**TP // Machine Learning – 2A**

# Practical Session

**Supervisor: Pierre-Alain MOELLIC**

- **Team**: 2 students
- **Validation**: A ZIP archive containing a PDF report + codes. ZIP must be put on the SEAFILE platform (no password) [https://seafile.emse.fr/]. SEAFILE access link is sent by email to pierre-alain.moellic@cea.fr and kevin.hector@cea.fr
- **STRICTLY Follow** the requirements below

**Deadline: to be announced…**

## Initial comment

**ML tools & libraries.** This Practical Session is essentially focused on Deep Learning. You may choose between two available platforms:

- **TENSORFLOW** (using KERAS) [doc & tutorials: https://www.tensorflow.org]
- **PYTORCH** [doc & tutorials: https://pytorch.org/]

Another useful Machine Learning platform is **SKLEARN**: https://scikit-learn.org/stable/

Moreover, additional resources are linked to each ML course. Please, be careful and think before copy & paste.

**Dev. environment.** It's better to use your own laptop with your PYTHON environment (with ANCADONCA it works). However, you can use GOOGLE COLAB: https://colab.research.google.com/

**Using chatGPT-like tools.** If you use chatGPT-like platforms to generate codes: (1) Say it ! (2) Have a critical review from what is generated, (3) Show how you improve the code.

**Cheating issue.** If one group copy from another one (same codes, same results…): note will be set to 0 for both groups (i.e., automatically retake exam…).

## Report requirements

Your report will be evaluated, **IF AND ONLY IF** the following requirements are met:

- Report must be written with PLM Latex following the LaTeX template available on eCAMPUS. Follow the instruction on eCAMPUS. **No other templates are allowed.** (Exception TYPST, cf eCAMPUS)
- Access to PLM Latex: Portail EMSE > "Mon Espace" > "Outils Collaboratifs" > "PLMlatex"
- NB: PLM Latex is exactly like OVERLEAF
- **MAX NUMBER OF PAGES = 10** (additional content will not be evaluated)
- Report could be in French or English. Use a spell checker (you have been warned… malus points are possible).

- Please avoid dirty screenshots of your results. Prefer clean and understandable tables.
- Your project = PDF report (generated with PLM Latex) + python codes. All sent in an (ZIP) archive file using SEAFILE (https://seafile.emse.fr/)

**You can do more than what is asking (bonus point).**

Table des matières

# 1. The datasets: CIFAR-3

## 1.1. Description

The dataset has been sampled from CIFAR-10, one of the most popular Machine Learning dataset. We name the dataset **CIFAR-3**, since it contains only 3 labels. It represents 18000 32x32 pixels images from 3 classes: 0=Automobile, 1=Deer, 2=Horse. A grayscale version is proposed and named **CIFAR-3-GRAY**.

## 1.2. Load the datasets

Download CIFAR-3 and CIFAR-3-GRAY on eCAMPUS: CIFAR-3.zip. The archive contains 3 numpy arrays:

- Color images: **X_cifar.npy**, a 18000x32x32x3 tensor
- Same images in grayscale (CIFAR-3-GRAY): **X_cifar_grayscale.npy**, a 18000x32x32 tensor.
- The labels: **Y_cifar.npy**, a 18000 one dimensional tensor. Labels are coded with integer: "0", "1" and "2".
- Note that CIFAR-3 and its grayscale counterpart CIFAR-3-GRAY have obviously the same labels.

Check the tensors' size:

```
>>> import numpy as np
>>> X=np.load('X_cifar.npy')
>>> np.shape(X)
(18000, 32, 32, 3)
```

## 1.3. Tips

Here is an example for displaying a picture with matplotlib:

```
######## Plot a sample picture
nb_sample = 123
plt.imshow(X[nb_sample])
img_title='Classe ' + str(y[nb_sample])
plt.title(img_title)
plt.show()
plt.clf()
```

NB: For the grayscale image, use imshow(X_gray[nb_sample],**cmap='gray'**)

## 1.4. Works / Questions

| | QUESTION / WORK | SCALE |
|---|---|---|
| #1 | What are the shape of the data? What are the min and max value of the pixels? Is it better to normalize the data to work in [0,1]? Display samples from the dataset | 1 point |
| #2 | Use the sklearn method ***train_test_split*** to split the dataset in one *train set* and one *test set*. Why this split is important in Machine Learning? | 2 points |
| #3 | Are the *train* and *test* sets well balanced (distribution of labels)? Why is it important for supervised Machine Learning? | 1 point |

# 2. DIMENSIONALITY REDUCTION WITH THE PCA

## Dataset

**CIFAR-3-GRAY (or CIFAR-3)**

## Objectives

Pictures from CIFAR-3-GRAY have 1024 pixels (32x32 grayscale picture), i.e. $x \in X \subset \mathbb{R}^{1024}$ we are in a (relatively) high-dimensional space. It is difficult to "see" how the data are distributed and if some intrinsic characteristics between the features exist that could help further analysis / tasks.

## Works / Questions

| | QUESTION / WORK | SCALE |
|---|---|---|
| #1 | Perform a *Principal Component Analysis* (PCA) with sklearn. You will need to reshape<br>Try to keep different *n_components*. | 1 point |
| #2 | An interesting feature is *PCA.explained_variance_ratio_*<br>Explain these values according to your understanding of PCA and use these values to fit a relevant value for *n_components* | 2 point |
| #3 | Display a CIFAR-3-GRAY picture with 5 values of *n_components* | 1 point |

## Resources

- https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html
- You will need to reshape your data to perform the PCA, because PCA expects a 2d matrix (for example: np.reshape(X,(18000,32*32)) ➔ you put all the pixels into a single row)
- For question #3, see the course #2 (slide 40).

# 3. Supervised Machine Learning

## 3.1. Logistic Regression & Gaussian Naïve Bayes Classifier

## Dataset

**CIFAR-3-GRAY**

With sklearn, and the train_test_split method, you already split the database into a train and test datasets.

## Objectives

The main objective is to perform a supervised classification task using two classical methods:

- (multiclass) Logistic Regression.
- Gaussian Naïve Bayes Classifier.

Apply these two methods on CIFAR-3-GRAY and on a compressed version of CIFAR-3-GRAY you obtained thanks to the CPA in the previous section.

### Resources

Here is the documentation to perform Logistic Regression and Gaussian Naïve Bayes classifier with sklearn:

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html

### Questions / Works

| | QUESTION / WORK | SCALE |
|---|---|---|
| #1 | What is the major difference between Naïve Bayes Classifier and Logistic Regression? (check course #2, a clue: what are we trying to predict?) | 1 point |
| #2 | With sklearn, train your LR and NBC models. Comment the results. If any, check how to modify the parameters and comment how it influences the results. | 2 points |
| #3 | With the score method, compute the accuracy of the model on the training and the test datasets. Why do we need to analyze the performance of the model at training and testing time? | 1 point |
| #4 | Same with a "compressed" version of CIFAR-3-GRAY with your PCA. Comment. | 2 points |

### Help, advices

What are the (strong) hypothesis for NBC (and LR)? For NBC, what happens with a very large value of var_smoothing (compare to the random-guess level)?

## 3.2. Deep Learning / MULTILAYER PERCEPTRON (MLP)

### Dataset

**CIFAR-3-GRAY**

### Objective and tool

Same as before: we want to perform a supervised image classification task.

For a model based on neural networks, the development platform will be TENSORFLOW (with KERAS included) or PYTORCH. The documentation and tutorials are here: https://www.tensorflow.org/tutorials/ for TensorFlow and here https://pytorch.org/ for PyTorch. More examples:

- TensorFlow: MLP https://www.tensorflow.org/datasets/keras_example, CNN https://www.tensorflow.org/tutorials/images/cnn
- PyTorch: MLP: https://colab.research.google.com/github/csc-training/intro-to-dl/blob/master/day1/optional/pytorch-mnist-mlp.ipynb , CNN: https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html

Both enable simple model design in a pure "Lego style": you just need to define each layer and connect one to the following (remember "feedforward network"?). For example, with TENSORFLOW:

```
input_layer = Input(shape=(nb_features,))
x = Dense(123)(inputs)
x = ReLU()(x)
x = Dropout(0.25)
x = Dense(54)(x)
x = ReLU()(x)
x = Dense(num_classes)(x)
output_layer = Softmax()(x)
```

```
model=Model(input_layer, output_layer)
```

Then, you need to **`compile`** with some important neural networks characteristics, **`fit`** with your data (here you can define the validation set) and evaluate on $X^{test}$.

Your first deep neural network is a MultiLayer Perceptron (MLP), i.e. a feedforward network only composed of fully connected layers (named "Dense" in Tensorflow and "Linear" in PyTorch). For your first attempt, use only one hidden layer.

### Questions / Works

| | QUESTION / WORK | SCALE |
|---|---|---|
| #1 | What is the size of the input tensor? What is the size of the output layer? | 1 point |
| #2 | How many epochs do you use? What does it mean? What is the batch_size? What does it means? | 1 point |
| #3 | Why do we need to define a validation set? | 1 point |
| #4 | Pick the most important hyper-parameters you have to set to run the training process (e.g., optimizer…). Briefly explain why are they important (i.e. their influence). | 2 points |
| #5 | Comment the training results. | 1 point |
| #6 | Is there any overfitting? Why? If yes, what could be the causes? How to fix this issue?<br>If you do not observe overfitting, how can you make your model overfit? Try and demonstrate the overfitting. | 4 points |
| #7 | According to this first performance, change the architecture of the MLP (change parameters, add/remove layers…) as well as hyper-parameters, explain why, what are the influence on the results…? | 3 points |

## 3.3. Deep Learning / CONVOLUTIONNAL NEURAL NETWORK (CNN)

### Dataset

**CIFAR-3**

### Objective and tool

Same as before: but now we use the color images and process a Convolutional Neural Network (CNN).

### Questions / Works

| | QUESTION / WORK | SCALE |
|---|---|---|
| #1 | What is the size of the input tensor? Why it is not the same as for your previous MLP model? | 1 point |
| #2 | Comment the training results. | 1 point |
| #3 | Is there any overfitting? Why? If yes, what could be the causes? How to fix this issue?<br>If you do not observe overfitting, how can you make your model overfit? Try and demonstrate the overfitting. | 4 points |
| #4 | According to this first performance, change the architecture of the CNN (add/remove kernels, add/remove layers…) as well as hyper-parameters, explain why, what are the influence on the results…? | 3 points |
| #5 | Final comment on the MLP/CNN comparison. | 2 points |