Deep Learning - MAI

Transfer Learning

THEORY

Dario Garcia Gasulla
dario.garcia@bsc.es

“*Don’t be a hero*” - Andrej Karpathy

The Transfer Learning philosophy

Learning from scratch

- Trying to learn from scratch is difficult and arduous
  - Learn the basics before learning complex stuff
- Easier to learn if you already know something
  - Some basic stuff is common to many tasks
  - e.g., learning to "see"

Why Transfer Learning

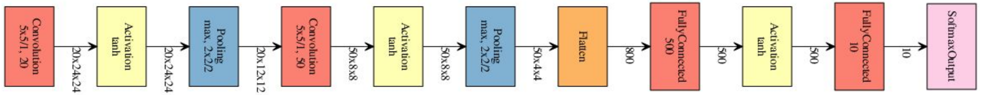
- You can learn **faster**
  - If I know that much, I'm that much closer to my goal
- You can learn **better**
  - Limited amount of things you can learn from data before finding spurious patterns.
    - What would you rather learn from your data?

Putting things in perspective

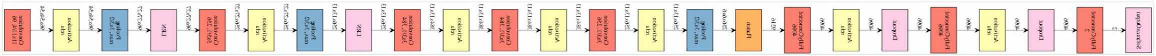
The ImageNet ¿success?

Growing up

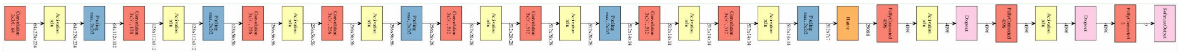
- **1998** LeNet-5



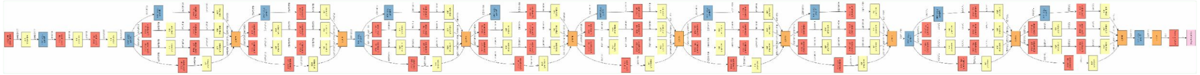
- **2012** AlexNet



- **2014** VGG19



- **2014** GoogLeNet



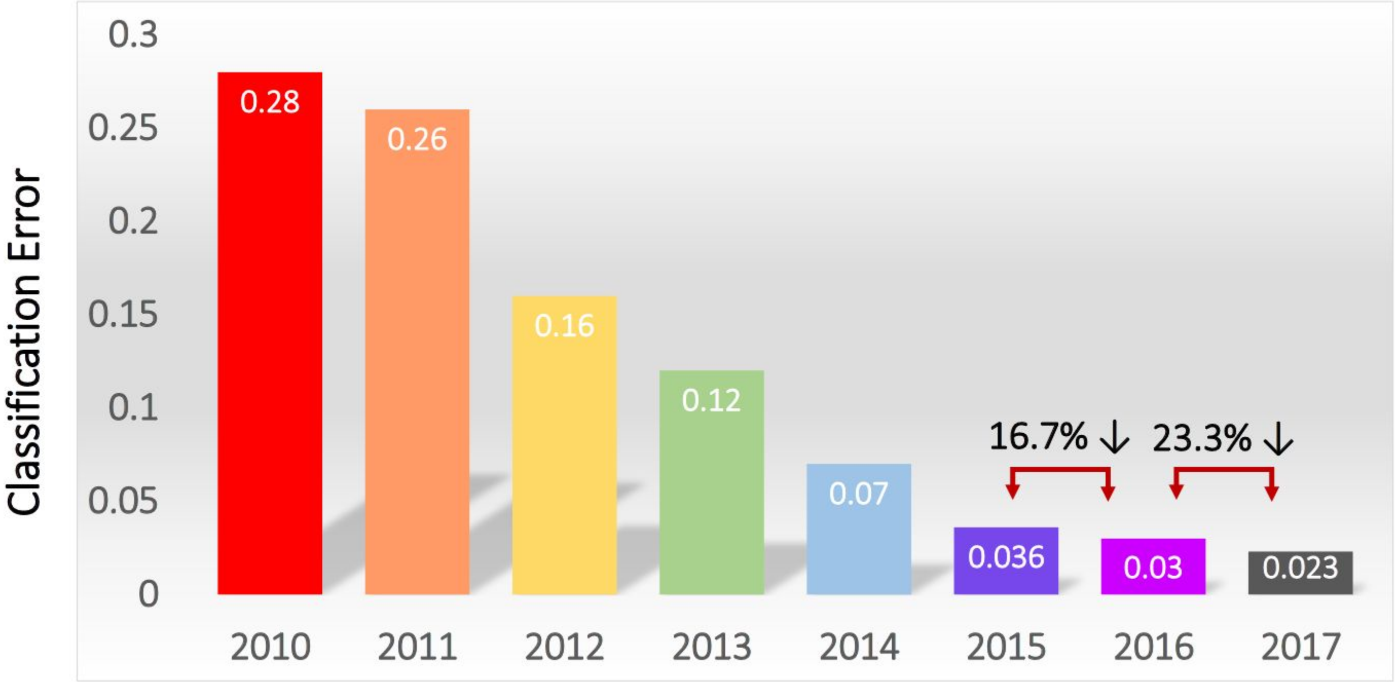
- **2015** Inception-V3

- **2015** ResNet-56

HIGH PERFORMANCE ARTIFICIAL INTELLIGENCE

What we get

- We solved ImageNet



What we pay

- Data labeling, transfer & storage
  - e.g., 1,000 images per class
- Training cost
  - Money (hardware, energy, salaries)
  - Environmental cost (CO_2 emissions)
  - Human effort
    - Highly skilled professionals
    - Architecture design
    - Hyper-parameter fine tuning

HPAI HIGH PERFORMANCE ARTIFICIAL INTELLIGENCE

The ImageNet way is no way

- We **cannot** do that for every single problem out there
    - The cost is too high. But more importantly…

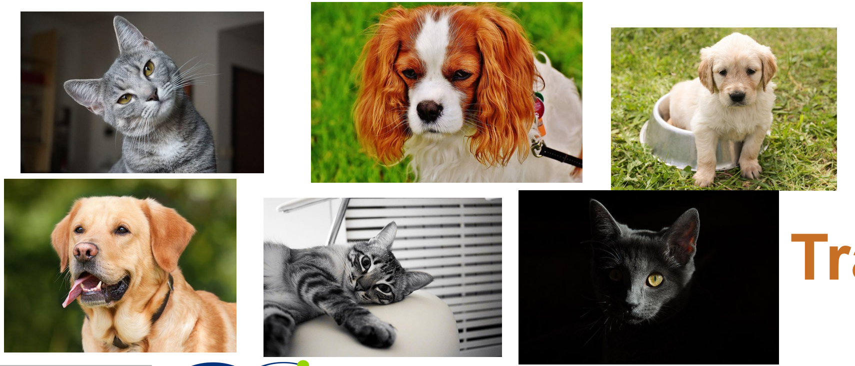
The ImageNet way is no way

- We **cannot** do that for every single problem out there
  - The cost is too high. But more importantly...
- We **do not want to** do that for every single problem out there
  - TL to the rescue
- Transfer learning reduces the requirements on...
  - Data (implicit reuse of data)
  - Cost (faster convergence)
  - Effort (initial design & parametrization)

Transfer Learning variables

- How much error can we expect?
- What does it depend on?

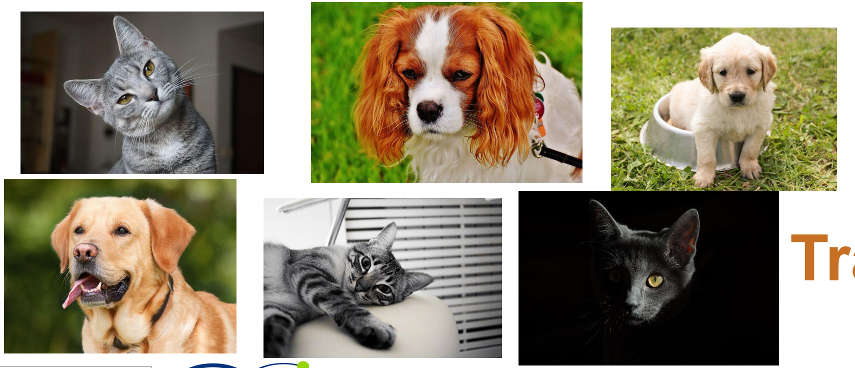
Test set



Train set

Transfer Learning variables

- How much error can we expect?
- What does it depend on?
  - Domain (must)
  - Task (should)
    - Intersect & size



Train set

Test set

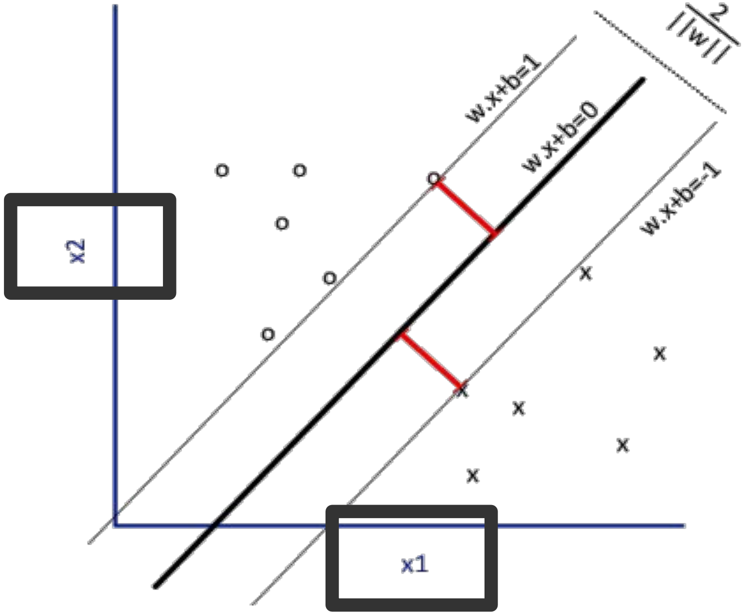
Representation Learning & Classifiers

Learning to describe

A typical classifier

- **Support Vector Machine** (SVM) is just a **classifier** (a very good one).

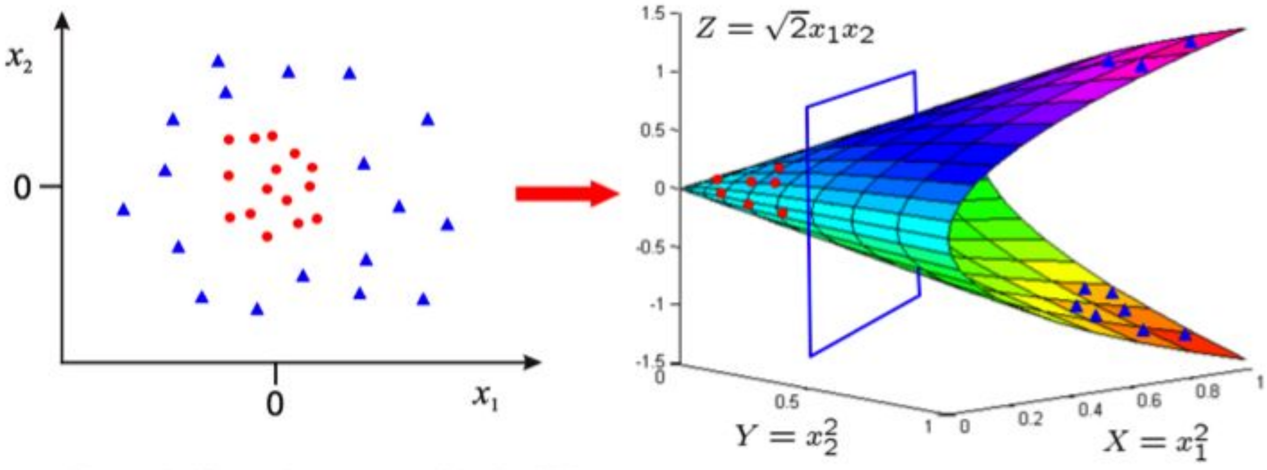
- SVM find the best boundary separating the data instances into different classes in a **given** feature space.



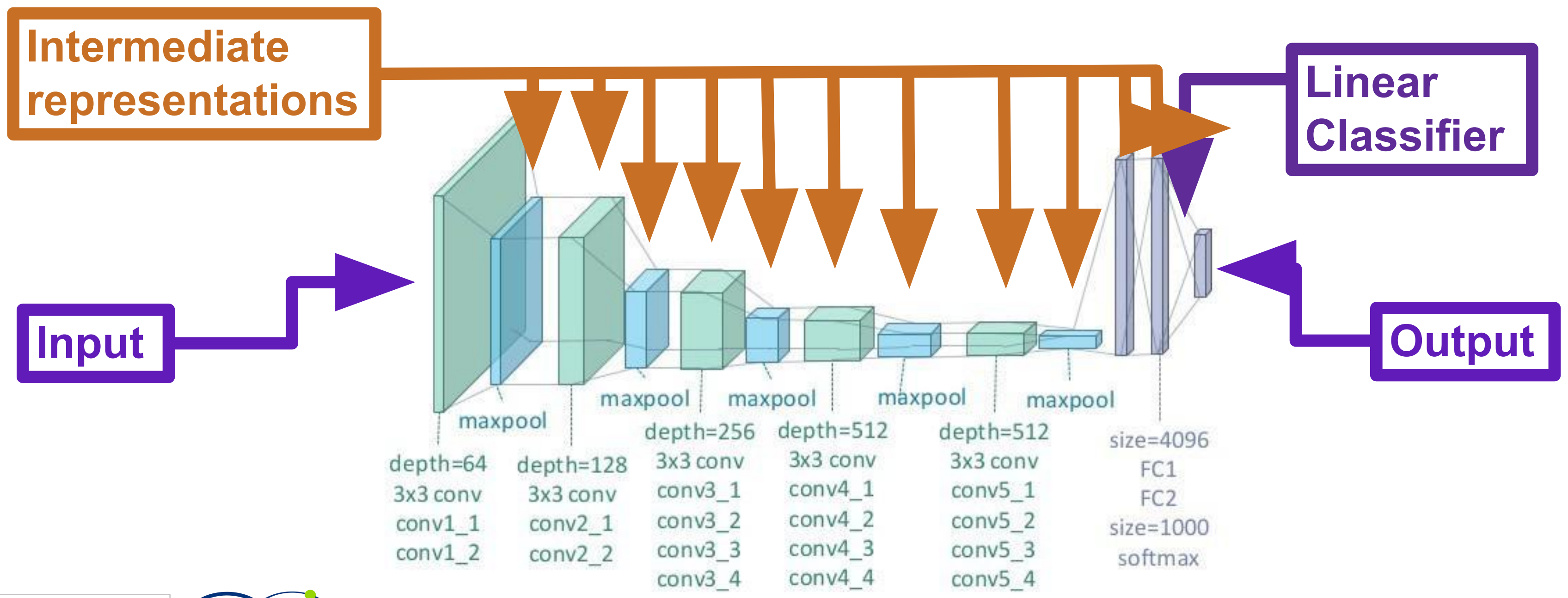
HIGH PERFORMANCE ARTIFICIAL INTELLIGENCE

A good classifier

- SVMs using the **kernel trick** can overcome the linear limitation through an **implicit** mapping to a higher dimensional feature space



Deep Neural Networks and classifiers



HIGH PERFORMANCE ARTIFICIAL INTELLIGENCE

Classifiers and Representations

- Classifiers are **Task**-specific
  - We can rarely reuse them for a different task, as they are bounded to the **label space**

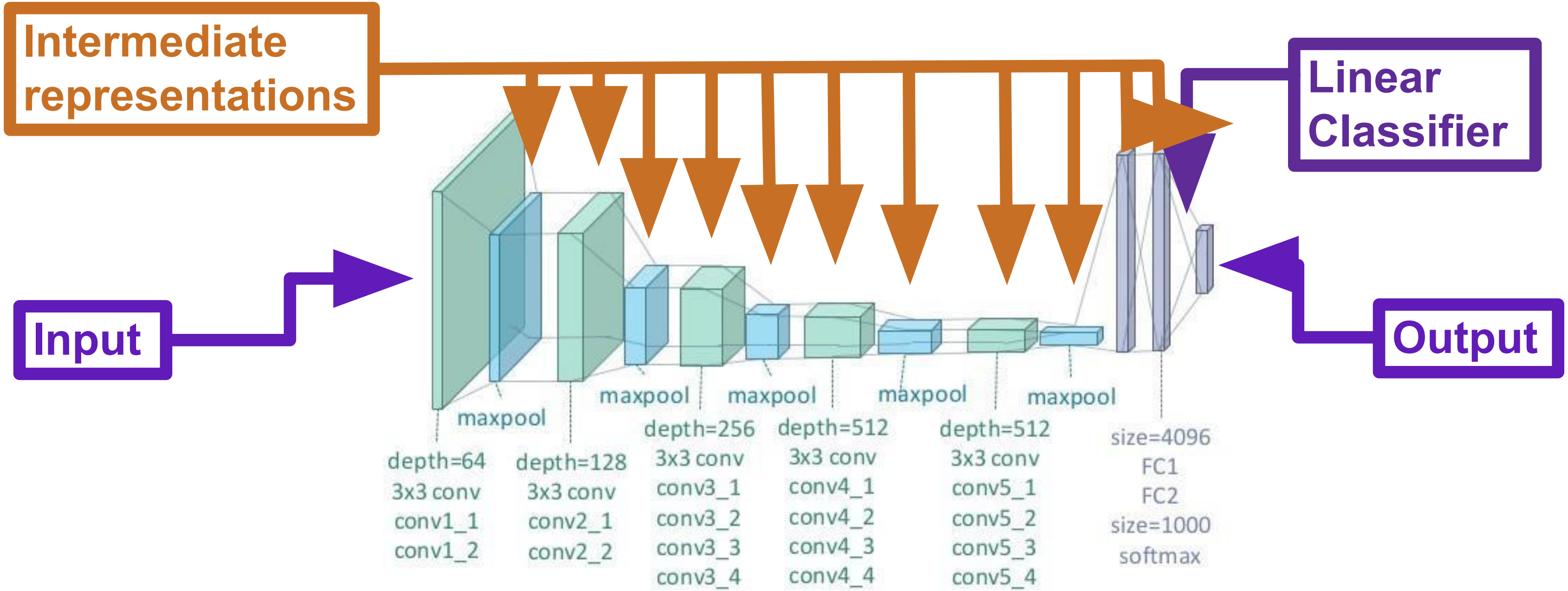
- Representations are **Domain**-specific
  - We can often reuse them for a different Task if we remain in the same **feature space**!

HIGH PERFORMANCE ARTIFICIAL INTELLIGENCE

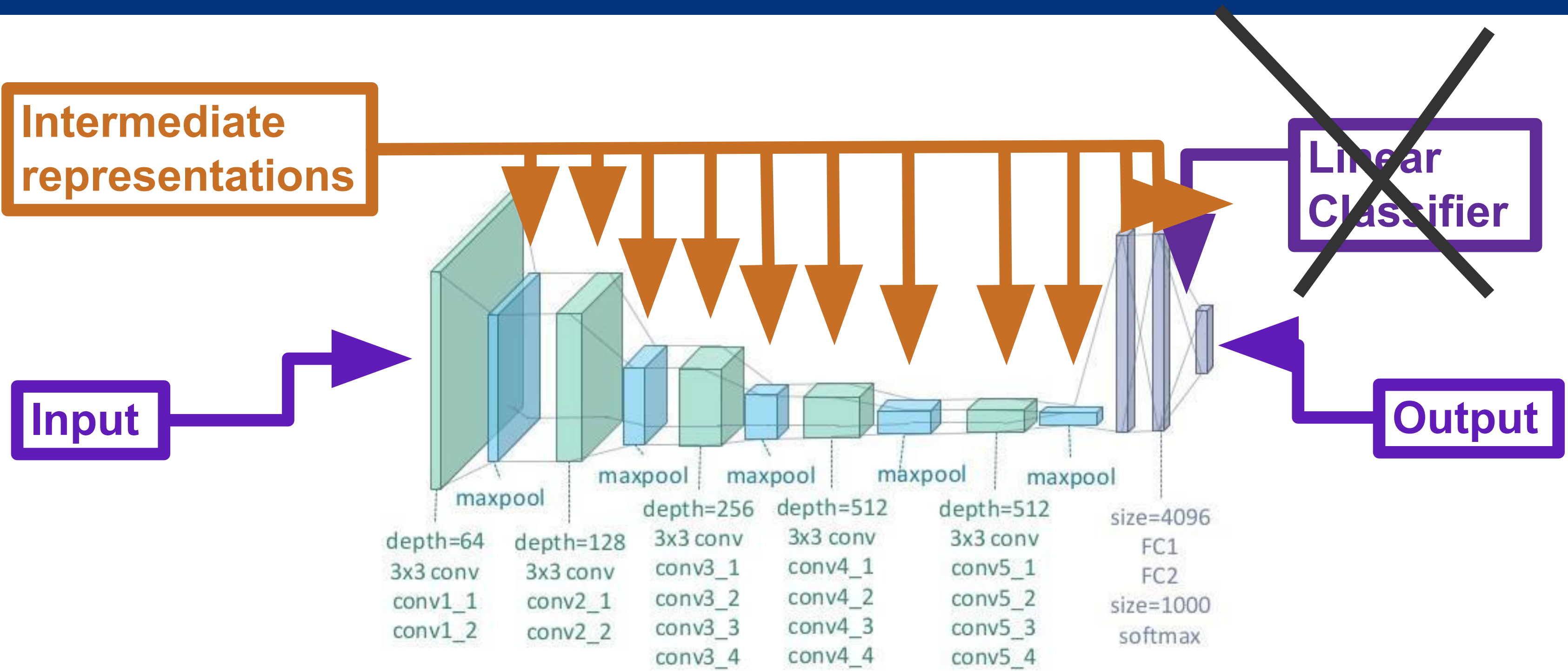
Reusing Deep Representations

Save the Earth - Reuse DNNs

What can be saved?

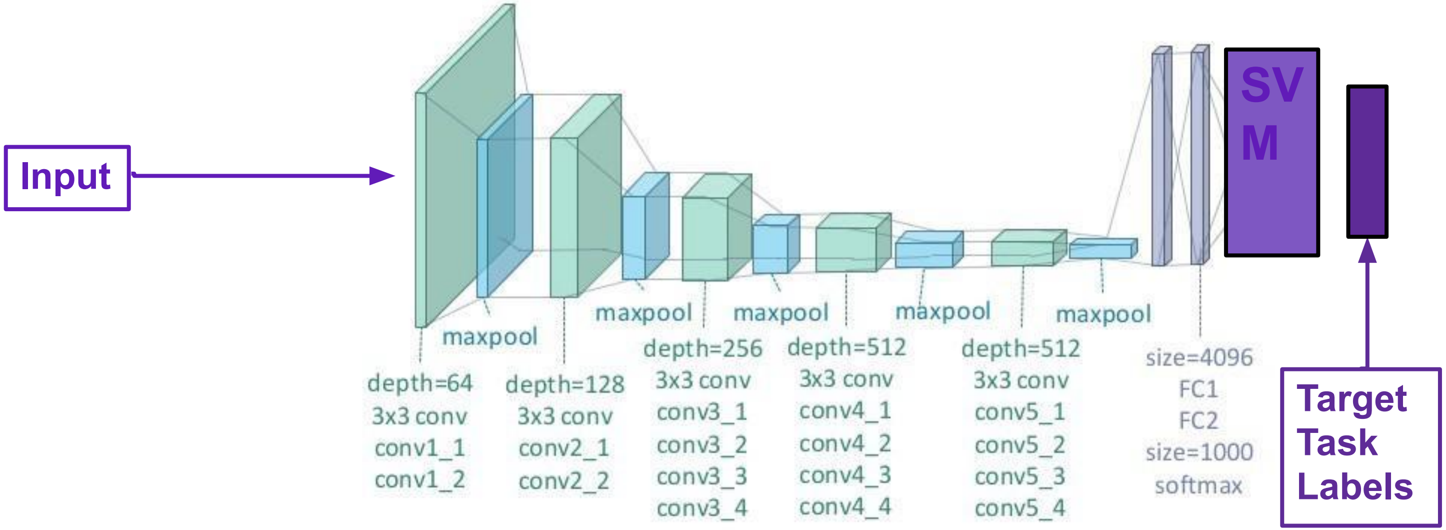


What can be saved?



Transfer Learning - Feature Extraction

- Extract output activations
- Pre-trained model is a feature extractor



When Feature Extraction

- Very scarce target data
- Very large source data
- Very similar task or direct subset

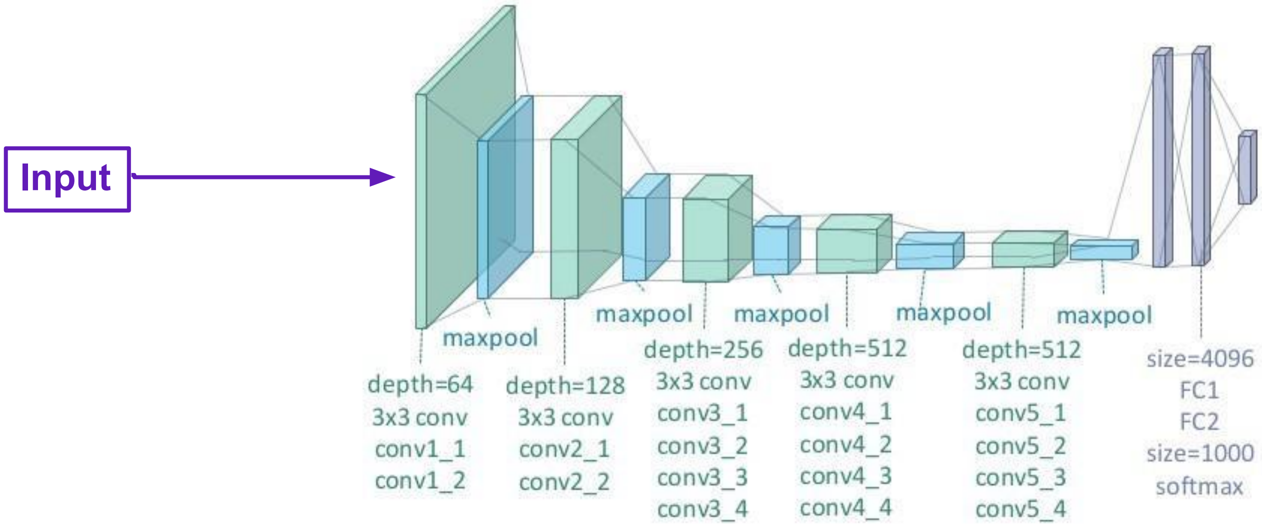
Better Feature Extraction

- Linear SVM usually enough
  - Beware dimensionality
  - Kernel likely to overfit

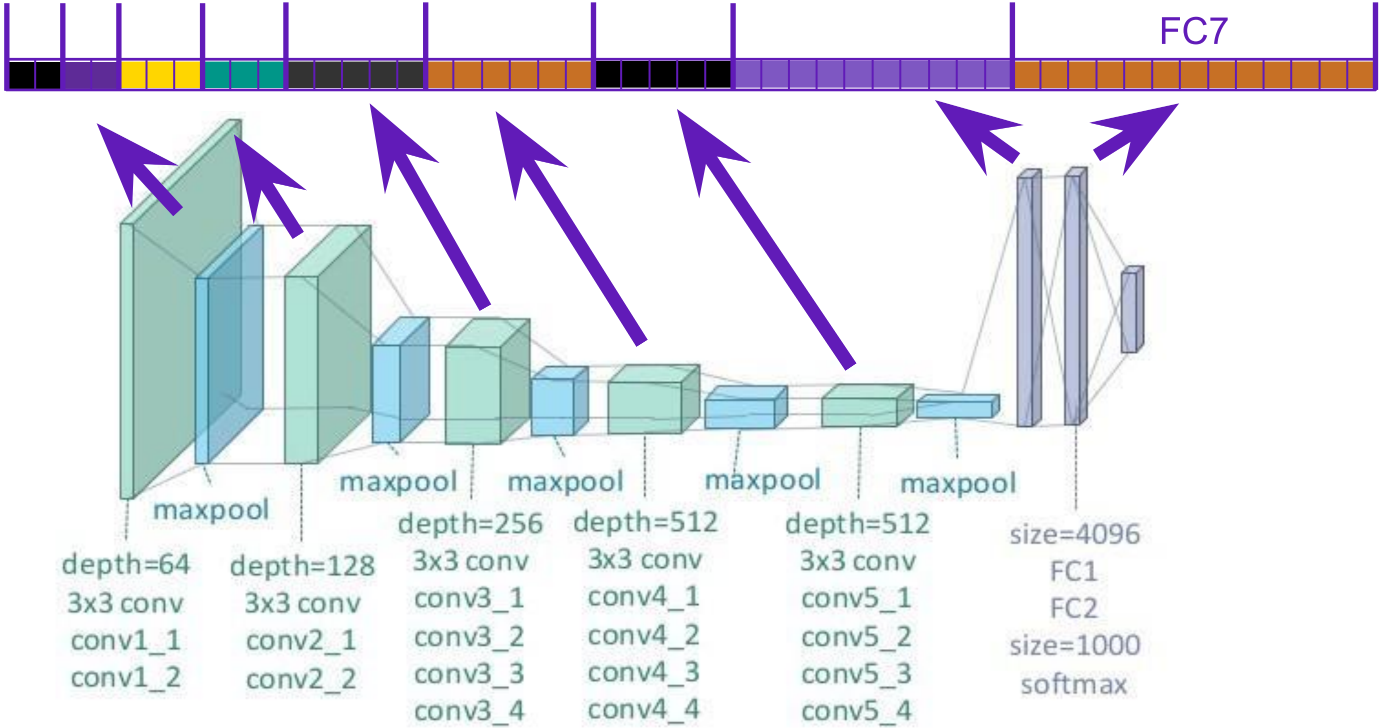
- If target data size allows
  - 1-3 FC layers

Advanced Feature Extraction

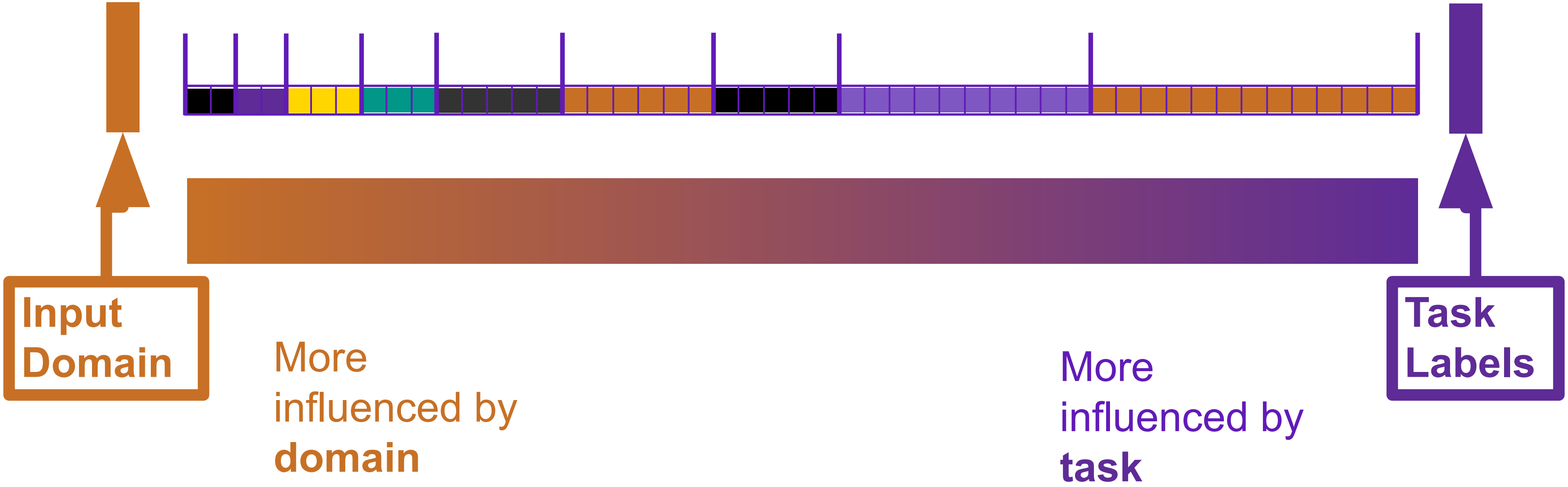
- What about the other activations?



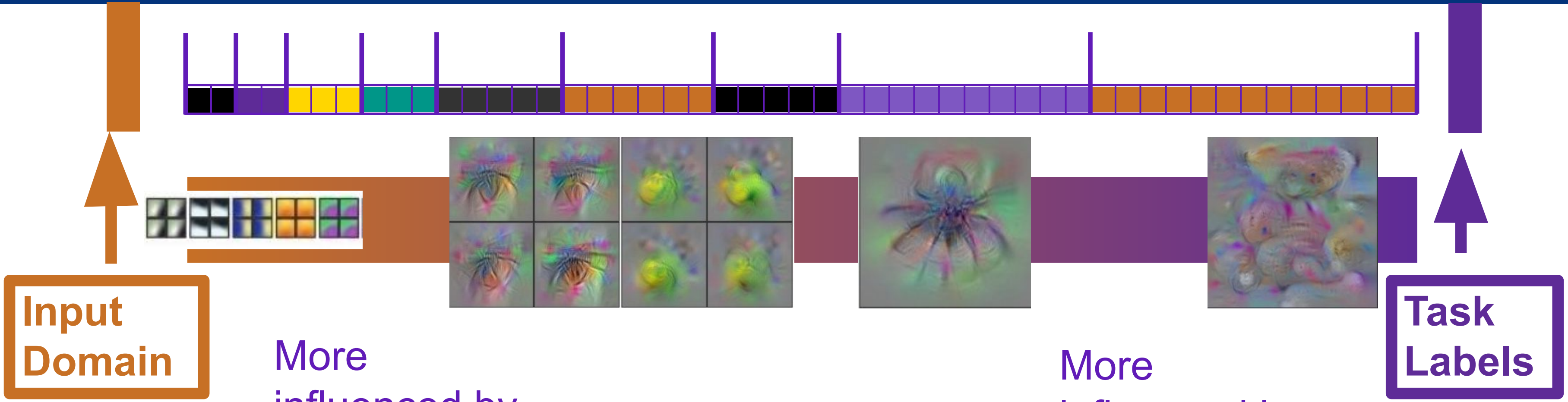
Knowledge inside DNN



Knowledge inside DNN



Knowledge inside DNN



More
influenced by
domain

**Simple
features**

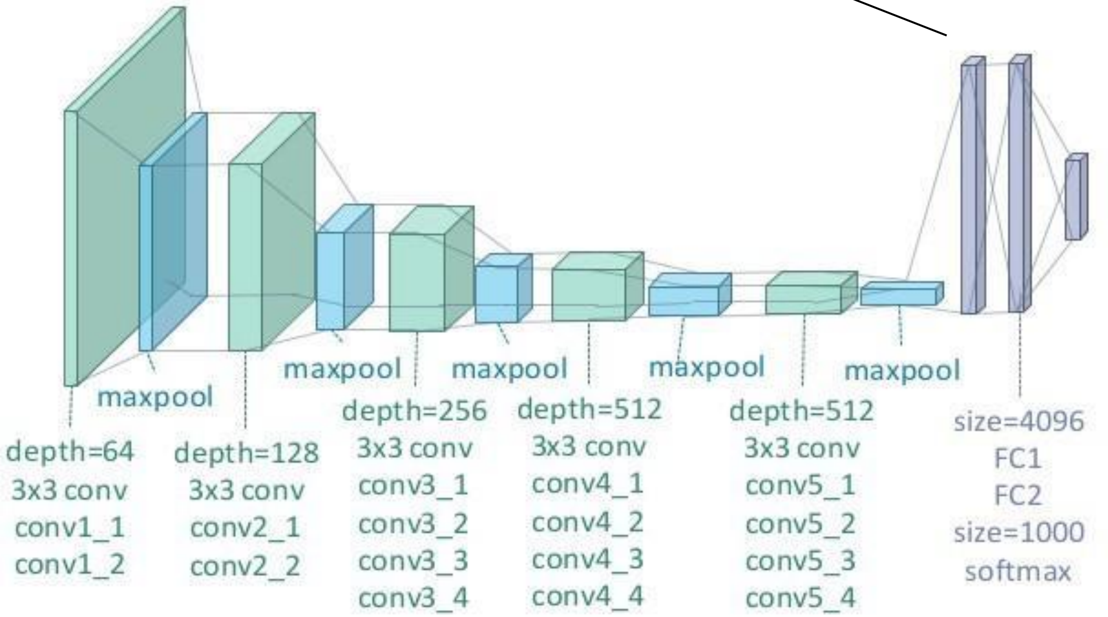
More
influenced by
task

**Complex
features**

Visualizations from: Yosinski, Jason, et al. "Understanding neural networks through deep visualization." *arXiv preprint arXiv:1506.06579* (2015).

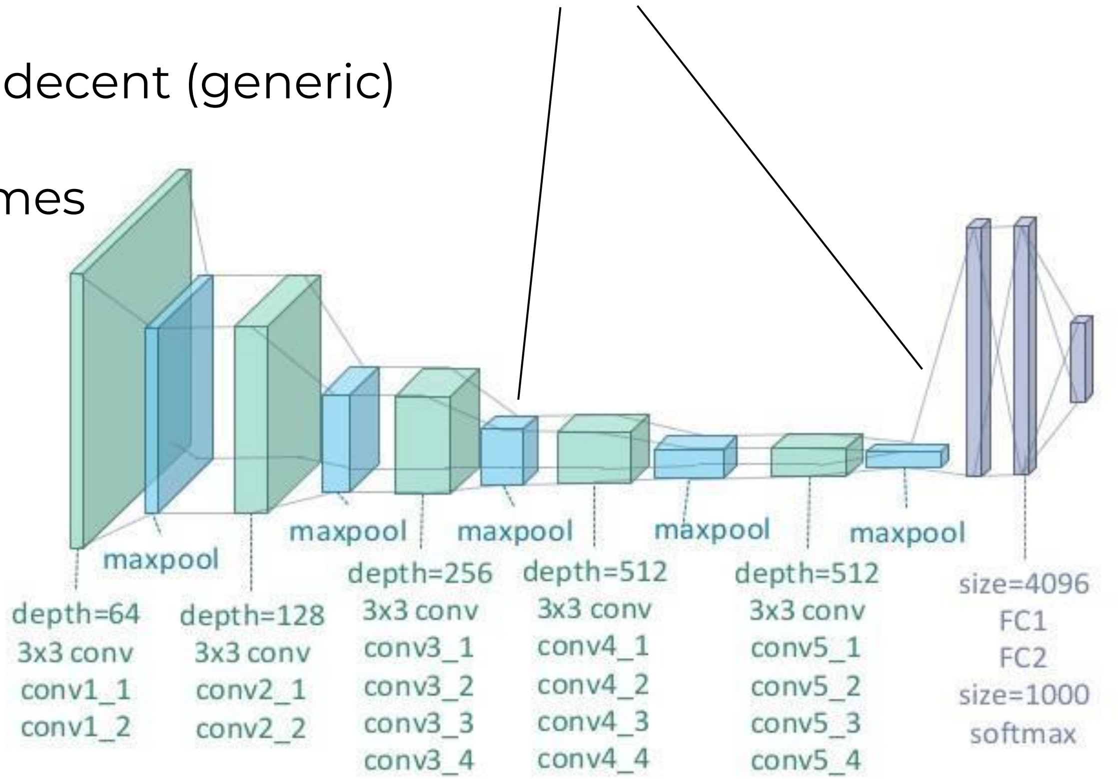
Which layers to use?

- If source & target task are **VERY** similar
  use the “classifier” layers



Which layers to use?

- If source & target task are not very similar broaden the scope
- Early layers are always decent (generic)
- Late layers are sometimes very good or very bad (discriminative)



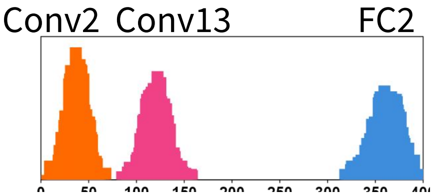
HIGH PERFORMANCE ARTIFICIAL INTELLIGENCE

Post-processing neural activations

- Beware of (relative) dimensionality
  - FC layers have lots of activations
  - Conv layers activations are spatially dependent

VGG16	*Convs*	*FCs*
# Layers:	14	2
Activations:	33%	66%

- Beware of scale
  - Different layers activate with different strength
  - BN?



Normalizing neural activations

- L2-norm by layer
  - Fixes scale (careful if mixing layers!)

- Feature standardization
  - By column
  - Representation relative to the rest of dataset
  - Statistics from train applied on val/test

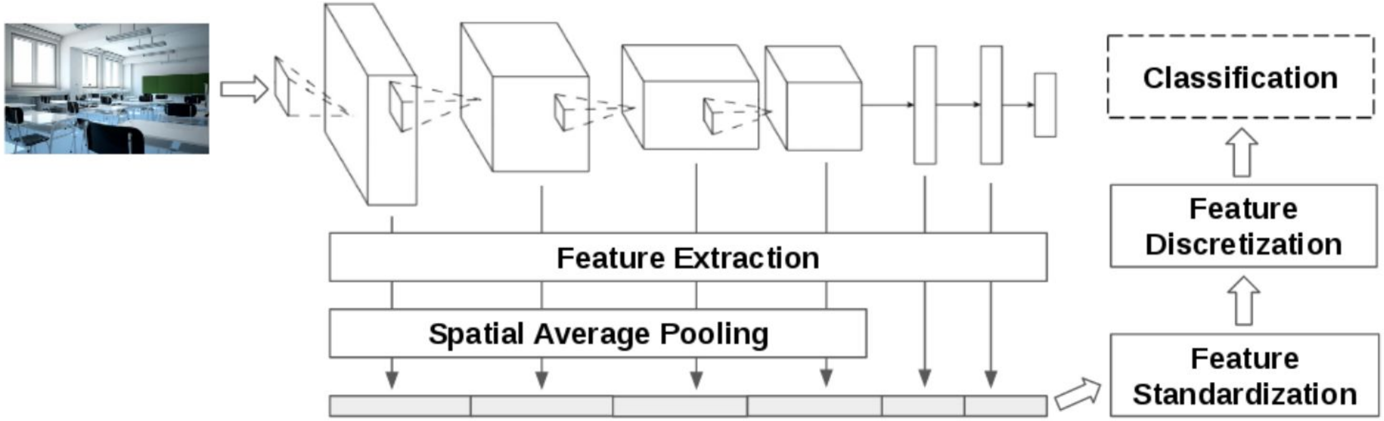
Reducing dimensionality

- Convolutional GAP
- PCA or others dim. red. techniques
  - Mixing of features
- Discretization of features
  - Reducing dimensional complexity, no dimensionality
    - e.g., (-1,0,1)

HIGH PERFORMANCE ARTIFICIAL INTELLIGENCE

Full-Network Embedding

- VGG16 FNE dimensionality: 12K



Feature extraction in action

- High similarity source - target

Network pre-trained on **Places2** for mit67 and on **ImageNet** for the rest.

Dataset	mit67	cub200	flowers102	cats-dogs	sdogs	caltech101	food101	textures	wood
Baseline fc6	80.0	65.8	89.5	89.3	78.0	91.4±0.6	61.4±0.2	69.6	70.8±6.6
Baseline fc7	81.7	63.2	87.0	89.6	79.3	89.7±0.3	59.1±0.6	69.0	68.9 ±6.8
Full-network	83.6	65.5	93.3	89.2	78.8	91.4±0.6	67.0±0.7	73.0	74.1±6.9
SotA	86.9 [5]	92.3 [10]	97.0 [5]	91.6 [6]	90.3 [5]	93.4 [31]	77.4 [4]	75.5 [17]	- -
ED	✓	✓	✓	✗	✓	✗	✗	✗	-
FT	✓	✓	✓	✓	✓	✓	✓	✗	-

HIGH PERFORMANCE ARTIFICIAL INTELLIGENCE

Feature extraction in action

- High similarity source - target

Network pre-trained on **Places2** for mit67 and on **ImageNet** for the rest.

Dataset	mit67	cub200	flowers102	cats-dogs	sdogs	caltech101	food101	textures	wood
Baseline fc6	80.0	65.8	89.5	89.3	78.0	91.4±0.6	61.4±0.2	69.6	70.8±6.6
Baseline fc7	81.7	63.2	87.0	89.6	79.3	89.7±0.3	59.1±0.6	69.0	68.9 ±6.8
Full-network	83.6	-0.3	93.3	-0.4	-0.5	91.4±0.6	67.0±0.7	73.0	74.1±6.9
SotA	86.9 [5]	92.3 [10]	97.0 [5]	91.6 [6]	90.3 [5]	93.4 [31]	77.4 [4]	75.5 [17]	- -
ED	✓	✓	✓	✗	✓	✗	✗	✗	-
FT	✓	✓	✓	✓	✓	✓	✓	✗	-

+2.9
+4.2

HIGH PERFORMANCE ARTIFICIAL INTELLIGENCE

Task similarity makes single layer l2-norm competitive

Feature extraction in action

- High similarity source - target

Network pre-trained on **Places2** for mit67 and on **ImageNet** for the rest.

Dataset	mit67	cub200	flowers102	cats-dogs	sdogs	caltech101	food101	textures	wood
Baseline fc6	80.0	65.8	89.5	89.3	78.0	91.4±0.6	61.4±0.2	69.6	70.8±6.6
Baseline fc7	81.7	63.2	87.0	89.6	79.3	89.7±0.3	59.1±0.6	69.0	68.9 ±6.8
Full-network	83.6	65.5	93.3	89.2	78.8	91.4±0.6	67.0±0.7	73.0	74.1±6.9
SotA	86.9 [5]	92.3 [10]	97.0 [5]	91.6 [6]	90.3 [5]	93.4 [31]	77.4 [4]	75.5 [17]	- -
ED	✓	✓	✓	✗	✓	✗	✗	✗	-
FT	✓	✓	✓	✓	✓	✓	✓	✗	-

+7.9

Data (external or not) can **make fine tuning worth the COST**

Feature extraction in action

- Low similarity source - target (*most real-world scenario!*)

Network pre-trained on **ImageNet** for mit67 and on **Places2** for the rest.

Dataset	mit67	cub200	flowers102	cats-dogs	caltech101	textures	wood
Baseline fc7	72.2	23.6	73.3	38.7	72.0	55.8	65.3
Full-network	75.5	35.5	88.7	56.2	80.0	65.1	74.0
	+3.3	+11.9	+15.4	+17.5	+8.0	+9.3	+10.6

Data (external or not) makes fine tuning worth the COST

Factors deep representations quality

- Source task
  - Total volume
  - Class variety
- Target task
  - Source-target similarity
- Starting Model
  - Capacity
  - Accuracy

Factors deep representations quality

- Source task
  - Total volume
  - Class variety
- Target task
  - Source-target similarity
- Starting Model
  - Capacity
  - Accuracy

If you have all of this, feature extraction plus a classifier will get you *close* to state-of-the-art in 10 minutes of CPU

HIGH PERFORMANCE ARTIFICIAL INTELLIGENCE

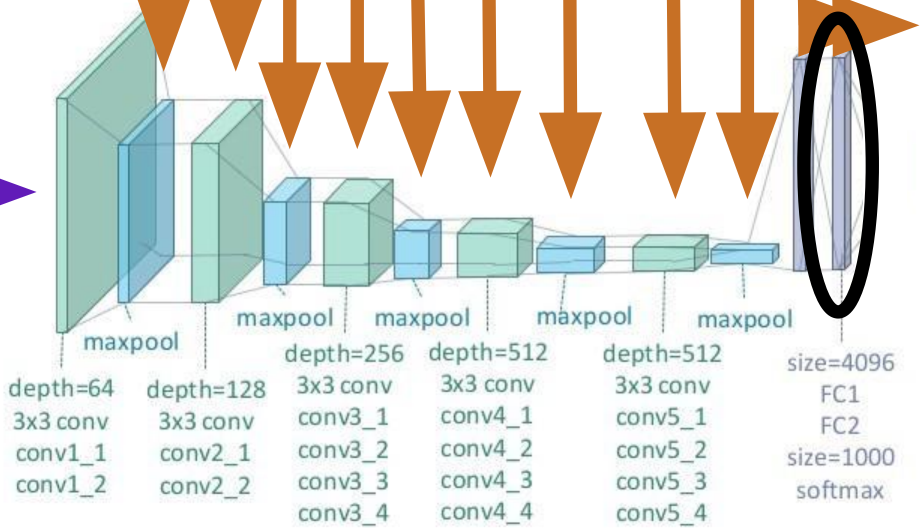
Fine Tuning

To improve, to remember, to forget

Fine tuning

Intermediate representations

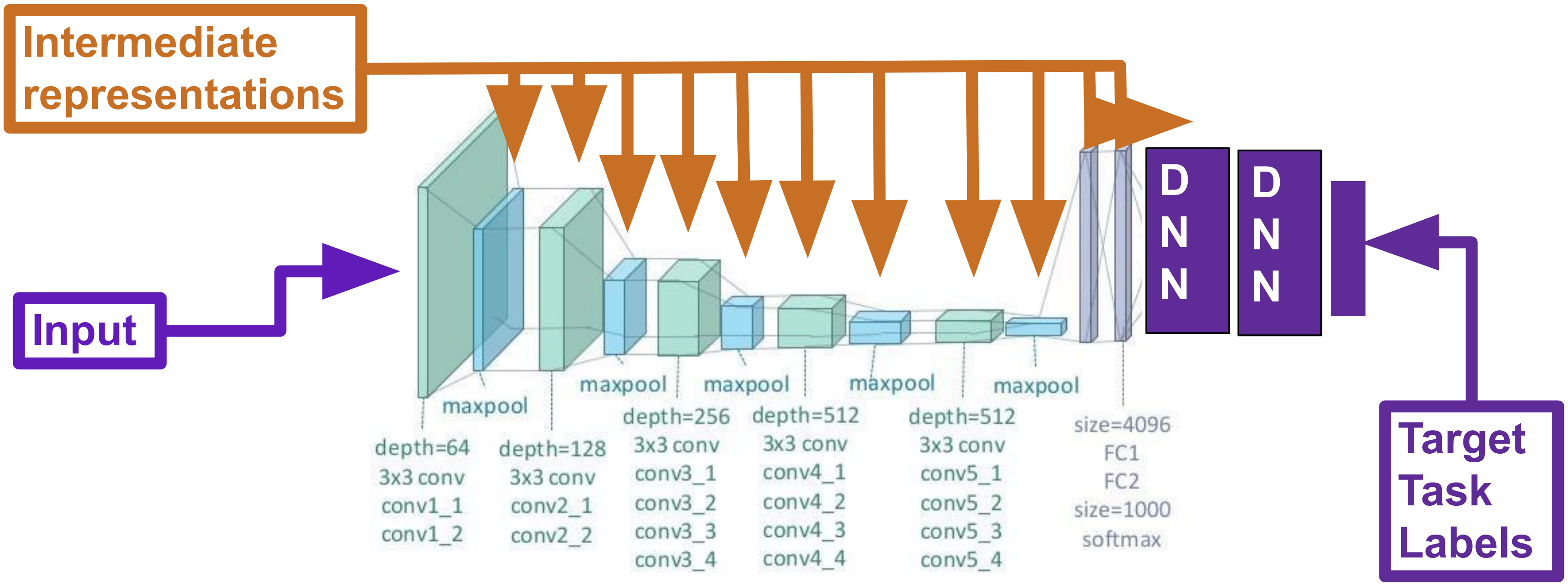
Input



Features learned for the **Source Task**

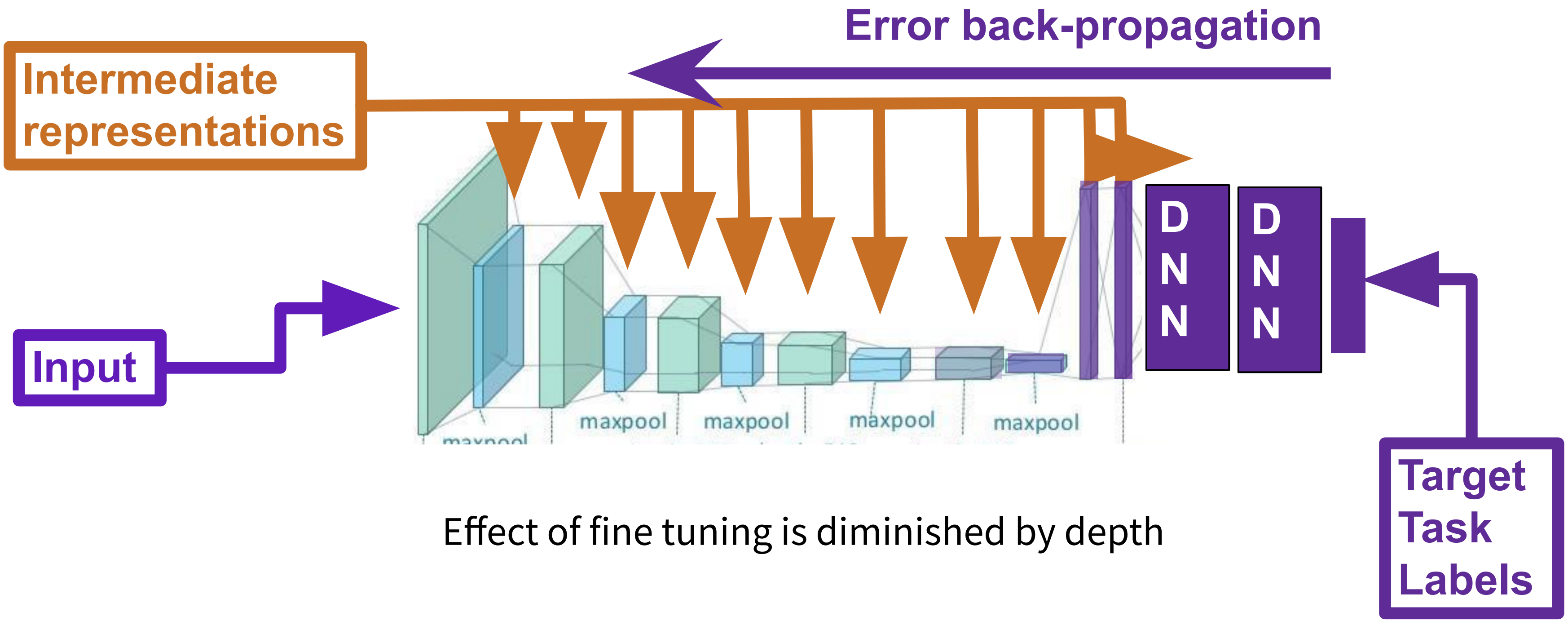
Can we make them better?

Fine tuning



HIGH PERFORMANCE ARTIFICIAL INTELLIGENCE

Fine tuning



Effect of fine tuning is diminished by depth

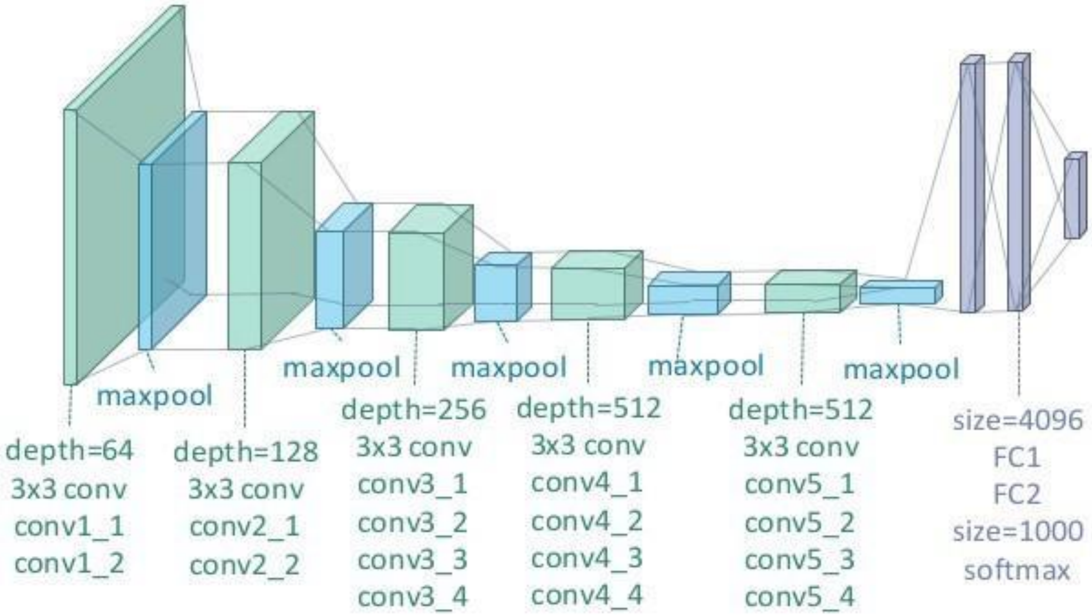
The choices in fine tuning

- Reuse and **freeze**
  - Use source task status
  - *“Its good as it is”*
- Reuse and **fine tune**
  - Start from source task status, adjust with target task
  - *“It’s a good starting point”*
- Train **from scratch**
  - Reinitialize weights randomly, train with target task only
  - *“It’s pretty much useless”*

HIGH PERFORMANCE ARTIFICIAL INTELLIGENCE

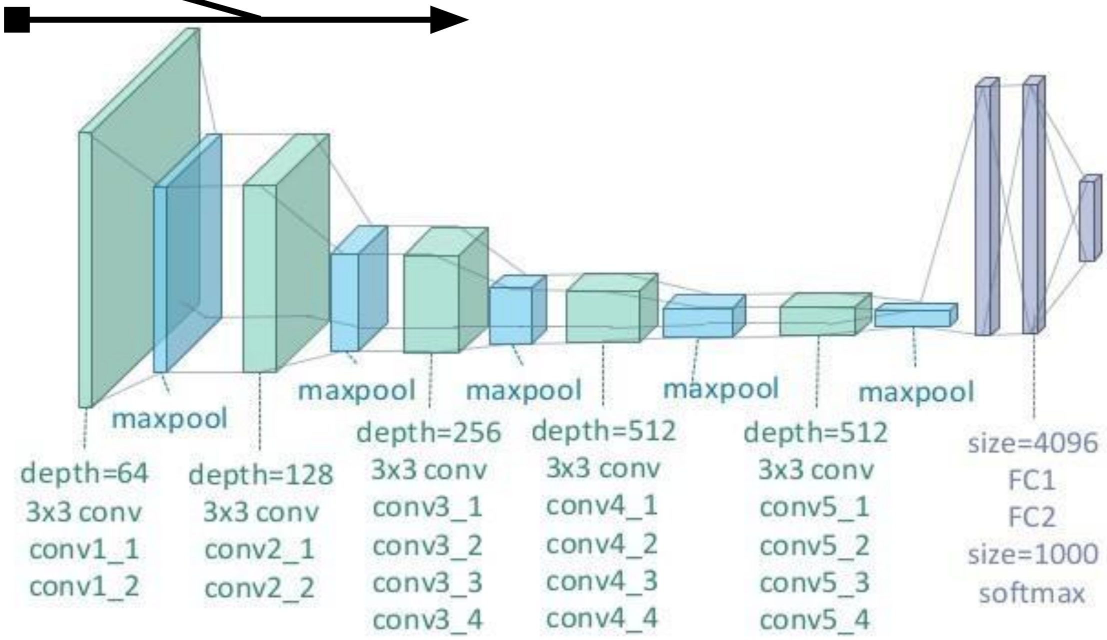
The order of fine tuning

Freeze **Fine tune** **Random**



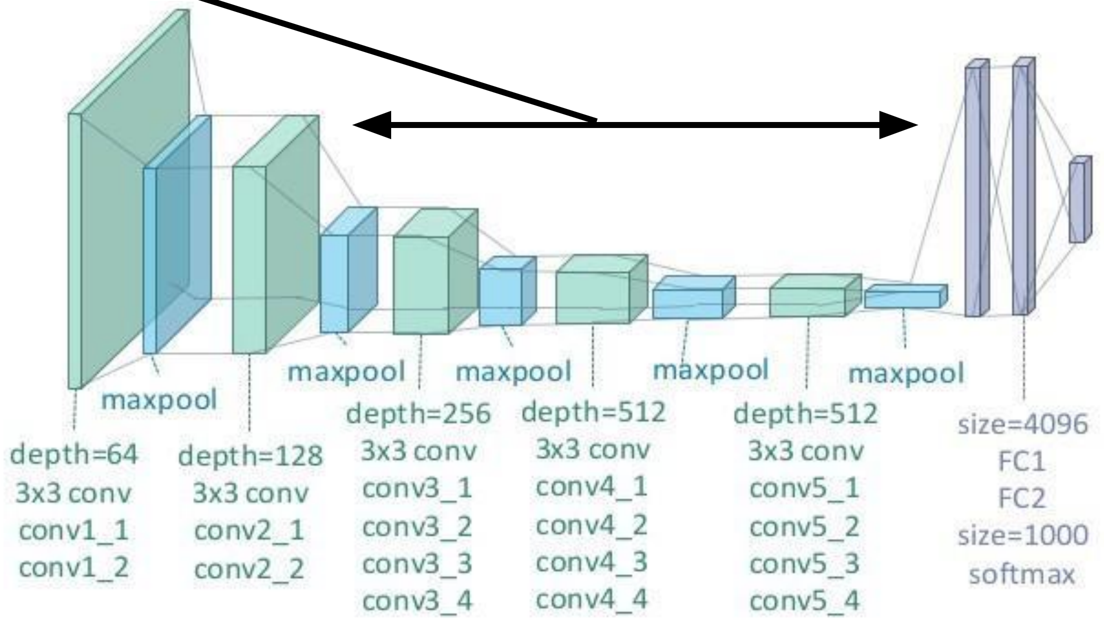
The order of fine tuning

Freeze Layers which are pretty stable and universal
Increase with source-target similarity



The order of fine tuning

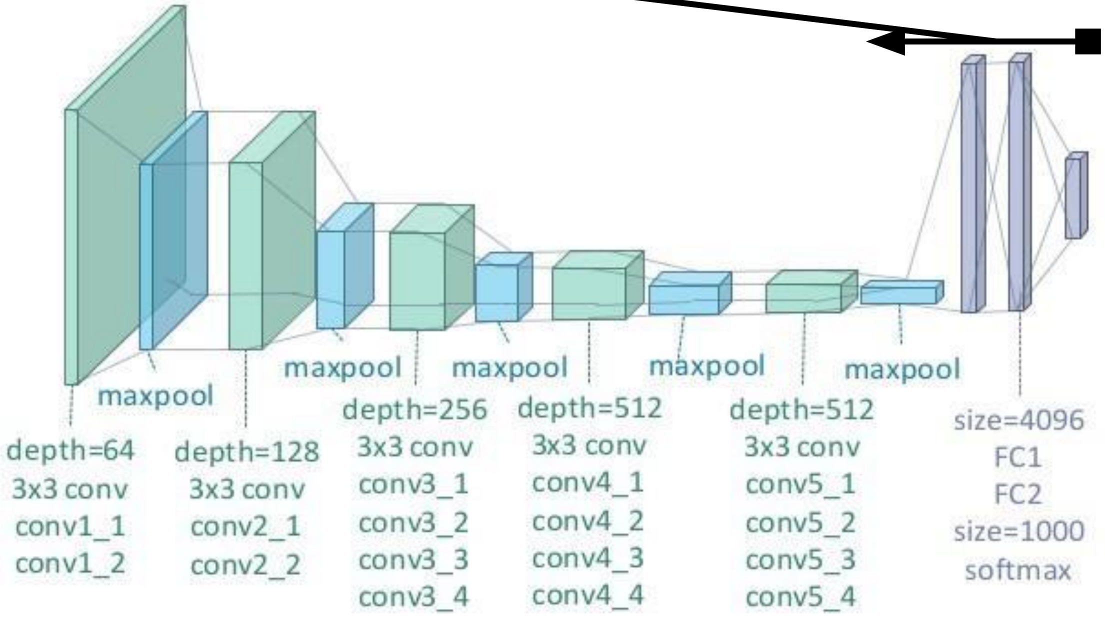
Fine tune Layers which are pretty similar but improvable
Increase with source-target similarity & data volume



HIGH PERFORMANCE ARTIFICIAL INTELLIGENCE

The order of fine tuning

Random Layers which are pretty dissimilar
Increase with source-target dissimilarity & data volume



HIGH PERFORMANCE ARTIFICIAL INTELLIGENCE

Trade-off of fine tuning

- Reuse and **freeze**
  - Remove parameters for target to learn (needs data but allows focus)
  - Adds noise
- Reuse and **fine tune**
  - Allows to focus learning (requires data)
  - Adds bias
- **Random** init
  - Again, from the top (cost, cost, cost)
  - Tailor made for target

FT vs FE

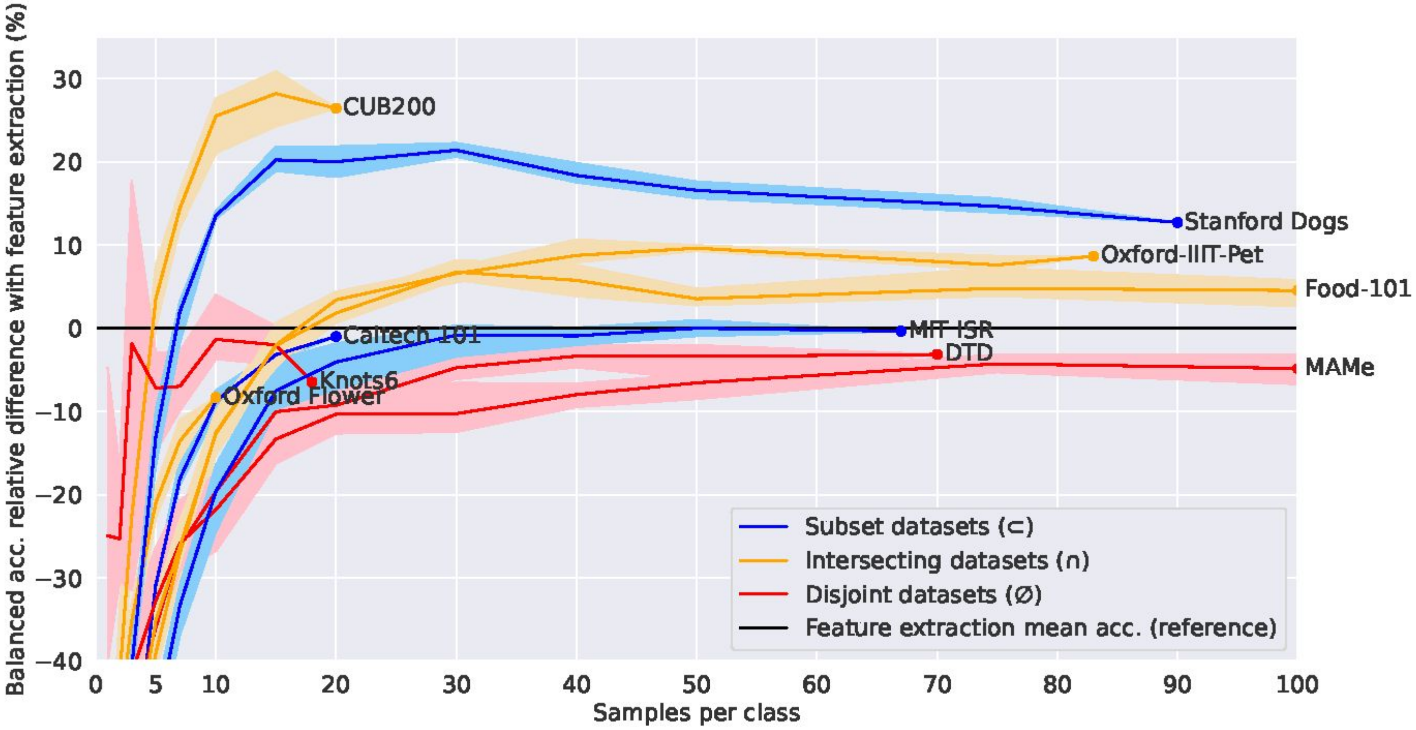
To improve, to remember, to forget

Trade-offs

	Performance		Footprint		Human cost		
	V_{ACC}	T_{ACC}	P_{AVG}	E_{CO_2}	T	n_{EXP}	A
FT	77.46	73.86	276.1W	201.54kg	1,825.72h	480	4-6h
FE	74.65	72.73	124.1W	3.84kg	60.02h	80	0-1h

HIGH PERFORMANCE ARTIFICIAL INTELLIGENCE

Training samples per class



HIGH PERFORMANCE ARTIFICIAL INTELLIGENCE

Key takeaways

- If possible, always use a pre-trained net
  - Don’t be a hero
- Consider the gradient of representations
  - From data to task
- Always analyze
  - Source/Target similarity
  - Data availability

Key takeaways

- Fine tune if possible
  - Freeze from the bottom
  - Fine tune the middle
  - Retrain from scratch at the top
- Feature extraction
  - Must-do baseline (cheap and easy!)
  - Best approach if data volume is short

HIGH PERFORMANCE ARTIFICIAL INTELLIGENCE

Parameter Efficient Fine-tuning

LoRA: Low Rank Adaptation

Fine-tuning modifying less parameters

- Freeze original weights W (dxk)
- Add a new set of weights D to be added to W
- Rank: Linearly independent columns in a matrix
- Decompose D: (dxk) = (dxr)(rxk)
    - Low rank: Lose of independent columns (Noisy approximation)
    - High rank: Keep dependent columns (No parameter gain)

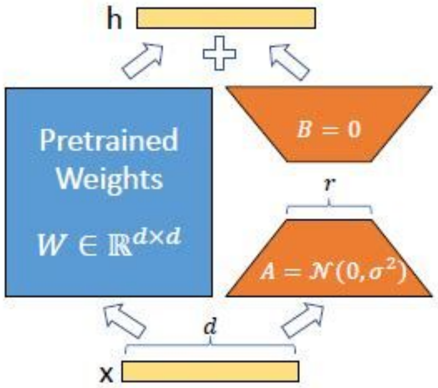


Figure 1: Our reparametrization. We only train A and B.

arxiv.2106.09685

Fine-tuning modifying less parameters

- Before training,
  - (rxk) vector initialized with zeros
  - (dxr) vector initialized with gaussian
- After training, apply delta to W
- One hyperparameter to choose = r

HIGH PERFORMANCE ARTIFICIAL INTELLIGENCE

Dario Garcia-Gasulla (BSC)

dario.garcia@bsc.es

HIGH PERFORMANCE ARTIFICIAL INTELLIGENCE