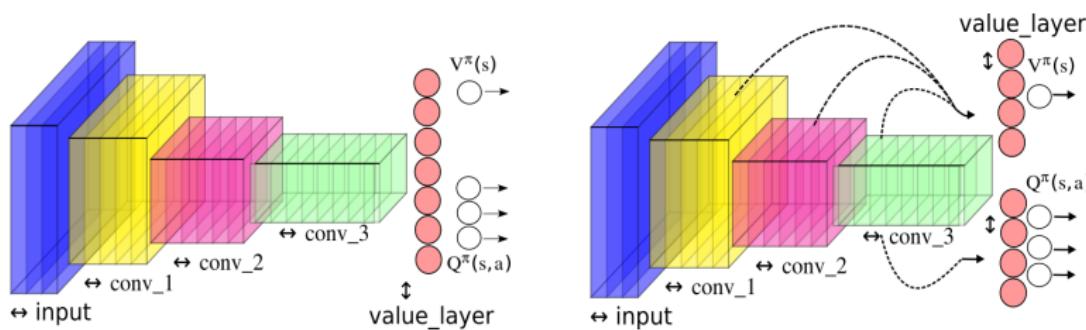
$$L(\phi) = \mathbb{E}_{(s_t, a_t, r_t, s_{t+1}) \sim U(D)} \left[ (r_t + \gamma V(s_{t+1}; \phi^-) - V(s_t; \phi))^2 \right],$$

or

$$L(\phi) = \mathbb{E}_{(s_t, a_t, r_t, s_{t+1}) \sim U(D)} \left[ (r_t + \gamma \max_{a \in \mathcal{A}} Q(s_{t+1}, a; \theta^-) - V(s_t; \phi))^2 \right],$$

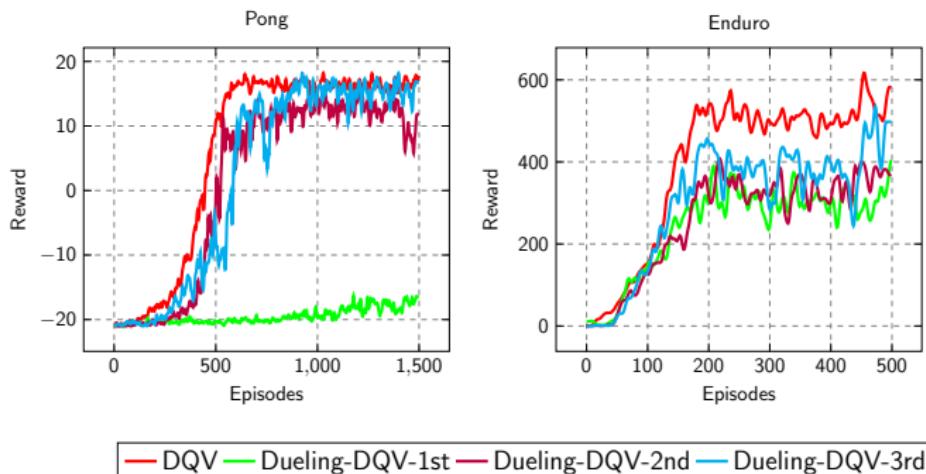
# Backup

2) We have also tried to train a DQV-Learning agent by following the Dueling neural architecture



## Backup

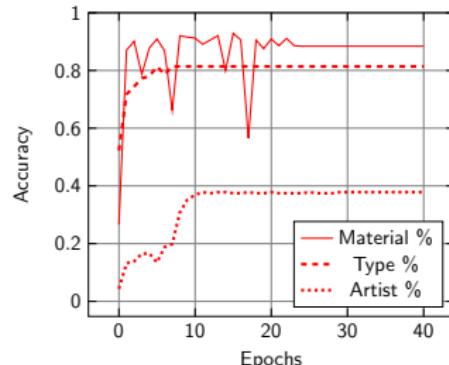
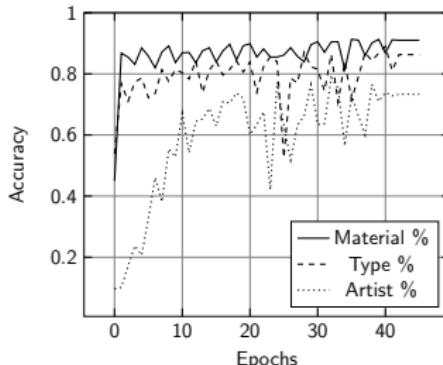
⇒ But the results were **never on par** with regular DQV-Learning



# Backup

*"Is there an alternative way of improving the performance of neural networks other than Transfer Learning?"*

We have experimented with [Multi-Task Learning](#), which for supervised learning problems showed promising results



# Backup

*"Does Transfer Learning never work when it comes to Deep Reinforcement Learning?"*

We believe that this is mostly the case for model-free Deep Reinforcement Learning only:

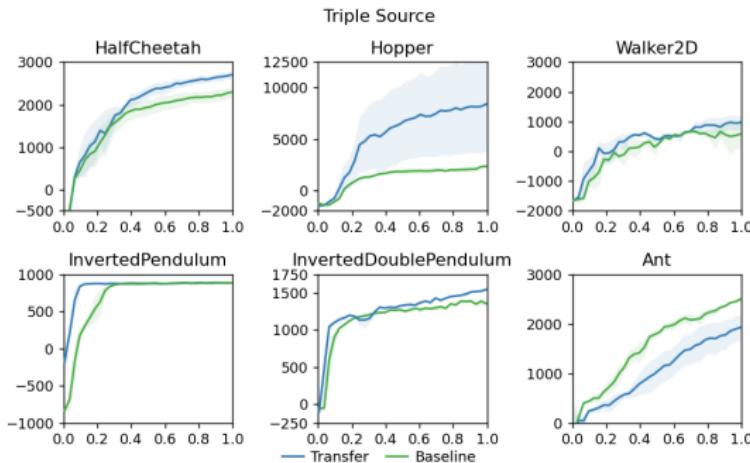
- As CNNs need to act as feature extractors as well as function approximators
- In fact, in model-based Reinforcement Learning transferring neural architectures works really well <sup>10</sup>

---

<sup>10</sup>Sasso, Remo, Matthias Sabatelli, and Marco A. Wiering. "Fractional transfer learning for deep model-based reinforcement learning." arXiv preprint arXiv:2108.06526 (2021).

# Backup

*"Does Transfer Learning never work when it comes to Deep Reinforcement Learning?"*



**Figure:** Results from Sasso, Remo, Matthia Sabatelli, and Marco A. Wiering. "Fractional transfer learning for deep model-based reinforcement learning." arXiv preprint arXiv:2108.06526 (2021)