



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
 - You have to learn many fundamental things before getting to learn complex aspects of your task
- It's easier to learn if you already know something beforehand
 - There are some basic things needed to learn anything
 - In image processing, learning to “see”: Characterize images based on fundamental visual features

Why Transfer Learning

- You can learn **faster**
 - If I know that much, I'm that much closer to my goal
- You can learn **better**
 - There is a limited amount of things you can learn from data before getting trapped in spurious patterns.
 - What would you rather learn from your data?

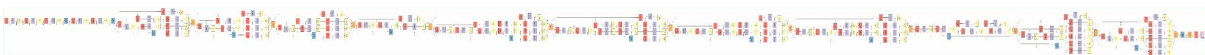
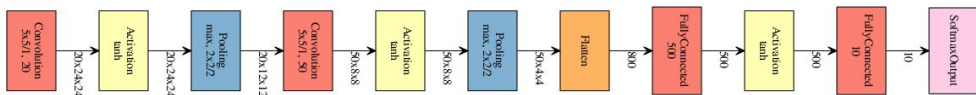


Putting things in perspective

The ImageNet ¿success?

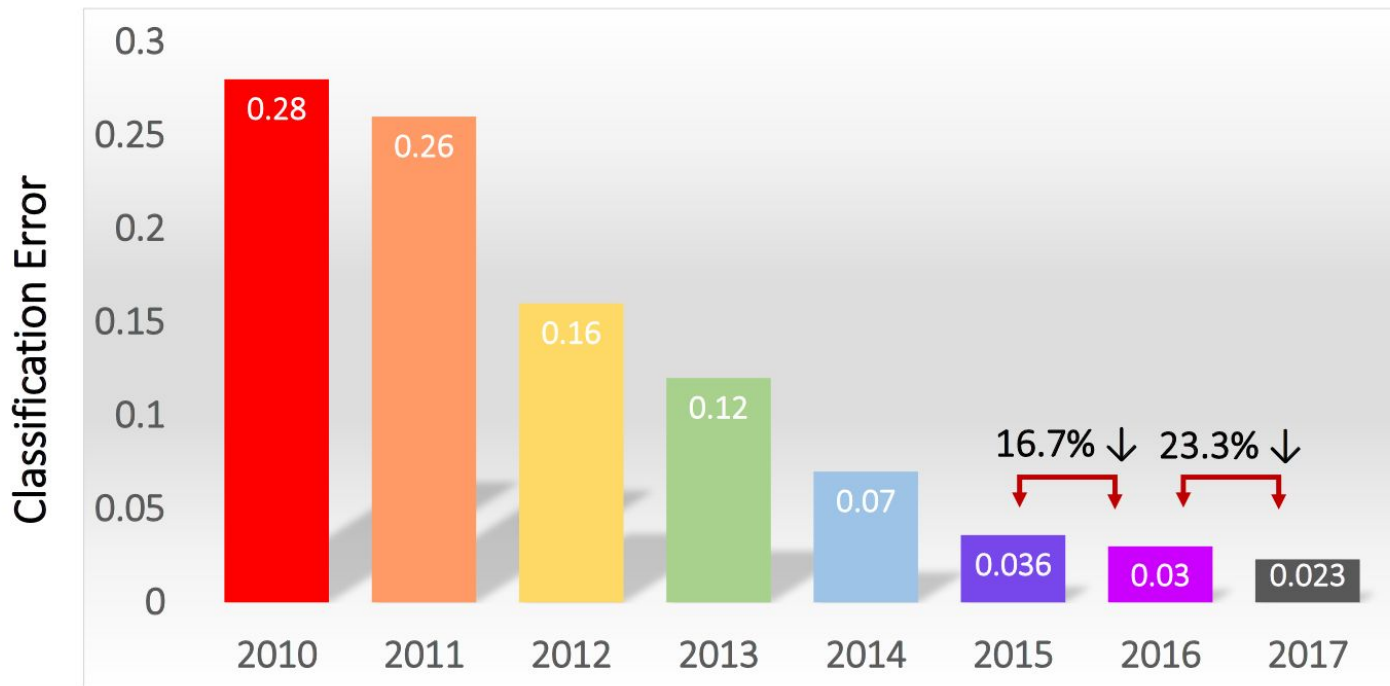
Growing up

- **1998** LeNet-5
- **2012** AlexNet
- **2014** VGG19
- **2014** GoogLeNet
- **2015** Inception-V3
- **2015** ResNet-56



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 (CO₂ emissions)
 - Human effort
 - Highly skilled professionals
 - Architecture design
 - Hyper-parameter fine tuning

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

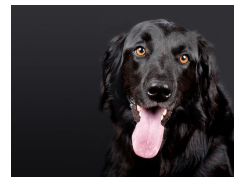
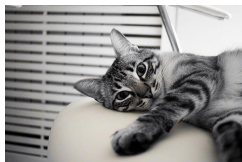
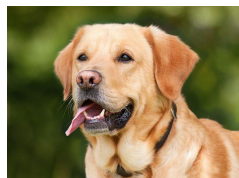
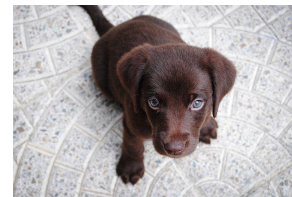
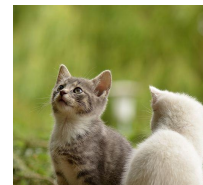
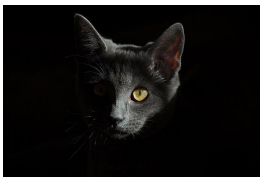
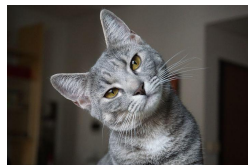
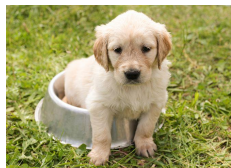
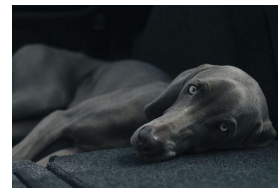
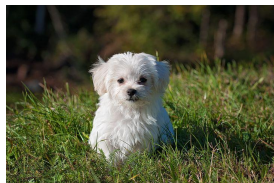
- We **cannot** do that for every single problem out there
 - The cost is too high. But more importantly...
- We **do not want to** do that for every single problem out there
 - TL to the rescue
- Transfer learning reduces the requirements on...
 - Data (implicit reuse of data)
 - Cost (faster convergence)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



The essence of Transfer Learning

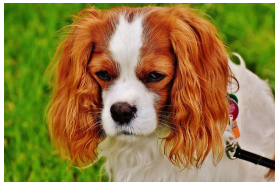
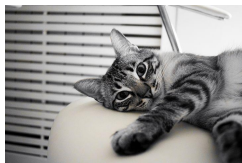
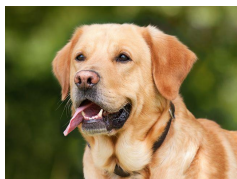
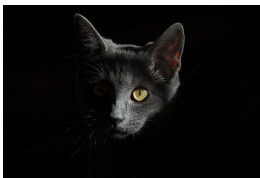
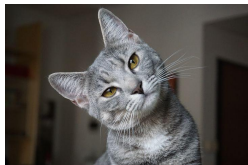
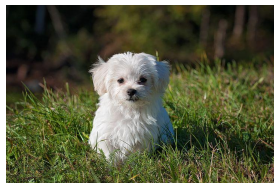
Learning it's all about generalization

What is learning about?

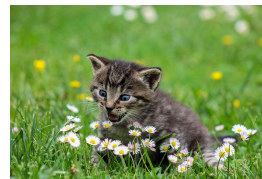
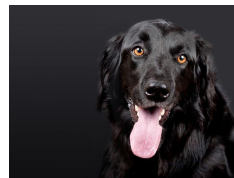
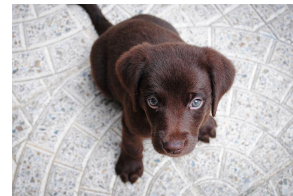
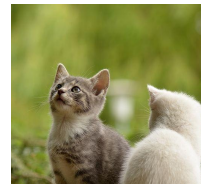
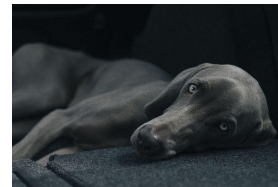


What is learning about?

Train set



Test set

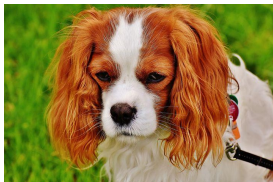
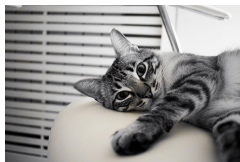
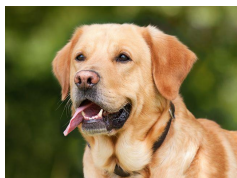
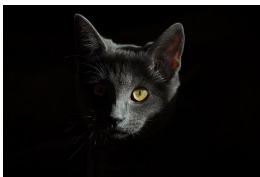
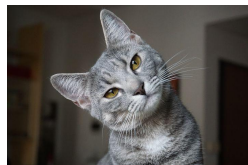
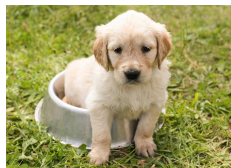


What is learning about?

MODEL

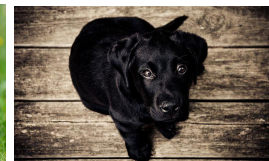
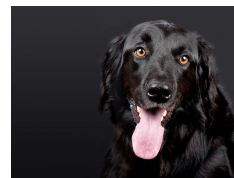
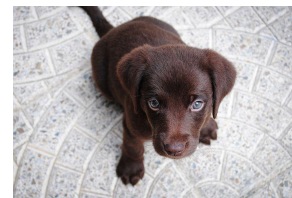
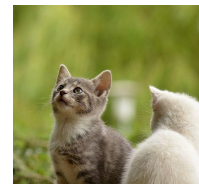
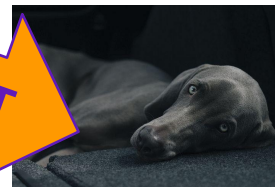
TRAIN

Train set



PREDICT

Test set



What is learning about?



What is the bias here?

Train set

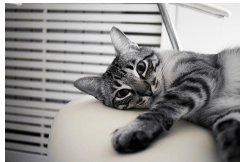


Test set

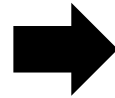


What is the bias here?

Train set



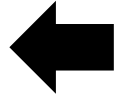
**GREEN
BACKGROUND**



DOG

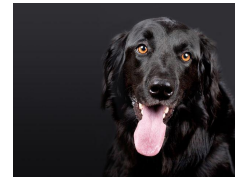
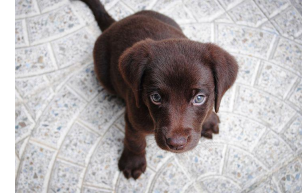
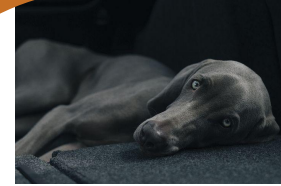
What is the bias here?

CAT

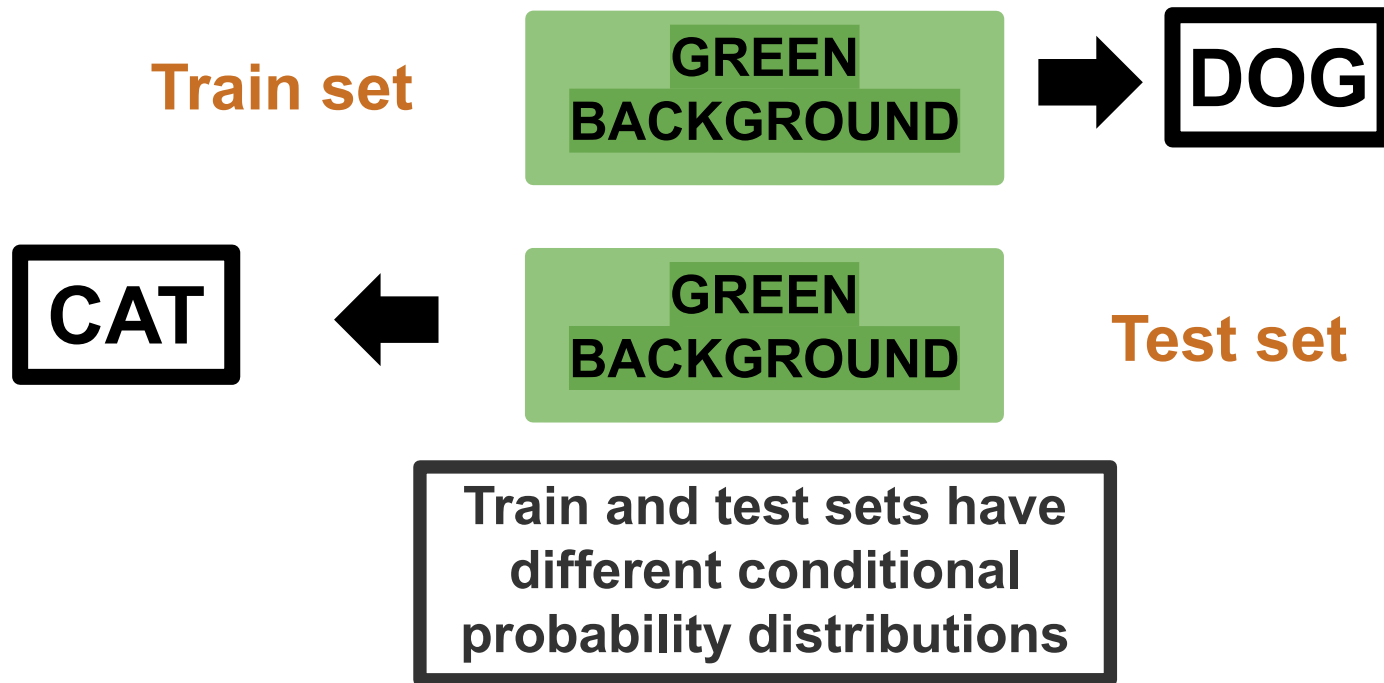


GREEN
BACKGROUND

Test set

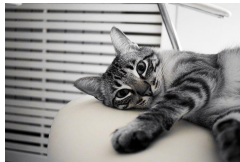


What is the bias here?



Fixing bias

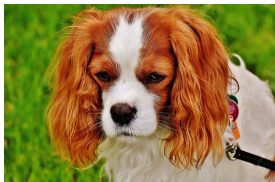
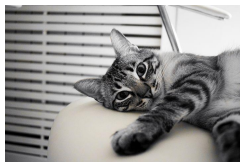
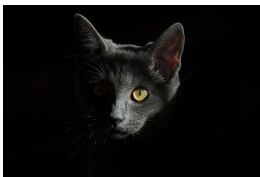
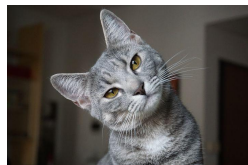
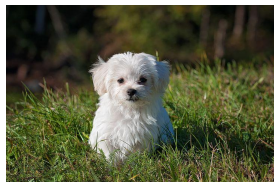
Train set



Solution?

Fixing bias

Train set

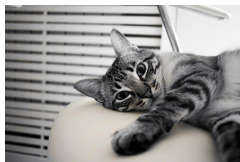
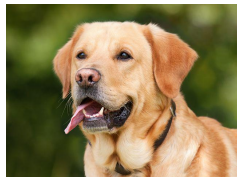
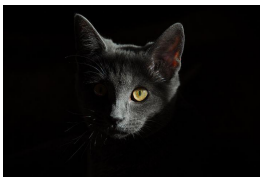
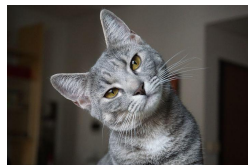
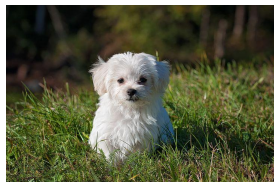


Solution?

Randomizing
ensures that train and
test sets have **similar**
conditional probability
distributions

Fixing bias

Train set

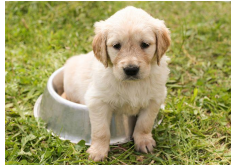


Solution?

optimizing and
**THEY WILL NEVER BE
EXACTLY EQUAL**
This is why overfitting exists

Fixing bias

Train set



**More similarity means
better generalization.
But generalization to
what?**

What is learning about?

Generalization between samples **from the same source** can be (approximately) ensured through randomization

Train set



GENERALIZATION

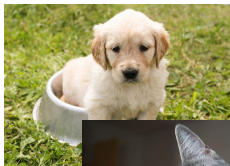
Test set



What is learning about?

Is the same source enough?
What do we really want to generalize to?
What is the **real purpose** of the model?
What should we test on?

Train set

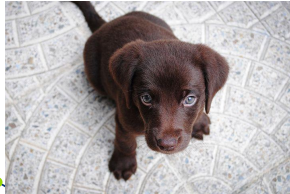


GENERALIZATION



A realistic scenario

Train set



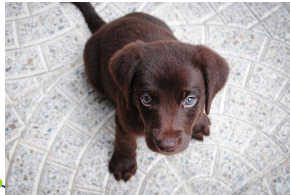
Test set



A realistic scenario

Generalization in this case is less certain
Error is expected to rise
Is it fixable?

Train set



Test set





Formalizing Transfer Learning

Tasks and Domains

Pan, Sinno Jialin, and Qiang Yang.
"A survey on transfer learning."
*IEEE Transactions on knowledge
and data engineering* 22.10 (2010):
1345-1359.

Formalizing transfer learning

Domain:

What is the nature of data?

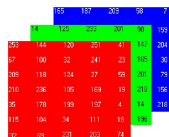
Which is it manifold?

Task:

Formalizing transfer learning

Domain: $\mathcal{D} = \{\mathcal{X}, P(X)\}$

- A feature space \mathcal{X}

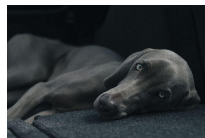
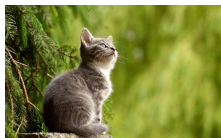


“The Elgar Concert
Hall at the University
of Birmingham for
our third conference”

→ Bag of words

→ Content vector

- A marginal probability distribution $P(X)$, where $X = \{x_1, \dots, x_n\} \in \mathcal{X}$



Task:

Formalizing transfer learning

Domain:

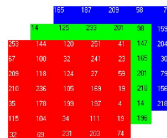
Task:

What is the mapping of data?
How is it computed?

Formalizing transfer learning

Domain: $\mathcal{D} = \{\mathcal{X}, P(X)\}$

- A feature space \mathcal{X}



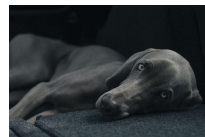
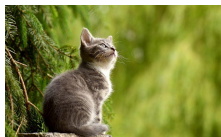
\neq

"The Elgar Concert
Hall at the University
of Birmingham for
our third conference"

→ Bag of words

→ Content vector

- A marginal probability distribution $P(X)$, where $X = \{x_1, \dots, x_n\} \in \mathcal{X}$



\neq

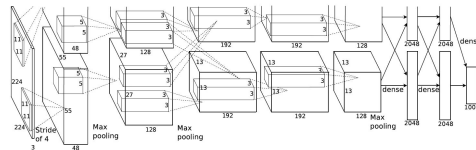


Task: $\mathcal{T} = \{y, f(\cdot)\}$

- A label space \mathcal{Y}

CAT, DOG \neq LION, WOLF

- An objective predictive function $f(\cdot) \Leftrightarrow P(y|x)$



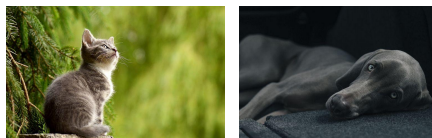
Formalizing transfer learning

Domain: $\mathcal{D} = \{\mathcal{X}, P(X)\}$

- A feature space \mathcal{X}
 - The Same (different)
- A marginal probability distribution $P(X)$
 - Different
 - Similar

Task: $\mathcal{T} = \{y, f(\cdot)\}$

Source



Target



Formalizing transfer learning

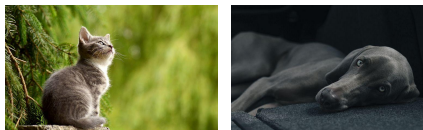
Domain: $\mathcal{D} = \{\mathcal{X}, P(X)\}$

- A feature space \mathcal{X}
 - The Same (different)
- A marginal probability distribution $P(X)$
 - Different
 - Similar

Task: $\mathcal{T} = \{y, f(\cdot)\}$

- A label space y
 - Different
 - The same
- An objective predictive function
 - Different (but similar?)

Source



{CAT, DOG}
{FELINE, CANINE}

$f_S(\cdot)$

Target

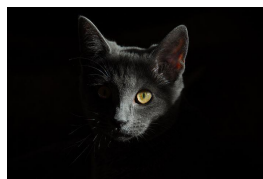
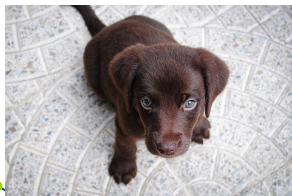
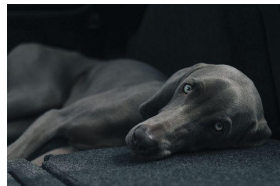
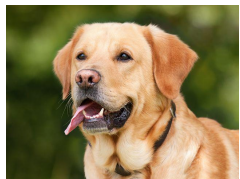
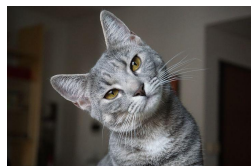


{LION, WOLF}
{FELINE, CANINE}

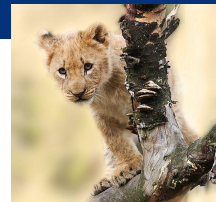
$f_T(\cdot)$

What is transfer learning about?

Train set

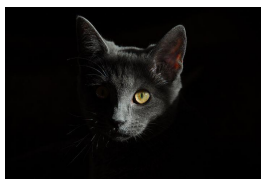
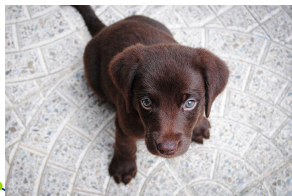
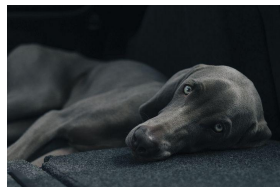
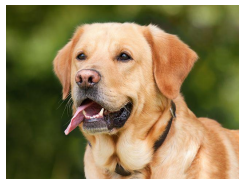
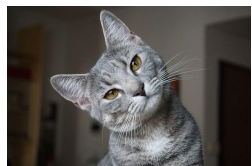


Test set

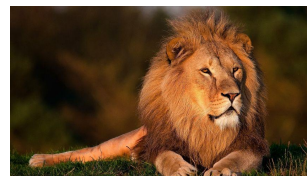
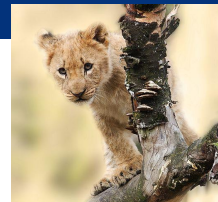


What is transfer learning about?

Source domain



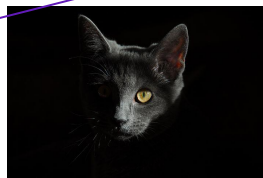
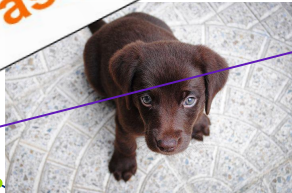
Target domain



What is transfer learning about?

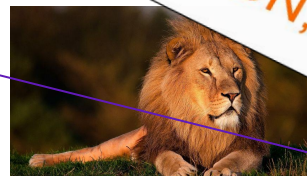
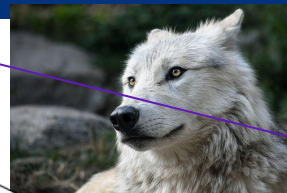
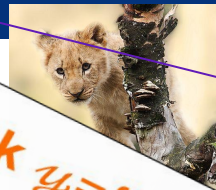
Source domain

Source Task $y=\{\text{CAT, DOG}\}, f_s(\cdot)$



Target domain

Target Task $y=\{\text{LION, WOLF}\}, f_T(\cdot)$



Formalizing transfer learning

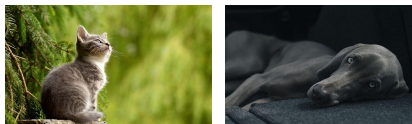
Domain: $\mathcal{D} = \{\mathcal{X}, P(X)\}$

- A feature space \mathcal{X}
 - The Same (different)
- A marginal probability distribution $P(X)$
 - Different
 - Similar

Task: $\mathcal{T} = \{y, f(\cdot)\}$

- A label space y
 - Different
 - The same

Source



{CAT, DOG}
{FELINE, CANINE}

Target



{LION, WOLF}
{FELINE, CANINE}

- An objective predictive function
 - Different (but similar?)

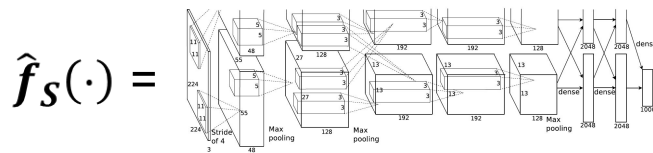
$f_S(\cdot)$

$f_T(\cdot)$

Formalizing transfer learning

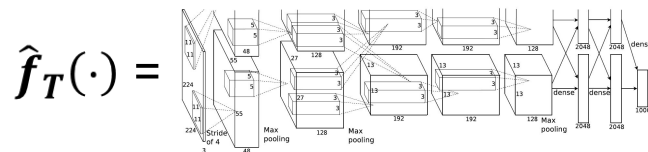
Source

$f_S(\cdot)$



Target

$f_T(\cdot)$



- Are they similar?
- Can we just use $\hat{f}_S(\cdot)$ to approximate $f_T(\cdot)$?
- Can we reuse part of it?

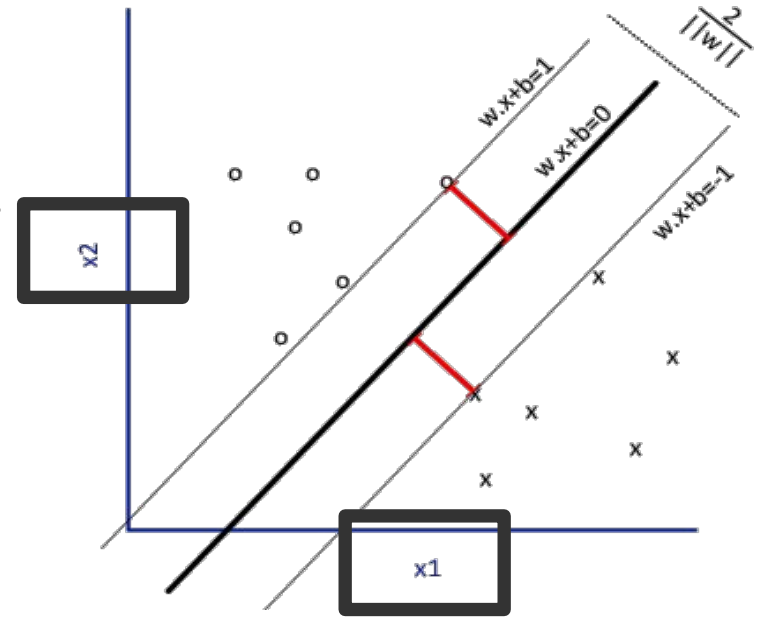


Representation Learning & Classifiers

Learning to describe

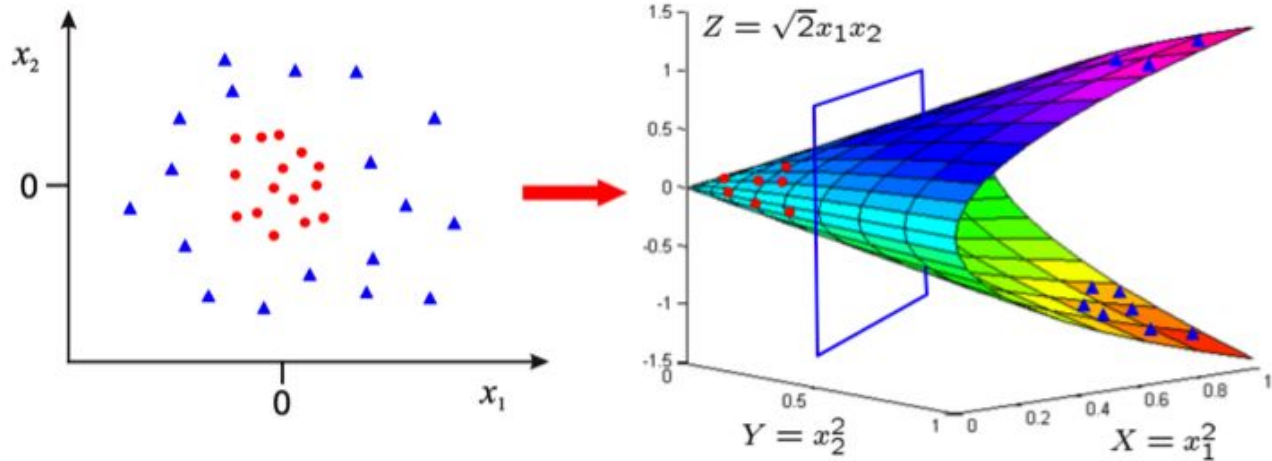
A typical classifier

- **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.

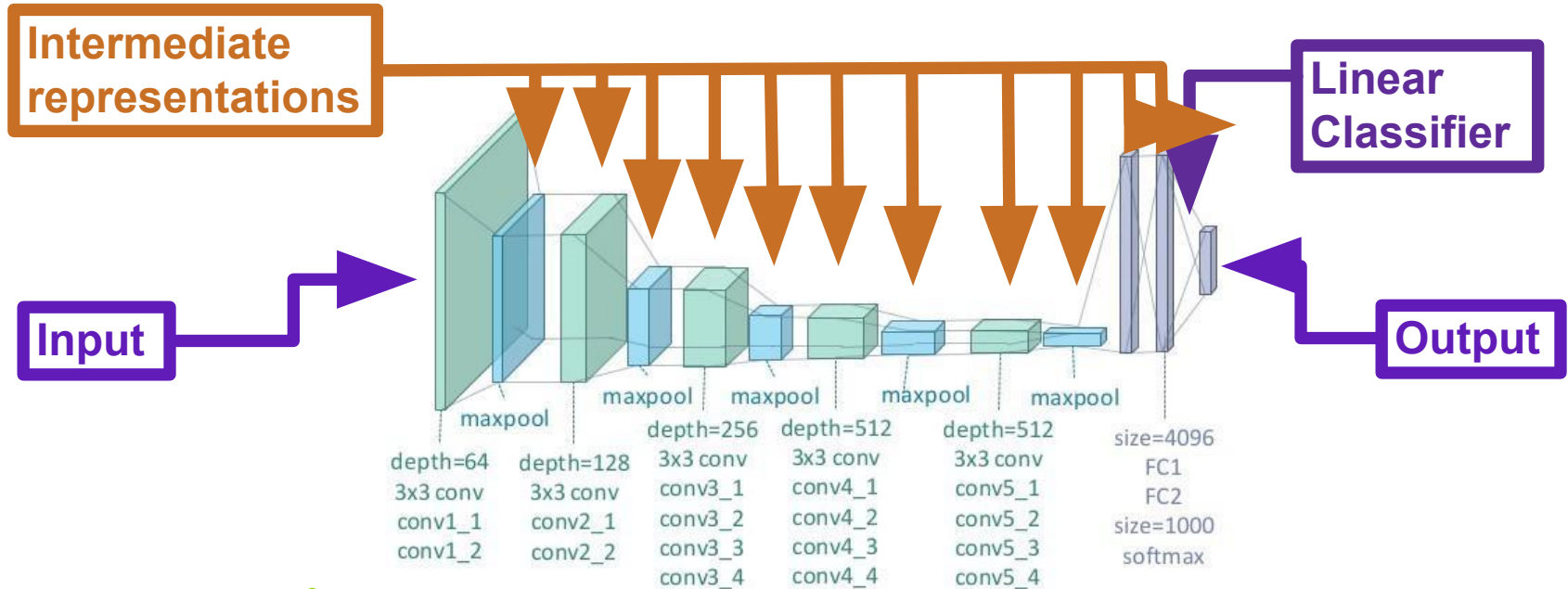


A good classifier

- 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



Classifiers and Representations

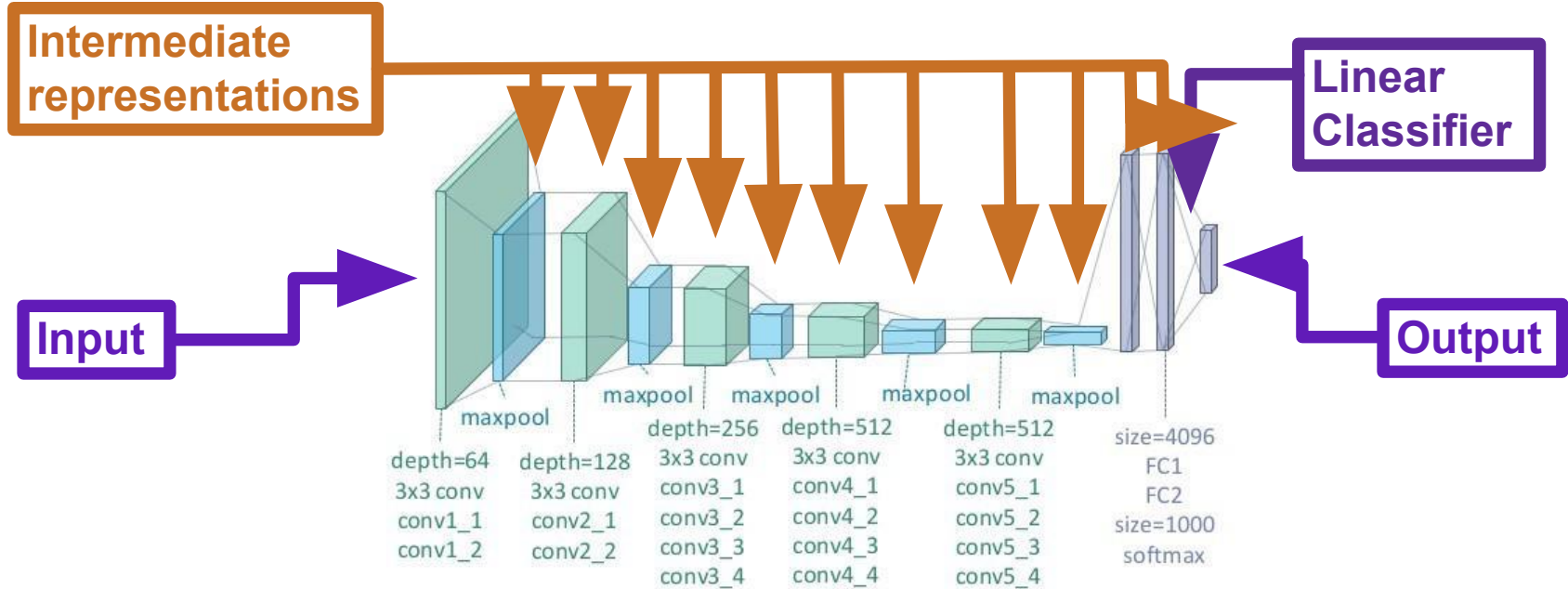
- Classifiers are **Task**-specific
 - We can rarely reuse them for a different task, as they are bounded to the **label space**
- Representations are **Domain**-specific
 - We can often reuse them for a different Task if we remain in the same **feature space**!



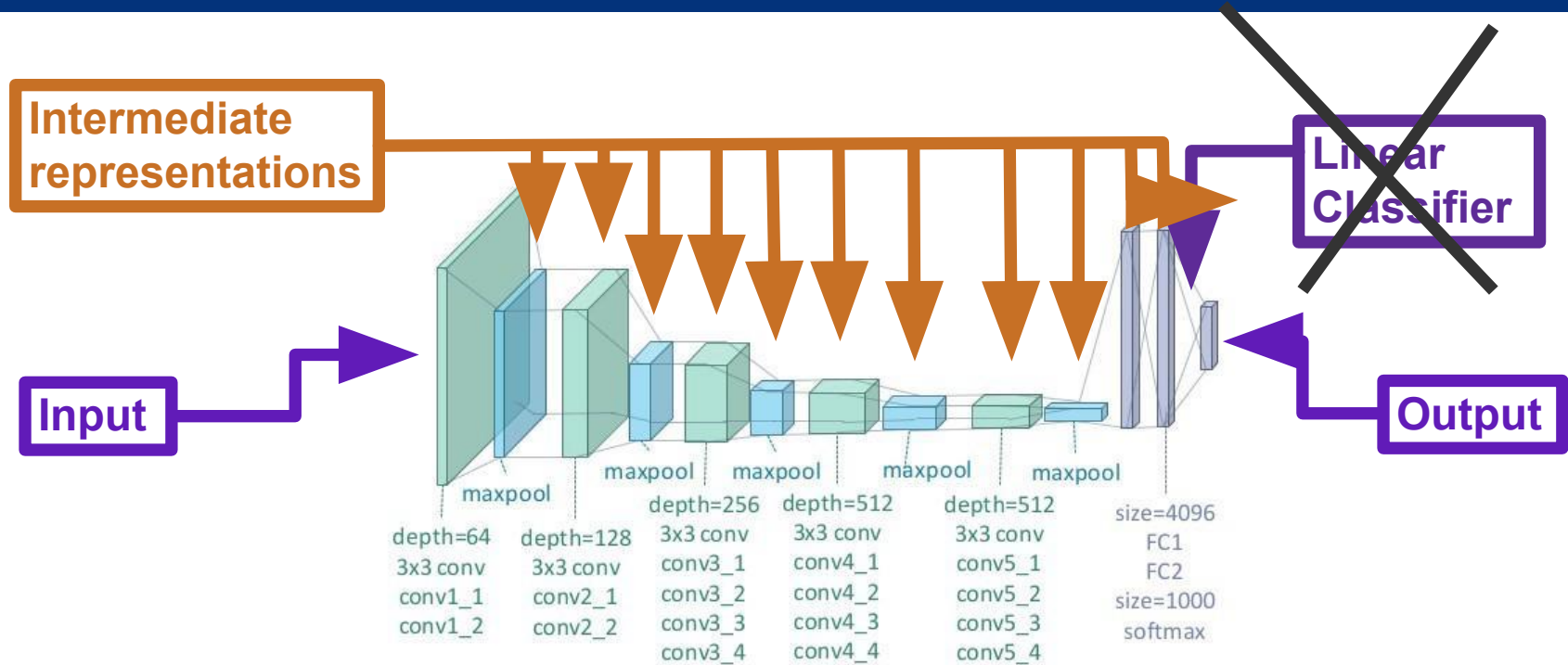
Reusing Deep Representations

Save the Earth - Reuse DNNs

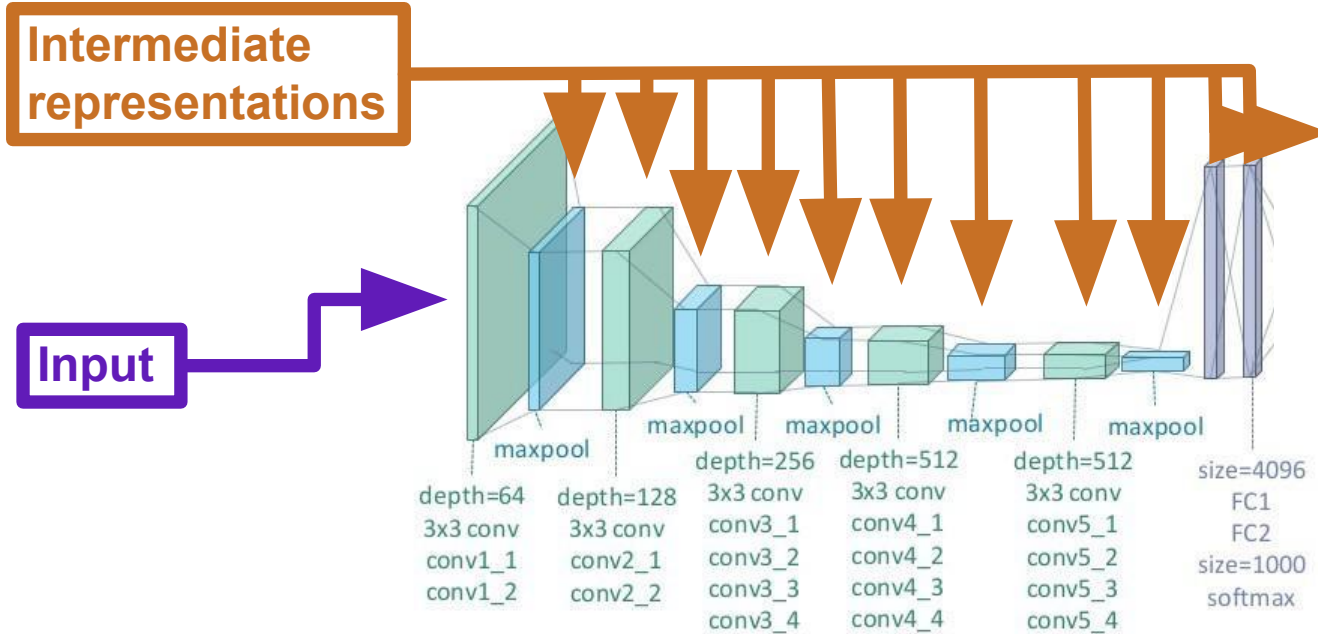
What can be saved?



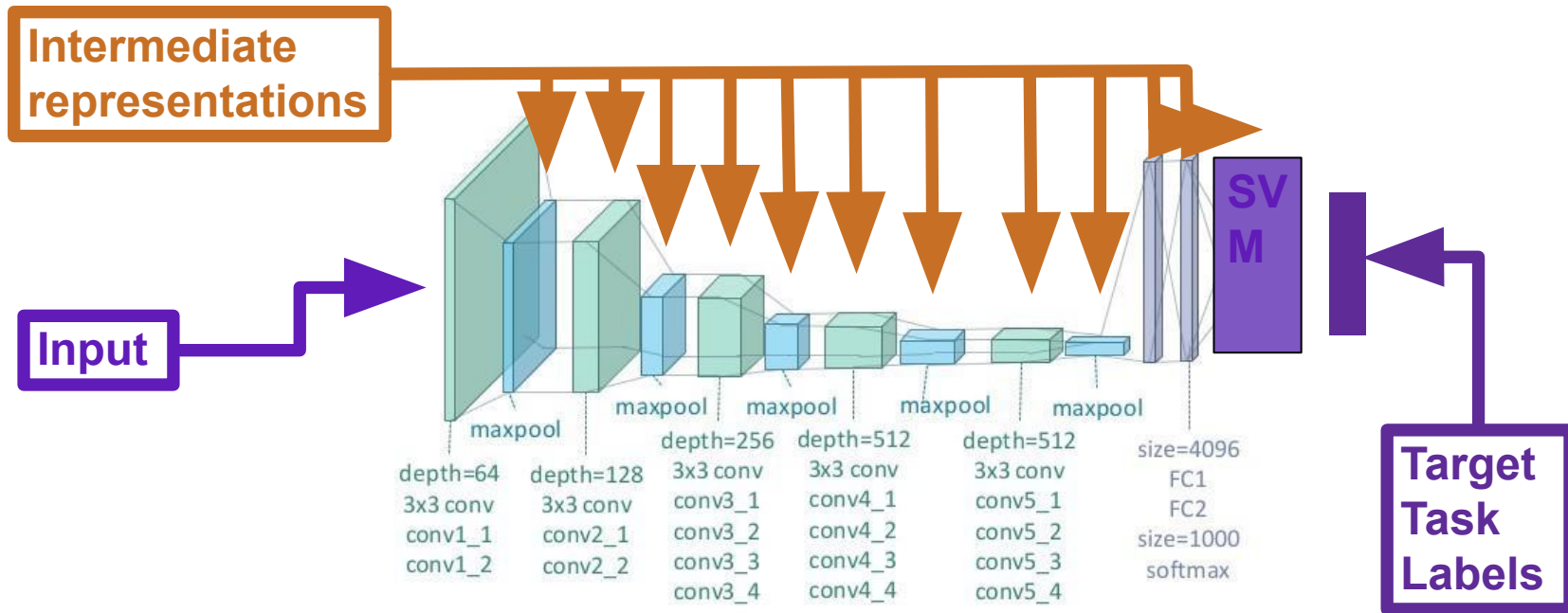
What can be saved?



Feature extraction

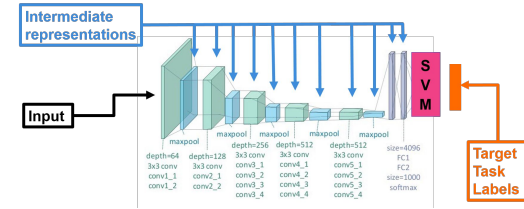


Feature extraction



Reuse All

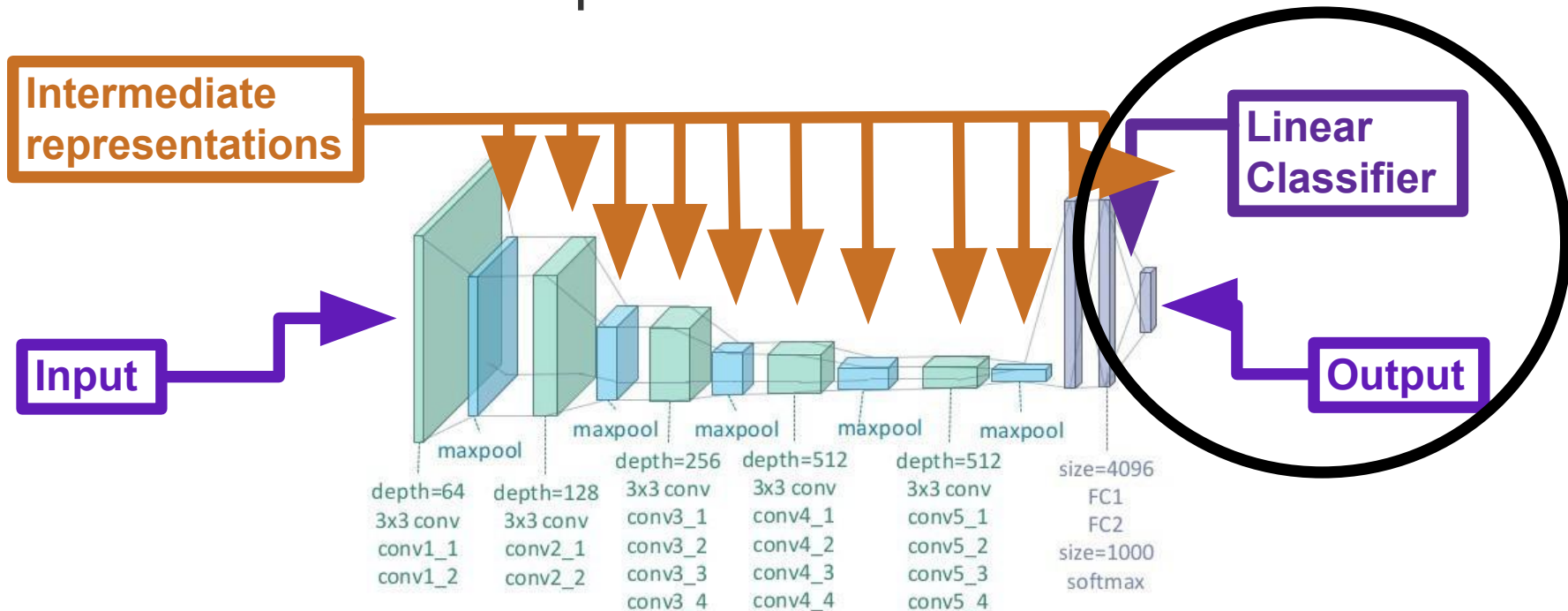
- ❖ DNN last layer features + SVM
 - *Feature extraction*
 - Very similar task and same domain



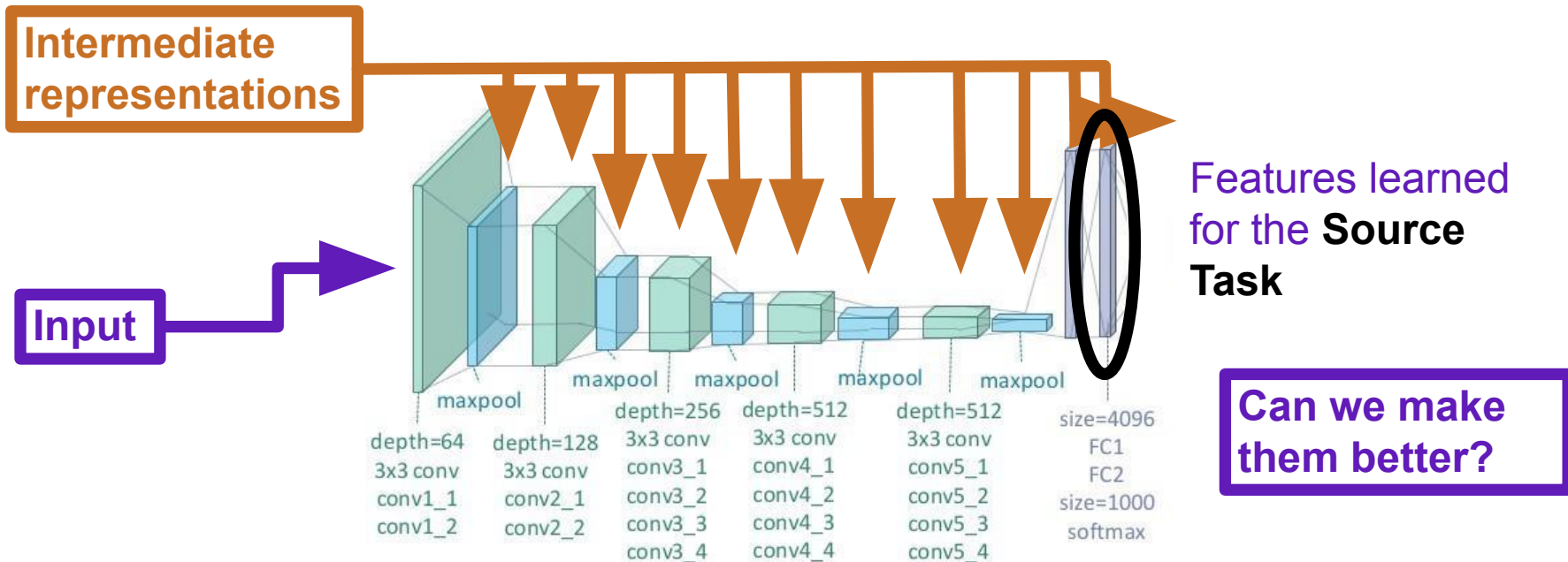
Fine tuning

What if the tasks are quite different?

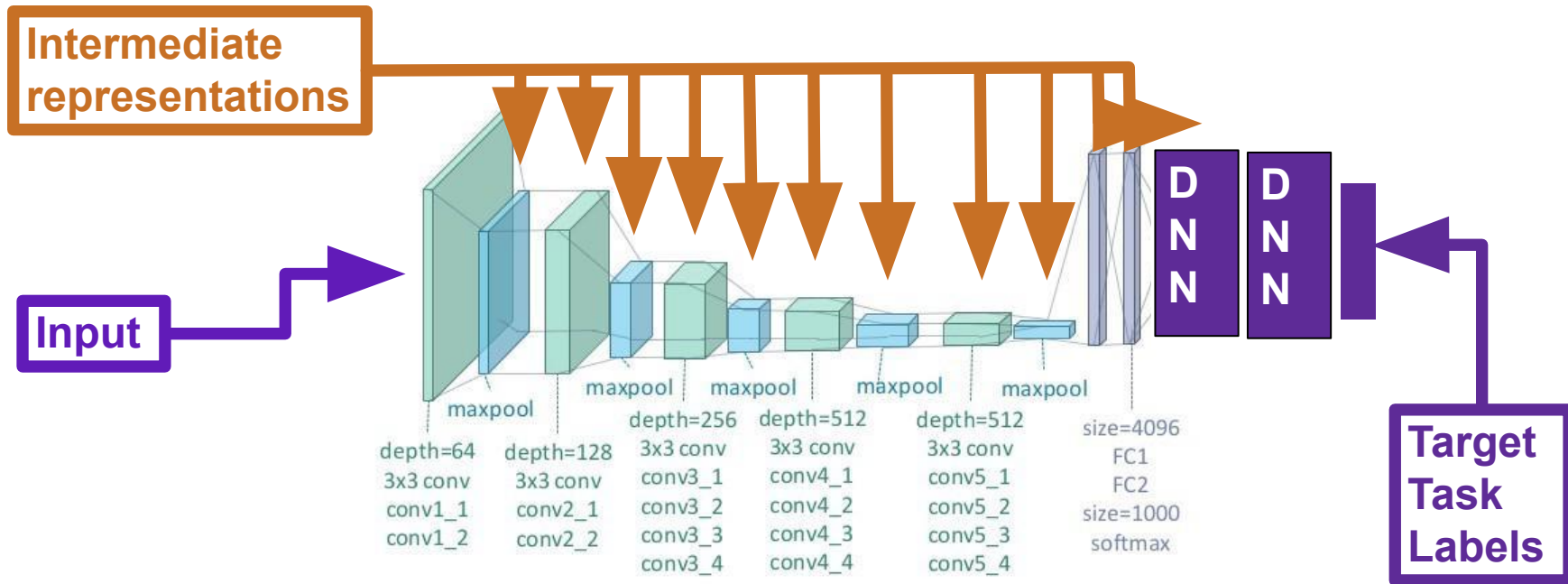
Source Task



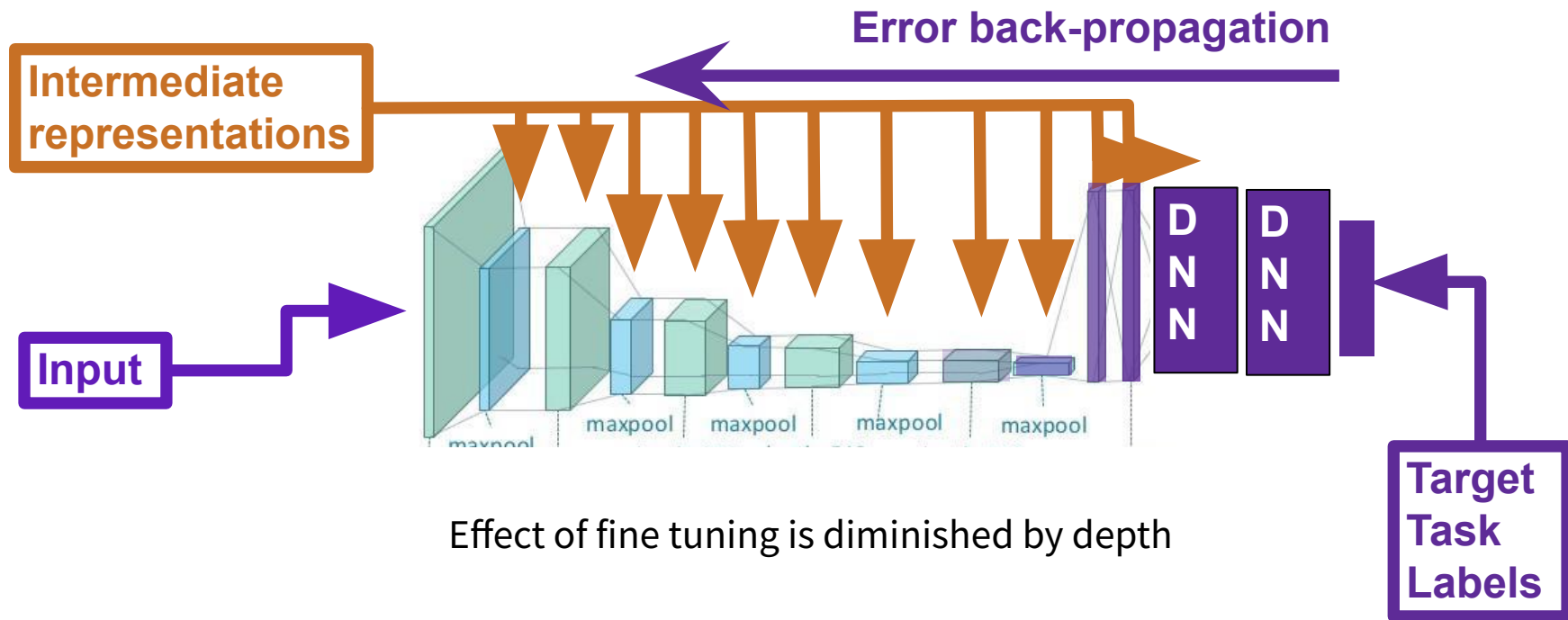
Fine tuning



Fine tuning

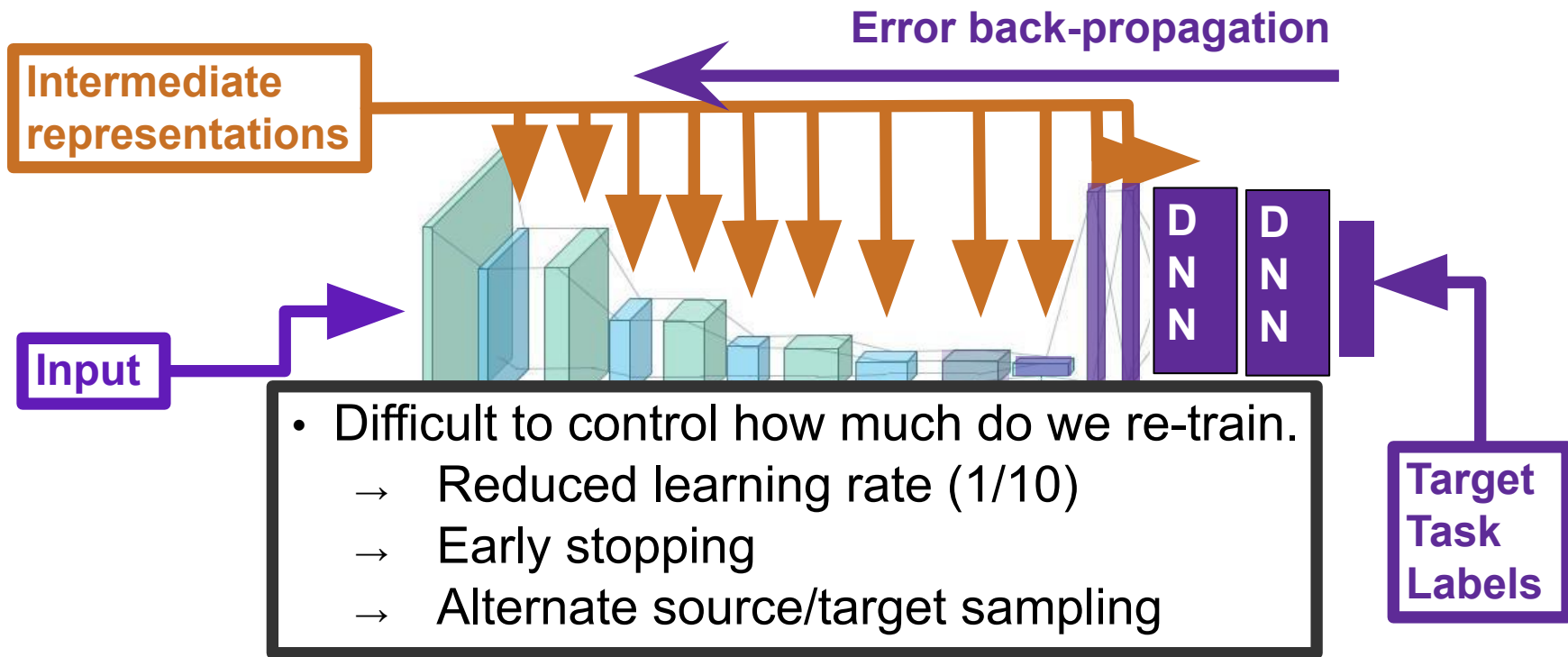


Fine tuning



Effect of fine tuning is diminished by depth

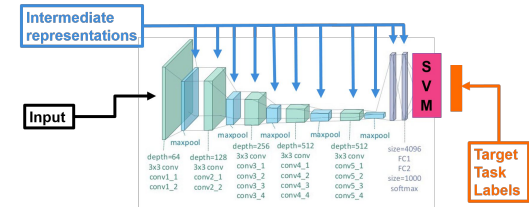
Fine tuning



Retrain

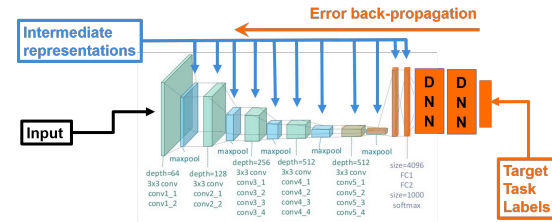
- ◆ DNN last layer features + SVM

- Feature extraction
- Very similar task and same domain

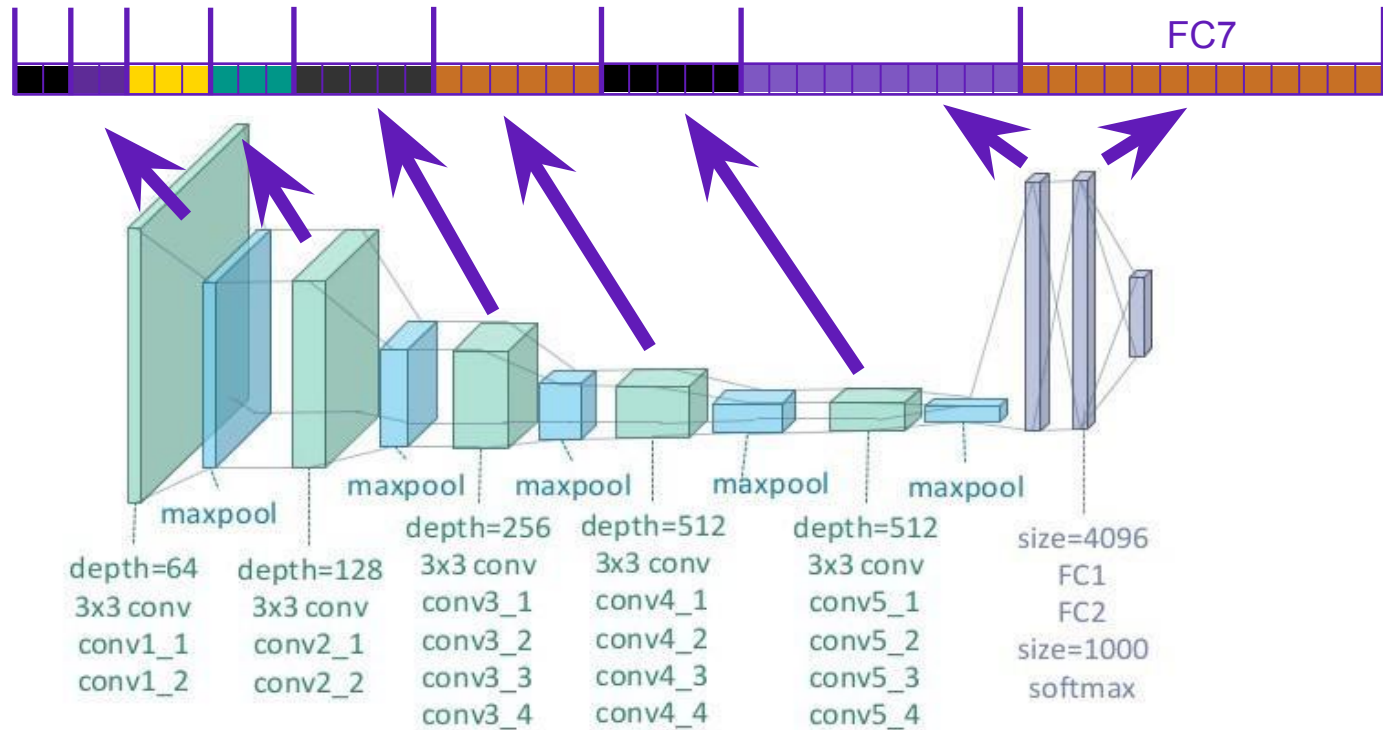


- ❖ Train one or several NN layers + pre-trained layers

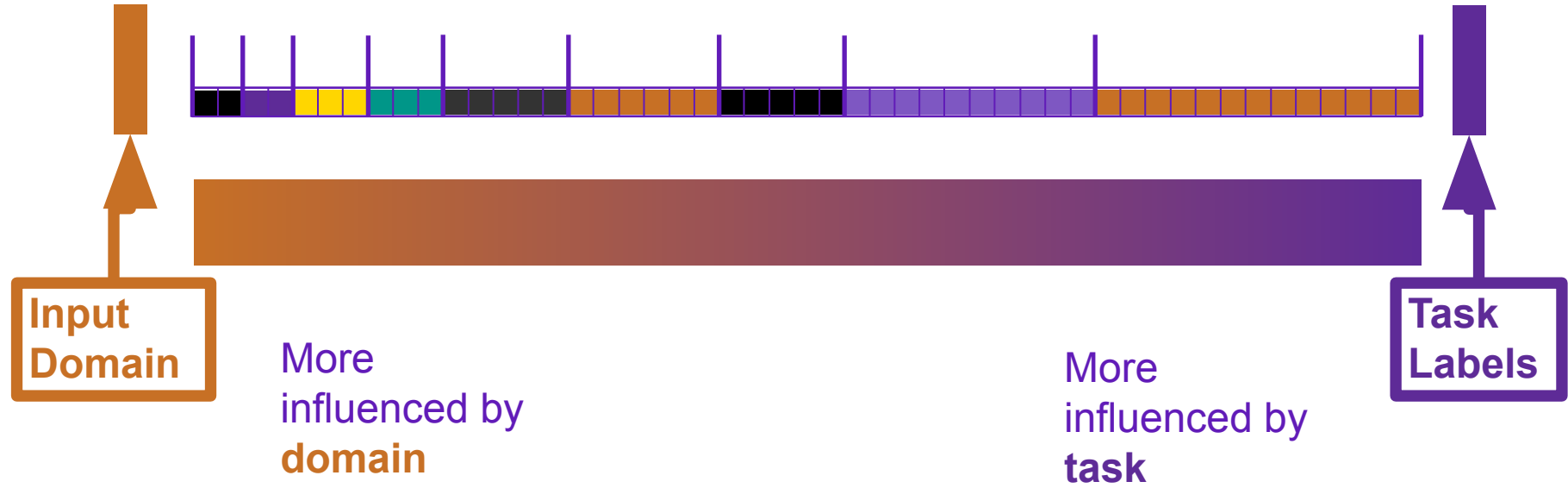
- Fine tuning
- Sort of similar task and same domain
- Data volume



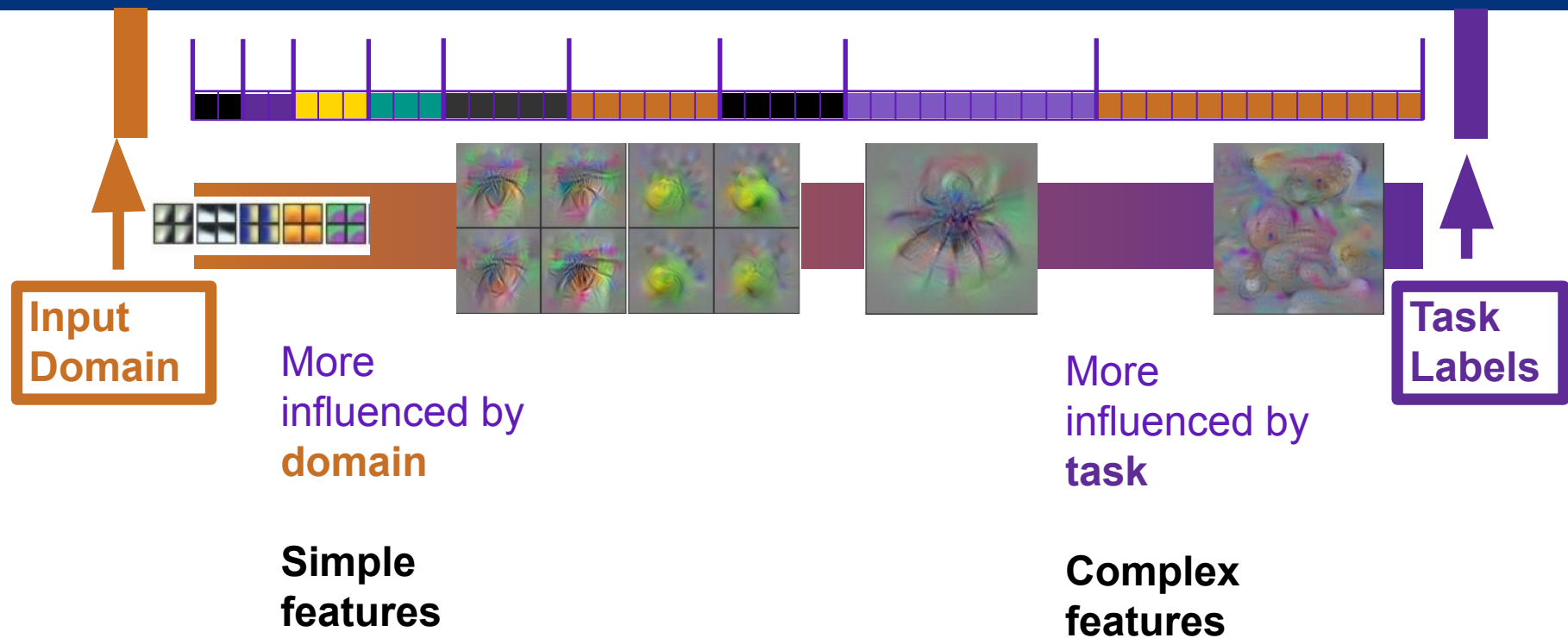
Knowledge inside DNN



Knowledge inside DNN



Knowledge inside DNN



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



Fine Tuning

To improve, to remember, to forget

The choices in fine tuning

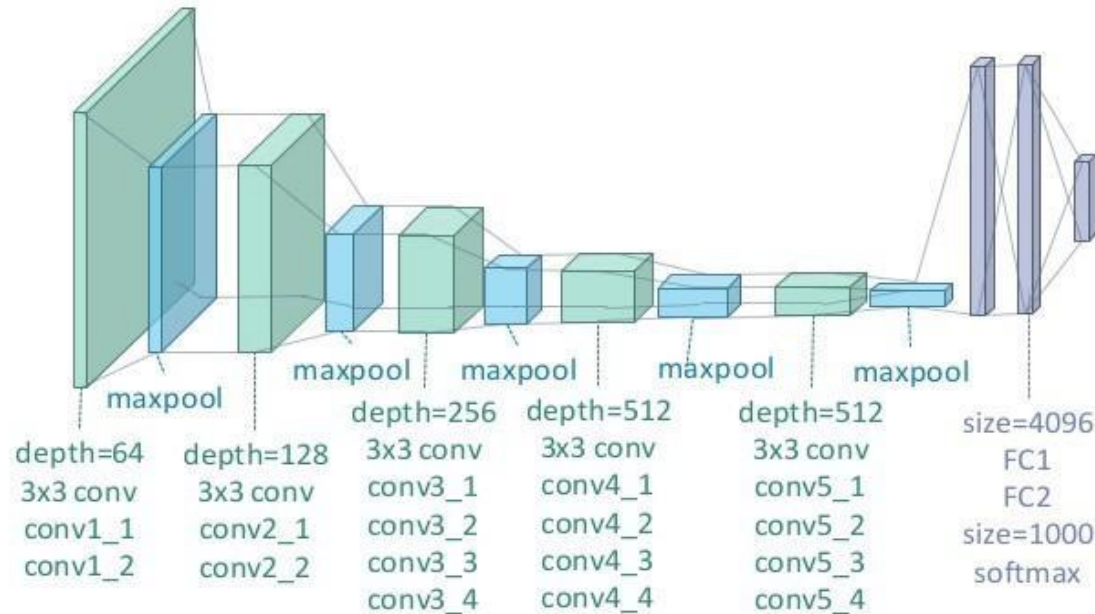
- Reuse and **freeze**
 - Use source task status
 - *"Its good as it is"*
- Reuse and **fine tune**
 - Start from source task status, adjust with target task
 - *"It's a good starting point"*
- Train **from scratch**
 - Reinitialize weights randomly, train with target task only
 - *"It's pretty much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

The order of fine tuning

Freeze

Fine tune

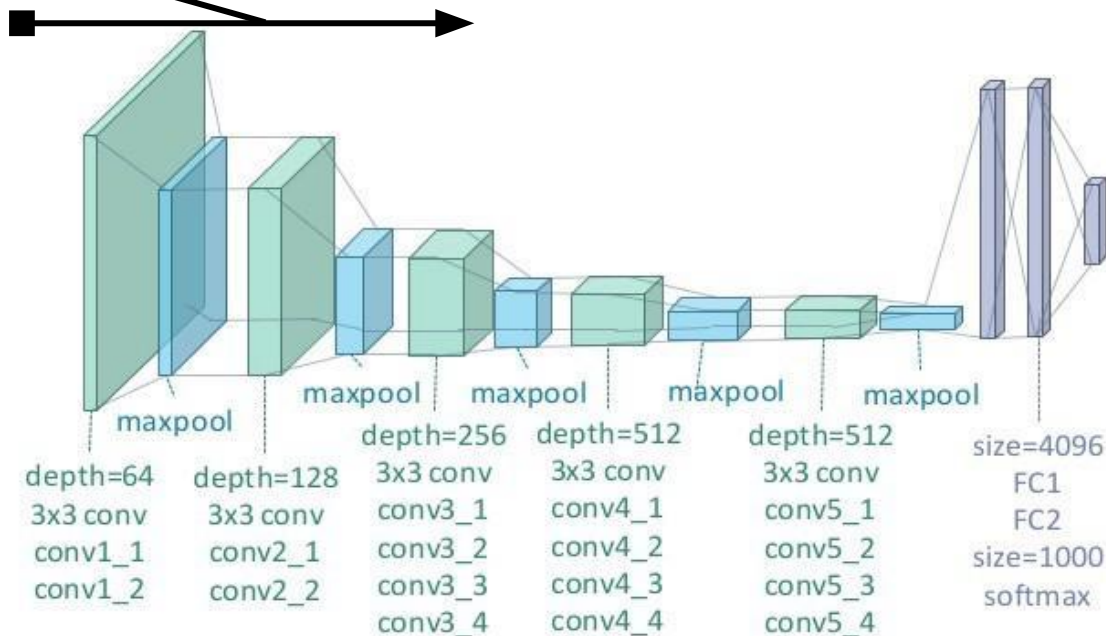
Random



The order of fine tuning

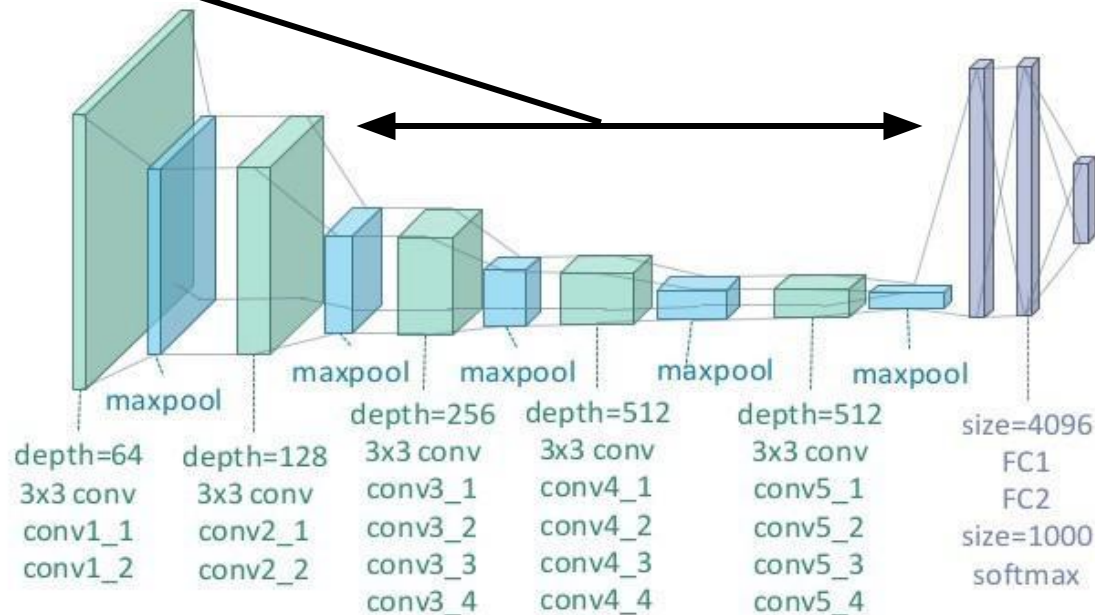
Freeze

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

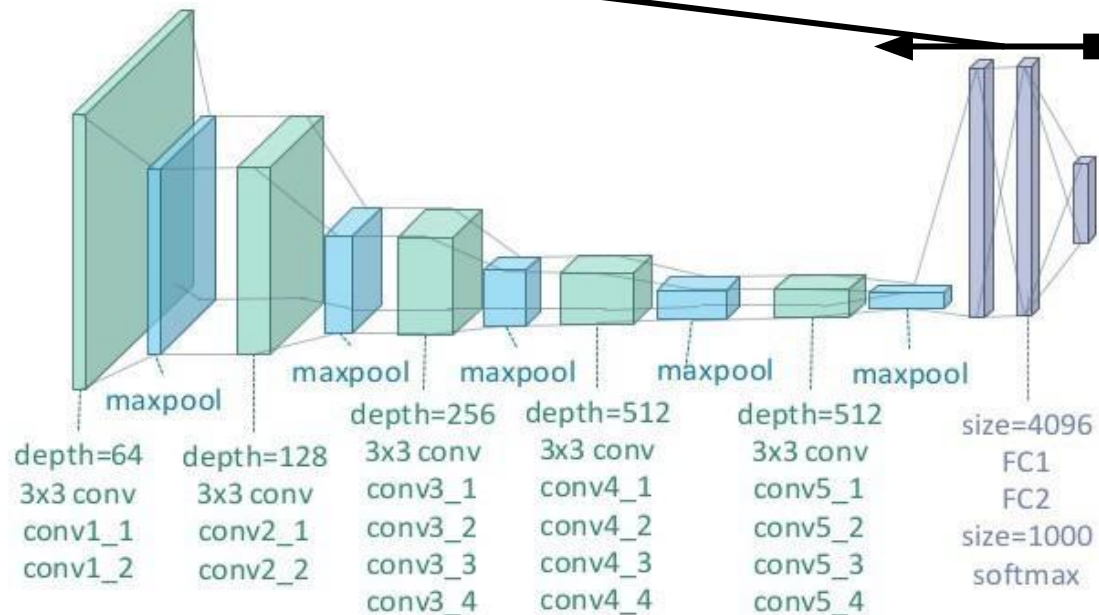


The order of fine tuning

Random

Layers which are pretty dissimilar

Increase with source-target dissimilarity & data volume



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



Feature Extraction

To improve, to remember, to forget

Factors deep representations quality

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

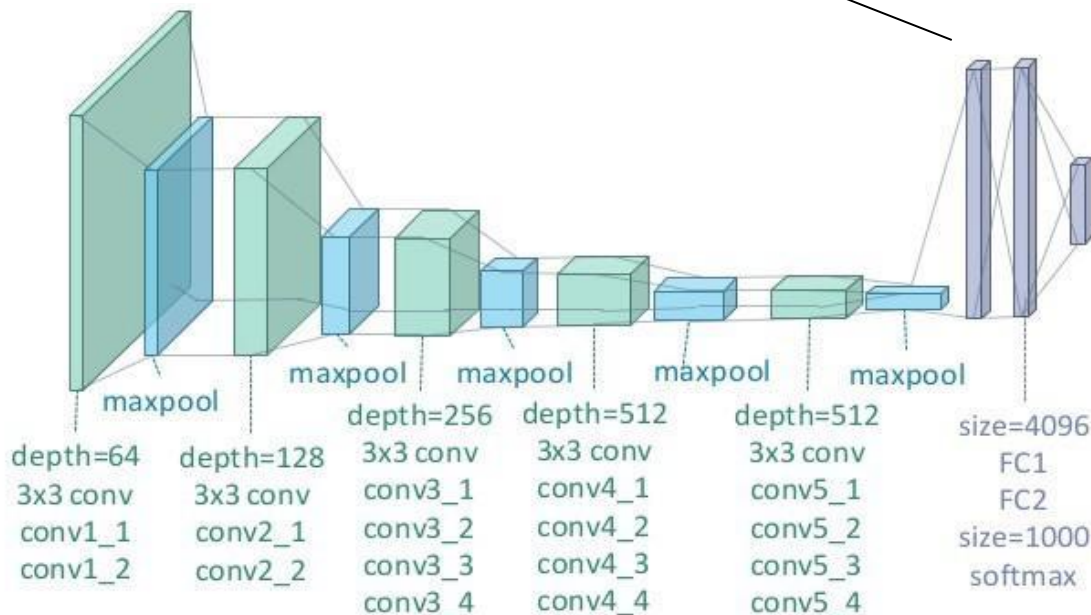
Factors deep representations quality

- 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

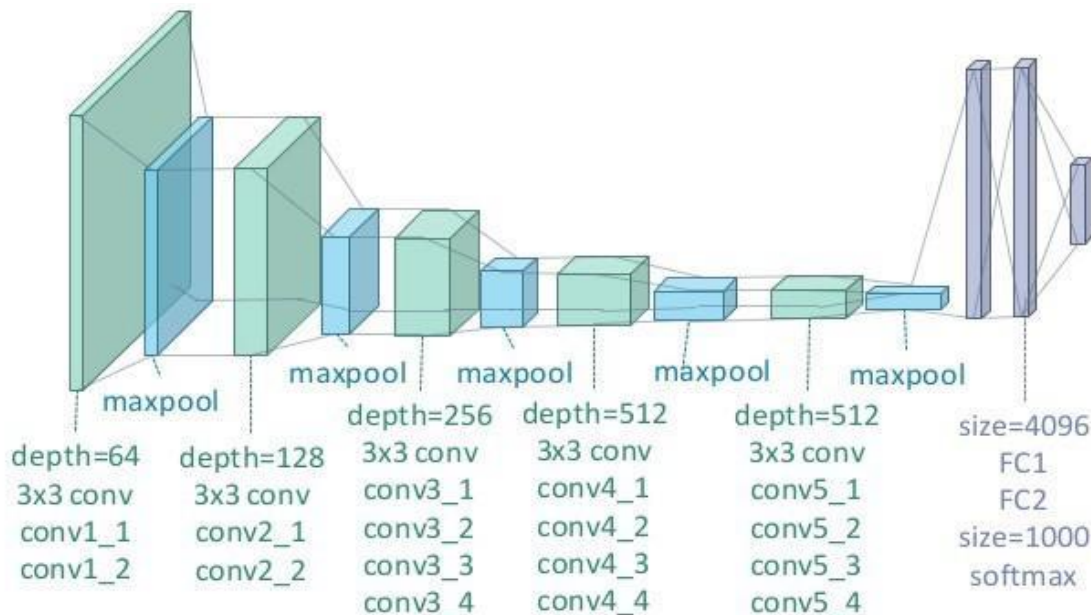
Which layers to use?

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



Which layers to use?

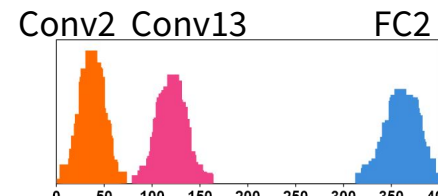
- If source & target task are NOT very similar
broaden the scope



Feature extraction normalization

- When doing feature extraction for a regular classifier (e.g., SVM) each feature is assumed to be i.i.d. (not even close!)
- Beware of size
 - FC layers have lots of activations
 - Conv layers activations are spatially dependent
- Beware of scale
 - Different layers activate with different strength
- Default solution: L2-norm (by layer)
 - Does not fix scale (careful if mixing layers!)

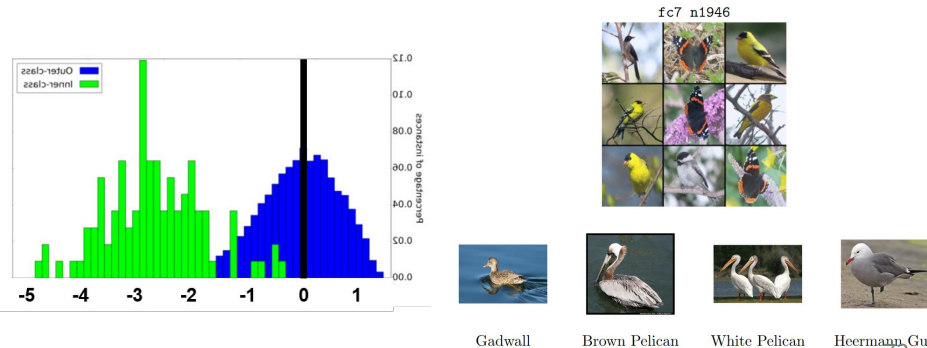
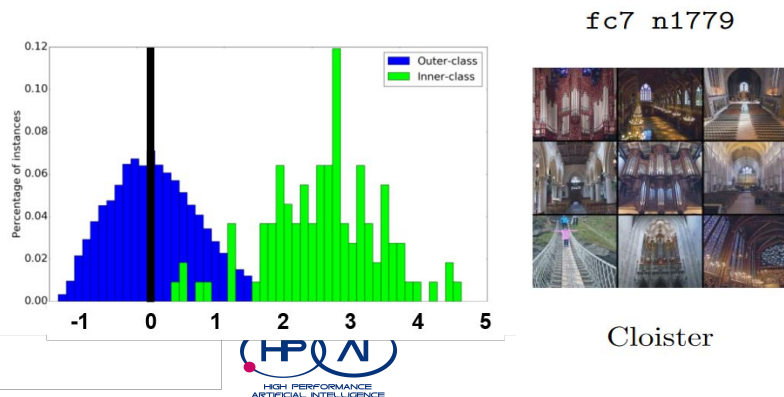
VGG16	Convs	FCs
# Layers:	14	2
Activations:	33%	66%



Advanced feature normalization

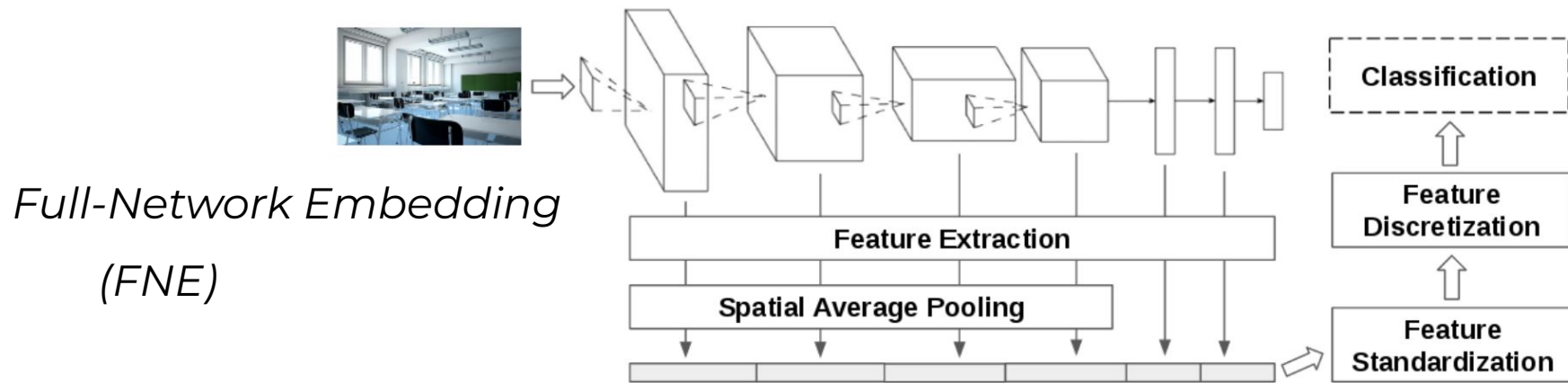
- Normalizing features considering the target
 - Feature standardization (vertically instead of horizontally)
- For each feature..
 - Compute mean and std dev. on *target* training set
 - Normalize feature-wise to zero mean, one std dev.
 - Features are adapted to *target domain*

Best for
multi-layer
feature
extraction



Advanced feature normalization

- Dimensionality of extracted features is an issue (12K in VGG16)
- Removing complexity without losing expressivity
 - Discretizing the space (-1,0,1)



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 \pm 0.6	61.4 \pm 0.2	69.6	70.8 \pm 6.6
Baseline fc7	81.7	63.2	87.0	89.6	79.3	89.7 \pm 0.3	59.1 \pm 0.6	69.0	68.9 \pm 6.8
Full-network	83.6	65.5	93.3	89.2	78.8	91.4 \pm 0.6	67.0 \pm 0.7	73.0	74.1 \pm 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	✓	✓	✓	✓	✓	✓	✓	✗	-

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 f c6	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 f c7	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

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

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

Dario Garcia-Gasulla (BSC)
dario.garcia@bsc.es

