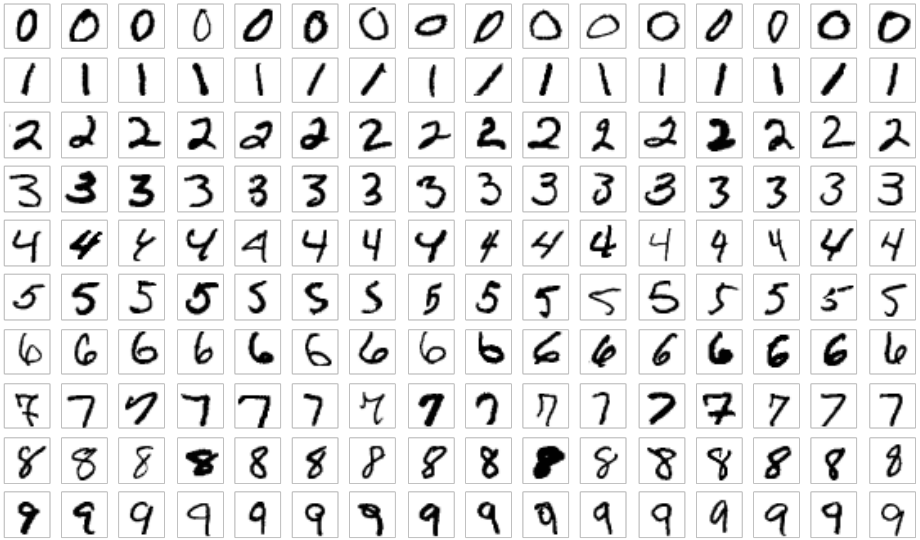

# Neural Networks with PyTorch

Battista Biggio

Department of Electrical and Electronic Engineering
University of Cagliari, Italy

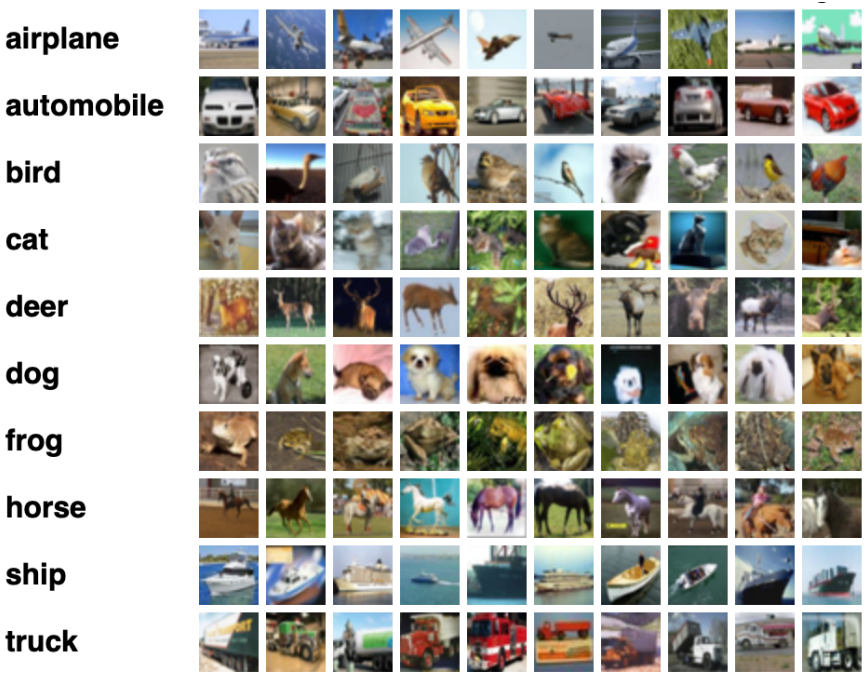
# MNIST Handwritten Digits

- Dataset of handwritten digits
  - 60,000 training images
  - 10,000 test images
- 10 classes: 0, 1, ..., 9
- 28x28 images corresponding to 784 feature (pixel) values
- Pixel values in {0, 1, ..., 255}



# CIFAR 10

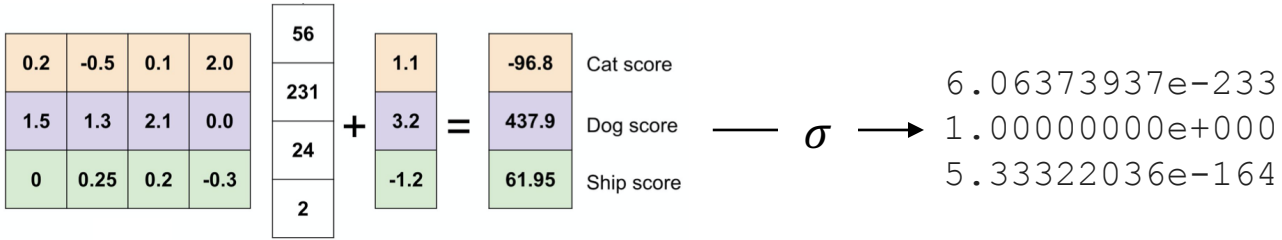
- Dataset of images
  - 50,000 training images
  - 10,000 test images
- 10 classes: 0, 1, ..., 9
  - 6,000 images per class
- 32x32x3 images
- Pixel values in {0, 1, ..., 255}




- Dataset available at:
  - https://www.cs.toronto.edu/~kriz/cifar.html

# Exercise 1

- Implement and train a softmax classifier on the MNIST data with PyTorch
  - Multiclass linear classifier $f(\boldsymbol{x}) = \mathbf{W}^{\mathrm{T}}\boldsymbol{x} + \boldsymbol{b}$
  - Outputs are scaled with $\sigma$, the softmax operator $\sigma_c = \frac{e^{fc}}{\sum_j e^{fj}}$

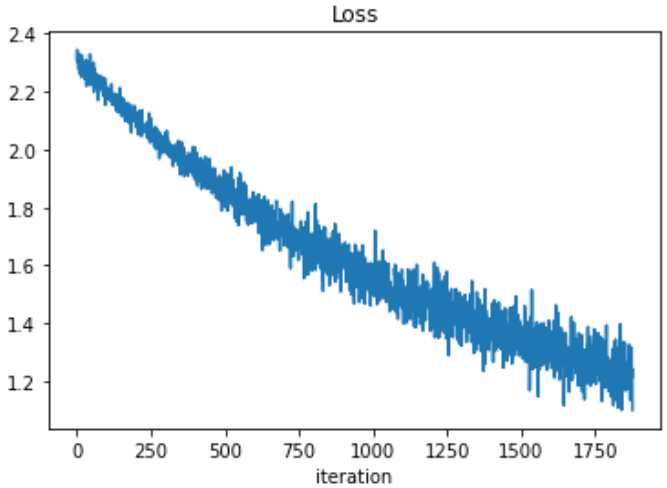



- Minimize the cross-entropy loss on softmax-scaled outputs
  - https://peterroelants.github.io/posts/cross-entropy-softmax/

$$\min_{\boldsymbol{\theta}} -\sum_{i=1}^{n}\sum_{c=1}^{k} y_{ic}\log(\sigma(f_c(\boldsymbol{x}_i;\boldsymbol{\theta}))$$

# Exercise 1: Solution

- Test accuracy of the model on the 10000 test images: 80.76 %


Open in Colab https://colab.research.google.com/github/unica-ml/ml/blob/master/notebooks/lab05.ipynb https://unica-ml.github.io

# Exercise 2

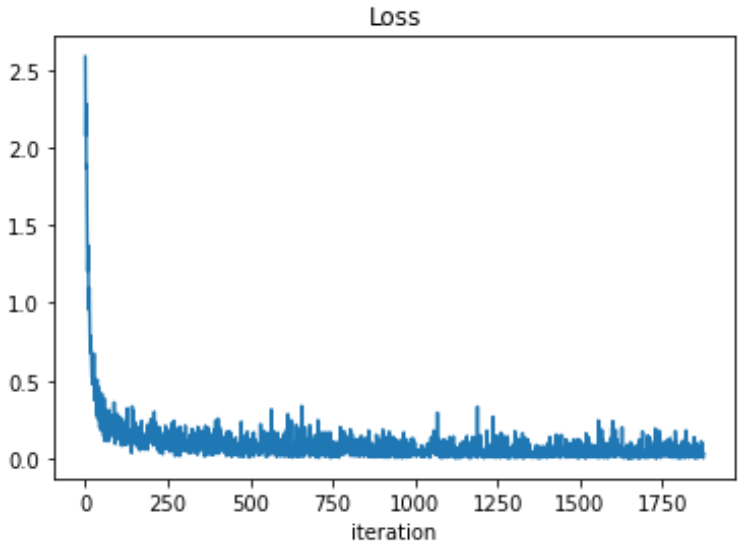
- Train a CNN on MNIST, again using the cross-entropy loss on softmax-scaled outputs

```
# Convolutional neural network (two convolutional layers)
class ConvNet(nn.Module):
    def __init__(self, num_classes=10):
        super(ConvNet, self).__init__()
        self.layer1 = nn.Sequential(
            nn.Conv2d(1, 16, kernel_size=5, stride=1, padding=2),
            nn.BatchNorm2d(16),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2, stride=2))
        self.layer2 = nn.Sequential(
            nn.Conv2d(16, 32, kernel_size=5, stride=1, padding=2),
            nn.BatchNorm2d(32),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2, stride=2))
        self.fc = nn.Linear(7*7*32, num_classes)

    def forward(self, x):
        out = self.layer1(x)
        out = self.layer2(out)
        out = out.reshape(out.size(0), -1)
        out = self.fc(out)
        return out
```

# Exercise 2: Solution

- Test accuracy of the model on the 10000 test images: 98.28 %

# Exercise 3

- Train a CNN on CIFAR10, again using the cross-entropy loss on softmax-scaled outputs


```
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.conv1 = nn.Conv2d(3, 6, 5)
        self.pool = nn.MaxPool2d(2, 2)
        self.conv2 = nn.Conv2d(6, 16, 5)
        self.fc1 = nn.Linear(16 * 5 * 5, 120)
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 10)

    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = x.view(-1, 16 * 5 * 5)
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x
```

# Exercise 3: Solution

- Accuracy of the network on the 10000 test images: 38 %
  - Accuracy of plane : 56 %
  - Accuracy of car : 52 %
  - Accuracy of bird : 23 %
  - Accuracy of cat : 28 %
  - Accuracy of deer : 16 %
  - Accuracy of dog : 35 %
  - Accuracy of frog : 57 %
  - Accuracy of horse : 35 %
  - Accuracy of ship : 37 %
  - Accuracy of truck : 41 %