Inteligencia Artificial

Ejercicio 7: Pytorch

Pytorch

Cronograma de Ej

Exercise 01: Organization Exercise 02: Math Recap	Intro
Exercise 03: Dataset and Dataloader Exercise 04: Solver and Linear Regression Exercise 05: Neural Networks Exercise 06: Hyperparameter Tuning	Numpy (Reinvent the wheel)
Exercise 07: Introduction to Pytorch Exercise 08: Autoencoder	Pytorch/Tensorboard
Exercise 09: Convolutional Neural Networks Exercise 10: Semantic Segmentation Exercise 11: Recurrent Neural Networks	Applications (Hands-off)

Deep Learning Frameworks

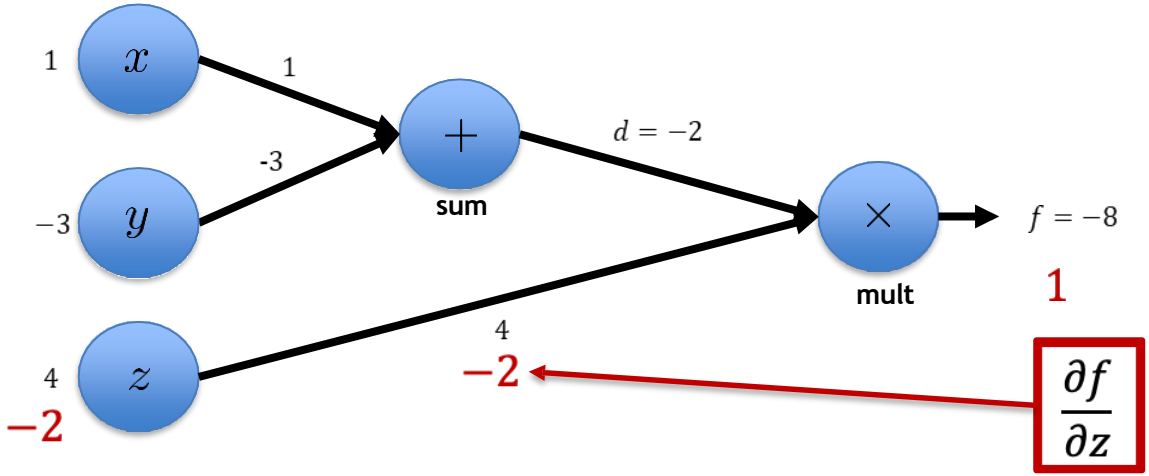
Dos grandes

- Tensorflow - Google
  - As well as Keras
- Pytorch - Facebook

Otros ejemplos

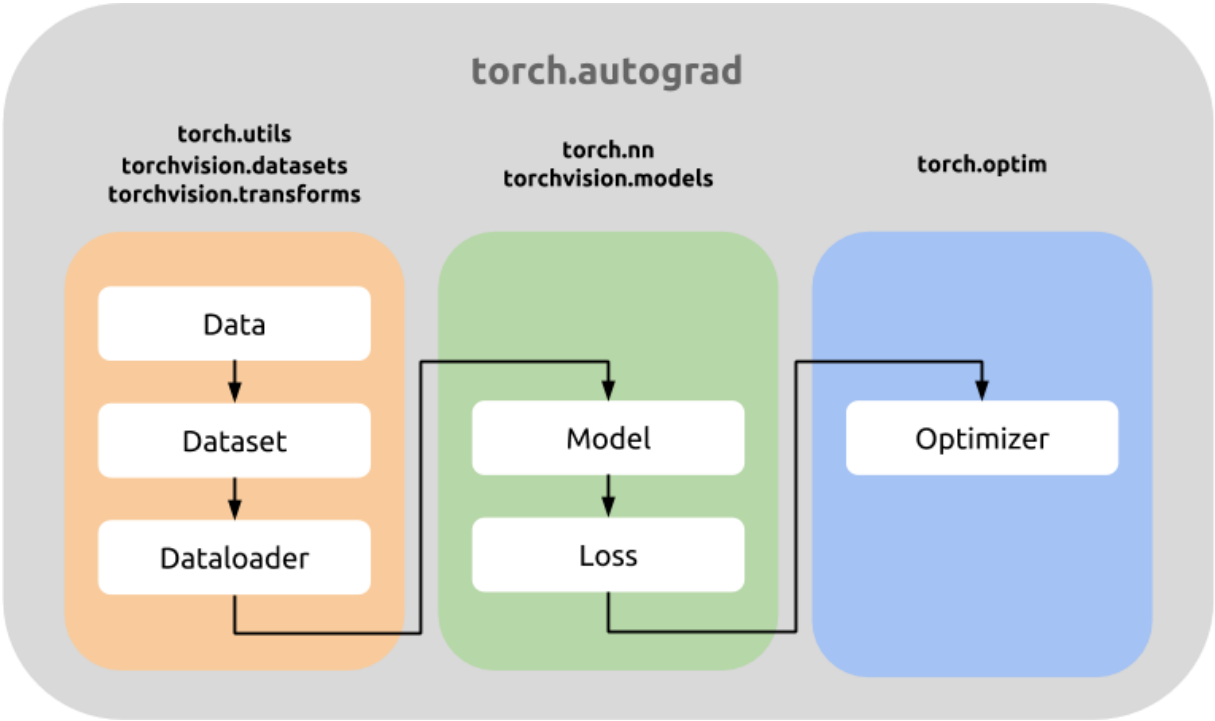
- CNTK - Microsoft
- Mxnet - Apache
- Jax - Google
- …

Different Paradigms



	Tensorflow	Pytorch
Graph Creation	Static/Eager	Dynamic/On Runtime
Similar to	C	Python

Pytorch: Overview

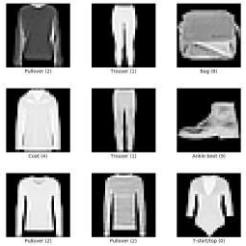


Características clave

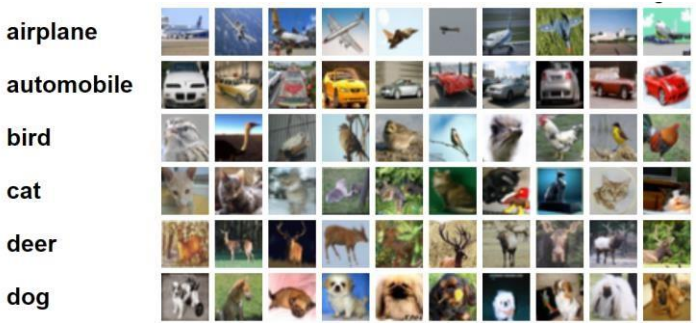
- **Simple device management**

```
device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
print(device)

print(f"Original device: {x.device}") # "cpu", integer

tensor = x.to(device)
print(f"Current device: {x.device}") #"cpu" or "cuda", double
```

```
cpu
Original device: cpu
Current device: cpu
```



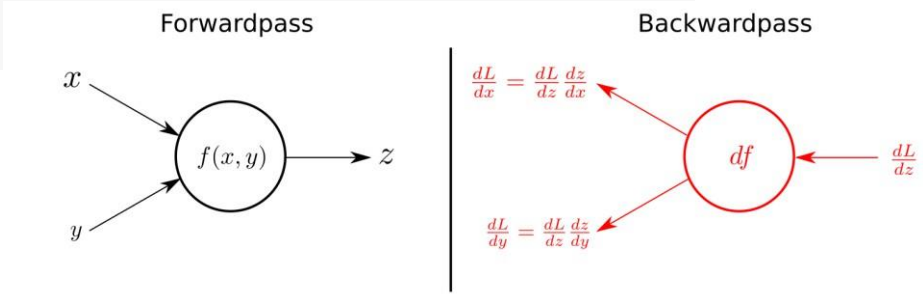
- **Implementación de:**
  - Optimizers, etc.
  - Datasets
  - Automatic gradients



Creación de redes fácilmente

```python
import torch.nn as nn
# defining the model
class Net(nn.Module):
    def __init__(self, input_size=1*28*28, output_size=100):
        super(Net, self).__init__()
        self.fc1 = nn.Linear(input_size, output_size)

    def forward(self, x):
        x = self.fc1(x)
        return x
net = Net()
net = net.to(device)
```

Donde está el backward pass?



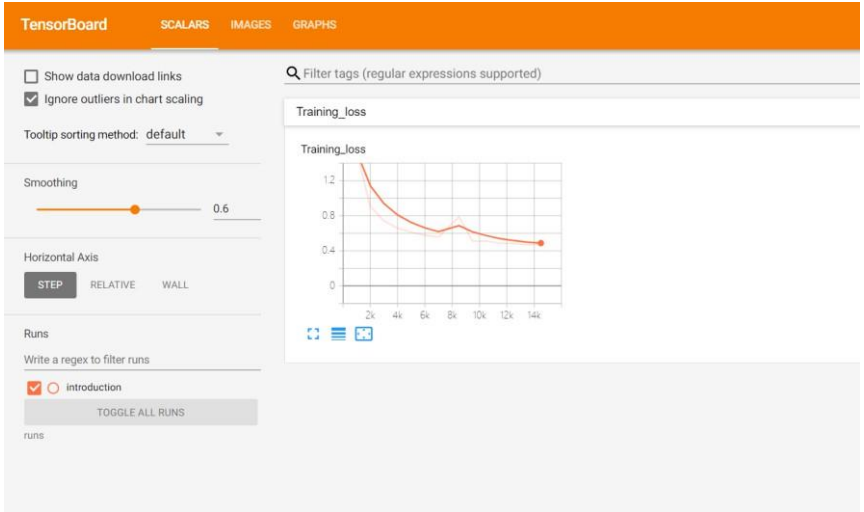
Referencias de Pytorch

- Repository: https://github.com/pytorch/pytorch
- Examples (recommendation): https://github.com/pytorch/examples
- PyTorch for NumPy users: https://github.com/wkentaro/pytorch-for-numpy-users
- Look up your own and share! ☺

Tensorboard (integrable con Pytorch)

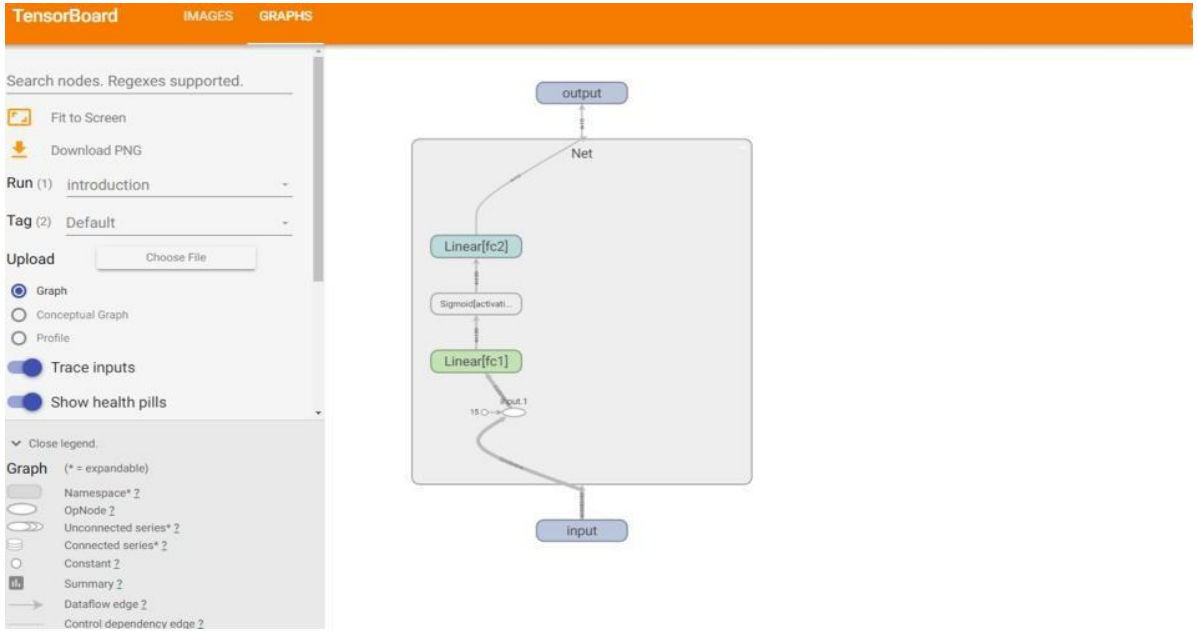
- Directamente accedes a tensorboard en tu training loop

- Tensorboard genera los graph/timestamps etc. a tu gusto

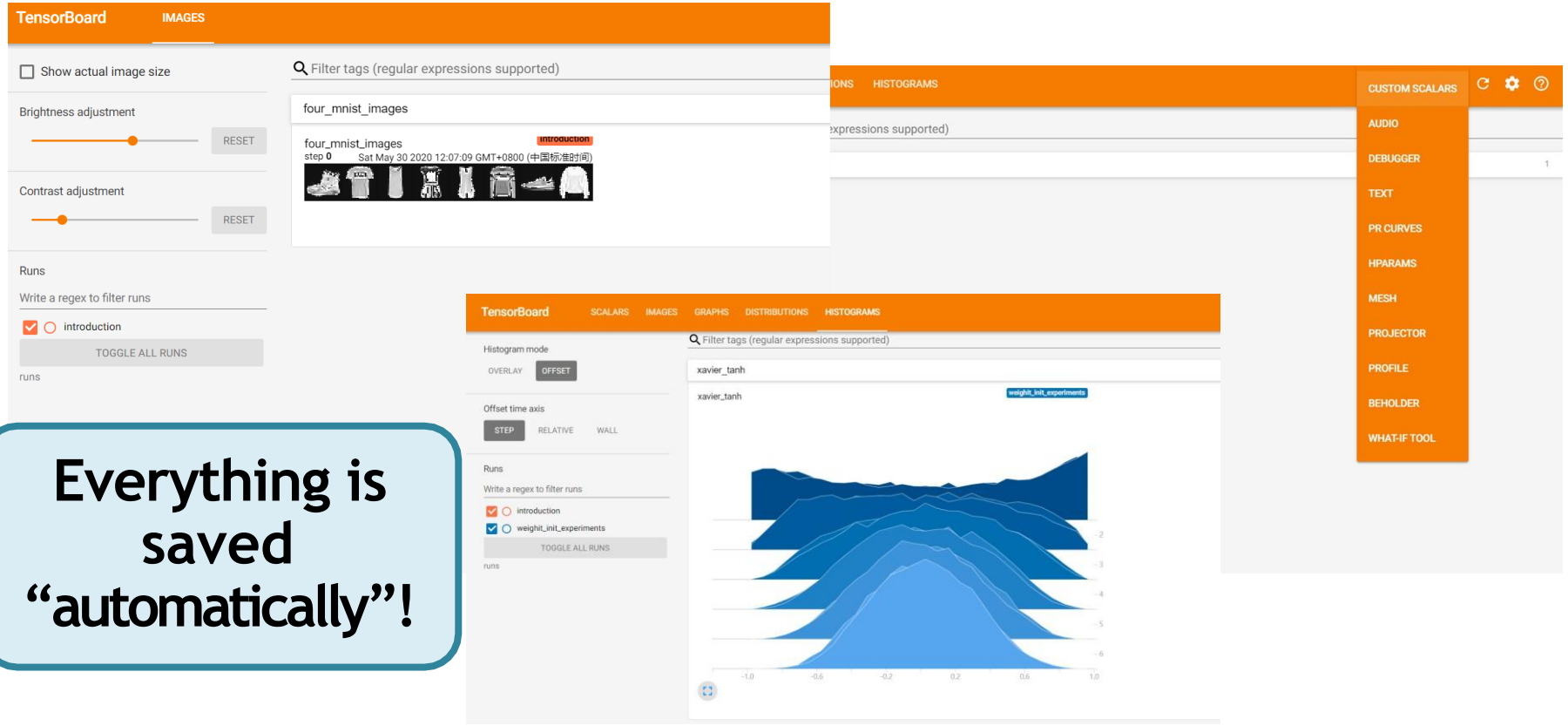


Visualize Networks

- Con un solo forward pass, tensorboard puede mapear y visualizar el gráfico de tu red



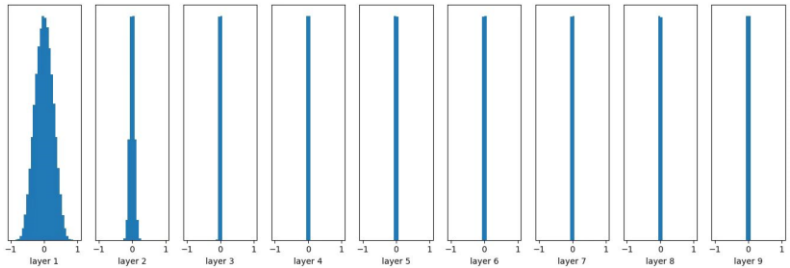
En resumen: ¡documenta todo!

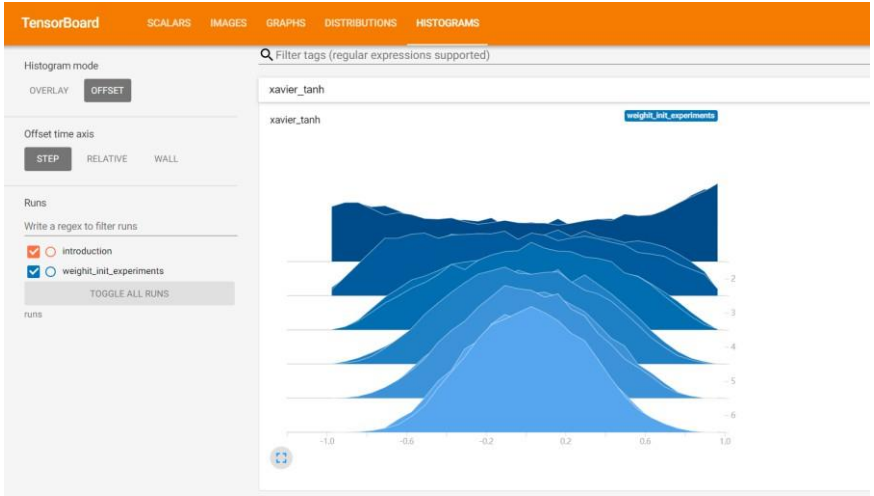


Everything is saved “automatically”!

Ejemplo: Weight Initialization

- La visualización del histograma para los layer outputs puede mostrar los efectos de la weight initialization, como se muestra en la lección.





Más abstracción: Pytorch Lightning

Clasificar nuestro código en tres categorías

1. **Research** code (¡la parte emocionante!, cambia con nuevas tareas, modelos etc.)

→ LightningModule

1. **Engineering** code (lo mismo para todos los proyectos y modelos)

→ Trainer

1. **Non-essential** code (logging, organización de ejecuciones)

→ Callbacks

PYTORCH

```
# models
encoder = nn.Sequential(nn.Linear(28 * 28, 64), nn.ReLU(), nn.Linear(64, 3))
decoder = nn.Sequential(nn.Linear(3, 64), nn.ReLU(), nn.Linear(64, 28 * 28))

encoder.cuda(0)
decoder.cuda(0)

# download on rank 0 only
if global_rank == 0:
    mnist_train = MNIST(os.getcwd(), train=True, download=True)

# split dataset
transform=transforms.Compose([transforms.ToTensor(),
                              transforms.Normalize(0.5, 0.5)])
mnist_train = MNIST(os.getcwd(), train=True, download=True, transform=transform)

# train (55,000 images), val split (5,000 images)
mnist_train, mnist_val = random_split(mnist_train, [55000, 5000])

# The dataloaders handle shuffling, batching, etc...
mnist_train = DataLoader(mnist_train, batch_size=64)
mnist_val = DataLoader(mnist_val, batch_size=64)

# optimizer
params = [encoder.parameters(), decoder.parameters()]
optimizer = torch.optim.Adam(params, lr=1e-3)

# TRAIN LOOP
model.train()
num_epochs = 1
for epoch in range(num_epochs):
  For train_batch in mnist_train:
    x, y = train_batch
    x = x.cuda(0)
    x = x.view(x.size(0), -1)
    z = encoder(x)
    x_hat = decoder(z)
    loss = F.mse_loss(x_hat, x)
    print('train loss: ', loss.item())

    loss.backward()
    optimizer.step()
    optimizer.zero_grad()

# EVAL LOOP
model.eval()
with torch.no_grad():
  val_loss = []
  for val_batch in mnist_val:
    x, y = val_batch
    x = x.cuda(0)
    x = x.view(x.size(0), -1)
    z = encoder(x)
    x_hat = decoder(z)
    loss = F.mse_loss(x_hat, x)
    val_loss.append(loss)
  val_loss = torch.mean(torch.tensor(val_loss))
  model.train()
```

PYTORCH LIGHTNING

Turn PyTorch into Lightning

Lightning is just plain PyTorch

Submission Opcional

CIFAR10... Again...

- **Tarea: Clasificación CIFAR10** (pero ahora en Pytorch)

- **Nuevo:**
  - Más conocimientos de la clase 7
  - Se puede utilizar todo, pero no: convolutional layers/transformers/ pre-trained networks

Nos vemos el próximo lunes ☺