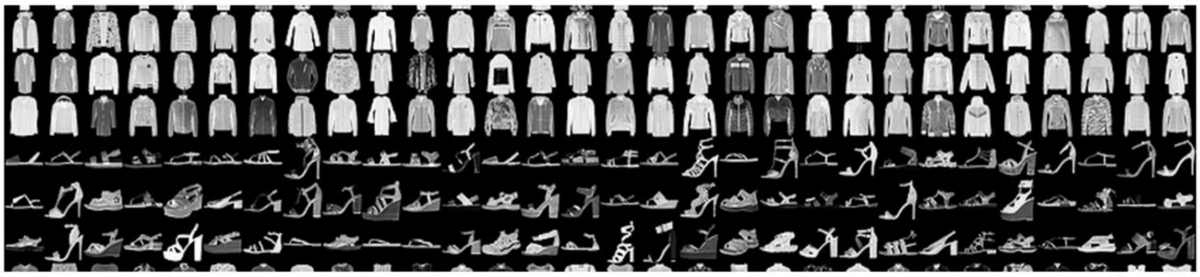
- **Deadline:** November 28, 23:59.
- **Objectives**
  - In this practice we will develop a classical neural network model for the Fashion-MNIST dataset (`https://keras.io/api/datasets/fashion_mnist/`).
- **Dataset**



  - Fashion-MNIST is a dataset of article images consisting of a training set of 60,000 examples and a test set of 10,000 examples.
  - Each example is a 28x28 grayscale image, associated with a label from the following 10 classes: (0) T-shirt/top, (1) Trouser, (2) Pullover, (3) Dress, (4) Coat, (5) Sandal, (6) Shirt, (7) Sneaker, (8) Bag and (9) Ankle boot.
  - The dataset can be easily downloaded from Keras using the following instructions:

```
from keras.datasets import fashion_mnist
(x_train , y_train), (x_test , y_test) = fashion_mnist.load_data()
```

- **Tasks to be carried out**

  1. **Preprocessing**.
     - Preprocess the dataset to prepare it to feed the neural network: flatten the images, convert integers to float values, encode the labels using the one-hot encoding, etc.
     - Divide the train data into train and validation, using this last one as a reference for hyperparameter tunning.
  2. **Develop a classical neural network to predict class of each image**.
     - Decide the structural hyperparameters of the network: layers, neurons per layer, activation functions, etc.
     - Decide the learning hyperparameters of the network: optimizer, learning rate, epochs, batch size, metrics, etc.
     - Justify the decisions made.
  3. **Regularization**.
     - Take some regularization techniques to avoid overfitting: dropout, batch normalization, weight regularization and initialization, etc.
  4. **Results**.
     - Comment the results obtained at each step.
     - Analyze the results using different metrics.

4. **Results.**

- Comment the results obtained at each step.
- Analyze the results using different metrics.
- Here you may find it useful to use the library `scikit-learn` y el método `classification_report()` (see `https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html`) that builds a text report showing the main classification metrics.
- Example:

https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html

```
from sklearn.metrics import classification_report

# Evaluate the model
predicted_values = model.predict(test_images)
predicted_classes = np.argmax(predicted_values, axis=1)

# Classification report for precision, recall, and f1-score
report=classification_report(test_labels, predicted_classes, target_names=[
    'T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',
    'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot'
])

# Print the evaluation metrics
print("Classification Report:")
print(report)
```

- Results example:

https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html

```
Classification Report:
             precision recall    f1-score  support

T-shirt/top  0.85      0.84      0.84      1000
Trouser      0.98      0.97      0.98      1000
Pullover     0.84      0.69      0.76      1000
Dress        0.87      0.91      0.89      1000
Coat         0.71      0.89      0.79      1000
Sandal       0.98      0.95      0.97      1000
Shirt        0.74      0.66      0.70      1000
etc.
```

5. **Conclusions.**

- Comment on the results obtained by each model.
- Make a reasoned comparison of the results obtained, where they have improved, worsened, etc.
- Comment advantages, disadvantages of the different methods, conclusions obtained and other aspects of interest.
- It is advisable to include a final graph or table summarizing all the results obtained by the different models.