Big Data Computing course
Teached by Prof. Gabriele Tolomei
Dipartimento di Informatica, I3S, Università di Roma Sapienza
AY 2020/21

SAPIENZA
UNIVERSITÀ DI ROMA

Course’s project presentation

Federico Fontana, 1744946, fontana.1744946@studenti.uniroma1.it
Romeo Lanzino, 1753403, lanzino.1753403@studenti.uniroma1.it

Sapienza Università di Roma

Search engine based on Distributed Semi-Supervised (Self-Training) Image Classification on STL-10

A **transfer learning** approach

Resources

- notebook of this project
  - editable version (if you have the permissions)
  - published version
- demo notebook of this project
  - editable version (if you have the permissions)
  - published version
- repo on GitHub with notebooks and presentation
- preprocessed dataset's folder on Google Drive

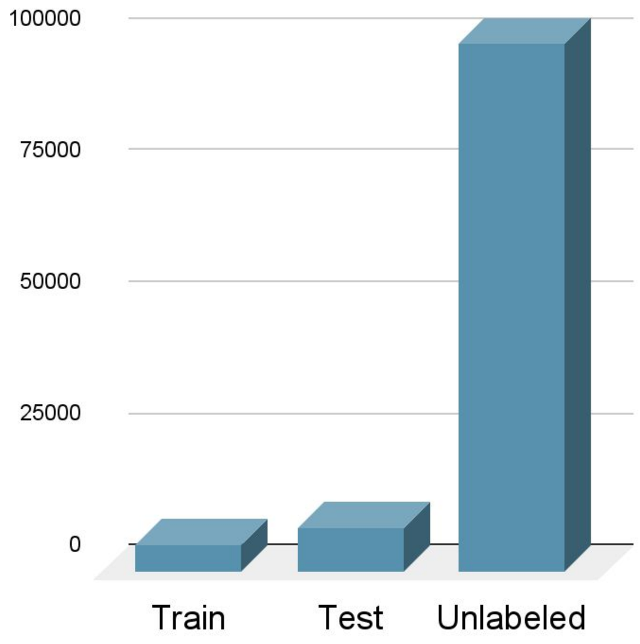
SAPIENZA
UNIVERSITÀ DI ROMA

Dataset's overview

STL10* is an image recognition dataset
with a corpus composed of:

- 100K unlabeled images
- 5K labeled training images
- 8K labeled test images

covering 10 different classes



Big-Data-Ness

$$SizeImage * BytesForPixel * n.OfImages = MemoryOnRAM$$
$$(96 * 96 * 3) * 4 * 113000 \approx 11.6GB$$

*Coates, Lee, Ng, 2011, An Analysis of Single Layer Networks in Unsupervised Feature Learning

SAPIENZA
UNIVERSITÀ DI ROMA

What are conventional small dataset solutions?*

- using a non-deep algorithm
- using a unsupervised algorithm on the unlabeled part
- using synthetic data 🤖
- finding new data 🔍

*Lateh et al., 2017, J. Phys.: Conf. Ser. 892 012016

How complex the task is?

Semi-supervised learning is an approach to machine learning that combines a **small amount** of **labeled data** with a **large amount** of **unlabeled data** during training.

The SSL is not a new topic but recently **(from 2018)*** the attention has growth: these approaches take care of problems where we don't have such big labeled dataset

*Padmanabha,Viswanath, Reddy, 2018, Semi-supervised learning: a brief review

How much effort would it take to get into a production system?

We have developed the system in a Python Notebook:

- we’ve chosen to give **privilege to readability** by reducing the usage of lambda functions only when indispensable
- the notebooks are divided into macro-**sections**
- the code has inline **comments** and before each cell there is its textual description
- at the beginning of the notebook we define multiple **environmental variables** that change notebook’s workflow
- the code is **available on git**

Thanks to these precautions, it’s straightforward to run the models using technologies like Amazon’s Sagemaker or EMR and setting environment variables to get the desired functioning

SAPIENZA
UNIVERSITÀ DI ROMA

Self-training

The approach we chosen is the **self-training***:

- Pseudo-label the unlabeled samples with a classifier trained with the labeled samples
- Train a big-classifier with pseudo-labeled and labeled samples
- Evaluate the final model with the test samples

*Wei, Shen, Chen, Ma, 2021, Theoretical Analysis of Self-Training with Deep Networks on Unlabeled Data

Challenging task

The impact of self-training is similar to that of entropy minimization; in both cases, the network is forced to output more confident predictions

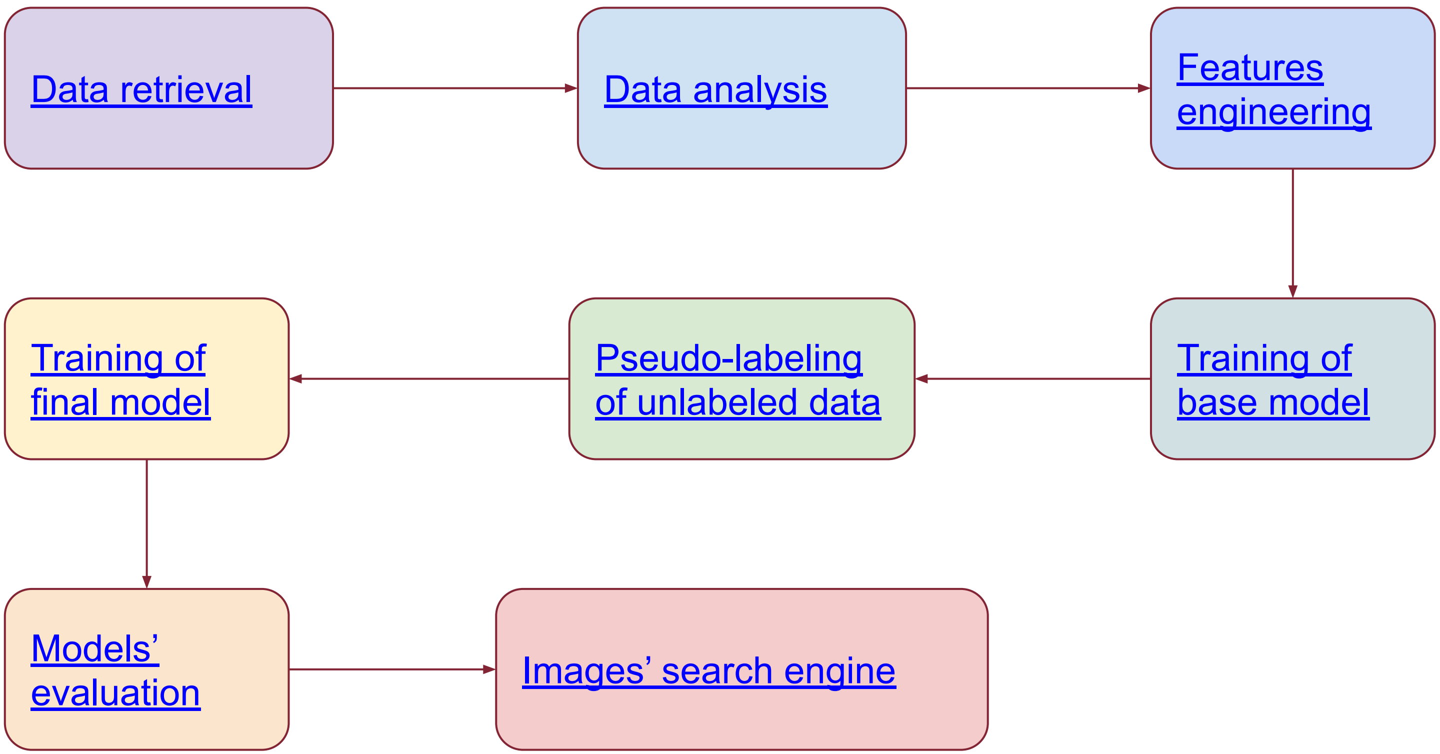
The main downside of such methods is that the model is unable to correct its own mistakes, and any biased and wrong classifications can be quickly amplified resulting in confident but erroneous proxy labels on the unlabeled data point

Training a network in such a way enhances the confidence on the predictions*, and it will be stronger to **adversarial attacks****

*Yassine, Hudelot, Tami, 2020, An Overview of Deep Semi-Supervised Learning

**Wu et al., 2018, Reinforcing Adversarial Robustness using Model Confidence Induced by Adversarial Training

Pipeline



Google Drive PyTorch databricks Community Edition APACHE Spark

SAPIENZA
UNIVERSITÀ DI ROMA

databricks
Community Edition

Material

- this presentation
- the (full) notebook
  - slow
  - implements each step in the pipeline
- a demo notebook
  - fast(er)
  - no data analysis
  - no training, since loads the pre-trained models

Sapienza Università di Roma

Pipeline description

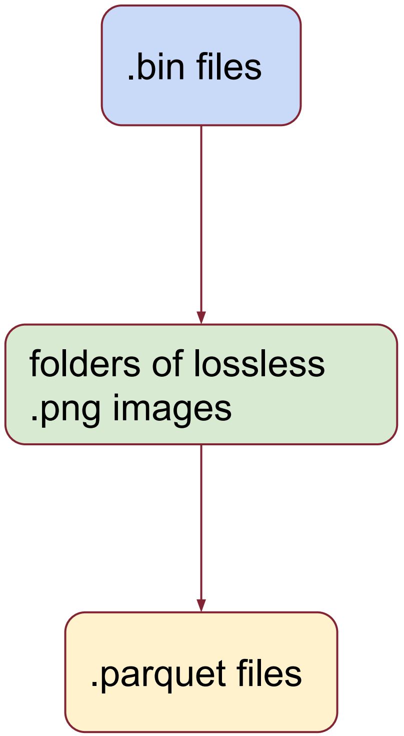
Sapienza Università di Roma

Pipeline description

Data Retrieval & Features engineering

Data retrieval

- dataset is available on its [website]
- .bin format is not PySpark-friendly 😅
- local conversion to [.parquet]
- uploaded on [Google Drive]



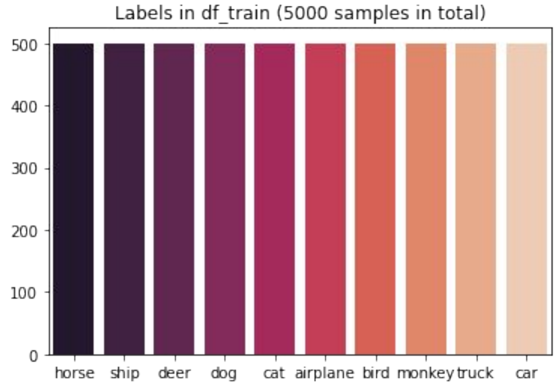
Nome	Proprietario	Ultima mo... ↑	Dimensioni file
stl10_test.parquet	io	11 mag 2021	199,5 MB
stl10_train.parquet	io	11 mag 2021	124,6 MB
stl10_unlabeled_part1.parquet	io	11 mag 2021	307,1 MB
stl10_unlabeled_part2.parquet	io	11 mag 2021	307 MB
stl10_unlabeled_part3.parquet	io	11 mag 2021	306,8 MB
stl10_unlabeled_part4.parquet	io	11 mag 2021	307,1 MB
stl10_unlabeled_part5.parquet	io	11 mag 2021	307,1 MB
stl10_unlabeled_part6.parquet	io	11 mag 2021	307,6 MB
stl10_unlabeled_part7.parquet	io	11 mag 2021	307,4 MB
stl10_unlabeled_part8.parquet	io	11 mag 2021	307,2 MB

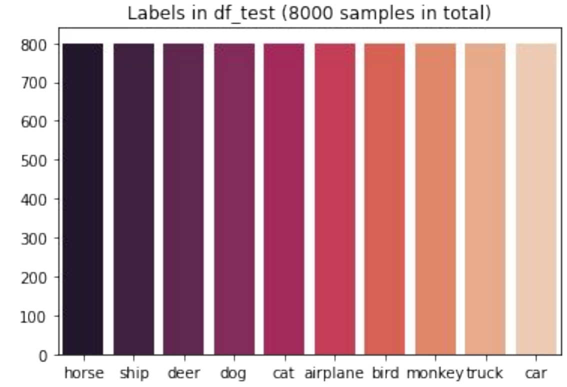
```
|df_train| = 5000 (in 2 partitions)
+--------------------+--------+---+
|               image|   label| id|
+--------------------+--------+---+
|[0, 2, 4, 0, 2, 4...|airplane|  0|
|[83, 115, 110, 83...|airplane|  1|
|[109, 112, 117, 1...|airplane|  2|
|[-58, -58, -48, -...|airplane|  3|
+--------------------+--------+---+
only showing top 4 rows
```

SAPIENZA
UNIVERSITÀ DI ROMA

Data analysis

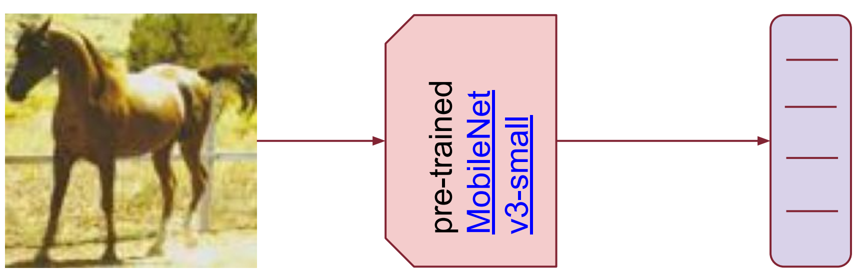
- the classes of the labeled samples are perfectly balanced
- unlabeled examples are extracted from a similar but broader distribution of images
  - for instance, it contains other types of animals (bears, rabbits, etc.) and vehicles (trains, buses, etc.) in addition to the ones in the labeled set





Features engineering

- added a unique id for each image
- casted pixels’ intensities to 1-byte integers
- dimensionality reduction
  - transfer learning using a pre-trained CNN
  - parallelized in PySpark thanks to @pandas_udf functions



shape = (96, 96, 3)

shape = (1024)

```
|df_train_emb| = 5000 (in 2 partitions)
+--------+---+--------------------+
|   label| id|          embeddings|
+--------+---+--------------------+
|airplane|  0|[-0.5840396285057...|
|airplane|  1|[-0.3303175568580...|
|airplane|  2|[-1.2865829467773...|
|airplane|  3|[-0.0749394893646...|
+--------+---+--------------------+
only showing top 4 rows
```

Pipeline description

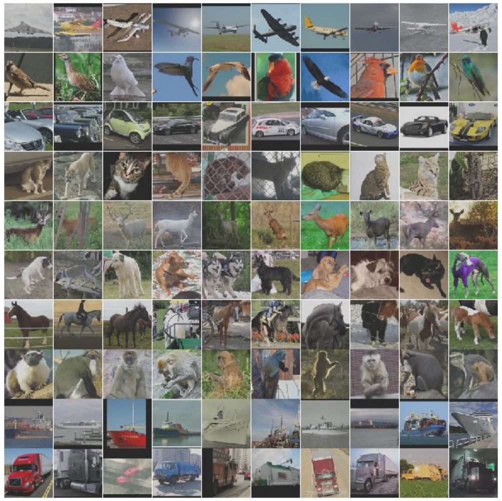
Models’ training

Training of base model

- labels’ mapping
- we’ve opted for [Multilayer Perceptron Classifier]
  - after some tests, the best-performing among other non-discrete multiclass classifiers available on PySpark (for our needs)
  - 50 iterations ⏳
  - learning rate = 0.001 👶
  - 4 layers 🎂
    - |input layer| = 1024
    - |hidden layer 1| = 512, |hidden layer 2| = 256
    - |output layer| = 10
- full training takes ~6 minutes
- saved on dbfs to ease access in the demo notebook

Pseudo-labeling of unlabeled data

- each unlabeled image is labeled according to the model trained in the previous step
  - some labels may be wrong
  - a model trained on this data may be biased
  - need a way to spot such errors 🧐



Training of final model

- same model architecture as before
- trained on:
  - labeled training part of the dataset
  - pseudo-labeled unlabeled part of the dataset
- k-fold cross validation* ⚔️
  - to reduce selection bias for “unlucky” splits
  - in each round, we split the dataset into k parts: one part is used for validation, and the remaining parts are merged into a training subset for model evaluation
  - performances computed as the arithmetic mean over the k performance estimates
- full training takes more than 60 minutes
- saved on dbfs to ease access in the demo notebook

*Raschka, 2018, Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning

Sapienza Università di Roma

Pipeline description

Models’ evaluation

Models' evaluation

- evaluation is done using
  - MulticlassClassificationEvaluator for accuracy, precision, recall, F1 score, TPR, FPR
  - sklearn.metrics for MCC, ROC and the other metrics
  - seaborn and matplotlib for plotting 📝
- done on the labeled **test** part of the dataset, never used before
- the two models have comparable performance, but the final model seems to be more resistant to **adversarial attack**

Models’ evaluation - numbers

Base model

```
Evaluation results (base model)
{'accuracy': 0.669375,
 'f1': 0.6717551753303268,
 'mcc': 0.6335526776740902,
 'weightedFalsePositiveRate': 0.036736111111111115,
 'weightedPrecision': 0.680093714461409,
 'weightedRecall': 0.669375,
 'weightedTruePositiveRate': 0.669375}
```

Final model

```
Evaluation results (final model)
{'accuracy': 0.660375,
 'f1': 0.6616909977888196,
 'mcc': 0.6229632711502258,
 'weightedFalsePositiveRate': 0.03773611111111111,
 'weightedPrecision': 0.6654451885476704,
 'weightedRecall': 0.660375,
 'weightedTruePositiveRate': 0.660375}
```

Models' evaluation - classification report

Base model

Classification report (base model)

	precision	recall	f1-score	support
airplane	0.74	0.81	0.77	800
bird	0.68	0.69	0.68	800
car	0.85	0.82	0.84	800
cat	0.54	0.47	0.50	800
deer	0.64	0.62	0.63	800
dog	0.42	0.59	0.49	800
horse	0.66	0.62	0.64	800
monkey	0.66	0.53	0.59	800
ship	0.86	0.76	0.81	800
truck	0.76	0.78	0.77	800
accuracy			0.67	8000
macro avg	0.68	0.67	0.67	8000
weighted avg	0.68	0.67	0.67	8000

Final model

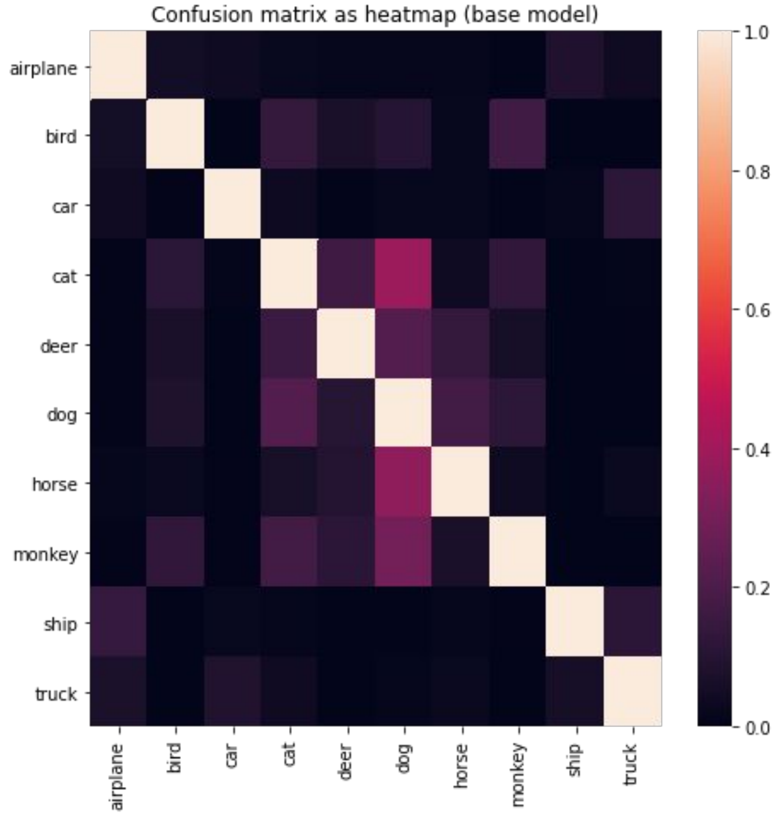
Classification report (final model)

	precision	recall	f1-score	support
airplane	0.73	0.74	0.74	800
bird	0.67	0.67	0.67	800
car	0.86	0.80	0.83	800
cat	0.53	0.49	0.51	800
deer	0.61	0.64	0.63	800
dog	0.44	0.54	0.48	800
horse	0.64	0.64	0.64	800
monkey	0.64	0.53	0.58	800
ship	0.79	0.80	0.79	800
truck	0.75	0.76	0.75	800
accuracy			0.66	8000
macro avg	0.67	0.66	0.66	8000
weighted avg	0.67	0.66	0.66	8000

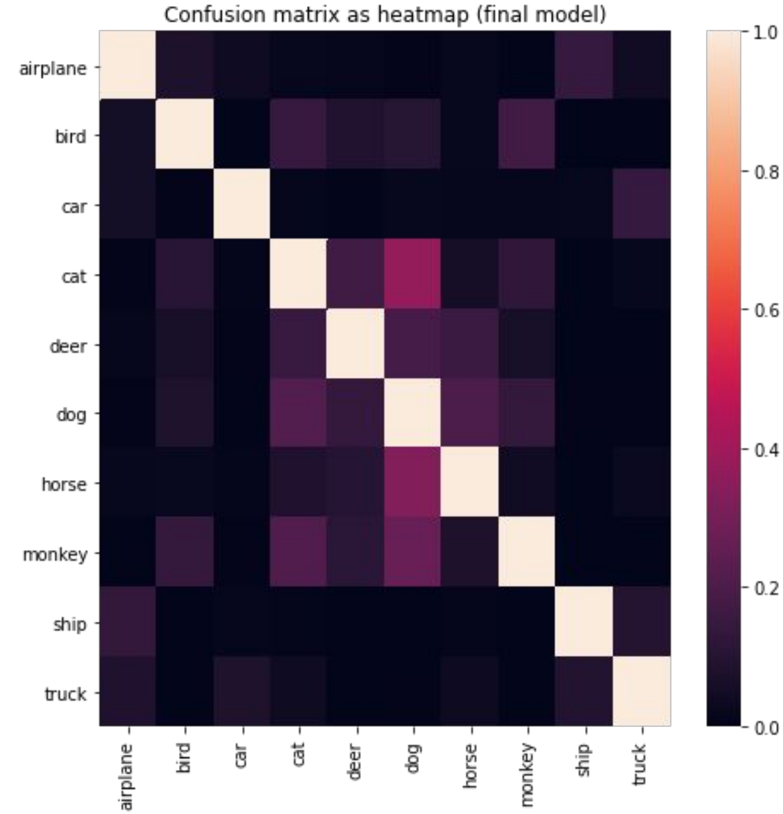
SAPIENZA
UNIVERSITÀ DI ROMA

Models' evaluation - confusion matrix (as heatmap)

Base model

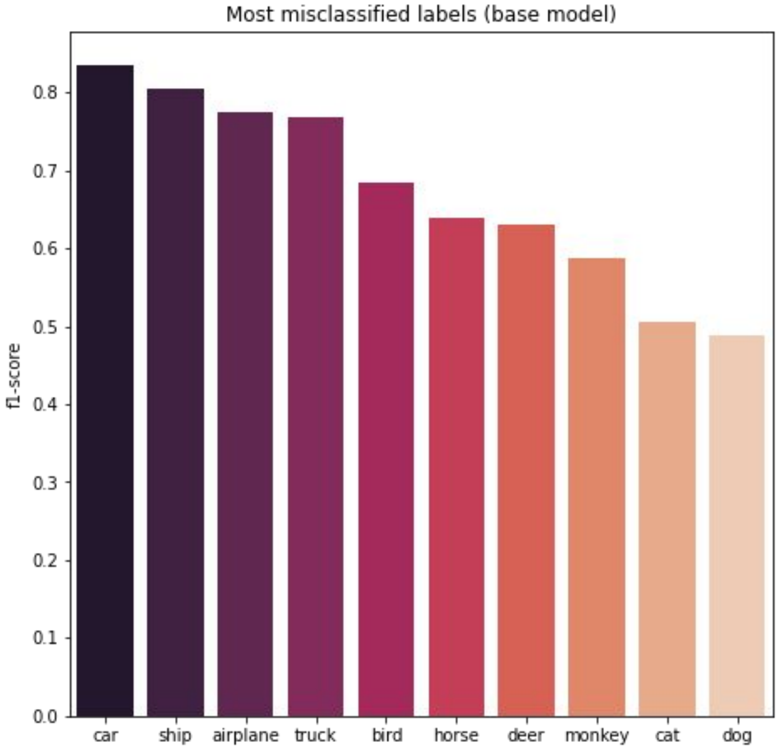


Final model

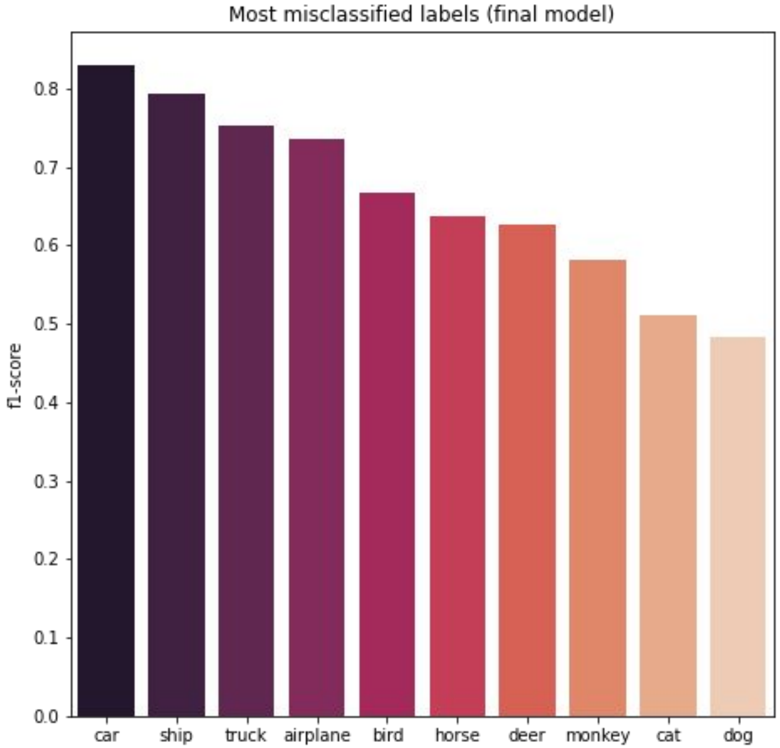


Models’ evaluation - most misclassified labels

Base model



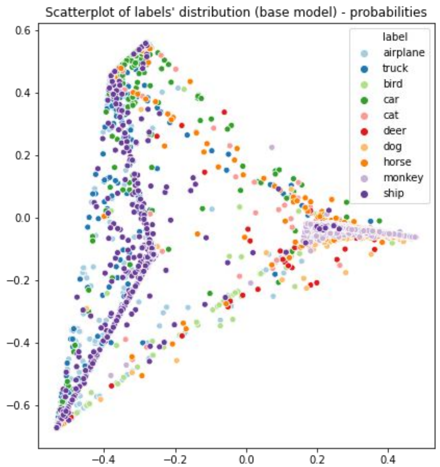
Final model

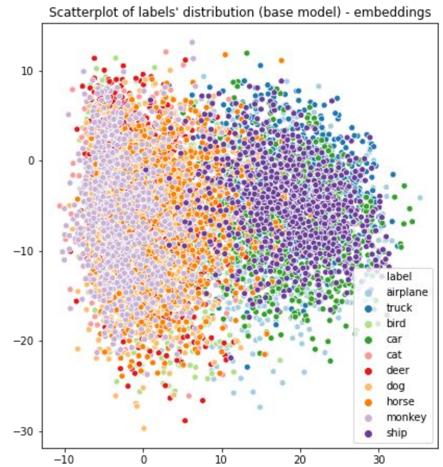


SAPIENZA
UNIVERSITÀ DI ROMA

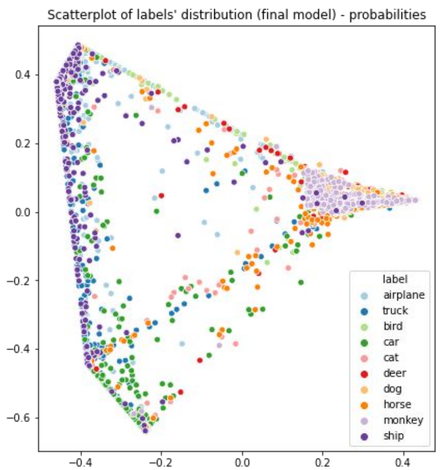
Models’ evaluation - labels’ distribution

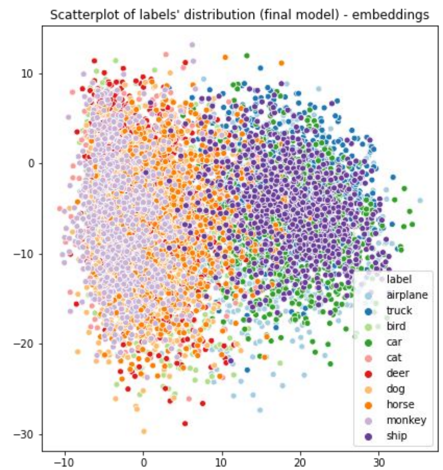
Base model





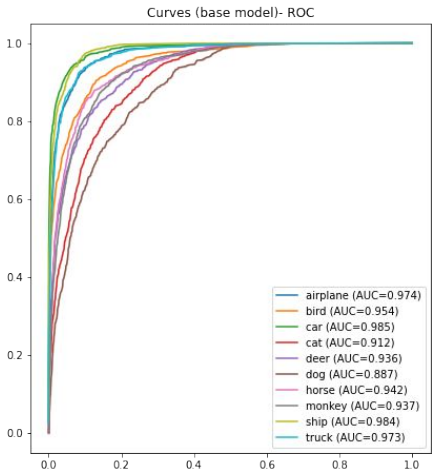
Final model

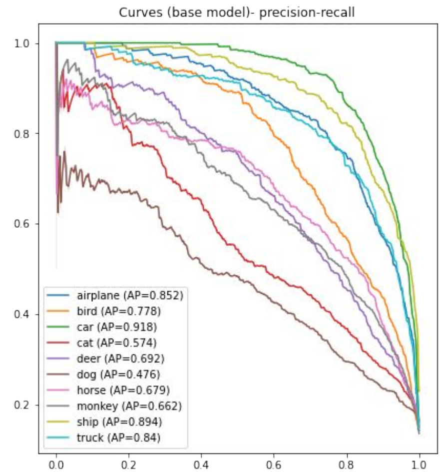




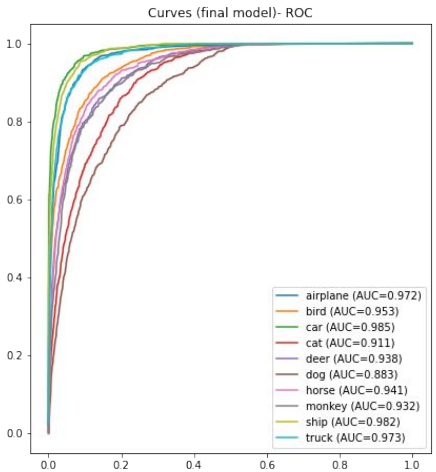
Models’ evaluation - ROC and precision-recall curves

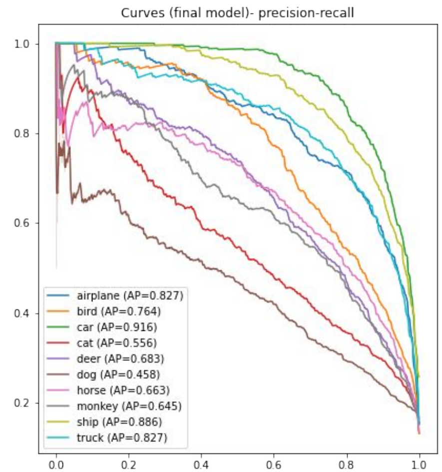
Base model





Final model





Sapienza
Università di Roma

Pipeline description

Images’ search engine

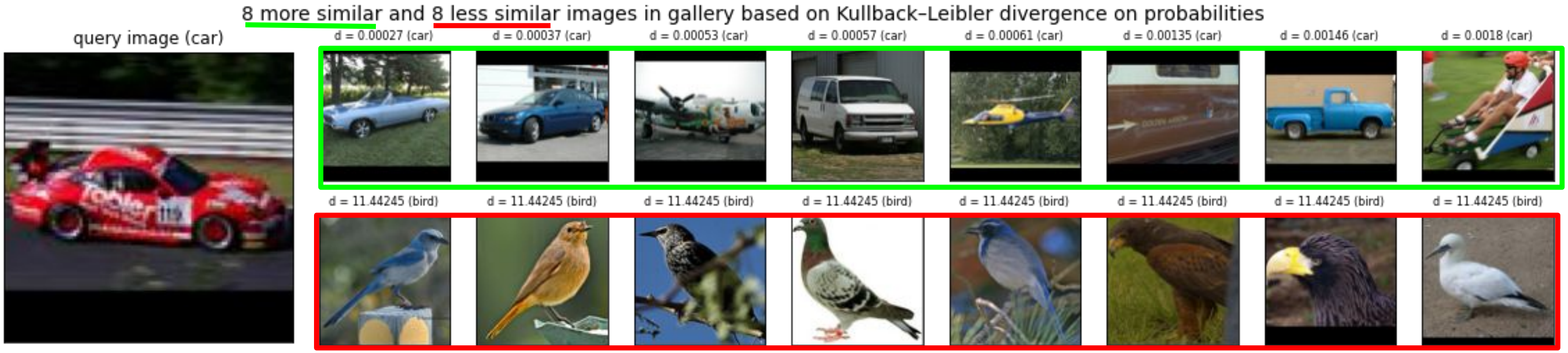
Images' search engine

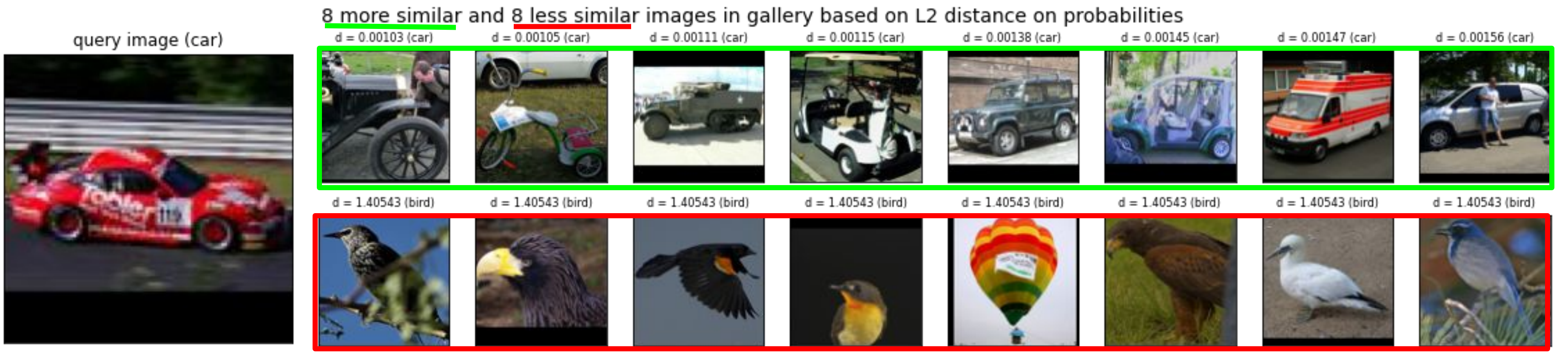
- two distinct disjoint sets:
  - **query set** 📷
    - images in the test part of the dataset, as predicted by final_model
    - images in this set are used as queries in the search engine
  - **gallery set** 🖼️
    - images in the unlabeled part of the dataset, as predicted by final_model
    - images in this set composes the gallery of the search engine, so the images that can be returned as results of a query
- four **distance metrics**:
  - Kullback-Leibler divergence* on probabilities
  - L2 distance on probabilities
  - L2 distance on embeddings
  - Cosine distance on embeddings

*Piro et al., 2008, Image Retrieval via Kullback-Leibler Divergence of Patches of Multiscale Coefficients in the KNN Framework

SAPIENZA
UNIVERSITÀ DI ROMA

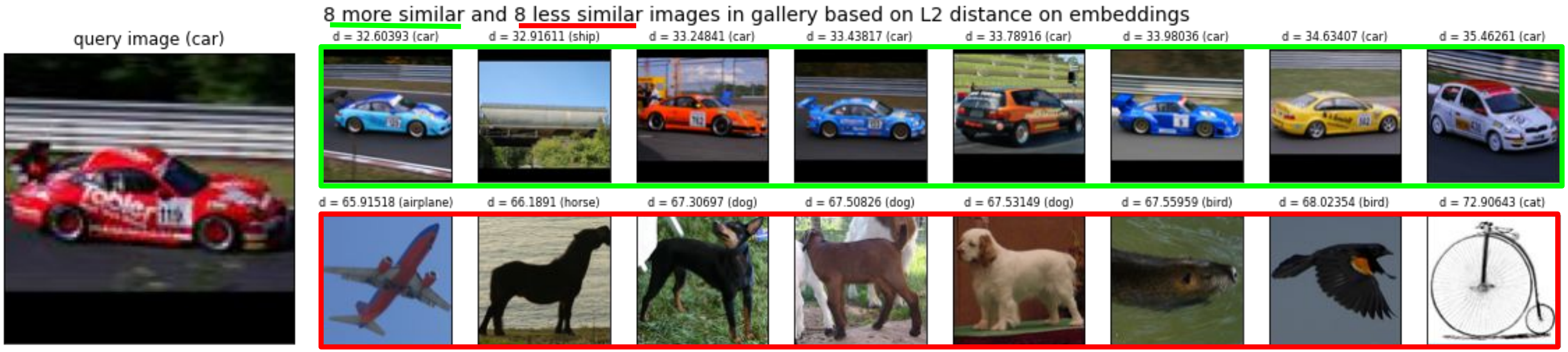
Images’ search engine - results on probabilities

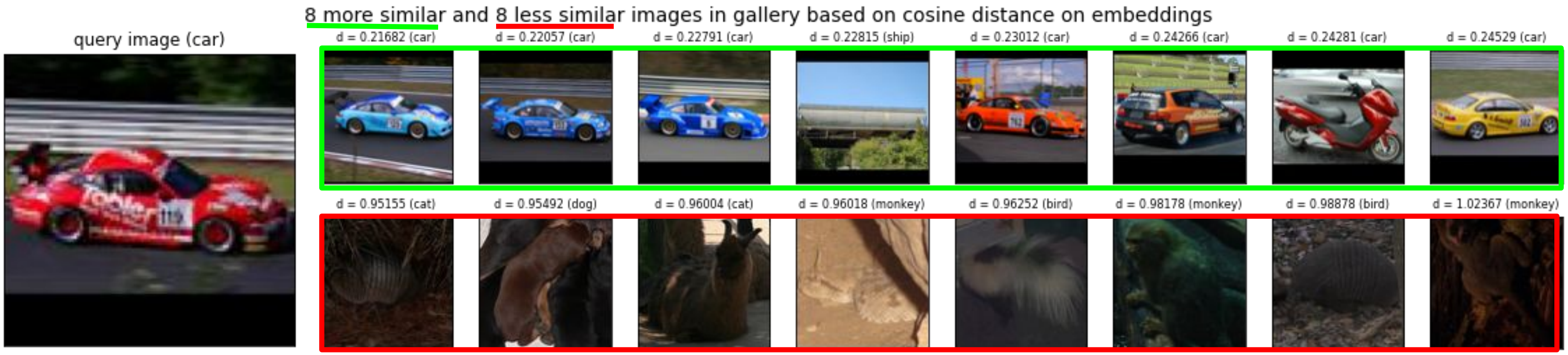




SAPIENZA
UNIVERSITÀ DI ROMA

Images' search engine - results on embeddings





Sapienza Università di Roma

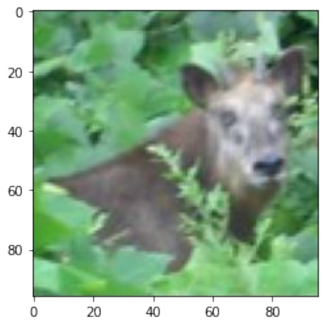
Pipeline description

Example of adversarial attack

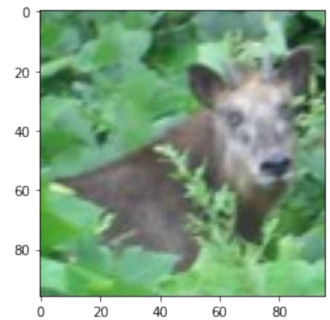
Sapienza Università di Roma

Black-Box Adversarial Attack

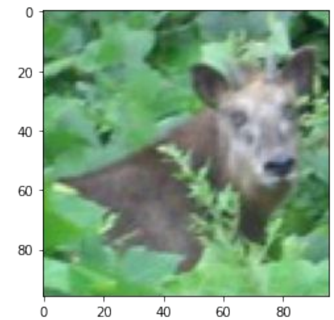
Original (deer)



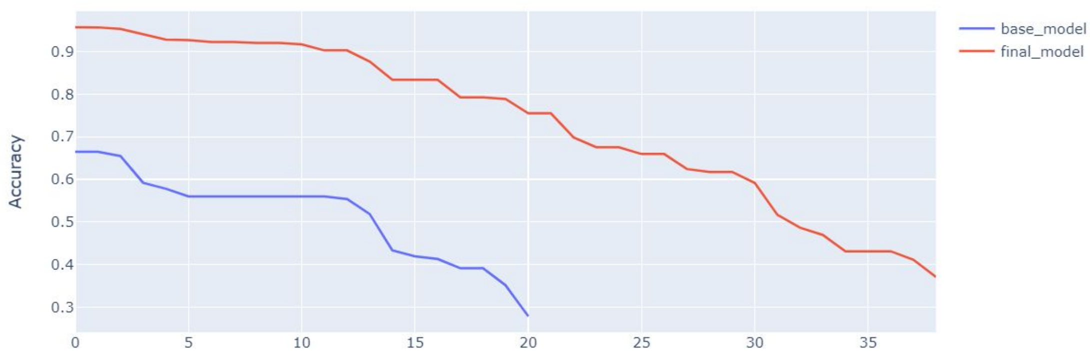
adversarial example:
base_model (dog)



adversarial example:
final_model (dog)



Accuracy over iterations of attack



*[Guo et al., 2019, Simple Black-Box Adversarial Attacks](#)

Conclusions

SAPIENZA
UNIVERSITÀ DI ROMA

Future work

- improve the classification models, maybe using a **state-of-the-art CNN** like ResNet (on GPU)
- perform an adversarial attack on the network to check **how the networks responds**
- **experiment with another approach** to semi-supervised classification such as virtual adversarial training*
- use **data augmentation**** techniques (random rotations, crops, jitter etc) to expand the labeled training set using the same images in a different perspective

*Miyato et al., 2018, Virtual Adversarial Training: A Regularization Method for Supervised and Semi-Supervised Learning
**Shorten, Khoshgoftaar, 2019, A survey on Image Data Augmentation for Deep Learning

Thank you for your attention! 🤗