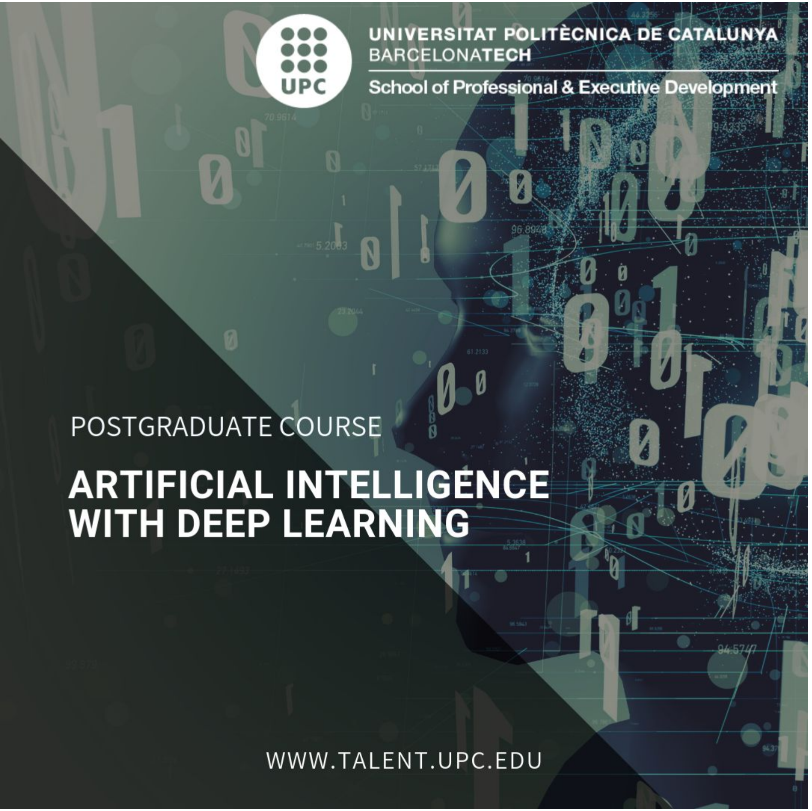



#DLUPC

Day 4 Lab 4

# Transfer Learning

Xavier Giro-i-Nieto
xavier.giro@upc.edu

Associate Professor
Universitat Politècnica de Catalunya
Technical University of Catalonia

Daniel Fojo
dani.fojo@gmail.com

Research Engineer
Disney Research

# The Concept

# Acknowledgements

Kevin McGuinness
kevin.mcguinness@dcu.ie

Research Fellow
Insight Centre for Data Analytics
Dublin City University

Ramon Morros
ramon.morros@upc.edu

Associate Professor
ETSETB TelecomBCN
Universitat Politècnica de Catalunya

# Videolectures


[Kevin McGuinness](#) ([UPC DLCV 2016](#))

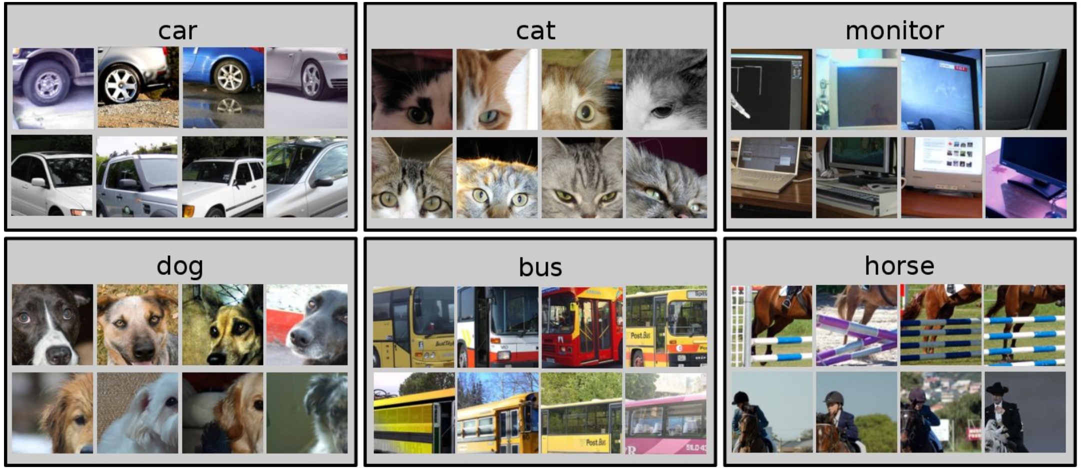

[Ramon Morros](#) ([UPC DLAI 2018](#))

# Transfer learning: the motivation

- In many cases, not enough training data is available to estimate the large amount of parameters required by a deep neural network.

# Example: PASCAL VOC 2007

- Standard classification benchmark, 20 classes, ~10K images, 50% train, 50% test
- Deep networks can have many parameters (e.g. 60M in Alexnet)
- Direct training (from scratch) using only 5K training images can be problematic. Model overfits.
- How can we use deep networks in this setting?

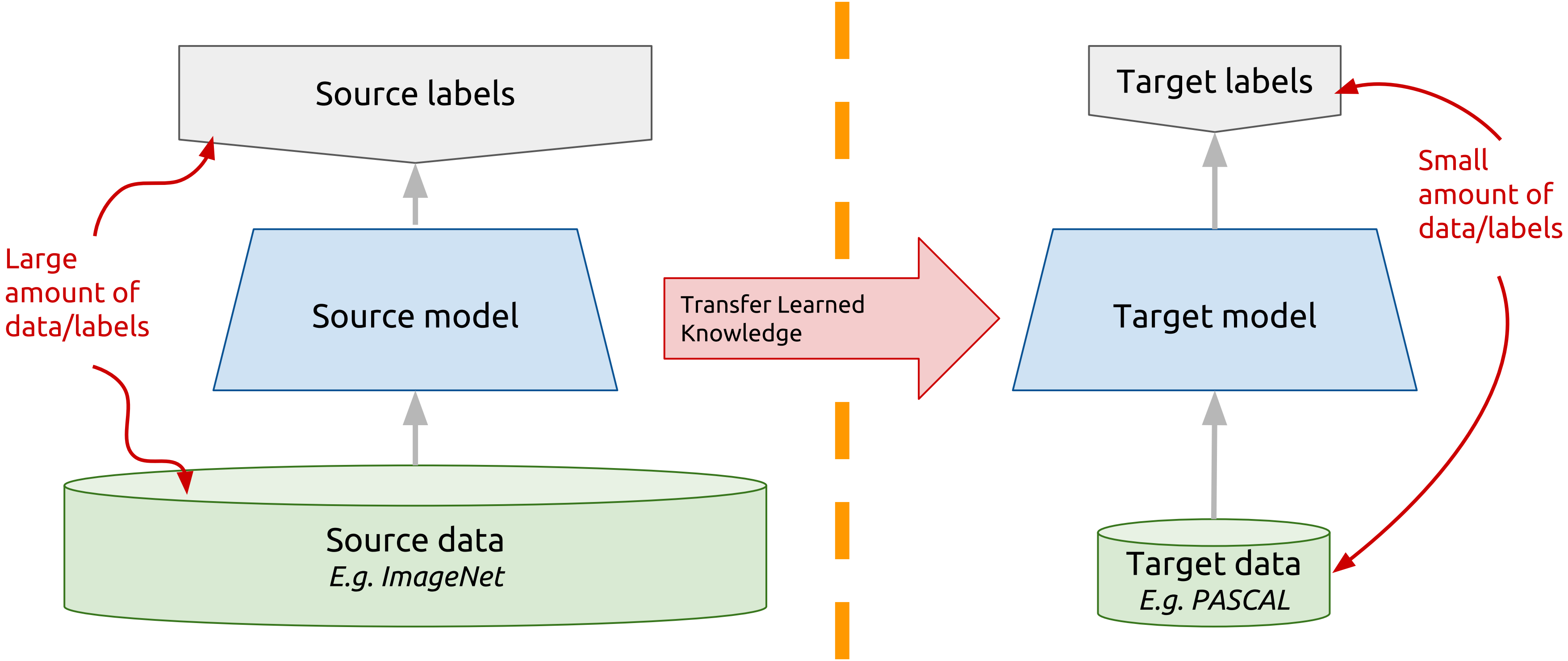


# Transfer learning: the solution

Instead of training a deep network from scratch for your task:

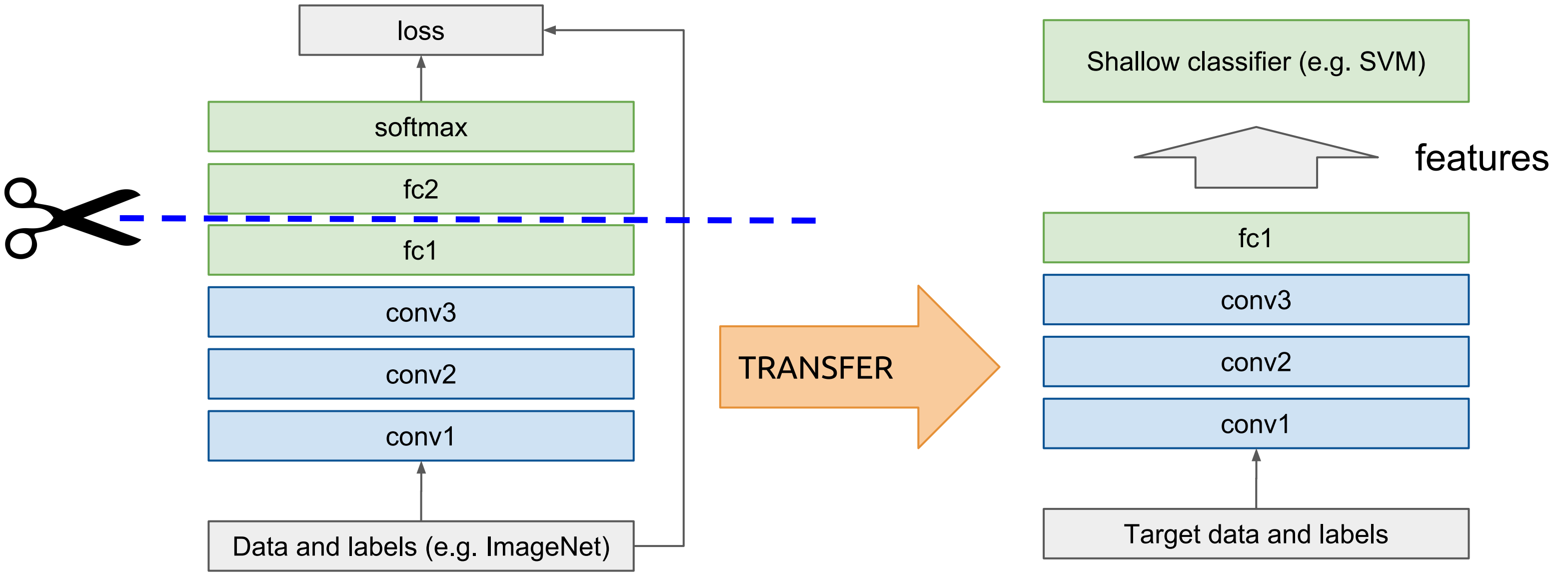
- Take a network trained on a different domain for a different **source task**
- Adapt it for your domain and your **target task**

# Transfer learning: idea



# “Off-the-shelf”

Idea: use outputs of one or more layers of a network trained on a differen task as generic feature detectors. Train a new shallow model on these features.



# Off-the-shelf features

Works surprisingly well in practice!

Surpassed or on par with state-of-the-art in several tasks in 2014

**Image classification:**

- PASCAL VOC 2007
- Oxford flowers
- CUB Bird dataset
- MIT indoors

**Image retrieval:**

- Paris 6k
- Holidays
- UKBench

| Method | mean Accuracy |
|---|---|
| HSV [27] | 43.0 |
| SIFT internal [27] | 55.1 |
| SIFT boundary [27] | 32.0 |
| HOG [27] | 49.6 |
| HSV+SIFTi+SIFTb+HOG(MKL) [27] | 72.8 |
| BOW(4000) [14] | 65.5 |
| SPM(4000) [14] | 67.4 |
| FLH(100) [14] | 72.7 |
| BiCos seg [7] | 79.4 |
| Dense HOG+Coding+Pooling[2] w/o seg | 76.7 |
| Seg+Dense HOG+Coding+Pooling[2] | 80.7 |
| CNN-SVM w/o seg | 74.7 |
| CNNaug-SVM w/o seg | **86.8** |

Oxford 102 flowers dataset

Razavian et al, **CNN Features off-the-shelf: an Astounding Baseline for Recognition,** CVPRW 2014 http://arxiv.org/abs/1403.6382

Can we do better than off the shelf features?

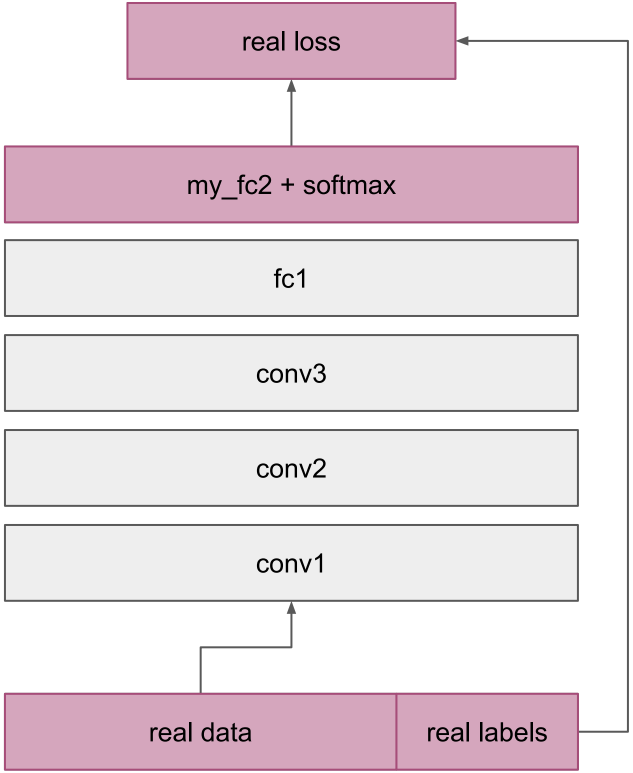
# Fine-tuning: supervised task adaptation

Train deep net on “nearby” task for which it is easy to get labels using standard backprop

- E.g. ImageNet classification
- Pseudo classes from augmented data
- Slow feature learning, ego-motion

Cut off top layer(s) of network and replace with supervised objective for target domain

**Fine-tune** network using backprop with labels for target domain until validation loss starts to increase



# Freeze or fine-tune?

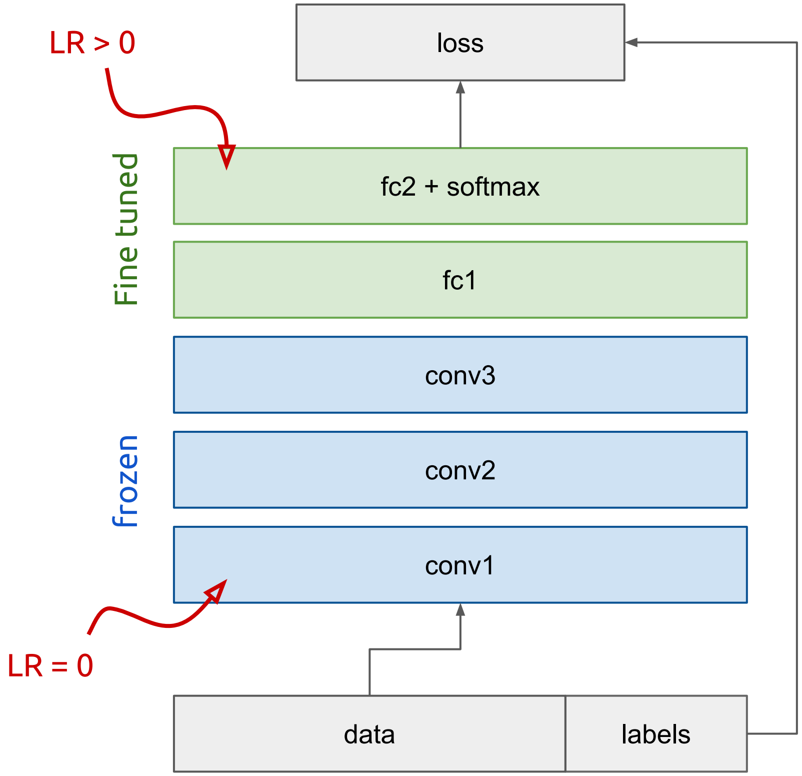
Bottom *n* layers can be frozen or fine tuned.

- **Frozen**: not updated during backprop
- **Fine-tuned**: updated during backprop

Which to do depends on target task:

- **Freeze**: target task labels are scarce, and we want to avoid overfitting
- **Fine-tune**: target task labels are more plentiful

In general, we can set learning rates to be different for each layer to find a tradeoff between freezing and fine tuning



# How are they related?

What do unsupervised learning, semi-supervised learning, and transfer learning have in common?

We often do not have complete access to the *y*-value that we care about.

Use a **surrogate objective** when *y*-value unavailable:

- Loss you do not care directly about.
- You expect that improving this loss will help with what you do care about.

**Transfer learning** uses different dataset in which there are lots of *y*-values to create a surrogate objective.

**Unsupervised** tries to formulate surrogate objectives without any manual labels.

# Summary

- Possible to train very large models on small data by using transfer learning and domain adaptation
- Off the shelf features work very well in various domains and tasks
- Lower layers of network contain very generic features, higher layers more task specific features
- Supervised domain adaptation via fine tuning almost always improves performance
- Possible to do unsupervised domain adaptation by matching feature distributions

# Additional resources


- Lluis Castrejon, "Domain adaptation and zero-shot learning". University of Toronto 2016.
- Hoffman, J., Guadarrama, S., Tzeng, E. S., Hu, R., Donahue, J., Girshick, R., ... & Saenko, K. (2014). LSDA: Large scale detection through adaptation. NIPS 2014. (Slides by Xavier Giró-i-Nieto)
- Yosinski, Jason, Jeff Clune, Yoshua Bengio, and Hod Lipson. "How transferable are features in deep neural networks?." In Advances in Neural Information Processing Systems, pp. 3320-3328. 2014.
- Shao, Ling, Fan Zhu, and Xuelong Li. "Transfer learning for visual categorization: A survey." Neural Networks and Learning Systems, IEEE Transactions on 26, no. 5 (2015): 1019-1034.
- Chen, Tianqi, Ian Goodfellow, and Jonathon Shlens. "Net2Net: Accelerating Learning via Knowledge Transfer." ICLR 2016. [code] [Notes by Hugo Larrochelle
- Gani, Yaroslav, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. "Domain-Adversarial Training of Neural Networks." arXiv preprint arXiv:1505.07818 (2015).

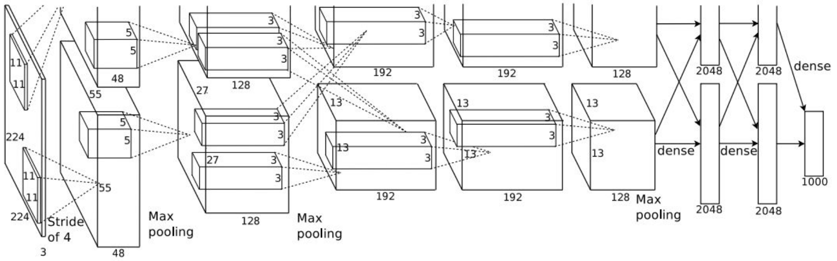
# Data Augmentation


Eva Mohedano

Research Fellow
Insight Centre for Data Analytics
Dublin City University

# Ways to reduce overfitting

- **Reduce network capacity**

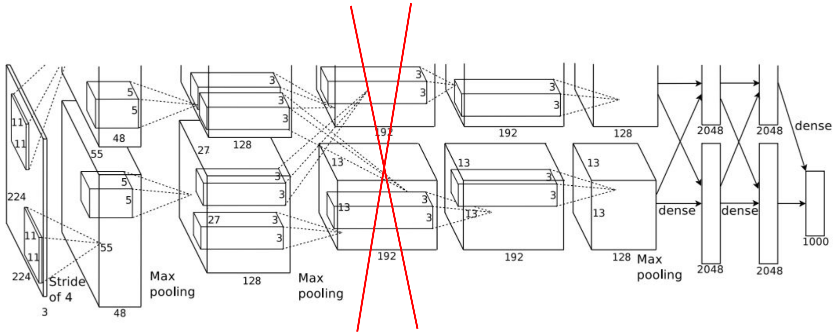



- Dropout

- Data augmentation

# Ways to reduce overfitting

- **Reduce network capacity**
- Dropout
- Data augmentation



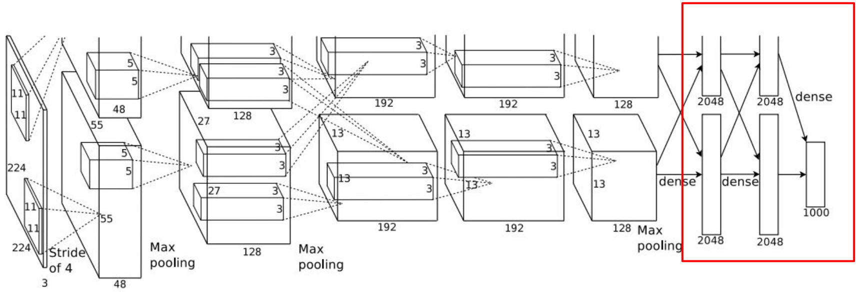

1% of total parameters (884K). Decrease in performance

# Ways to reduce overfitting

- **Reduce network capacity**

- Dropout

- Data augmentation



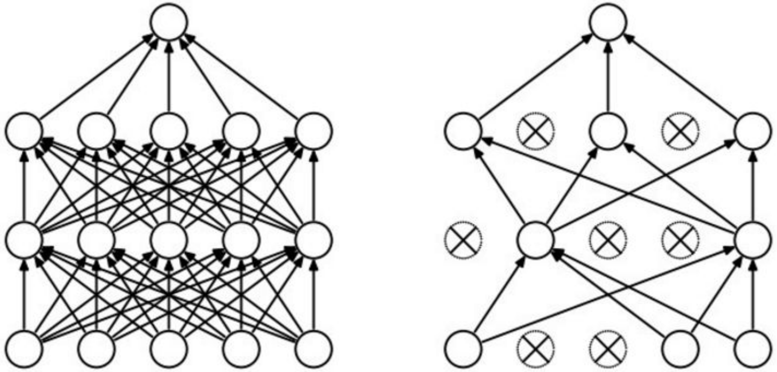

37M, 16M, 4M parametes!! (fc6,fc7,fc8)

# Ways to reduce overfitting

- Reduce network capacity

- **Dropout**

- Data augmentation



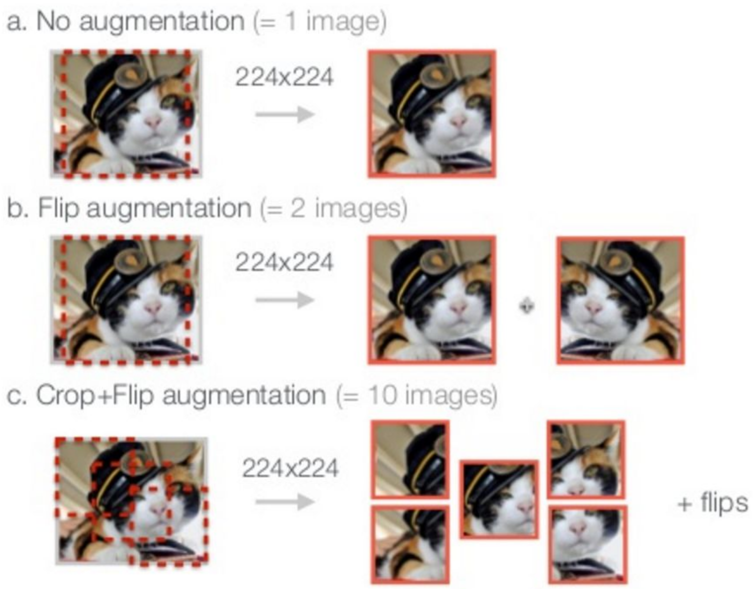
(a) Standard Neural Net (b) After applying dropout.

Every forward pass, network slightly different.
Reduce co-adaptation between neurons
More robust features

More interations for convergence

# Ways to reduce overfitting

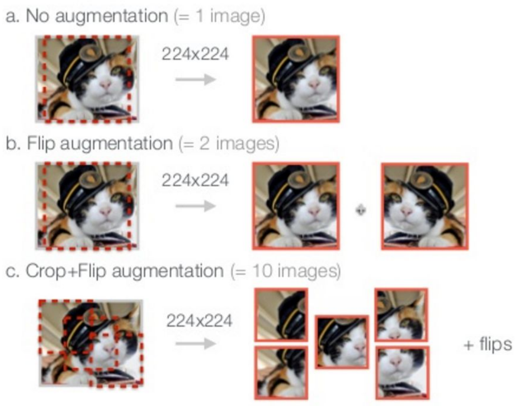
- Reduce network capacity
- Dropout
- **Data augmentation**



# Data Augmentation

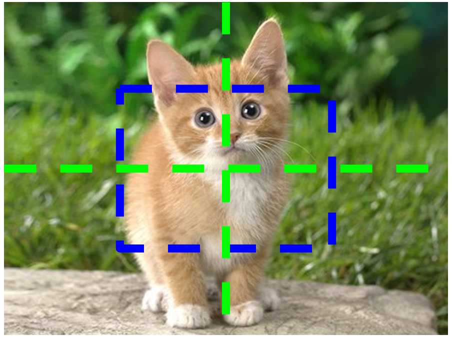
During training, modify the input image (Krizhevsky A., 2012)

- Random crops on the original image
- Translations
- Horizontal reflections
- Increases size of training x2048
- On-the-fly augmentation



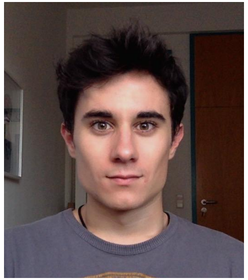

During testing

- Average prediction of image augmented by the four corner patches and the center patch + flipped image. (10 augmentations of the image)



# The Lab

# Acknowledgements




Víctor Campos
victor.campos@bsc.es

PhD Candidate

Barcelona Supercomputing Center



Míriam Bellver
miriam.bellver@bsc.edu

PhD Candidate
Barcelona Supercomputing Center



Amaia Salvador
amaia.salvador@upc.edu

PhD Candidate
Universitat Politècnica de Catalunya



Santiago Pascual de la Puente
santi.pascual@upc.edu

PhD Candidate
Universitat Politecnica de Catalunya
Technical University of Catalonia

# Lab created by

Míriam Bellver
miriam.bellver@bsc.edu

PhD Candidate
Barcelona Supercomputing Center

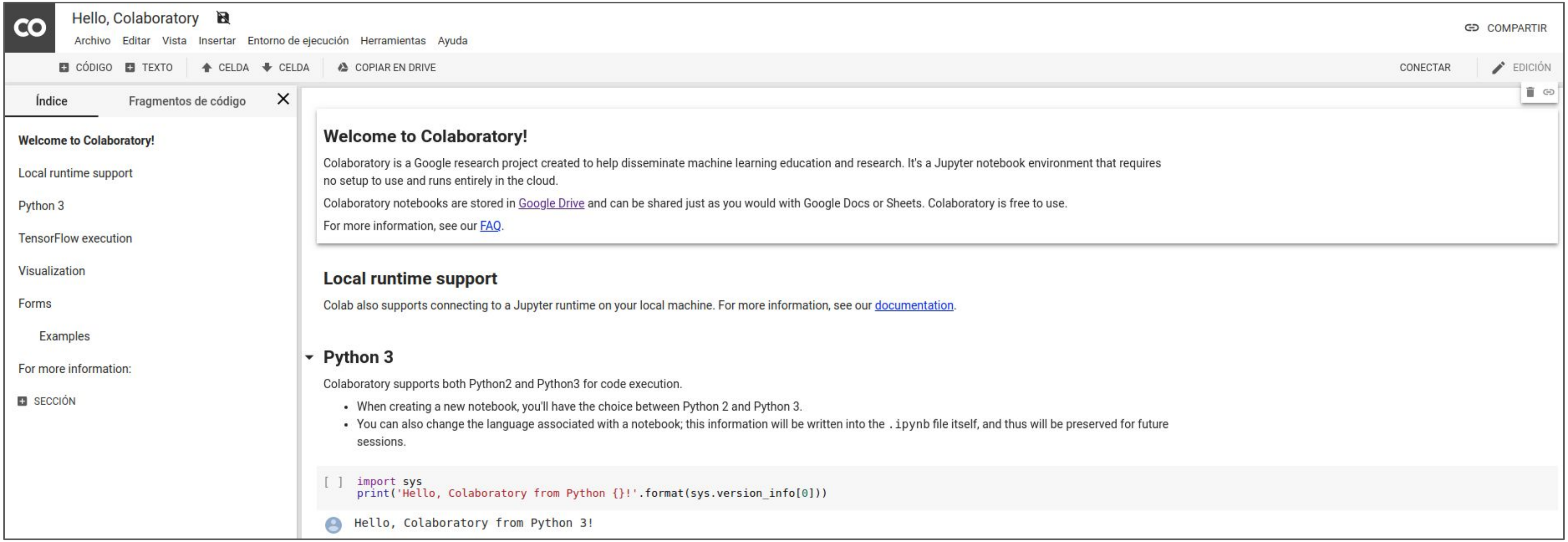
# Today’s objectives

- Tricks when working with small dataset
- Training network from scratch with and without data augmentation
- Using pretrained network just to extract features
- Using pretrained network and finetune it

# Cats vs Dataset

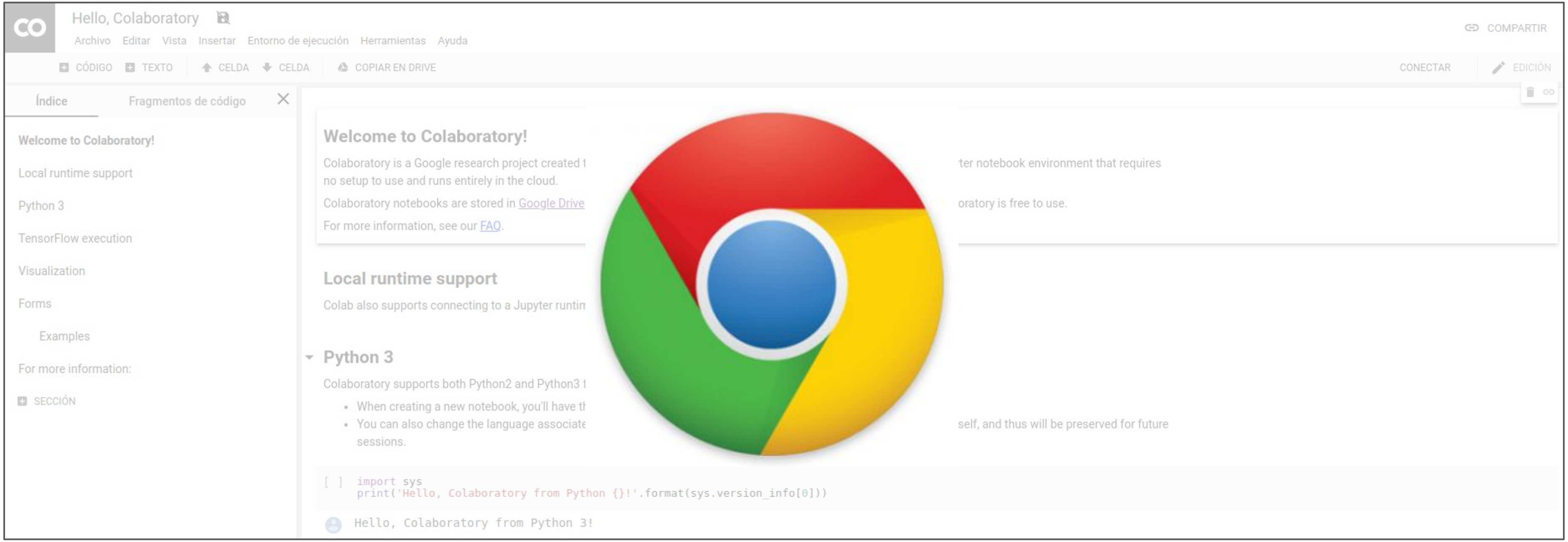
- Cats vs. Dogs dataset
  - Binary Classification between cats and dogs
  - Does **not** come packaged with Keras

# Google Colab




https://colab.research.google.com/

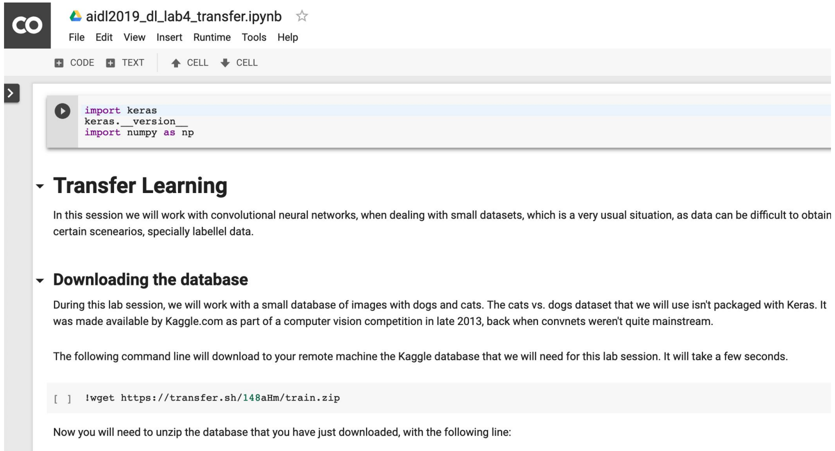
# Google Colab

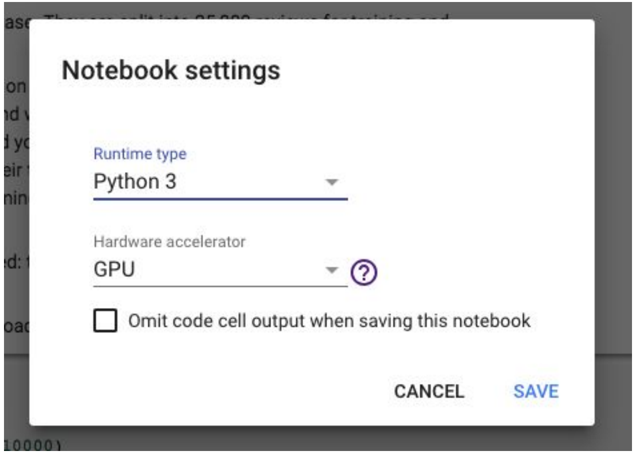



https://colab.research.google.com/

# Google Colab

1. Login in Colab with a Google account: yours or aidlupc2019@gmail.com (talentcenter)
2. Open the notebook of this lab session
3. Copy this notebook to your Drive to be able to run it (or open in draft mode if using aidlupc2019@gmail.com )
4. Change runtime type to work with GPU! Your trainings will be much faster :)



# Final Questions