Learning from Images

Transfer Learning and Domain Adaptation

Master DataScience

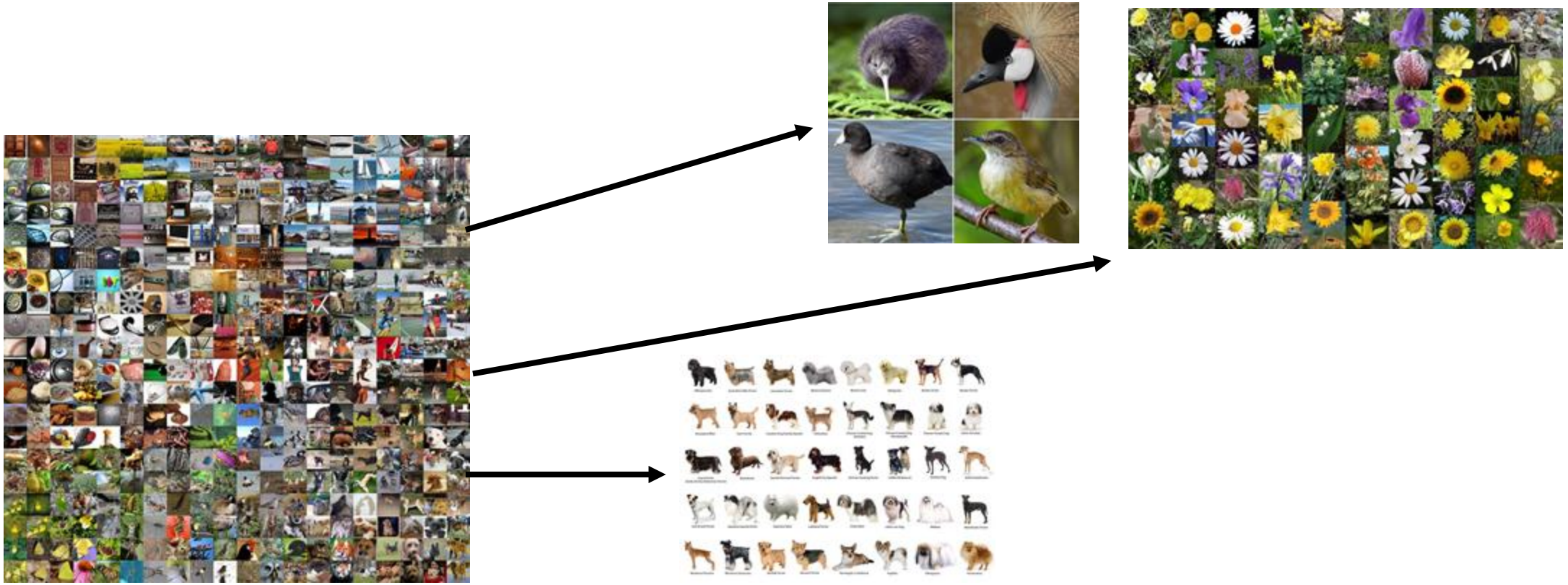
Prof. Dr. Kristian Hildebrand
khildebrand@beuth-hochschule.de

Berliner Hochschule für Technik
Studiere Zukunft

Transfer Learning…

is a machine learning technique where a model **trained on one task is reused or adapted for a related task**. It leverages knowledge gained from one domain to improve learning in another domain.

Use a network pre-trained on **ImageNet21K** and **fine-tune** it to perform multiple specialised tasks.



ImageNet image: https://cs.stanford.edu/people/karpathy/cnnembed/
Bird image: https://ieee-dataport.org/sites/default/files/Design%20sans%20titre%20(1).png
Dogs image: https://user-images.githubusercontent.com/26468713/34918645-fbfe6bc2-f97b-11e7-9f89-548b508db905.jpg
Flowers Image: https://community.wolfram.com//c/portal/getImageAttachment?filename=Imagecollageof17flowersdataset.jpg&userId=1556395

Transfer Learning - advantages

Efficiency: Reduces the need for large amounts of data to train new models by reusing learned knowledge.

Faster Adaptation: Accelerates learning in new domains or tasks as the model begins with a solid knowledge foundation.

Improved Performance: Often leads to better model accuracy and generalization, especially in scenarios with limited available training data.

Cost and Resource Savings: Lowers costs and time requirements for model development since there's no need to train from scratch, especially in computationally intensive tasks.

Transfer Learning - procedure

1. **Pre-trained Model Selection**: Choosing a model previously trained on a large dataset for a specific task, often in a related domain.
2. **Feature Extraction or Fine-tuning**: Adapting the pre-trained model by either:
   a. Using its learned features directly for the new task (feature extraction), or
   b. Fine-tuning the model's parameters on a smaller dataset from the target domain to improve its performance for the new task.
3. **Applying to Target Task**: Using the adapted model on the target domain to perform the desired task, leveraging the transferred knowledge to achieve better performance with potentially less data than training from scratch.

Transfer Learning - pre-trained model selection

- **Task compatibility:** Ensure the pre-trained model addresses a task related to your target task
- **Availability of pre-trained weights:** Check that the dataset used for pre-training is somehow related to your target data
  - generally, bigger pre-training dataset results in better target performance
- **Model Architecture:** Select a model architecture suitable for the size of your dataset and computational resources available

Transfer Learning - training

Standard approach: *Fine-tune* the pre-trained model by training the model on the target dataset

1. **Select hyperparameters and setup**: data augmentations, freezing layers, learning rate, scheduler, warm-up, … *(or define a hyperparameter search)*
2. **Adjust network:** Replace the **classification head** (last layer) of the network by a new classification head matching the number classes of your target dataset
3. **Train network:** Perform training on target data

- in this example the pre-training dataset has **5 classes**
- consequently, our network has **5 nodes in its output layer**

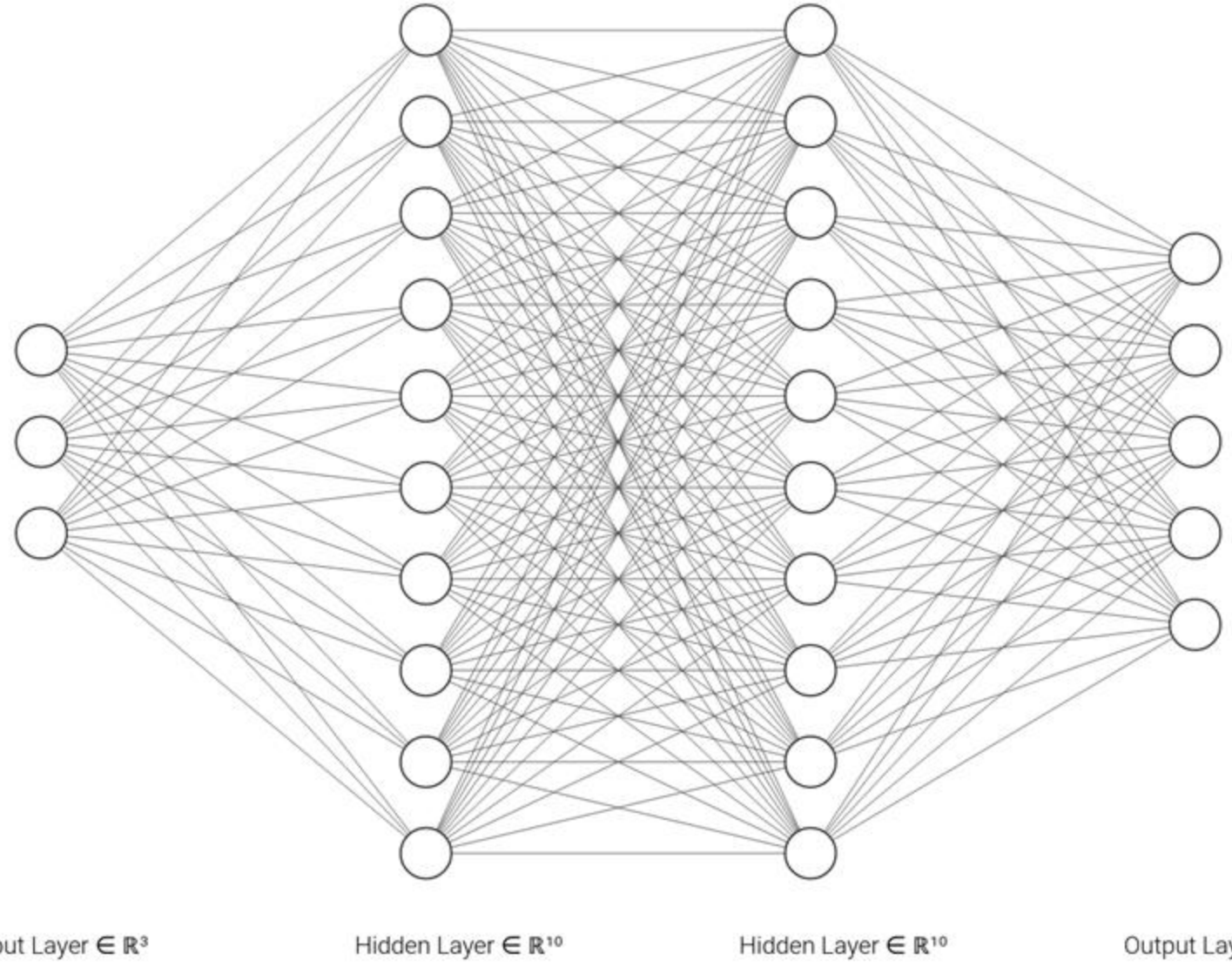


Image created with: http://alexlenail.me/NN-SVG/index.html

- our target dataset has only **2 classes**
- so we **remove** the output layer and replace it by an new output layer with **2 nodes**

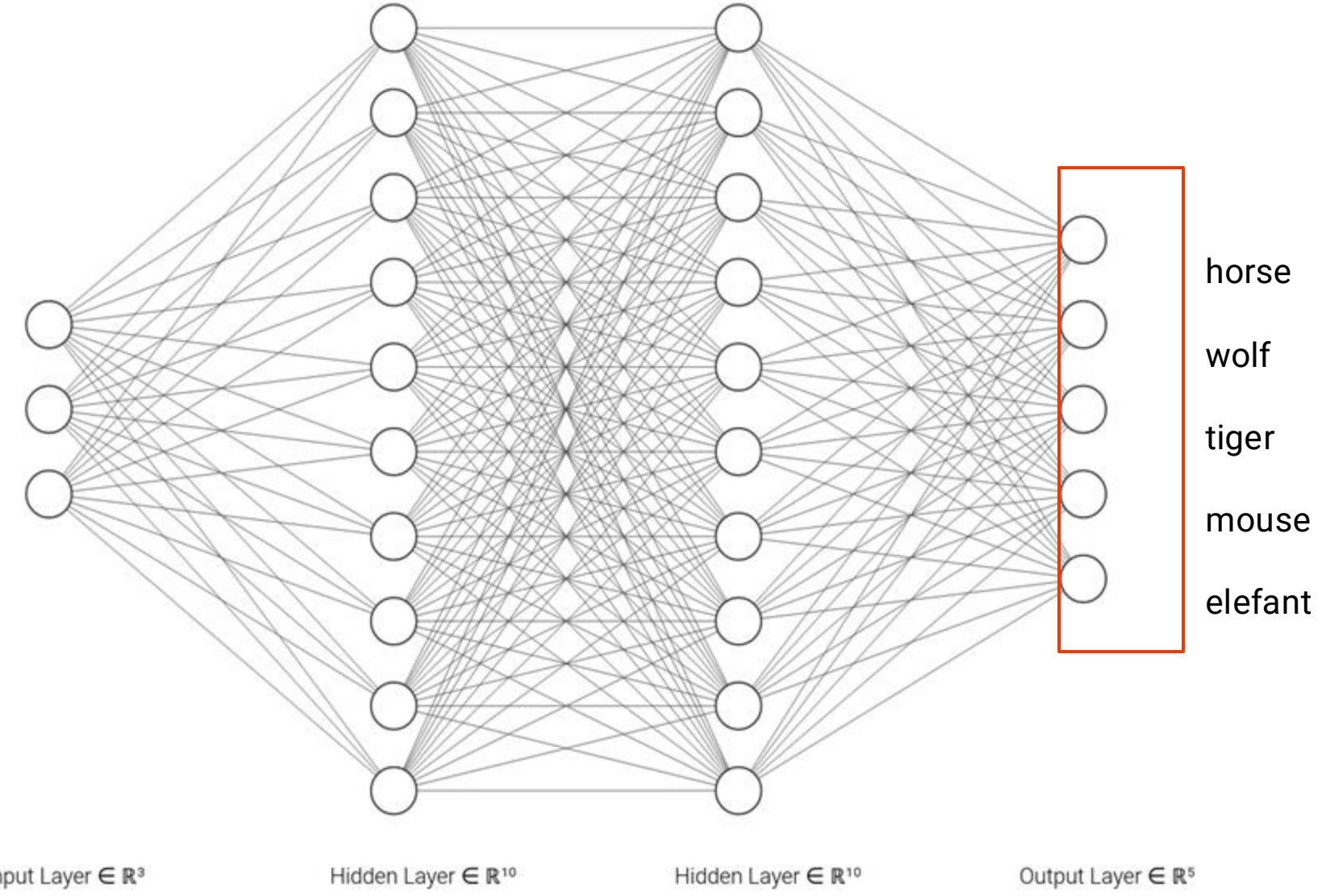


Image created with: http://alexlenail.me/NN-SVG/index.html

- we can now fine-tune the pre-trained model to separate cats and dogs

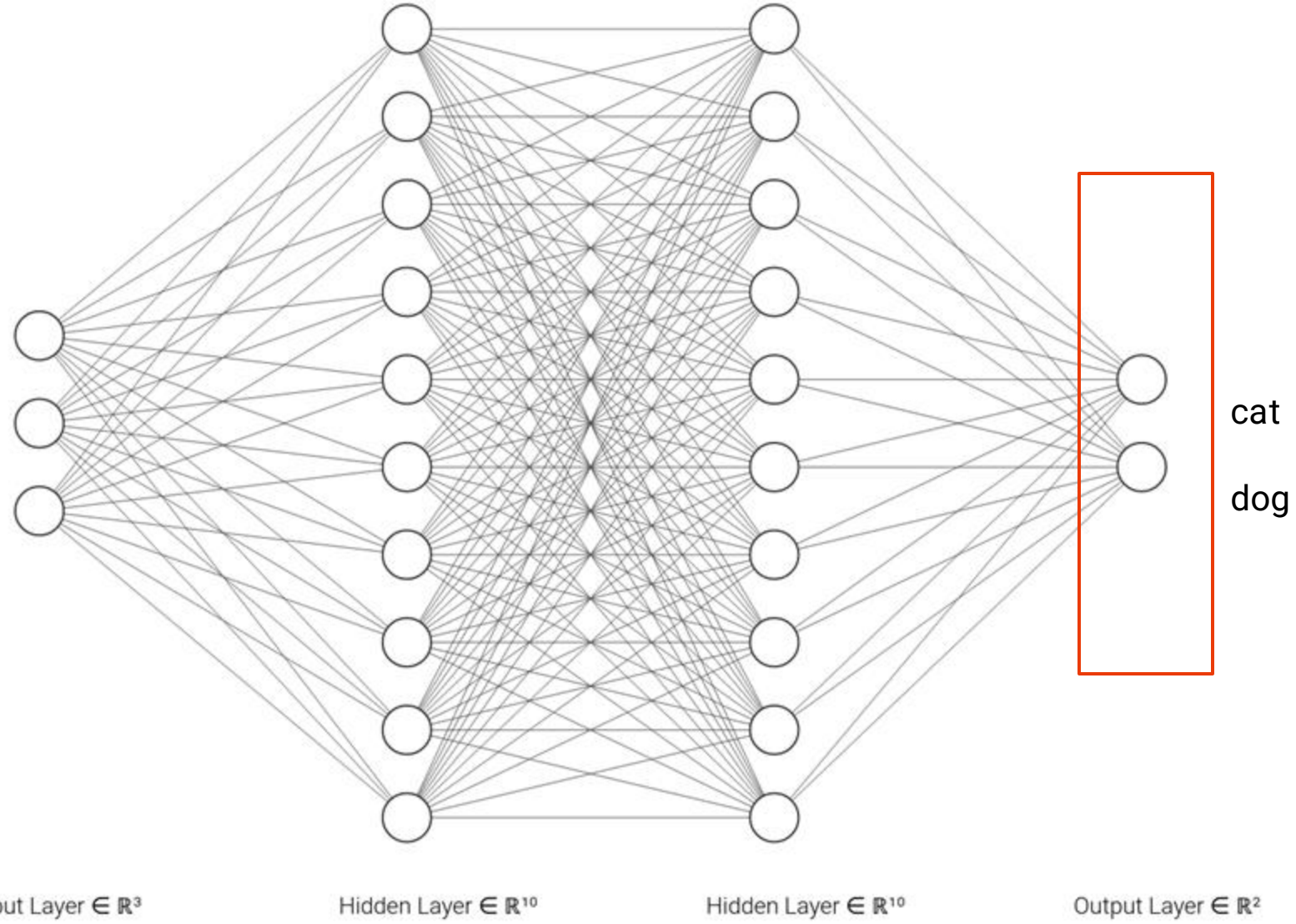
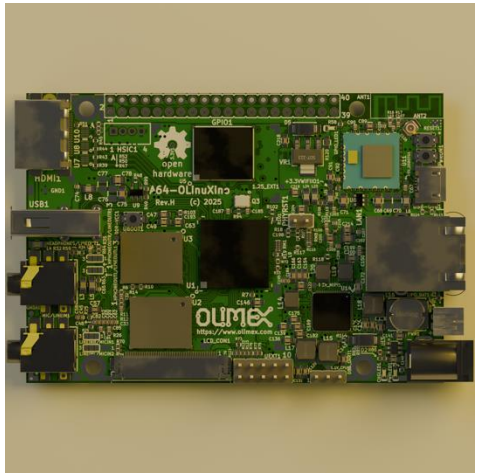
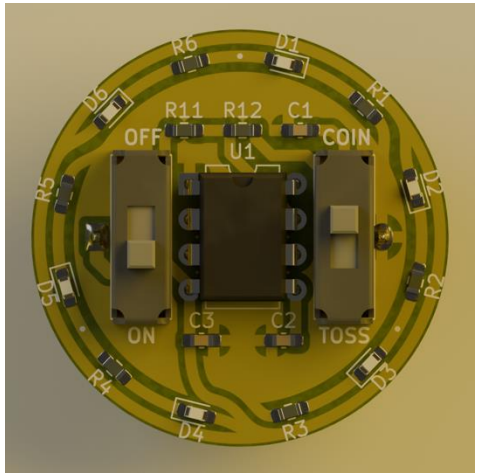


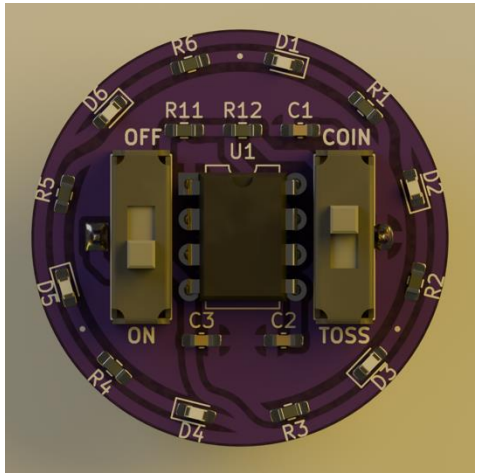
Image created with: http://alexlenail.me/NN-SVG/index.html

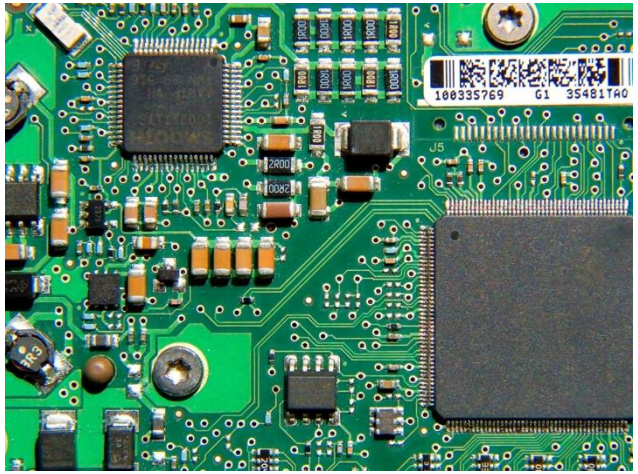
```
import torch.nn as nn


# Note: We assume, that the classification layers name is “class_layer”
num_ftrs = pretrained_model.class_layer.in_features # get number of inputs
num_classes = 2 # set new number of classes
pretrained_model.class_layer = nn.Linear(num_ftrs, num_classes) # set new layer
```

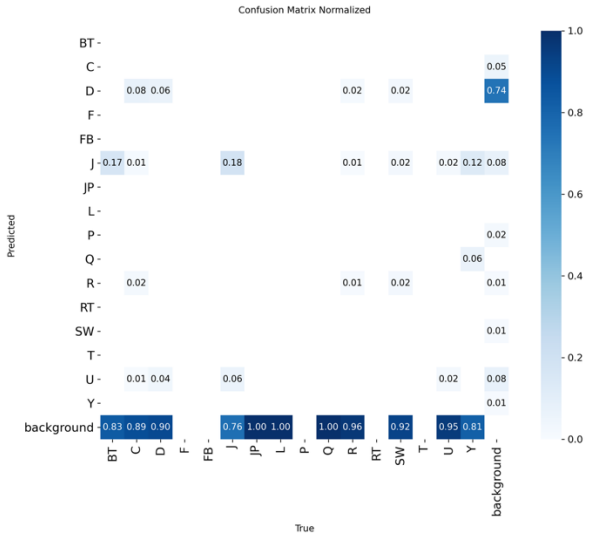
Domain Gap/Shift











Domain Transfer…

adjusts a model trained on one dataset to perform better on a different dataset with similar but not identical characteristics, addressing differences in data distributions between the two domains

domainnet image: https://production-media.paperswithcode.com/datasets/DomainNet-0000002698-42f300d8_nfcNml7.jpg

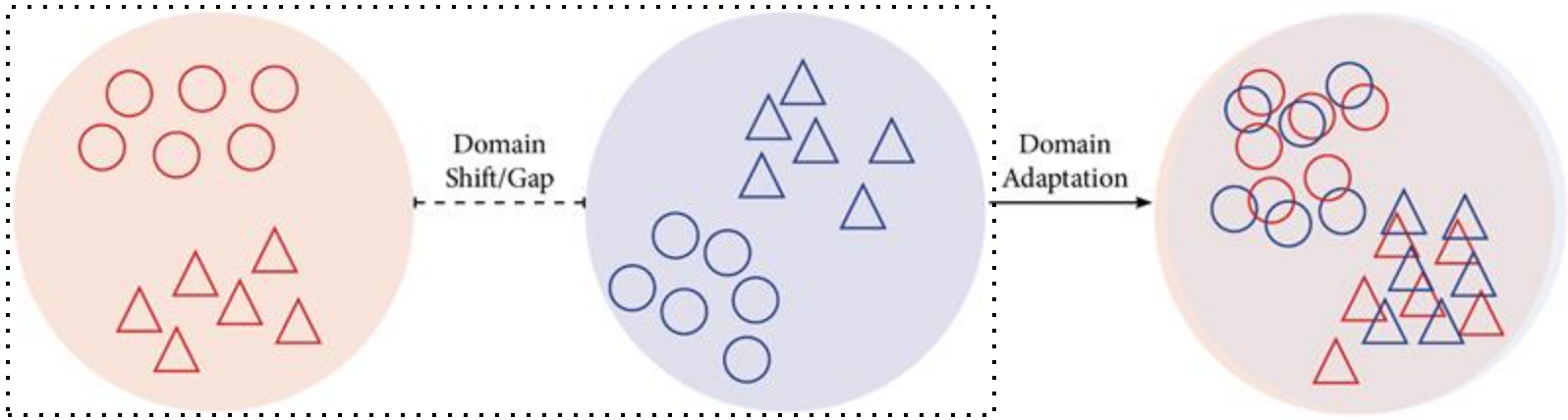
Domain Transfer

Domain Adaptation:
Transferring a Network from a source domain to a target domain while having **access to labeled source data and unlabeled target data**.

Domain Generalisation:
Transferring a Network from a source domain to a target domain while having **access to labeled source data but no target data**.

Domain Adaptation

- the goal is to train the network to produce **domain-invariant features**
  - classifier can’t distinguish between source and target domain (eg. real photos and sketches)



Domain Adaptation - metrics

- the **Domain Gap** or **Domain Shift** describes the distance between source and target data distributions in the N-dimensional feature space
- some metrics to quantify the *Domain Gap* can be:
  - **distance metrics:** Euclidean distance, Wasserstein distance, ..
  - **divergence measures:** Kullback-Leibler (KL) divergence, Maximum Mean Discrepancy (MMD), Kernel Inception Distance (KID)
- But the main focus lies on your target task metric (Accuracy for classification tasks for example)

Domain Adaptation - methods

- the two most prominent groups of Domain Adaptation methods are **adversarial based** and **discrepancy minimization based** methods
- **adversarial methods** utilize a **domain discriminator** to distinguish between source and target domains while simultaneously training the main model to *confuse* the discriminator
  - generate features so that the discriminator network can no longer distinguish between the different domains
- **discrepancy minimization methods** utilize distance and discrepancy metrics like MMD and penalize the model for not producing domain-invariant features
  - generate features that lie closer together in the feature space

Domain Adaptation - datasets

- **VisDa-2017:**
  Arbitrary objects, 10 classes (real, 3d renders)
- **DomainNet:**
  Arbitrary objects, 345 classes (real, sketch, painting, clipart, quickdraw, infograph)
- **Office-Home:**
  Everyday objects, 65 classes (real, art, clipart, product)

Domain Adaptation - resources

- **Huggingface**
  - easy access to pre-trained models and datasets
  - https://huggingface.co/
- **tllib**
  - **Transfer learning algorithm library**
  - **https://github.com/thuml/Transfer-Learning-Library**
- **“Everything you need for transfer learning”**
  - **Collection of useful and informative material for transfer learning**
  - **https://github.com/jindongwang/transferlearning**

SynthNet - Intro

Anwendung von **synthetischen Daten** zur Entwicklung von KI

3D Model

KI-unterstütztes Rendering

Training der Neuronalen Netze & Indexierung

Visuelle Suche

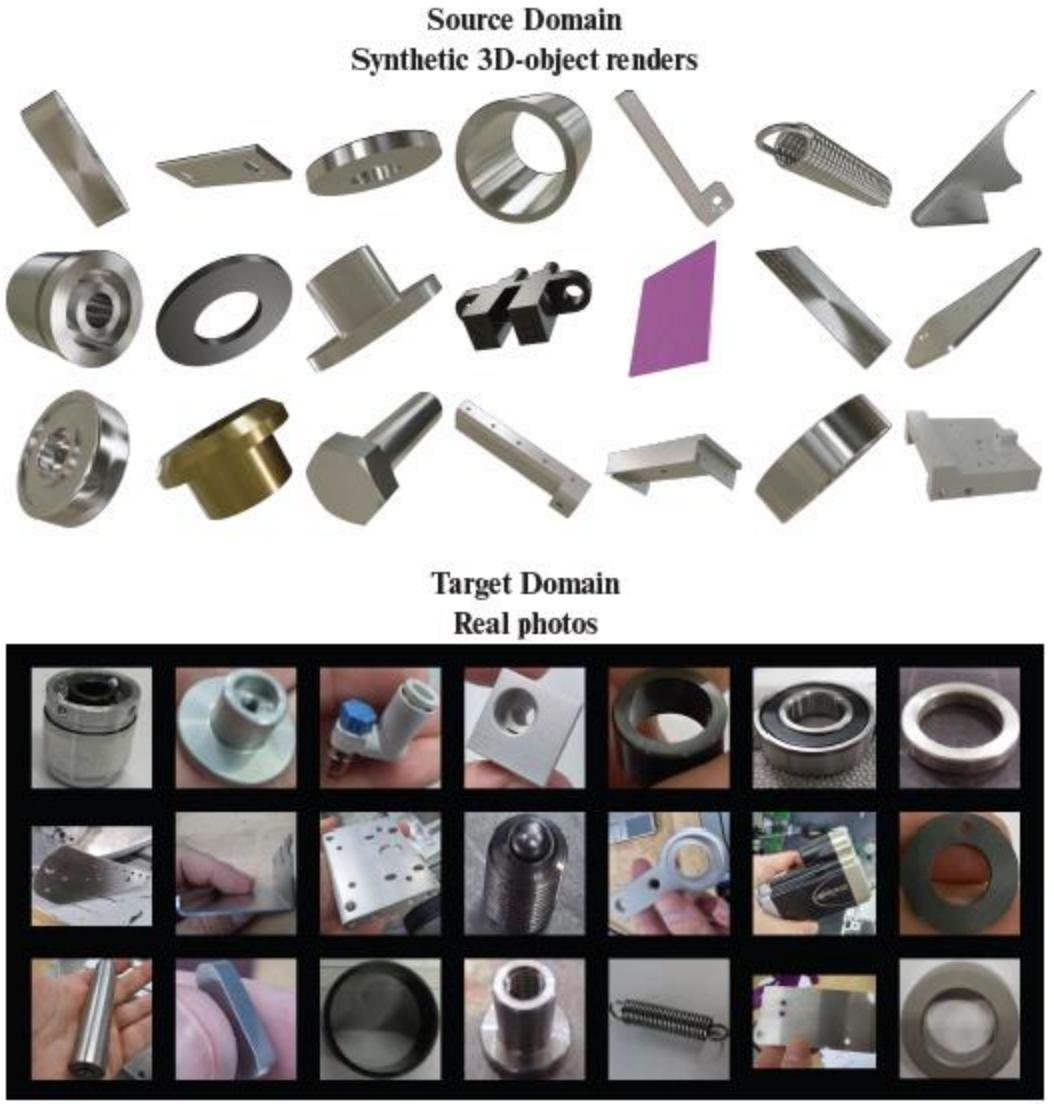
SynthNet - motivation

- Machine-specific specialists are often required to **quickly identify components**
- Reduce personal support effort
- Train a classifier using photos?
  - High cost of creating a enough labeled images for training
- Companies own the computer-aided design (CAD) data of the parts

⇒ Generate as much Synthetic render data from CAD as needed and train a classifier using Domain Adaptation methods

SynthNet - topex-printer dataset

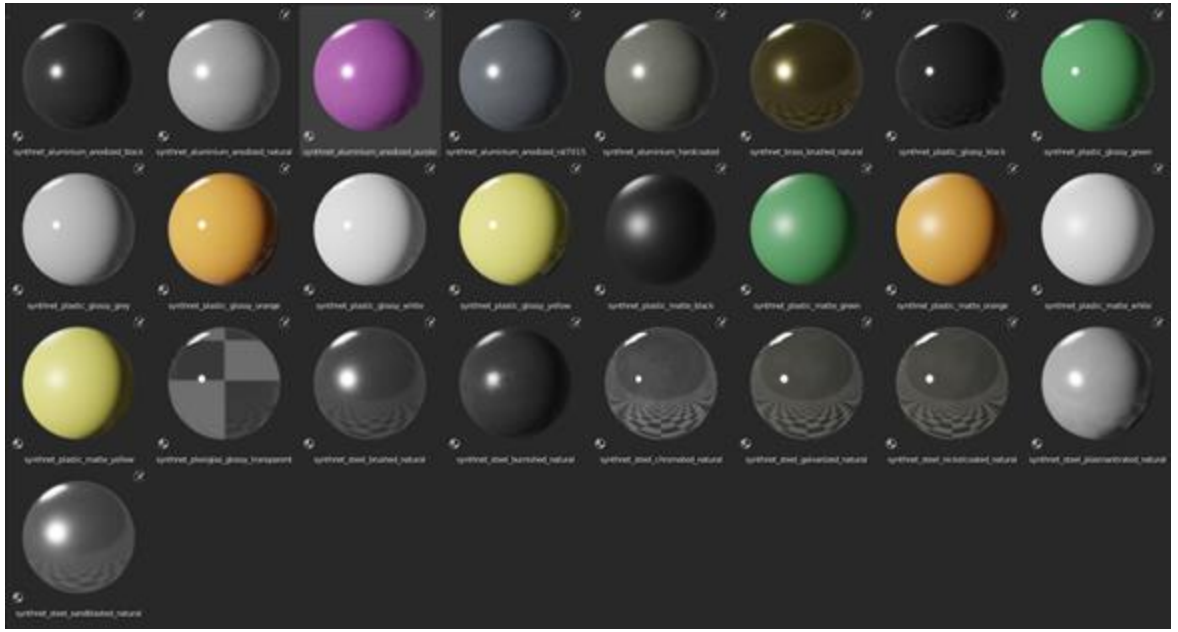
- Two-Domain dataset
  - CAD rendered images
  - Real Photos
- Machine parts of a labeling machine from german manufacturing company
- Closely related classes mimic real-world complexities
  - bearings
  - plates
  - cylinders
  - …



SynthNet - topex-printer dataset

CAD Renders - Source Domain

- Blender-Python render pipeline
- Assign Blender-Materials based on metadata information about each part
  - Material
  - Surface
  - Color
- Assign Environment map
- Render *n* (n=32) images for each part
  - 512x512
  - **3.264 images** in total



SynthNet - topex-printer dataset

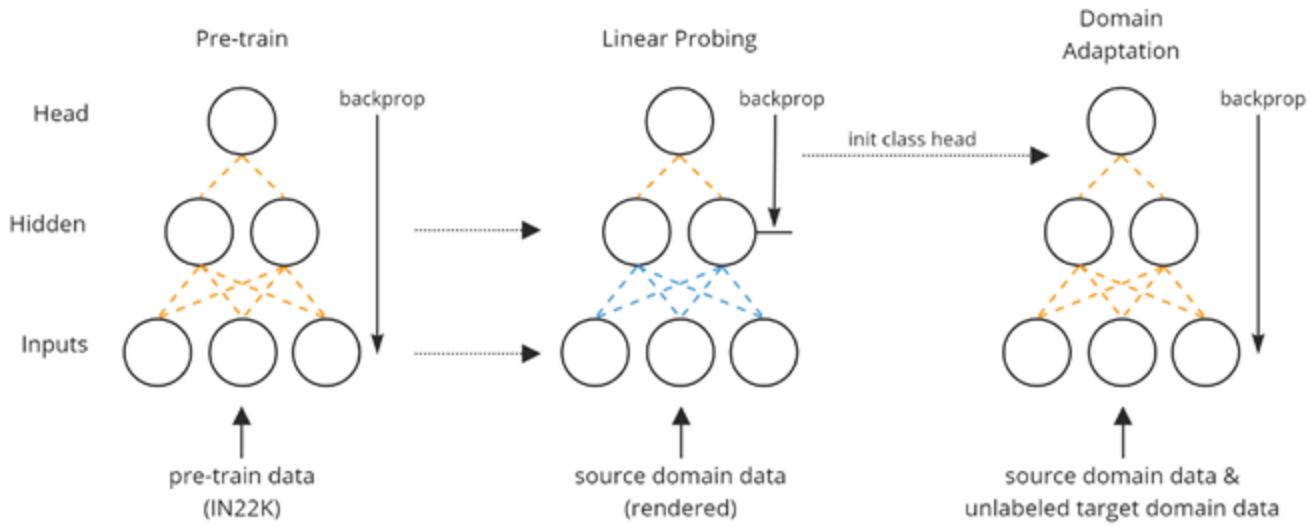
Real Images - Target Domain

- **6.146 real photos** taken by constructors and engineers during the assembly process
- **6 to 137 images** for each part
- Cut out single parts and export rectangular images
  - Expand smaller dimension
  - If no expansion possible, fill with black

SynthNet - Pipeline

- Start from IN22K pre-trained model checkpoint

1. Linear Probing
   a. Train **classification layer** on **source domain data only** (Freeze other layers)
2. Domain Adaptation
   a. init classification head from stage 1
   b. Perform Unsupervised Domain Adaptation using **labeled source domain** data and **unlabeled target domain** data on whole network



SynthNet - Pipeline

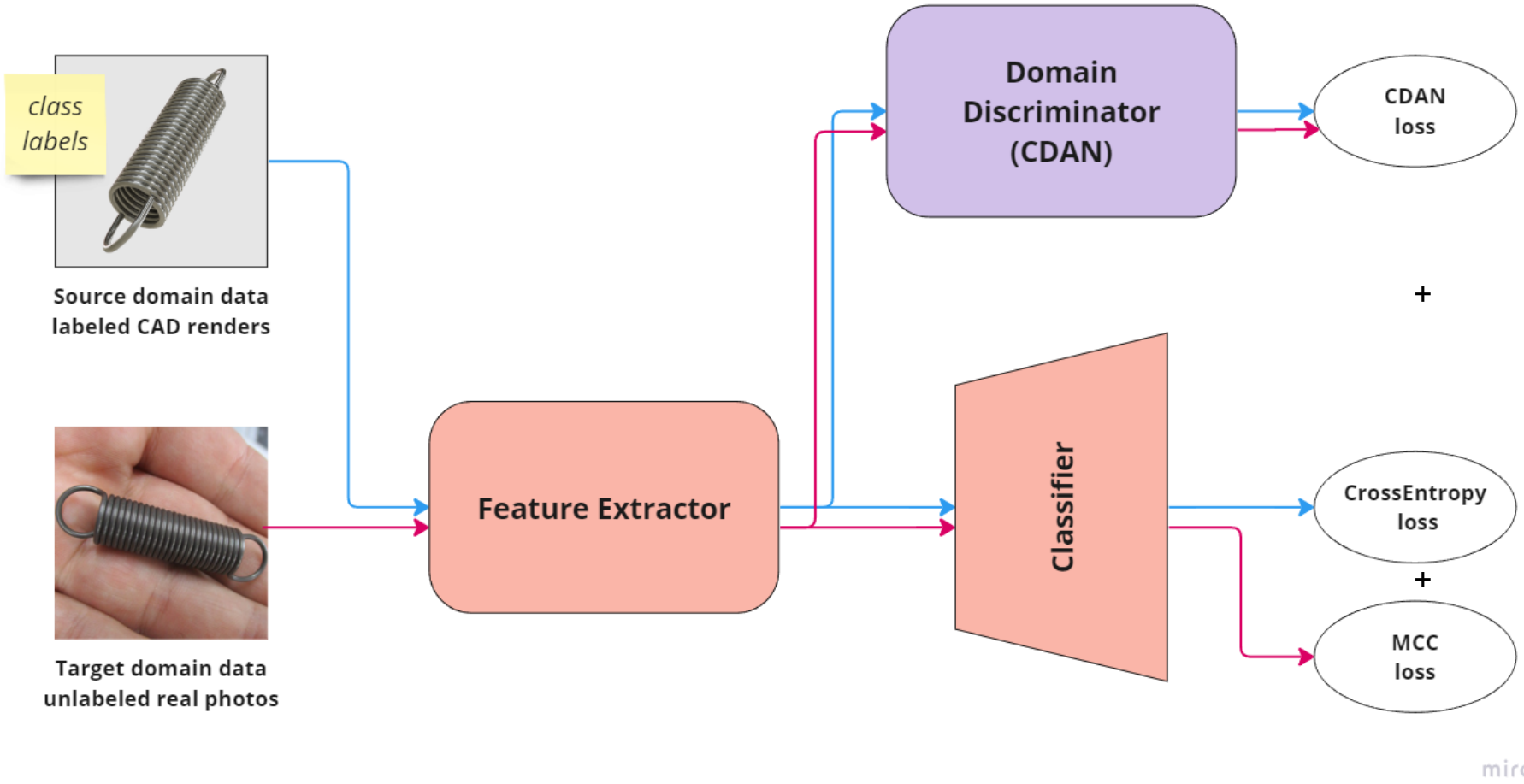
Stage 1 (src only)

a. Train **classification layer** on **source domain data only** (Freeze other layers)
b. Full fine-tuning across all layers for comparison to linear probing and hyperparameter search for stage 2

Stage 2 (src + unlabeled target)

- Use best parameters from stage 1b
- Init network with best checkpoint from stage 1a
- Train full network using unsupervised domain adaptation techniques (CDAN, MCC, CDAN-MCC)

SynthNet - Pipeline



Synthnet - Evaluation
VisDA-2017

- Our approach slightly **outperforms SoTA** (MIC) by 0.67% using **SwinV2** architecture
- Our best ViT run performs comparable to SDAT
- improvement on *hard* classes Bus and Truck
- Decrease on Person and Motorcycle

Table 1. Image classification top-1 accuracy in % on VisDA-2017 target domain (real images) across all classes compared to literature. We report our best source-domain-only and UDA runs for the ViT and SwinV2 architecture.

Method		Pl	Bcl	Bus	Car	Hrs	Knf	Mcy	Per	Plt	Skb	Trn	Tck	Mean
CDAN [19]	ResNet	85.2	66.9	83.0	50.8	84.2	74.9	88.1	74.5	83.4	76.0	81.9	38.0	73.9
MCC [10]		88.1	80.3	80.5	71.5	90.1	93.2	85.0	71.6	89.4	73.8	85.0	36.9	78.8
SDAT [23]		95.8	85.5	76.9	69.0	93.5	97.4	88.5	78.2	93.1	91.6	86.3	55.3	84.3
MIC [8]		96.7	88.5	84.2	74.3	96.0	96.3	90.2	81.2	94.3	95.4	88.9	56.6	86.9
TVT [29]	ViT	92.9	85.6	77.5	60.5	93.6	98.2	89.3	76.4	93.6	92.0	91.7	55.7	83.9
CDTRANS [28]		97.1	90.5	82.4	77.5	96.6	96.1	93.6	88.6	97.9	86.9	90.3	62.8	88.4
SDAT [23]		98.4	90.9	85.4	82.1	98.5	97.6	96.3	86.1	96.2	96.7	92.9	56.8	89.8
MIC [8]		**99.0**	93.3	86.5	87.6	**98.9**	**99.0**	**97.2**	**89.8**	98.9	**98.9**	96.5	68.0	92.8
Ours w/o UDA		96.48	71.82	90.14	**99.20**	94.66	77.71	87.28	44.45	95.12	83.64	94.05	40.76	80.54
Ours		94.82	93.49	92.80	95.89	90.95	88.51	77.46	75.42	96.27	97.32	94.74	88.03	89.38
Ours w/o UDA	Swin	97.09	80.48	85.35	98.12	92.39	83.54	94.85	19.89	89.13	78.89	**97.03**	55.18	80.12
Ours		97.96	**95.15**	**95.81**	98.64	98.34	95.68	80.12	83.87	**99.39**	94.68	96.61	**93.85**	**93.47**

Synthnet - Evaluation
VisDA-2017

- **Person** class mixed up with several classes
  ⇒ Other objects in *Person* image samples?
- **Motorcycle** mixed up with **bicycle & Skateboard**
- **Truck** class mixed up with **Train** or **Bus** only 1%-2%
- **Bus** class mixed up with **Train** or **Truck** only 1%-2%

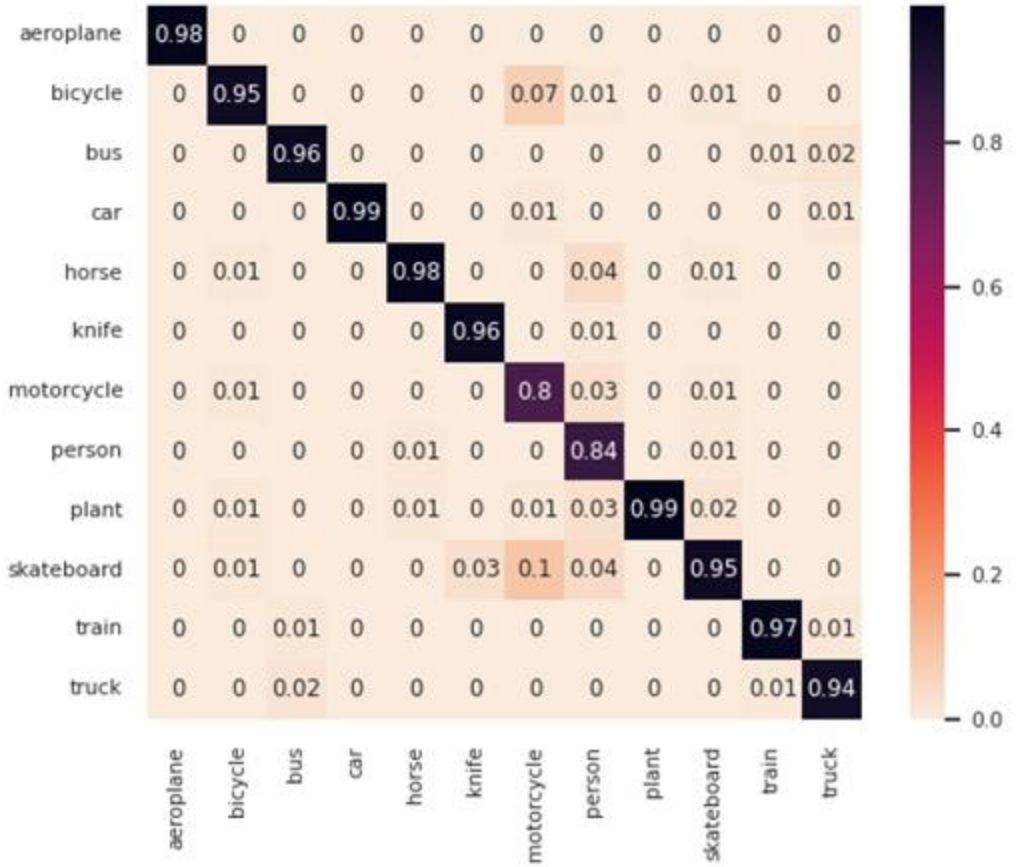


Fig. 4. Confusion matrix for our best-performing model on VisDA-2017: SwinV2-CH-CDAN-MCC

Synthnet - Evaluation

Topex-Printer

- **SwinV2 > ViT**
- **Class head tuning improves performance** compared to UDA fine-tuning only
- **CDAN & MCC combined** outperforms CDAN or MCC isolated

Table 6. Acc@1 in % on target domain (real images) for all UDA experiments on the Topex-Printer dataset. Note that *init checkpoint* describes the model checkpoint used for the UDA experiments. *CH* refers to the best-performing CH training scheme from our source-domain-only training experiments respecting the used model architecture and IN22K refers to the respective Huggingface model checkpoints described in section A.2.

Model	DA method	init checkpoint	Acc@1
ViT-b	CDAN	IN22K	43.31
ViT-b	CDAN	CH	47.51
ViT-b	MCC	IN22K	32.95
ViT-b	MCC	CH	61.36
ViT-b	CDAN-MCC	IN22K	43.33
ViT-b	CDAN-MCC	CH	61.08
SwinV2	CDAN	IN22K	65.51
SwinV2	CDAN	CH	61.94
SwinV2	MCC	IN22K	72.86
SwinV2	MCC	CH	71.14
SwinV2	CDAN-MCC	IN22K	73.74
SwinV2	**CDAN-MCC**	**CH**	**74.86**