Neural Networks at Work

How to implement and manage a Neural Network

Ing. Zese Riccardo
Ing. Bizzarri Alice
alice.bizzarri@unife.it

AIDA4Edge
UK Research and Innovation
Funded by the European Union
University of Ferrara

NN Frameworks

- There are many frameworks available for working with NNs.
    - Tensorflow
    - PyTorch
    - CNTK
    - Theano
    - Caffe
    - …

AIDA4Edge UK Research and Innovation Funded by the European Union University of Ferrara

NN Frameworks

An interesting library is Keras (https://keras.io/).
From the website:

Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. It was developed with a focus on enabling fast experimentation. Being able to go from idea to result with the least possible delay is key to doing good research.

AIDA4Edge UK Research and Innovation Funded by the European Union University of Ferrara

Keras

- Contains a set of classes that abstracts the underlying NN library giving a unified interface to the systems Tensorflow, CNTK, and Theano.
- In practice, you can use Keras to define a network and run it using one of the three systems above without changing anything in the code.
- Written in Python, allows users to configure complicated models directly in Python
- Uses either CPU or GPU for computation
- Uses numpy data structures and a similar command structure to scikit-learn (model.fit , model.predict, etc.)

AIDA4Edge
UK Research and Innovation
Funded by the European Union
University of Ferrara

Keras and Tensorflow

- Recently, Tensorflow adopted Keras as its official frontend to simplify the definition and the management of the code.
  - It contains keras as a submodule.
- Before version 2, users could choose whether to use the original Tensorflow interface and functions or use the Keras module
- From version 2, the actual one, Tensorflow has been redesigned to use Keras only. Old code that does not use Keras cannot be executed if a previous version of Tensorflow is not installed.

AIDA4Edge
UK Research and Innovation
Funded by the European Union
University of Ferrara

Tensorflow and PyTorch

To install Tensorflow (https://www.tensorflow.org/) simply run the command

```
pip install tensorflow
```

To install PyTorch (https://pytorch.org/) simply run the command

```
pip install torch
```


AIDA4Edge UK Research and Innovation Funded by the European Union University of Ferrara

Implementing the network

Tensorflow vs PyTorch

AIDA4Edge
UK Research and Innovation
Funded by the European Union
University of Ferrara

Common concepts

- We can define both as *multi concept libraries*:
    - Deep Learning Libraries
    - Tensor computation libraries
        - With a strong GPU capability
- A **TENSOR** can be seen as:
    - A mathematical object that we can manipulate using linear algebra
    - A software object representing a data structure
- Pytorch defines a class called ***torch.Tensor*** that defines these tensor objects (very similar to Numpy Arrays) that can operate on a device supporting CUDA (Nvidia GPUs)
- Tensorflow defines a class called ***tensorflow.Tensor*** similar to Pytorch

AIDA4Edge
UK Research and Innovation
Funded by the European Union
University of Ferrara

Tensor

Scalar

Rank-0 tensor

```
import torch

a = torch.tensor(1.)
a.shape
✓ 2.6s

torch.Size([])
```

Vector

Rank-1 tensor

```
import torch

a = torch.tensor([1., 2., 3.])
a.shape
✓ 0.0s

torch.Size([3])
```

Matrix

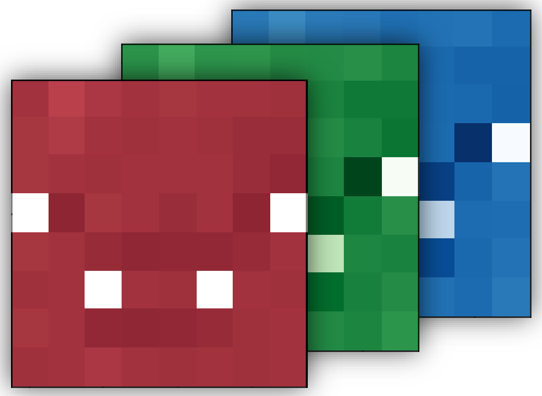
Rank-2 tensor

```
import torch

a = torch.tensor([[1., 2., 3.],
                  [4., 5., 6.]])
a.shape
✓ 0.0s

torch.Size([2, 3])
```


AIDA4Edge
UK Research and Innovation
Funded by the European Union
University of Ferrara

Tensor

An RGB image is nothing more than a stack of matrices
A 3D Rank-3 tensor



```python
import torch

a = torch.tensor([[[1., 2., 3.],
                   [4., 5., 6.]],
                  [[7., 8., 9.],
                   [10., 11., 12.]]],)
a.shape
```

✓ 0.0s

```
torch.Size([2, 2, 3])
```

Pipeline

With both frameworks, the following pipeline is implemented:

1. Definition of the dataset
2. Model definition
3. Definition of the training cycle
4. Train the model
5. Evaluate the model

AIDA4Edge UK Research and Innovation Funded by the European Union University of Ferrara

TensorFlow

Introduction and overview

AIDA4Edge
UK Research and Innovation
Funded by the European Union
University of Ferrara

Tensorflow

```
import tensorflow as tf

(x_train, y_train),(x_test, y_test) =
tf.keras.datasets.mnist.load_data()

model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(input_shape=(28, 28)),
  tf.keras.layers.Dense(512, activation=tf.nn.relu),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(10, activation=tf.nn.softmax)
])
model.compile(optimizer='adam',loss='sparse_categorical_crossentropy',
            metrics=['accuracy'])
model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test)
```


AIDA4Edge UK Research and Innovation Funded by the European Union University of Ferrara

Tensorflow

```
import tensorflow as tf

(x_train, y_train),(x_test, y_test) = tf.keras.datasets.mnist.load_data()

model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(input_shap
  tf.keras.layers.Dense(512, activat
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(10, activati
])
model.compile(optimizer='adam',loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test)
```

Downloads the MNIST dataset and loads the dataset dividing it in training and test sets

AIDA4Edge UK Research and Innovation Funded by the European Union University of Ferrara

Tensorflow

```
import tensorflow as tf

(x_train, y_train),(x_test, y_test) = tf.kera                    a()

model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(input_shape=(28, 28)),
  tf.keras.layers.Dense(512, activation=tf.nn.relu),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(10, activation=tf.nn.softmax)
])
model.compile(optimizer='adam',loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test)
```

Definition of the model

AIDA4Edge
UK Research and Innovation
Funded by the European Union
University of Ferrara

Tensorflow

```
import tensorflow as tf

(x_train, y_train),(x_test, y_test) = tf.keras.datasets.mnist.load_data()

model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(input_shape=(28, 28
  tf.keras.layers.Dense(512, activation=tf.nn
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(10, activation=tf
])
model.compile(optimizer='adam',loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test)
```

Definition of the optimizer, loss function and metrics to consider.

Tensorflow

```
import tensorflow as tf

(x_train, y_train),(x_test, y_test) = tf.keras.datasets.mnist.load_data()

model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(input_shape=(28, 28)),
  tf.keras.layers.Dense(512, activation=tf.nn.relu),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(10, activation=tf.n
])
model.compile(optimizer='adam',loss='sparse                          ,
              metrics=['accuracy'])
model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test)
```

Training of the net. We don't have to compute loss or any other metric. The fit method will do the work for us.

Tensorflow

```
import tensorflow as tf

(x_train, y_train),(x_test, y_test) = tf.keras.datasets.mnist.load_data()

model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(input_shape=(28, 28)),
  tf.keras.layers.Dense(512, activation=tf.nn.relu),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(10, activation=tf.nn.softmax)
])
model.compile(optimizer='adam',loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test)
```

The test set is used to evaluate the trained model.

AIDA4Edge UK Research and Innovation Funded by the European Union University of Ferrara

Tensorflow

```
import tensorflow as tf

(x_train, y_train),(x_test, y_test) = tf.keras.datasets.mnist.load_data()

model = tf.ke
  tf.keras.la
  tf.keras.la
  tf.keras.la
  tf.keras.layers.Dense(      activation=tf.nn.softmax)
])
model.compile(optimizer='adam',loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test)
```

As you can see the number of lines and operations is reduced to the minimum, making the definition and use of models a simple task. We will see each of these parts in detail.

TensorFlow – Output

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step

Epoch 1/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 22s 10ms/step - accuracy: 0.8436 - loss: 8.5909
Epoch 2/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 16s 8ms/step - accuracy: 0.9048 - loss: 0.4297
Epoch 3/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 16s 8ms/step - accuracy: 0.9104 - loss: 0.4110
Epoch 4/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 16s 9ms/step - accuracy: 0.9163 - loss: 0.3868
Epoch 5/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 16s 8ms/step - accuracy: 0.9245 - loss: 0.3479

313/313 ━━━━━━━━━━━━━━━━━━━━ 2s 5ms/step - accuracy: 0.9260 - loss: 0.3047
[0.277653306722641, 0.9340999722480774]
```


AIDA4Edge
UK Research and Innovation
Funded by the European Union
University of Ferrara

Typical Command Structure in Tensorflow

1. Build the structure of your network.
2. Compile the model, specifying your loss function, metrics, and optimizer (which includes the learning rate).
3. Fit the model on your training data (specifying batch size, number of epochs)
4. Predict on new data
5. Evaluate your results

AIDA4Edge UK Research and Innovation Funded by the European Union University of Ferrara

Building the model in TensorFLow

- Tensorflow provides two approaches for building the structure of your model:
    - Sequential Model: it allows a linear stack of layers – simpler and more convenient if the model has this form
    - Functional API: more detailed and complex, but it allows more complicated architectures
- We saw the Sequential Model.

AIDA4Edge UK Research and Innovation Funded by the European Union University of Ferrara

Sequential Model VS Functional API

- Add some slide to compere it

AIDA4Edge UK Research and Innovation Funded by the European Union University of Ferrara

PyTorch

Introduction and overview

AIDA4Edge
UK Research and Innovation
Funded by the European Union
University of Ferrara

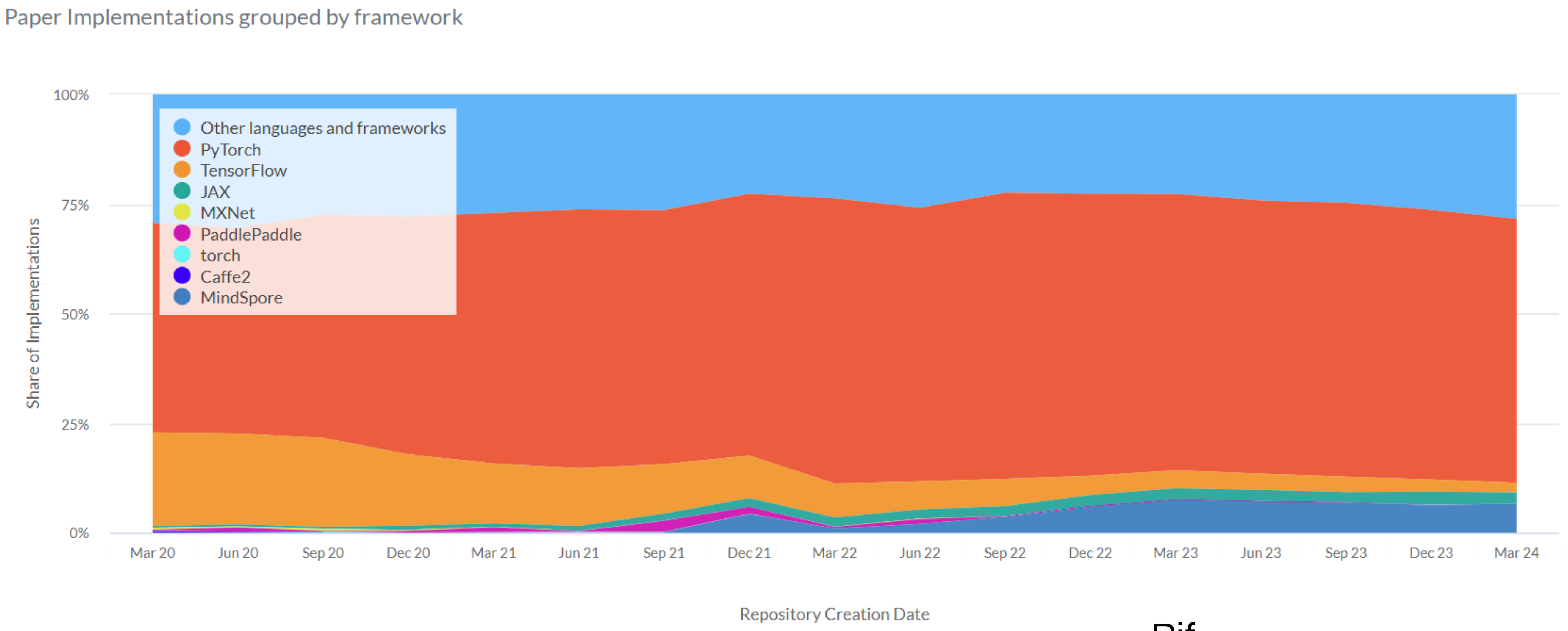
Pytorch

Libreria di machine learning open-source sviluppata inizialmente da Meta AI (Facebook)
Attualmente parte di Linux Foundation
E’ una delle più note e usate libreria di sviluppo di sistemi di machine e deep learning alla pari di TensorFlow (Google)
Tra i più noti progetti sviluppati in Pytorch c’è l’**autopilot di Tesla**

La libreria è scritta in Python

AIDA4Edge
UK Research and Innovation
Funded by the European Union
University of Ferrara

Perché Pytorch?



Rif. https://paperswithcode.com/trends

Pipeline

With both frameworks, the following pipeline is implemented:

1. Definition of the dataset
2. Model definition
3. Definition of the training cycle
4. Train the model
5. Evaluate the model

AIDA4Edge
UK Research and Innovation
Funded by the European Union
University of Ferrara

Definition of the dataset

1. Loading data
2. (optional) split for training, validation and testing
3. Dataloader creation to cycle through the dataset

AIDA4Edge UK Research and Innovation Funded by the European Union University of Ferrara

Loading data (basic)

```python
from torchvision.datasets import ImageFolder
from torchvision import transforms
from torch.utils.data.dataset import random_split

data_augment =  transforms.Compose([
                transforms.Resize((256, 256), interpolation=transforms.InterpolationMode.BICUBIC),
                transforms.ToTensor(),
            ])

train_dataset = ImageFolder(root="path/to/the/dataset", transform=data_augment)
train_dst, valid_dst = random_split(train_dataset, lengths=[50000, 5000])
```

Data Augmentation

Loading data

Split for training and validation

AIDA4Edge UK Research and Innovation Funded by the European Union University of Ferrara

Lettura dei dati (advanced)

- A class is created that inherits from the Dataset class of torch
- In the constructor you put all paths, data lists, default transformations, etc.
- There must always be 2 methods:
  - **__len__:** Serves to get the size of the dataset
  - **__getitem__:** Used to actually get the data from the dataset to be used in the training phase

```python
class SegmentDataLoader(Dataset):
    def __init__(self, root1, root2, root3, mask_folder, img_size, transform):
        self.img_folderC = root1
        self.img_folderM01 = root2
        self.img_folderM02 = root3
        self.mask_folder = mask_folder
        self.transform = transform
        self.img_size = img_size

        transform2 = transforms.Compose([
            transforms.Resize((self.img_size, self.img_size)),
            transforms.ToTensor(),
            transforms.Grayscale()
        ])

        self.transform2 = transform2

        self.folders, self.images = [], []

        self.imagesC = os.listdir(self.img_folderC)
        self.imagesM01 = os.listdir(self.img_folderM01)
        self.imagesM02 = os.listdir(self.img_folderM02)
        self.labels = os.listdir(self.mask_folder)

    def __len__(self):
        return len(self.imagesC)

    def __getitem__(self, index):
        imgC = os.path.join(self.img_folderC, self.imagesC[index])
        imgM01 = os.path.join(self.img_folderM01, self.imagesM01[index])
        imgM02 = os.path.join(self.img_folderM02, self.imagesM02[index])
        label = os.path.join(self.mask_folder, self.labels[index])

        img_c = self.transform(Image.open(imgC).convert("RGB"))
        img_m1 = self.transform(Image.open(imgM01).convert("RGB"))
        img_m2 = self.transform(Image.open(imgM02).convert("RGB"))
        label_t = self.transform2(Image.open(label).convert("RGB"))

        return {'c': img_c, 'm1': img_m1, 'm2': img_m2, 'label': label_t}
```


AIDA4Edge
UK Research and Innovation
Funded by the European Union
University of Ferrara

Creation of Dataloader

- The Dataloader is a torch class that allows you to have a dataset loaded and used in an iterable way

- It allows you to do various things with the read data:
    - Batch data
    - Shuffling
    - Parallel loading of data
    - Data augmentation

```python
from torchvision.datasets import ImageFolder
from torchvision import transforms
from torch.utils.data.dataset import random_split
from torch.utils.data import DataLoader

data_augment =  transforms.Compose([
                transforms.Resize((256, 256), interpolation=transforms.InterpolationMode.BICUBIC),
                transforms.ToTensor(),
            ])

train_dataset = ImageFolder(root="path/to/the/dataset", transform=data_augment)
train_dst, valid_dst = random_split(train_dataset, lengths=[50000, 5000])

dataloader = DataLoader(train_dst, batch_size=32, shuffle=True)
```


AIDA4Edge UK Research and Innovation Funded by the European Union University of Ferrara

Model definition (basic)

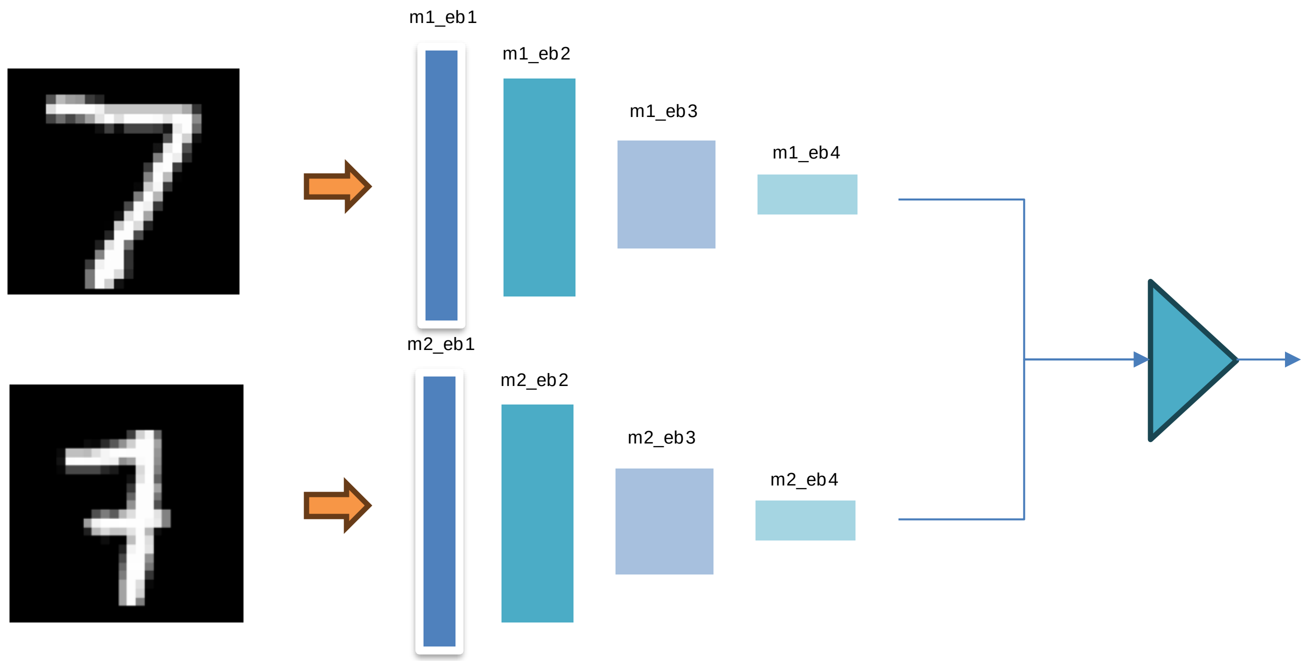
Network structure in the class constructor

Network forward pass

```python
class FirstCNN(torch.nn.Module):
    def __init__(self, num_classes):
        super(FirstCNN, self).__init__()

        self.conv1 = torch.nn.Conv2d(in_channels=3, out_channels=32)
        self.conv2 = torch.nn.Conv2d(in_channels=32, out_channels=64)
        self.conv3 = torch.nn.Conv2d(in_channels=64, out_channels=128)
        self.conv4 = torch.nn.Conv2d(in_channels=..., out_channels=...)

        self.fullyC = torch.nn.Linear(..., num_classes)
        self.activation

    def forward(self, input_data):
        x = self.conv1(input_data)
        x = self.conv2(x)
        x = self.conv3(x)
        x = self.conv4(x)
        ...
        output = self.fullyC(x)
        return output
```


AIDA4Edge UK Research and Innovation Funded by the European Union University of Ferrara

Model definition (advanced)




```python
class Encoder(nn.Module):
    def __init__(self, in_channels=3, base_width=64, tiler = None):
        super(Encoder, self).__init__()
        # Encoder 1 ---------------------------------------------------------------------------
        self.m1_eb1 = nn.Sequential(
            nn.Conv2d(in_channels, base_width, kernel_size=5, stride=1, padding=2, bias=False),
            nn.BatchNorm2d(base_width),
            nn.GELU(),
            nn.MaxPool2d(kernel_size=2, stride=2),
        )
        self.m1_eb2 = nn.Sequential(
            nn.Conv2d(base_width, base_width*2, kernel_size=5, stride=1, padding=2, bias=False),
            nn.GELU(),
            nn.Conv2d(base_width*2, base_width*4, kernel_size=5, stride=1, padding=2, bias=False),
            nn.BatchNorm2d(base_width*4),
            nn.GELU(),
            nn.MaxPool2d(kernel_size=2, stride=2),
        )
        self.m1_eb3 = nn.Sequential(
            nn.Conv2d(base_width*4, base_width*8, kernel_size=5, stride=1, padding=2, bias=False),
            nn.GELU(),
            nn.MaxPool2d(kernel_size=2, stride=2),
        )
        self.m1_eb4 = nn.Sequential(
            nn.Conv2d(base_width*8, base_width*8, kernel_size=3, stride=1, padding=1, bias=False)
        )
        # Encoder 2 ---------------------------------------------------------------------------
        self.m2_eb1 = nn.Sequential(
            nn.Conv2d(in_channels, base_width, kernel_size=3, stride=1, padding=1, bias=False),
            nn.BatchNorm2d(base_width),
            nn.GELU(),
            nn.MaxPool2d(kernel_size=2, stride=2),
        )
        self.m2_eb2 = nn.Sequential(
            nn.Conv2d(base_width, base_width*2, kernel_size=3, stride=1, padding=1, bias=False),
            nn.GELU(),
            nn.Conv2d(base_width*2, base_width*4, kernel_size=3, stride=1, padding=1, bias=False),
            nn.BatchNorm2d(base_width*4),
            nn.GELU(),
            nn.MaxPool2d(kernel_size=2, stride=2),
        )
        self.m2_eb3 = nn.Sequential(
            nn.Conv2d(base_width*4, base_width*8, kernel_size=3, stride=1, padding=1, bias=False),
            nn.GELU(),
            nn.MaxPool2d(kernel_size=2, stride=2),
        )
        self.m2_eb4 = nn.Sequential(
            nn.Conv2d(base_width*8, base_width*8, kernel_size=1, stride=1, padding=0, bias=False)
        )

    def forward(self, x1, x2):
        # enc1
        m1_s1 = self.m1_eb1(x1)
        m1_s2 = self.m1_eb2(m1_s1)
        m1_s3 = self.m1_eb3(m1_s2)
        m1_ls = self.m1_eb4(m1_s3)
        # enc2
        m2_s1 = self.m2_eb1(x2)
        m2_s2 = self.m2_eb2(m2_s1)
        m2_s3 = self.m2_eb3(m2_s2)
        m2_ls = self.m2_eb4(m2_s3)
        # fusion
        lsc = torch.cat([m1_ls, m2_ls], dim=1)
        ls = self.combo_latent(lsc)
        return ls
```


AIDA4Edge
UK Research and Innovation
Funded by the European Union
University of Ferrara

Training loop definition

1. Instantiate the model
2. Move it to the GPU
3. Instantiate the optimiser
4. Loss definition
5. For looping on data

AIDA4Edge UK Research and Innovation Funded by the European Union University of Ferrara

Training loop definition

```
model = FirstCNN(num_classes=10)
model = model.cuda()
optimizer = torch.optim.Adam(model.parameters(), lr=learning_rate)
CE_loss = torch.nn.CrossEntropyLoss()

for epoch in range(num_epochs):
    model.train()
    for batch in train_dataloader:
        img = batch[0].cuda()
        labels = batch[1].cuda()

        # forward pass
        output = model(img)
        loss = CE_loss(img, labels)

        # backpropagation
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
```

1. Instantiate the model
2. Move it to the GPU
3. Instantiate the optimiser
4. Loss definition
5. For looping on epochs
    1. Network in training mode
    2. Move the data to the GPU
    3. Pass the batch through the network
    4. Calculate loss
    5. I reset the gradients to zero so they don't accumulate (so they are calculated correctly on each iteration)
    6. I calculate the gradients
    7. I update the weights

AIDA4Edge
UK Research and Innovation
Funded by the European Union
University of Ferrara

PyTorch

```
import torch
import torch.nn as nn
import torch.optim as optim
import torch.nn.functional as F
from torchvision import datasets, transforms
from torch.utils.data import DataLoader

transform = transforms.Compose([transforms.ToTensor(),
transforms.Normalize((0.5,), (0.5,))])
train_dataset = datasets.MNIST(root="./data", train=True,
transform=transform, download=True)
test_dataset = datasets.MNIST(root="./data", train=False,
transform=transform, download=True)

train_loader = DataLoader(train_dataset, batch_size=64, shuffle=True)
test_loader = DataLoader(test_dataset, batch_size=64, shuffle=False)
```

Downloads the MNIST dataset and loads the dataset dividing it in training and test sets

AIDA4Edge UK Research and Innovation Funded by the European Union University of Ferrara

PyTorch

```
class NeuralNet(nn.Module):
    def __init__(self):
        super(NeuralNet, self).__init__()
        self.flatten = nn.Flatten()
        self.fc1 = nn.Linear(28 * 28, 512)
        self.dropout = nn.Dropout(0.2)
        self.fc2 = nn.Linear(512, 10)

    def forward(self, x):
        x = self.flatten(x)
        x = F.relu(self.fc1(x))
        x = self.dropout(x)
        x = self.fc2(x)
        return F.log_softmax(x, dim=1)
```

Definition of the model

AIDA4Edge
UK Research and Innovation
Funded by the European Union
University of Ferrara

PyTorch

```
# Initialize model, loss function, and optimizer
model = NeuralNet()
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)
```

Definition of the optimizer, loss function and metrics to consider and print.

PyTorch

Definition of the training loop.

```
# Training loop
def train(model, train_loader, criterion, optimizer, epochs=5):
    model.train()
    for epoch in range(epochs):
        for images, labels in train_loader:
            optimizer.zero_grad()
            outputs = model(images)
            loss = criterion(outputs, labels)
            loss.backward()
            optimizer.step()
        print(f"Epoch {epoch+1}, Loss: {loss.item():.4f}")
```


AIDA4Edge UK Research and Innovation Funded by the European Union University of Ferrara

PyTorch

Definition of the evaluation loop.

```
# Evaluation loop
def evaluate(model, test_loader):
    model.eval()
    correct = 0
    total = 0
    with torch.no_grad():
        for images, labels in test_loader:
            outputs = model(images)
            _, predicted = torch.max(outputs, 1)
            total += labels.size(0)
            correct += (predicted == labels).sum().item()
    print(f"Accuracy: {100 * correct / total:.2f}%")
```


AIDA4Edge UK Research and Innovation Funded by the European Union University of Ferrara

PyTorch

```
# Training loop
def train(model, train_loader, validation_loader, criterion,
optimizer, epochs=5):
    model.train()
    for epoch in range(epochs):
        for images, labels in train_loader:
            optimizer.zero_grad()
            outputs = model(images)
            loss = criterion(outputs, labels)
            loss.backward()
            optimizer.step()
        print(f"Epoch {epoch+1}, Loss: {loss.item():.4f}")
        evaluate(model, validation_loader)
```

We can use also evaluate function in definition of the training loop.

AIDA4Edge
UK Research and Innovation
Funded by the European Union
University of Ferrara

PyTorch

Training of the net.

```
# Train and evaluate
train(model, train_loader, criterion, optimizer, epochs=5)
evaluate(model, test_loader)
```

The test set is used to evaluate the trained model.

AIDA4Edge
UK Research and Innovation
Funded by the European Union
University of Ferrara

PyTorch – Output

```
Downloading https://ossci-datasets.s3.amazonaws.com/mnist/train-images-idx3-ubyte.gz
Downloading https://ossci-datasets.s3.amazonaws.com/mnist/train-images-idx3-ubyte.gz to ./data/MNIST/raw/train-images-idx3-ubyte.gz
100%|██████████| 9.91M/9.91M [00:00<00:00, 56.8MB/s]
Extracting ./data/MNIST/raw/train-images-idx3-ubyte.gz to ./data/MNIST/raw


Downloading https://ossci-datasets.s3.amazonaws.com/mnist/train-labels-idx1-ubyte.gz
Downloading https://ossci-datasets.s3.amazonaws.com/mnist/train-labels-idx1-ubyte.gz to ./data/MNIST/raw/train-labels-idx1-ubyte.gz
100%|██████████| 28.9k/28.9k [00:00<00:00, 2.07MB/s]
Extracting ./data/MNIST/raw/train-labels-idx1-ubyte.gz to ./data/MNIST/raw

Downloading https://ossci-datasets.s3.amazonaws.com/mnist/t10k-images-idx3-ubyte.gz
Downloading https://ossci-datasets.s3.amazonaws.com/mnist/t10k-images-idx3-ubyte.gz to ./data/MNIST/raw/t10k-images-idx3-ubyte.gz
100%|██████████| 1.65M/1.65M [00:00<00:00, 14.5MB/s]
Extracting ./data/MNIST/raw/t10k-images-idx3-ubyte.gz to ./data/MNIST/raw

Downloading https://ossci-datasets.s3.amazonaws.com/mnist/t10k-labels-idx1-ubyte.gz
Downloading https://ossci-datasets.s3.amazonaws.com/mnist/t10k-labels-idx1-ubyte.gz to ./data/MNIST/raw/t10k-labels-idx1-ubyte.gz
100%|██████████| 4.54k/4.54k [00:00<00:00, 1.65MB/s]
Extracting ./data/MNIST/raw/t10k-labels-idx1-ubyte.gz to ./data/MNIST/raw

Epoch 1, Loss: 0.1447
Epoch 2, Loss: 0.0435
Epoch 3, Loss: 0.1497
Epoch 4, Loss: 0.0357
Epoch 5, Loss: 0.2335
Accuracy: 97.19%
```

Typical Command Structure in PyTorch

1. Create a new class for your network.
    1. Initialize the layers in the contructor (**`__init__`**)
    2. Define how the layers are connected in method **`forward`**
2. Specify the loss function, metrics, and optimizer (which includes the learning rate).
3. Define the methods for training and evaluating your model
4. Fit the model on your training data (specifying batch size, number of epochs)
5. Predict on new data evaluating your results

AIDA4Edge
UK Research and Innovation
Funded by the European Union
University of Ferrara

Exercise

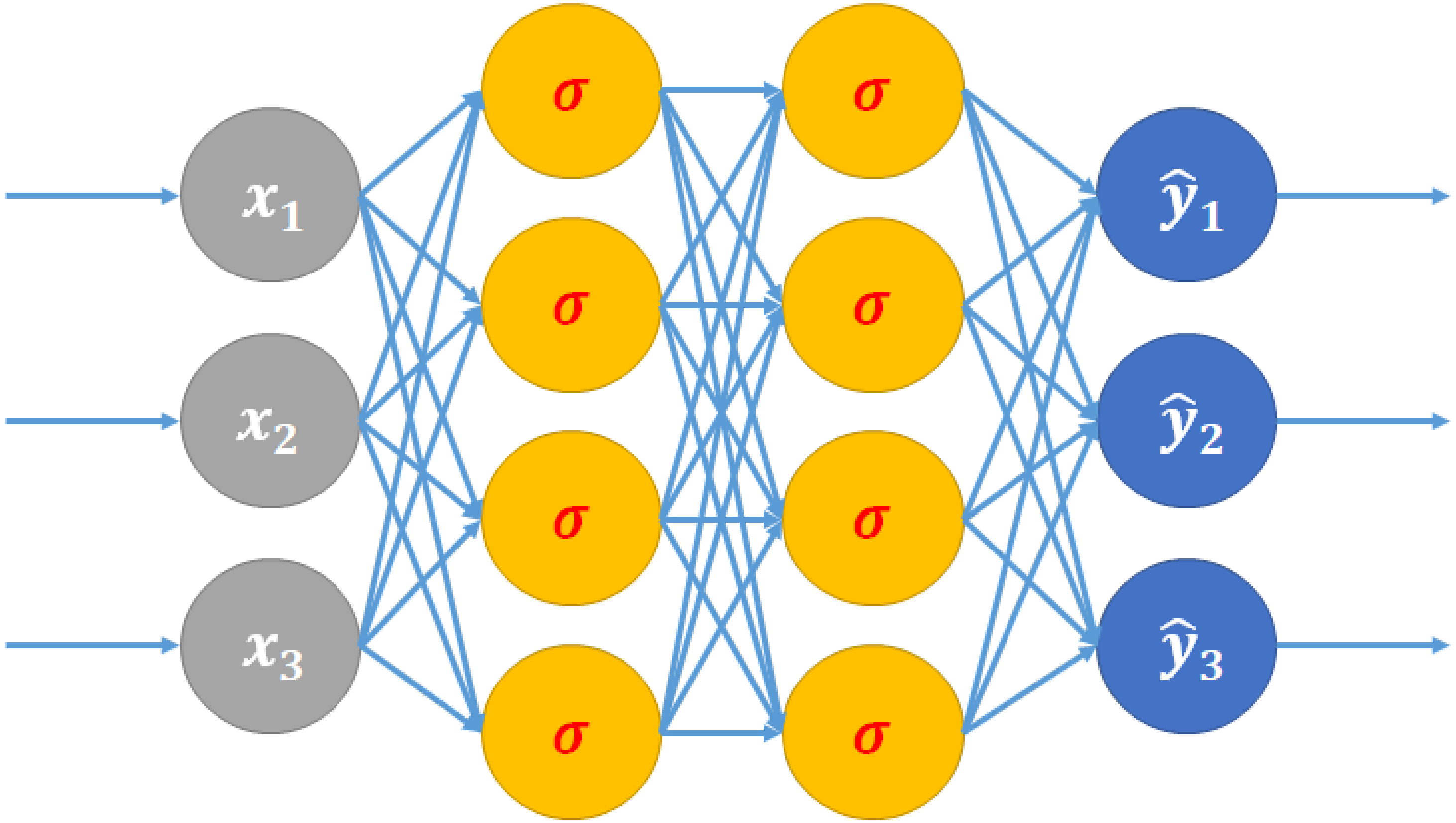
AIDA4Edge
UK Research and Innovation
Funded by the European Union
University of Ferrara

Network Design

Let’s go to https://colab.research.google.com/drive/1QyNIHgW44Z9Usj7J3sdKaypVrFzohG6T notebook to see how we can define a neural network.

AIDA4Edge
UK Research and Innovation
Funded by the European Union
University of Ferrara

Exercise



Exercise

First, import the Sequential function and initialize your model object:

```
from keras.models import Sequential
model = Sequential()
```


AIDA4Edge UK Research and Innovation Funded by the European Union University of Ferrara

Exercise

Then we add layers to the model one by one.

```
from keras.layers import Dense, Activation

# For the first layer, specify the input dimension
model.add(Dense(units=4, input_dim=3))

# Specify an activation function
model.add(Activation(sigmoid'))

# For subsequent layers, the input dimension is # presumed from the
previous layer
model.add(Dense(units=4))
model.add(Activation(sigmoid'))
model.add(Dense(units=3))
model.add(Activation('softmax'))
```


AIDA4Edge
UK Research and Innovation
Funded by the European Union
University of Ferrara

Train a NN

Re-open the notebook https://colab.research.google.com/drive/1QyNIHgW44Z9Usj7J3sdKaypVrFzohG6T and go to section **Train and evaluate**

AIDA4Edge UK Research and Innovation Funded by the European Union University of Ferrara

Tensorflow Training

- One note about how training / test loss are calculated in Tensorflow.
- You may see that training loss is higher than test loss in Tensorflow models.
- This behavior stems from difference in procedure on how Tensorflow calculates training loss vs test loss
    - Training loss is averaged over many batches in an epoch, whereas test loss is based on the final parameters of that epoch and consequently may turn out to be lower.

AIDA4Edge UK Research and Innovation Funded by the European Union University of Ferrara

Tensorflow in details

Let's see an example:

https://colab.research.google.com/drive/1gSKvOoyQISKghL0-QV2woIl2kcf6FcPp

AIDA4Edge UK Research and Innovation Funded by the European Union University of Ferrara