Focus on supervised Deep Learning to classify images of waste in the wild

From Machine Learning to Deep Learning: a concise introduction

Dr. Khatuna Kakhiani

26.03.- 28.03.2024, HLRS, Universität Stuttgart

Agenda - March 27, 2024

9:00 - 17:30 CEST

- A Brief Introduction to Deep Learning
  - Tutorial 1: Image exploration with Jupyter Notebook
- Data Processing
  - Tutorial 2: Dataset exploration and preprocessing
- Waste Image Classification - Neural Networks
  - Tutorial 3: Sigmoid Neural Model; Simple Neural Networks
  - Tutorial 4: Image Classification with Multilayer Perceptron (MLP)
- Convolutional Neural Networks (CNN)
  - Tutorial 5: Image Classification with CNN

Lunch break: 13:00 – 14: 00 PM; 15 min Breaks @ 11:00 AM & 15:45 PM

What is Deep Learning (DL) used for?

Increasingly use of DL

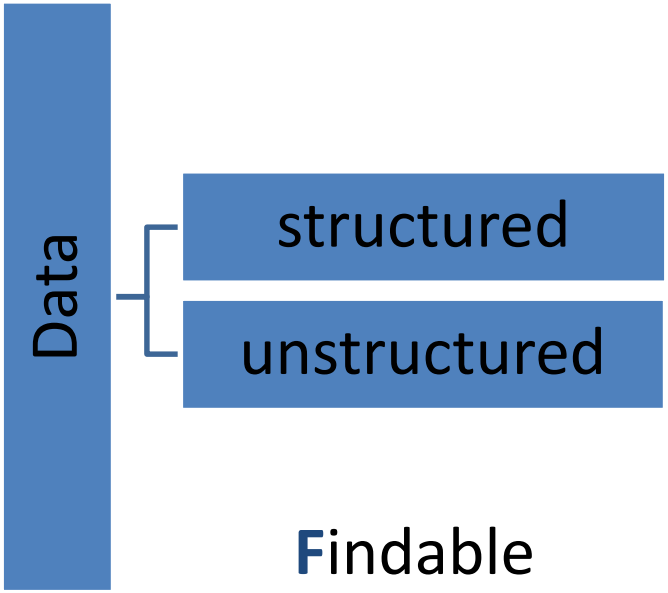
- Identify objects in images
- Transcribe speech into text
- Match news items
- Web search
- Content filtering on social networks
- Recommendations on e-commerce websites
- Microscopic Imaging; Medical Imaging
- Drug Discovery
- Search for new Functional Materials

- **AI-based content generators**
- **AI image generators, Chatbots**

Generating new data

Learning from Data

Machines Learn from Data



- Collection of Raw Fact
- Unprocessed
- Recorded or stored

Findable

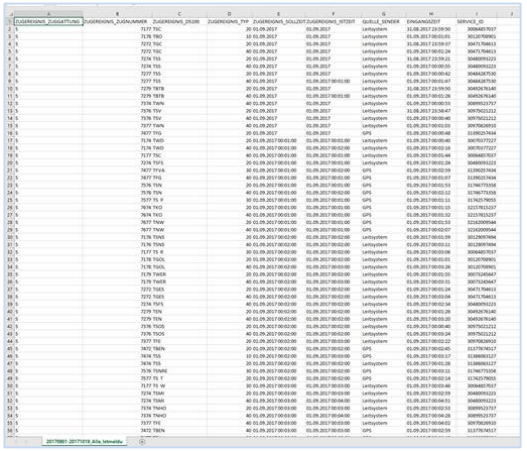
Accessible

Interoperable

Reusable

Data

- **Structured Data:**
  - organized and formatted
  - Examples: data held in spreadsheets and databases



- **Unstructured Data:**
  - no predefined construction or systemization
  - challenging to analyze
  - Examples: text, audio, images, videos & any combinations

Annotated Data

ML/DL models learn recurring patterns in **annotated data**

Data annotation:

- key part of the training dataset development for a supervised ML/DL
- adding metadata to a dataset - i.e., tags added

- Dataset for tutorial is annotated

Data

- TACO dataset- http://tacodataset.org/
- Contains images of waste in the wild
- User annotated - annotations in COCO* format
- Size: 1500 annotated images
- 60 categories of objects and 28 super categories

[*] https://cocodataset.org/#format-data for object detection

```
annotation{                 : int,
  "id"                      : int,
  "image_id"                : int,
  "category_id"             : RLE or [polygon],
  "segmentation"            : float,
  "area"                    : [x,y,width,height],
  "bbox"                    : 0 or 1,
  "iscrowd"
}

categories[{                : int,
  "id"                      : str,
  "name"                    : str,
  "supercategory"
}]
```

Learning Machines (LM)

The training of programs developed
by allowing a computer to learn
from its Experience,
rather than through manually coding the individual steps

Machine Learning (ML)

- Arthur Samuel (**1959**): “Field of study that gives computer ability to learn without being explicitly programmed”
- Tom Mitchell (**1998**): "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure **P**, if its performance at tasks in T, as measured by P, improves with experience E“

E = playing many games of chess

T = the task of playing chess

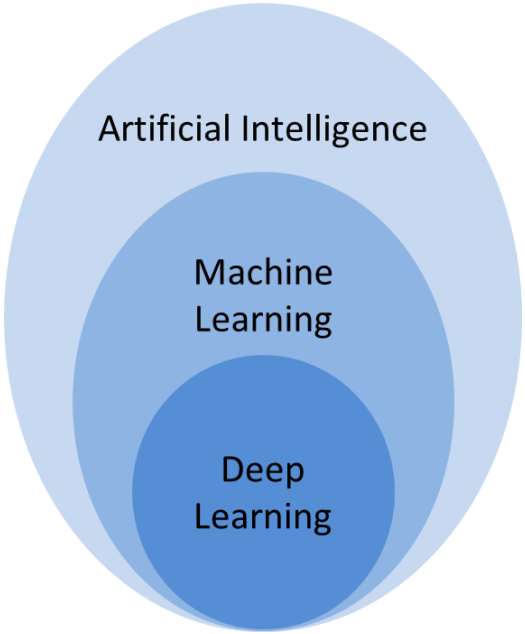
P = the probability that the program will win the next game

DL, ML and Artificial Intelligence

Techniques enable computer/machine to

- mimic human intelligence - Artificial Intelligence (AI)
- improve at task with experience (ML)
- train itself to perform a task (DL)

  (based on Deep Neural Networks (DNN))



The Learning Problems

- Supervised Learning
  - continues target variable
    - **Regression**
      - Prediction of age of a person based on a picture
  - categorical target variable
    - **Classification**
      - **Image classification**
      - Diagnostics
        - Microscopic imaging
        - Biomedical imaging
      - Data-Driven Computational Fluid Dynamics
      - Ocean/ Weather/Geosciences
- Unsupervised Learning
  - target variable not available
    - **Clustering**
      - Customer Segmentation
      - Astronomical data analysis
    - **Dimensionality reduction**
- Reinforcement Learning
  - categorical target variable
    - **Classification**
      - Optimized Marketing
  - target variable not available
    - **Control**
      - Driverless car

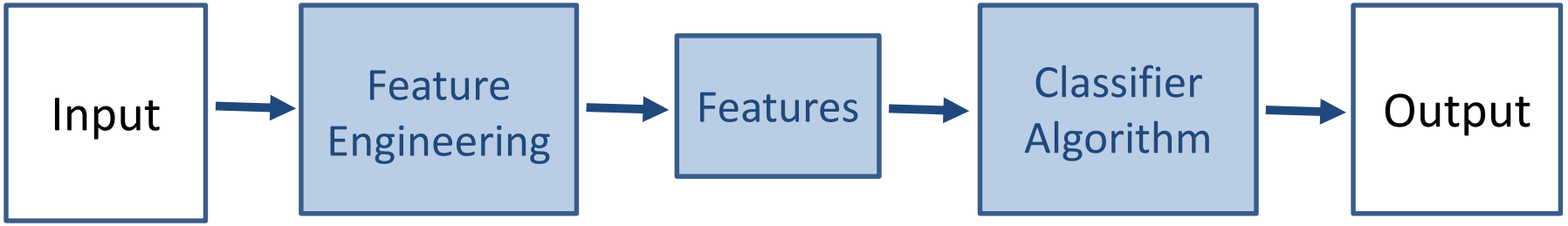
The Learning Problem

- **Supervised learning**
  - **Waste image classification**
- **Classification:** The goal is to predict discrete values
  - {Bottle, No bottle}
  - {1, 0}

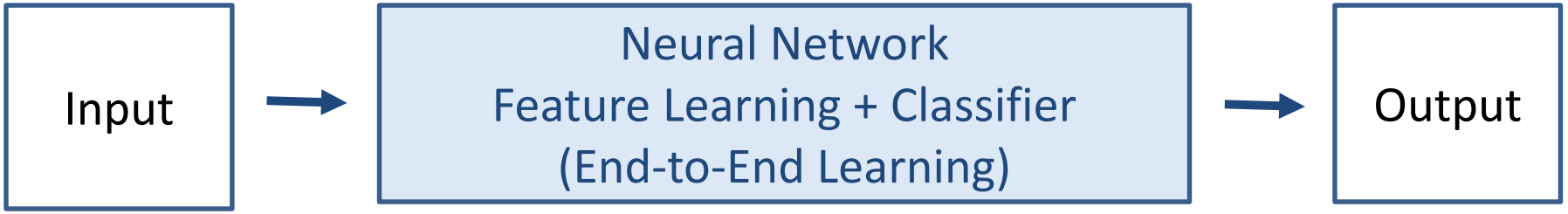
Why Deep Learning?

Simplified workflow

Classical Machine Learning



Deep Learning



Why Deep Learning?

- Works better on **unstructured data**
- **End-to-End training**
- **Model reuse**: learned features and trained models
  - transfer learning in case of a small dataset
- **Automatic Feature Extraction**
- **Eliminate domain-specific processing**
  - remove boundaries between CV, NLP etc.
  - unified set of tools for diverse problems
- Cons: Black-Box models

Scientific ML/DL

- **Data-driven surrogate modeling**
  - Replacing a **computationally expensive** numerical simulator by a **fast** data-driven model

- **Phisics-Informed machine learning**
  - **Regularizing** a data-driven machine learning model using a physics-based model

- **ML-enhanced model discovery**

- **Hybrid modeling and simulation**

DL in the field of CFD

Towards Data-Driven Computational Fluid Dynamics

A. Schwarz, A. Beck (Day3)

An overview about recent advances and failures in the application of deep learning techniques to enhance the solution of nonlinear, hyperbolic PDEs, from supervised to reinforcement learning.

Examples from the field of CFD

- Turbulence modeling &
- Shock capturing

Data - Image

Your very complex sence

"All you are given are two tiny upside down two dimensional images inside your eyeballs, but you perceive a single panoramic, right side up, three dimensional world. How does this miraculous transformation come about?"- Richard L Gregory

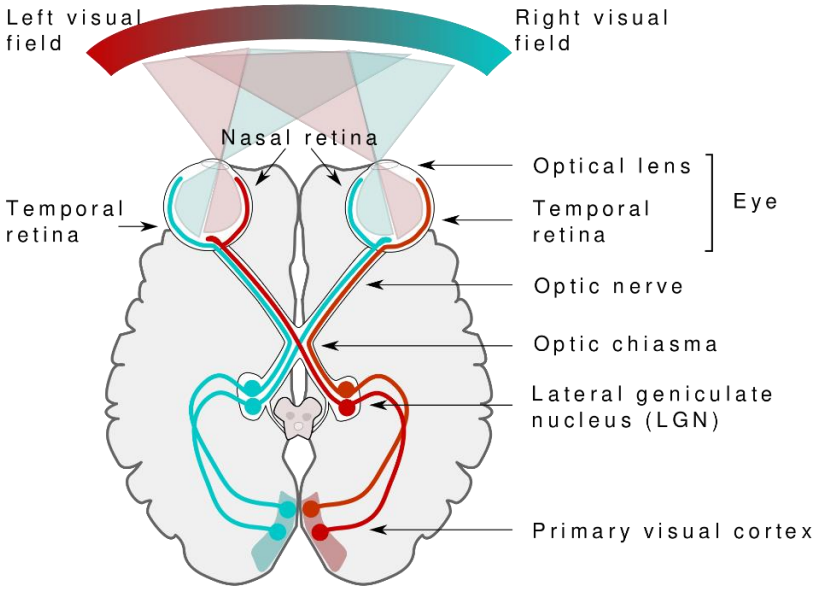
Eyes:

- our wonderful kind of camera
- accommodate ~ 70 % of body sensory receptors detect Visible Light (VL) (fireflies & stars emit VL)
- look around & dynamically adjust
- signal to the brain (our LM)

The Brain

We see a mind's reconstruction* of objects

- Millions of light receptors signal to the primary visual cortex
- The primary visual cortex makes sense of what we see

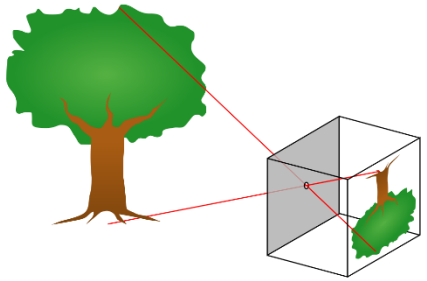


Shen G, Horikawa T, Majima K, Kamitani Y (2019) Deep image reconstruction from human brain activity. PLoS Comput Biol 15(1): e1006633.

Picture Source: https://commons.wikimedia.org/wiki/File:Human_visual_pathway.svg

How does camera work?

- Pointed at an object
- The **lens** in a digital camera **focuses light** from an object onto a **light sensor** inside the camera
- A sensor collects **VL** (400 ~ 700nm)
- Other Tasks:
  - convert into electrical signal
  - organize information to store in **raw format** or remove some information during **compression**
  - render images (x,y pixel size) and video streams
- Tutorial Image dataset (images in **JPEG** format)
- Data Compression with **BigWhoop library** for compression of numerical data sets (**CoE Excelerat**)



Digital Compression

- Compression reduces the size of any file
  - **Lossless** Compression
    - **raw** files saved in i.e., TIFF file format
  - **Lossy** Compression
    - discarding some information, reduce quality
    - file format mostly used JPEG
    - Tutorial Image dataset (images in **JPEG** format)

- Data Compression of numerical data sets with the BigWhoop library

How do Computer Store Images?

- **A matrices of numbers** representing **the discrete color** or **intensity values** present in every **image pixel**

- Every photograph, in digital form, is made up of pixels
- **Pixel** - smallest unit of information
- Each pixel is using a combination of Red, Green, Blue colors
- This is what we call an RGB image
- typically arranged in a 2-dimensional grid

Image color mode

A grayscale color mode

- An image, comprised of pixels, a function **f ?**
- Each 8 - bit pixel has its own value:
  i.e., intensity range [0, 255]; 0 - black & 255 - white
- f(x, y) → the color intensity at pixel position (x, y)
  assuming a rectangle with a finite range:
  f: [a, b] x [c, d] → [0, 255]

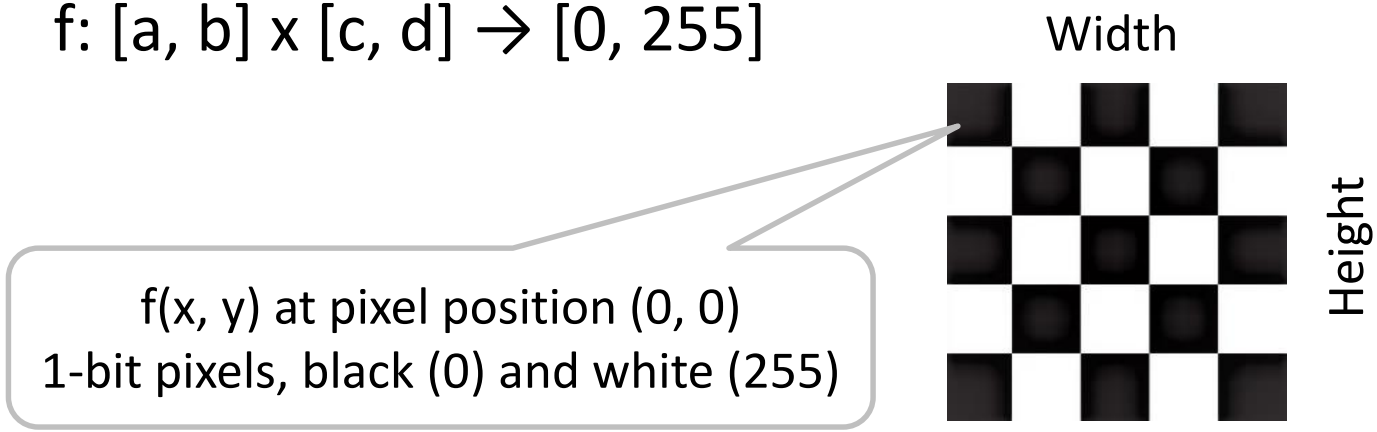
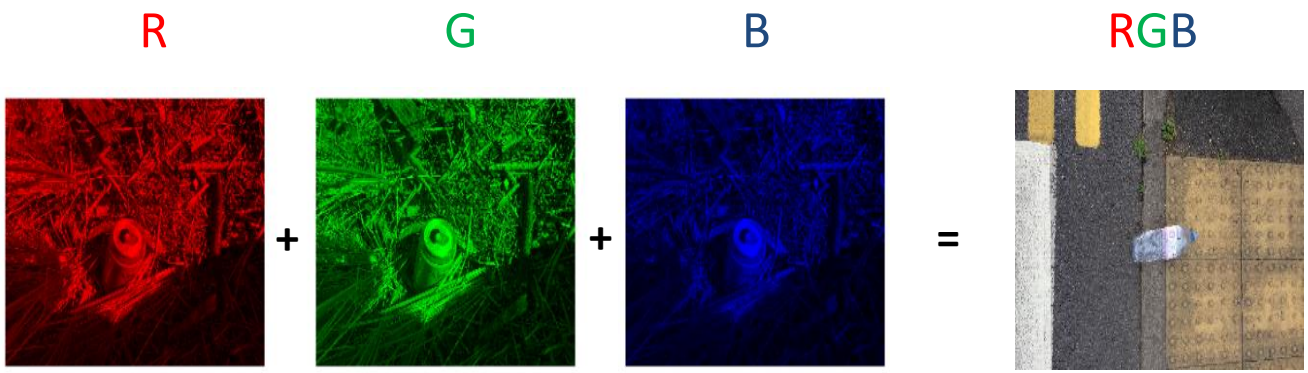


Image color mode

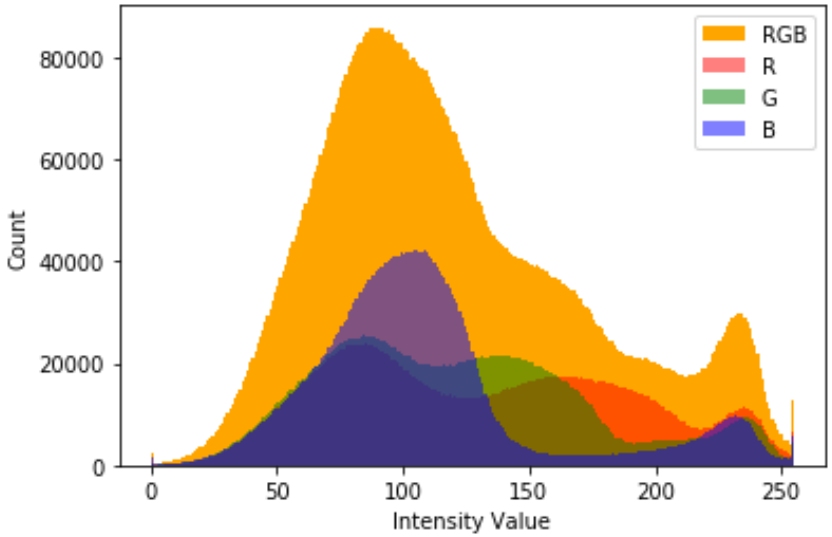
A **RGB color mode**

- RGB colors - a combination Red, Green, Blue channels
- Pixel range [0, 255] of each 3, 8 - bit color channel
- 256 * 256 * 256 = 16,777,216 combinations or color



Histogram

- The distribution of pixel values of an image



- Image width x height: 2049 x 1537 (tutorial image)
- Color channel intensity values: Red [67 49 38 … 145 152 160]
  Green [64 46 35 … 119 126 134]
  Blue [71 53 42 … 96 103 109]

Tutorial

Tutorial - Image exploration

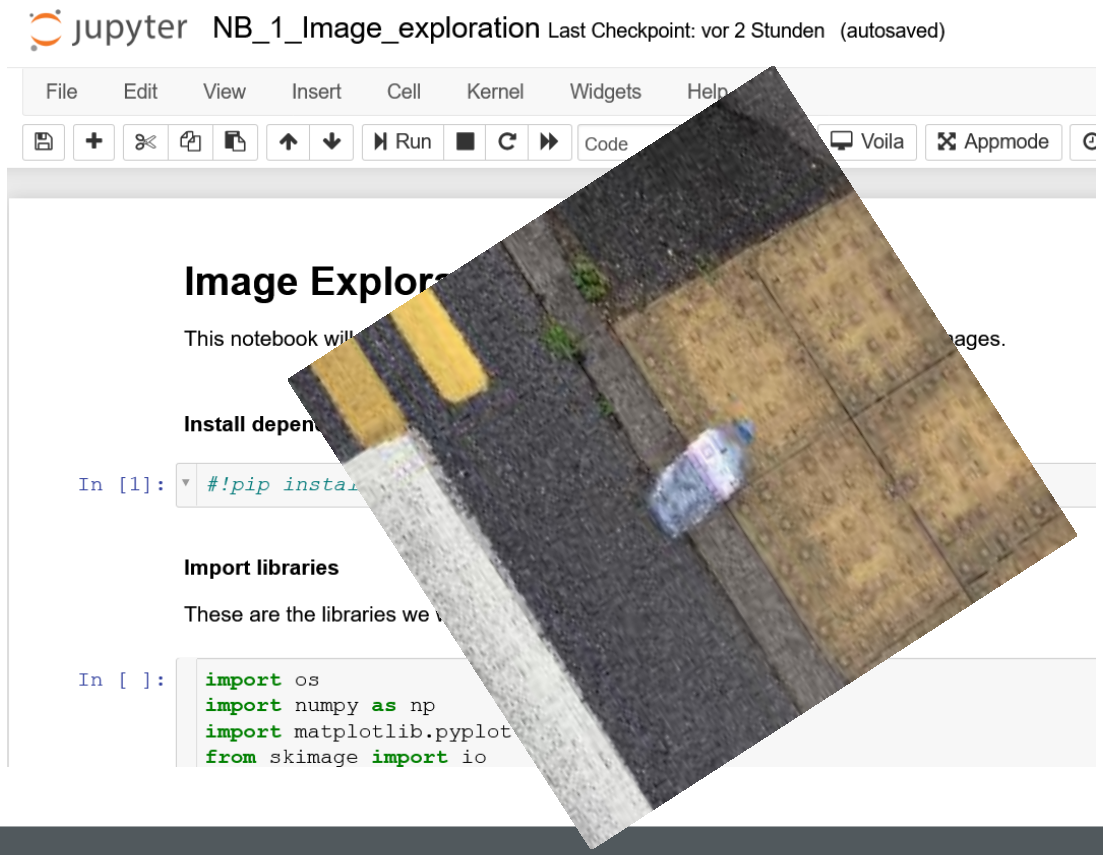
- **Hands-on:** Image exploration with Jupyter Notebook
- **Outcome:** Basic understanding of *image data,* ability to manipulate image instances and visualize results.

Tutorial

Jupyter notebook

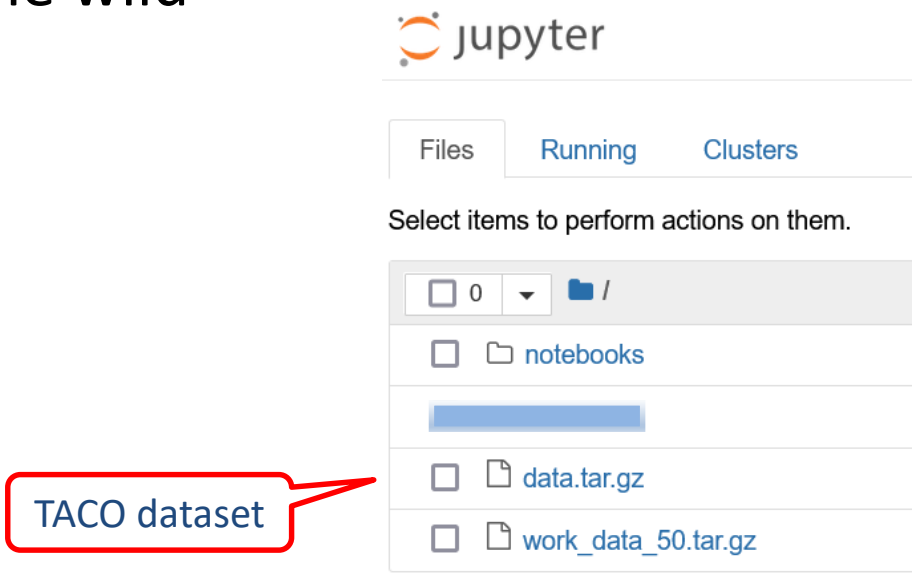
10 min

- Open a notebook notebooks/NB-1_Image_exploration.ipynb



Tutorial - Image exploration

- TACO dataset - http://tacodataset.org/
- Contains images of waste in the wild
- User annotated 1500 images

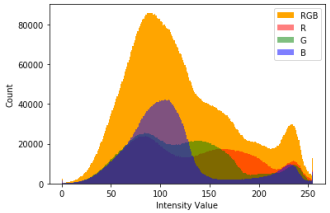


Tutorial - Image exploration

Tasks to complete in Notebook: NB-1_Image_exploration.ipynb

- Import libraries
- Load and handle one image data using PIL/Pillow
- Get original image size and color mode
- Split image from dataset into its RGB channels

- Explore Histogram of an Image



- Manipulate image instances

Tutorial - Image exploration

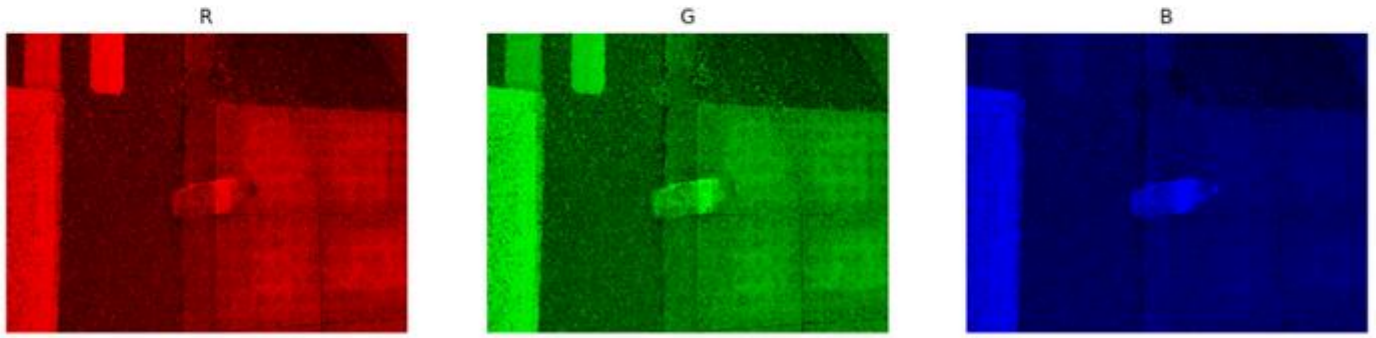
Example of a task:

```
In [ ]: #Exercise 5: display each color mode on a separat axes
fig, axes = plt.subplots(nrows=1, ncols=3, figsize=(15,5))

for v, c, ax in zip(["----", "G", "----"], range(3), axes):
    tmp_im = np.zeros(im.shape, dtype="uint8")
    tmp_im[:,:,c] = im[:,:,c]
    ax.imshow(tmp_im)
    ax.set_title(v)
    ax.set_axis_off()
```

to be completed

Expected result:



Deep Learning

Problem

- Huge amount of plastic waste
- Impact to wildlife and on humans
- Can we help build robots to classify waste automatically?

Source: https://www.onegreenplanet.org/

Data

- TACO dataset- http://tacodataset.org/
- Contains images of waste in the wild
- User annotated - annotations in COCO* format
- Size: 1500 annotated images
- 60 categories of objects and 28 super categories

[*] https://cocodataset.org/#format-data for object detection


Pop tab
Drink can


Single-use carrier bag


Plastic straw


Plastic Film


Paper cup
Plastic lid


Disposable plastic cup


Other plastic wrapper
Glass bottle


Styrofoam piec


Plastic Film
Meal carton


Foam food container


Plastic drink bottle
Plastic bottle cap
Wrapping paper


Drink can

Learning from a small dataset?

- Are huge datasets for every kind of real-world problem feasible?
- What are Issues of small number of observations?
- How can one avoid overfitting and get accurate results?
- Are small datasets balanced?
- What are model's limitations when drawing conclusions from its predictions?
- Is one aware about evaluation metrics for imbalanced data?

Image Classification Task

- **Given Dataset** → annotated RGB **images of litter**

- **The Task:**
  - predict a single label (or a distribution over labels to indicate confidence) for a given image

➢ turn numbers into a single label, such as *“bottle”*

The Learning Problem

- **Supervised learning**
  - **Waste image classification**
- **Classification:** The goal is to predict discrete values
  - {Bottle, No bottle}
  - {1, 0}

Image Classification Pipeline

- **Input:**
  - a set of **m** RGB **images of litter**
  - each labeled with one of **2 different classes {1, 0}**
- **Learning:**

  step of **training a classifier**, or **learning a model** using training set to learn what every one of the classes looks like
- **Evaluation:**

  classifier quality is evaluated by asking it to predict labels for a new set of images that it has never seen before

Input

- We distinguish 3 different sets of the **data**:

Train set	Validation set	Test set
• Model is trained • ~ 80% of the dataset	• Model is assessed • ~ 10% of the dataset	• Model is tested to make a prediction • ~ 10% of the dataset

- **The ground truth**: train, validation & test **label** sets

Learning a model

Configure the Model Architecture	Compile the model	Train the Model
• Artificial Neural Networks (ANN) • Multilayer Perceptron (MLP) • Convolutional Neural Networks (CNN)	• Loss (to measures how accurate the model is during training) • Optimizer (to minimize Loss with respect of parameters) • Metrics (to evaluate performance)	• Performance at Task improves with an Experience • Train to classify images • Track epochs, let model see every pictures many times; babysit process

Learning

- The **model train itself** from **experience E** with respect to some class of **tasks T** and performance **measure P**, if its performance at tasks in T, as measured by P, improves with experience E

E = perform a task multiple times (on train images & train labels)

T = the task to predict a single label for a given image

P = the probability that the trained model predict bottle

Evaluation

Evaluate

- Model performance is evaluated on validation set
- Trained model gives predictions on unseen data
- Chosen metrics suffice

Original Annotated Data

DL models learn recurring patterns in **annotated data**

- A set of 1500 images with annotations
- Original purpose: an object detection
- Annotations.json: a **nested dictionary** of data annotation

  Keys are mapped to another dictionary within original dictionary → preprocessing needed
- There are more annotations than images. Why?

Data Preprocessing

Create a simplified dictionary w.r.t. each image and its associated categories

```
Labels at instance {'file_name': 'batch_12/000019.jpg', 'category_ids':
{36, 5, 7, 45, 49, 22}, 'image_id': 320, 'height': 5312, 'width': 2988, '
category_names': {'Plastic bottle cap', 'Foam cup', 'Disposable food cont
ainer', 'Plastic film', 'Plastic utensils', 'Clear plastic bottle'}, 'sup
er_categories': {'Bottle', 'Bottle cap', 'Cup', 'Plastic container', 'Pla
stic bag & wrapper', 'Plastic utensils'}}
9.jpg', 'category_ids': {36, 5, 7, 45, 49, 22}, 'image_id': 320, 'height
```

Original

Data Preprocessing

- **Construct a binary classification problem**
  in a one vs all setting, i.e., does this image contain a specific supercategory (“bottle”) or not.

- **Split data**
  into three sets (**train**, **evaluation and test)** of **images and labels** to train, evaluate and test the model.

Data Preprocessing

- Summarize train, validation, and test data
- Notice the datasets are not balanced
  (e.g. proportion in supercategory < 50%)
- Choose evaluation metrics for imbalanced data

Training Number of instances: 1200, Proportion in supercategory: 19.50 %
Validation Number of instances: 150, Proportion in supercategory: 37.33 %
Test Number of instances: 150, Proportion in supercategory: 21.33 %

Data Preprocessing

Input parameters:

- Number of images
- Dataset images of varying sizes
- Number of RGB image channels (3)
- Pixel range [0, 255]

width x height in px: 2049 x 1537

Data Preprocessing

- Convert to gray scale:
  - color channel (3 → 1)
- Scale image size:
  - down-scaling of the width and height
  - uniform aspect ratio
  - most DNN models assume a square shape input image
- Normalize/standartize:
  - scale the pixel values to the range [0, 1]
  - each input parameter (pixel) has a similar data distribution
  - faster convergence while training the network

Deep Learning Tutorial

Tutorial - Data Preprocessing

- **Hands-on:** Dataset exploration & preprocessing
- **Outcome:** Basic understanding of issues and challenges of image data preprocessing; ability to preprocess data for the image classifier training and evaluation.

Tutorial

10 min

Jupyter notebook

> Open a **notebooks/**NB-2_preprocess.ipynb

1.2 Explore annotations dictionary

In [ ]:
```
#Exercise 1.2.1: Explore each annotation key to understand the dataset
```

In [ ]:
```
#Number of images
print("Number of images:", len(annotations["images"]))
```

2. Preprocessing

For simplicity, we create a simplified dictionary w.r.t to each image and its associated categories. We store only a subset of information, e.g. 'image_id', 'file_name', 'height', 'width', 'category_ids', 'category_names', and 'super_categories'.

In [ ]:
```
# 2.1 Open annotation file and read into memory
with open(anno_file, "r") as f:
    annotations = json.load(f)
```

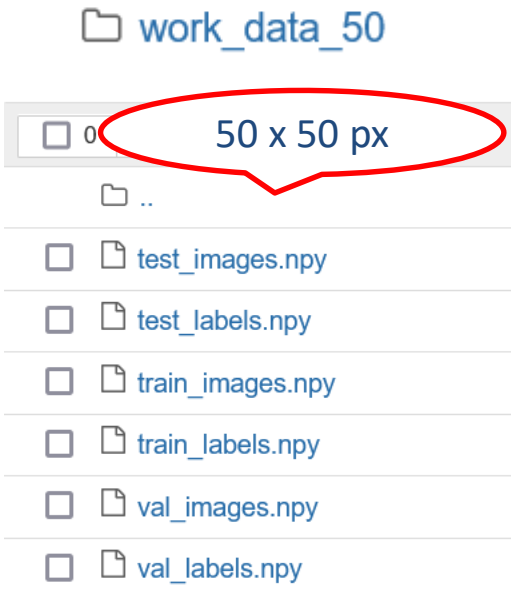
In [ ]:
```
# 2.2 Prepare category id to name mappings. Items are ordered by category_id, so you can get the
# category name of a category_id via the category_id, e.g.
# via annotations["categories"][category_id]
categories = annotations["categories"]
```

In [ ]:
```
#2.3 Create new python dictionary with subset of relevant information
#(e.g. image -> category data)
```

Tutorial - Data preprocessing

Tasks to complete in Notebook: NB-2_preprocess.ipynb

- Explore annotations dictionary & create a new one
- Construct a binary classification problem
- Split data and labels into 3 different sets (training, validation & test)
- Preprocess: resize, channel (3→1), normalize
- Check for the data distribution (summary)
- Verify the format of data
  - ✓view image pixel representation before & after normalization
  - ✓check annotations & label corresponding to this image



Tutorial - Data preprocessing

Some ToDo Tasks

Image source: http://tacodataset.org/

In [ ]:
```
#Image information
annotations["images"][320]
```

In [ ]:
```
#Exercise 1.2.2: explore annotation for image_id 6
for anno in annotations ["annotations"]:
    if anno["image_id"]==------:
        print(anno)
```

ToDo task

Expected result:

{'id': 17, 'image_id': 6, 'category_id': 11, 'segmentation': [[799.0, 652.0, 802.0, 670.0, 804.0, 699.0, ...

In [ ]:
```
#Exercise 1.2.3: print Category information for image_id = 6
print("Category information:", annotations["categories"][------])
```

ToDo task

Tutorial - Data preprocessing

3. Binary Classification

We can construct a binary classification problem in a one vs all setting, e.g. does this image contain a specific supercategory or not.

Let's create the numpy arrays corresponding to the images and labels that we can use for training in Network.

In [ ]:
```
#3.1 Split data into training, validation, and test

data_ids = list(data.keys())
#random.shuffle(data_ids) #Shuffle list of keys and store;  #for workshop will not be used

#Configure proportion of training, validation, and test data
train_perc = 0.8
val_perc = 0.1
test_perc = 0.1
train_size=int(len(data_ids)*train_perc)
val_size=int(len(data_ids)*val_perc)
train_ids, val_ids, test_ids = (
    data_ids[:train_size],
    data_ids[train_size : train_size + val_size],
    data_ids[train_size + val_size :],
    )

print("Number of images in training dataset:", len(train_ids))
print("Training image_ids:", train_ids)

print("Number of images in validation dataset:", len(val_ids))
print("Validation image_ids:", val_ids)

print("Number of images in  dataset:", len(test_ids))
print("test image_ids:", test_ids)
```

Tutorial

In [ ]:
```
#3.5 Summarize training, validation, and test data.

#Exercise 3.5 determine proportion of positive ('Bottle') class in train,
#validation, and test sets
```

Expected result:
Training Number of instances: 1200, Proportion in supercategory: 19.5%
Validation Number of instances: 150, Proportion in supercategory: 37.3%
Test Number of instances: 150, Proportion in supercategory: 21.3%

Notice the datasets are not balanced (e.g. proportion in supercategory <50%).
This is especially important for what evaluation metrics to use.

In [ ]:
```
#3.6 End-to-end Exercise: Verify that the data is in the correct format
#and ready to build and train the network.

#3.6.1 View colored training set image, example_index = 0.
example_index = 0
example_id = train_ids[example_index]
img_path = data[example_id]["file_name"]
image = Image.open(os.path.join(data_dir, img_path))
image
```

In [ ]:
```
# Normalization here for visualisation purpose of corresponding pixel representation
train_images = train_images / 255.0
```

In [ ]:
```
#3.6.2 Compare to grayscaled and resized version
image = image.convert("L")
image = image.resize(size=(500, 500))
plt.imshow(image, cmap="gray")
```

Expected result:

Outlook

What can you do more?

Data Augmenting (DA)

Augment - artificially expand small labeled training datasets

- to prevent overfitting
- initial dataset is too small to train on
- to “squeeze” better performance from model

- Findable Accessible Interoperable Reusable

Data

DA Techniques

- **Geometric transformations**
  - Affine Transformations, simple linear transformations
  - shift, scale, rotate, flip, translate images…

- **Color space transformations**
  - change, intensify any color

- **Kernel filters** - sharpen or blur an image
- **Random Erasing** - delete a part of the initial image
- **Mixing images** - mix images with one another

Generative AI

- Involves training computers **to generate new data,** similar to data that it has been trained on, and to what a human might produce.
- Those algorithms can:
  - **improve the efficiency and accuracy of existing AI systems** (NLP, CV).
  - **produce various types of content, including text, imagery, audio and synthetic data**.

- **AI-based content generators**
- **AI image generators, Chatbots**
- Other useful tools that can make people's lives easier and grabbed the business headlines lately.

Tutorial

Requirements for the exercises

Deep Learning – day2

https://fs.hlrs.de/projects/par/events/2024/dl-hlrs/commands-part2.txt

- Access your course account on Vulcan

  > ssh username@training.hlrs.de

- Navigate to your **day2** workspace

  > ws_list

  > MYSCR = $(ws_find wspart2)

  > cd $MYSCR (if **ws** path is stored) or

  > cd /shared/username/**wsday2**

Requirements for the exercises

Deep Learning – day2

- Copy all data for day2 to your workspace:

  > cp -r cp /shared/akad-dl-hlrs/day2/***.tar.gz** .

- Extract the content of the tar file:***.tar.gz** archives:

  > tar -xzf data.tar.gz

  > tar -xzf work_data_50.tar.gz

  > tar -zxvf notebooks.tar.gz

Requirements for the exercises

- **data** - for Litter classification (*.json & co)
- **work_data_50** - image data as numpy array
- **notebooks** - tutorial exercises
- **solutions (you wont need hopefully)**

[*] http://tacodataset.org (dataset)

Requirements for the exercises

Options to work on exercises:

- **Jupyter Notebook**

  and/or

- **Job scheduler** in terminal

Requirements for the exercises

To work with **Jupyter notebook**

- Open a **second** terminal window

- Connect Vulcan via **ssh** again:

  > ssh username@training.hlrs.de -D 8080 -q

Requirements for the exercises

To work with **Jupyter notebook**

- Reserve computing time:

Max: until 18:00 CEST

as day1

```
qsub -I -q R_DL -l select=1:node_type= skl:mem=12gb,walltime=09:00:00 -q smp
```

- Navigate to the workspace:

Special prompt of interactive batch job

s12345 n011101 000$ **ws_list**

> MYSCR = $(ws_find wsday2)

> cd $MYSCR (if **ws** path is stored/full path for wspart2)

(please check that you are in your workspace!)

Requirements for the exercises

To work with **Jupyter notebook** on **compute node &**

- Initialize your HPC working environment in your **ws**:

  > `. notebooks/modules.sh`

- To access the notebook, **copy URL**:

  > `http://n110010:8888/?token=...`

See next slide

Requirements for the exercises

Login information and password for laptops

- Browser for the Jupyter Notebooks:

  Jupyter-Chrome

  DO NOT use it as an Internet browser.

  Use e.g. **Firefox** instead

- Website with all the course material:

https://fs.hlrs.de/projects/par/events/2024/DL-HLRS/

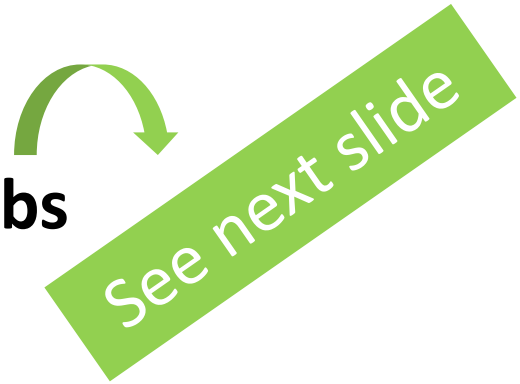
Requirements for the exercises

To work in **terminal** & HPC environment

> cd **notebooks**

- Submit job:

  > **qsub job_test.pbs**



- Check status:

  > **qstat** # status of your job

  > **qstat -q** # available queues