# Tensorboard (Pytorch)

PyTorch BIVL²ab

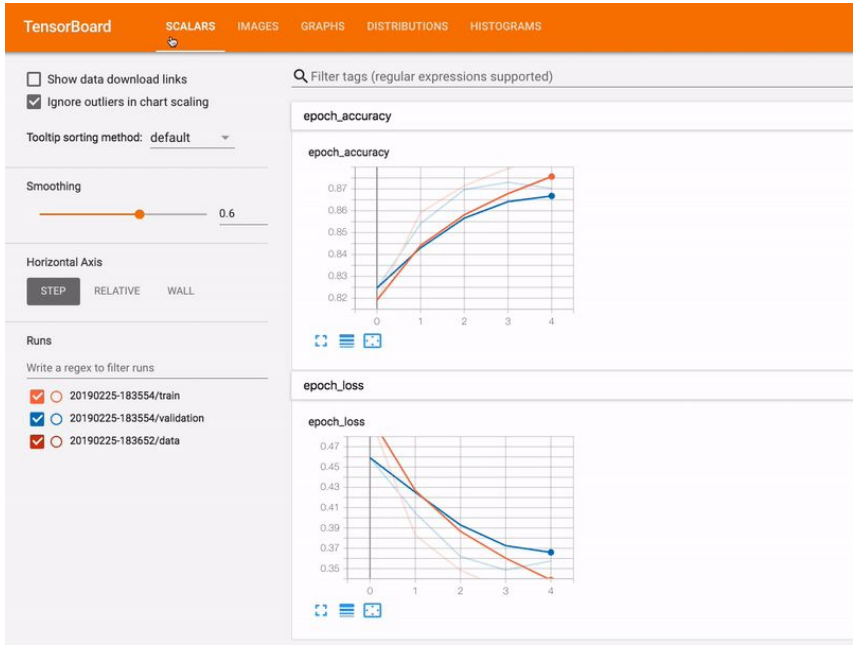
# Agenda

- Use the BIVL²ab server
- Set up an environment for the course
- Access the course resources
- `transforms` API

# TensorBoard

- TensorFlow's **visualization toolkit.**
- We can track and visualize metrics, project embeddings, histograms, display images, etc.

TensorBoard UserInterface (UI)




https://www.tensorflow.org/tensorboard

# Before going further…

- Install TensorBoard.
- from **torch.utils.tensorboard** import SummaryWriter (Class to log Pytorch models and metrics into a directory for )

- Pytorch Tensorboard tutorial:

https://pytorch.org/docs/stable/tensorboard.html

# Example No.1

Directory location, by default:
"runs/CURRENT_DATETIME_HOSTNAME"

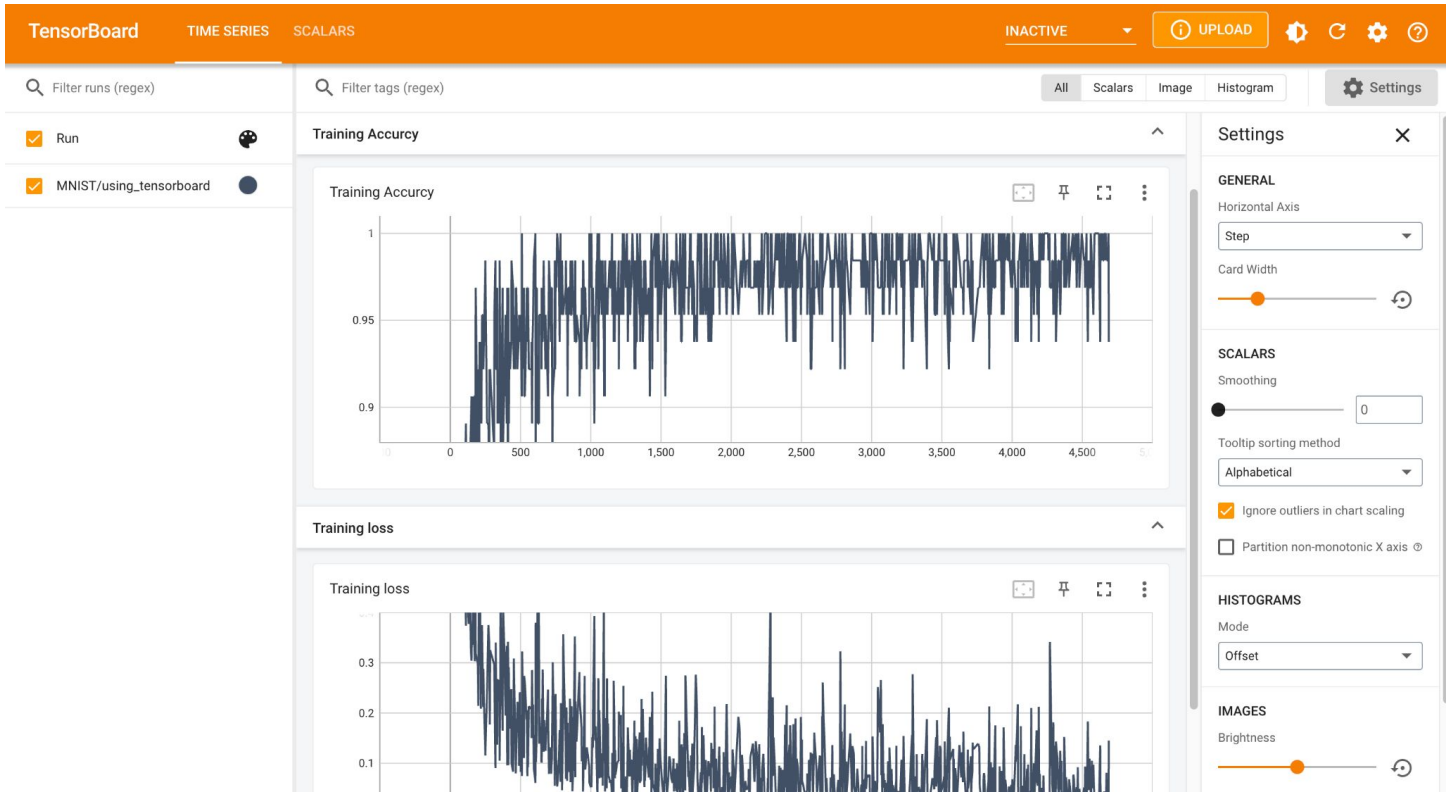
```
"""
TENSORBOARD WRITER
"""
writer = SummaryWriter(f'runs/MNIST/using_tensorboard')
step = 0
for epoch in range(num_epochs):
    for batch_idx, (data, targets) in enumerate(train_loader):
        # Get data to cuda if possible
        data = data.to(device=device)
        targets = targets.to(device=device)

        # forward
        scores = model(data)
        loss = criterion(scores, targets)

        # backward
        optimizer.zero_grad()
        loss.backward()

        # gradient descent or adam step
        optimizer.step()

        #Calculate running trining accuracy
        _, predictions = scores.max(1)
        num_correct = (predictions == targets).sum()
        running_train_acc = float(num_correct)/float(data.shape[0])

        # tensorboard
        writer.add_scalar('Training loss', loss, global_step = step)
        writer.add_scalar('Training Accurcy', running_train_acc, global_step = step)
        step += 1 #step is the number of batches
```

**MNIST. Saving loss and accuracy during training.**

Add scalars to the log file

# Example No.1

1. **tensorboard –logdir runs**
2. **Enter to url**



# Example No.2

```python
# Hyperparameters
in_channels = 1
num_classes = 10
num_epochs = 5
bacth_sizes = [1,64,128,1024]
learning_rates = [0.1, 0.01, 0.001, 0.0001]


# Load Data
train_dataset = datasets.MNIST(root='dataset/', train=True, transform = transforms.ToTensor(), download=True)

for batch_size in tqdm(bacth_sizes):
    for lr in learning_rates:
        step = 0

        #dataloader
        train_loader = torch.utils.data.DataLoader(dataset=train_dataset, batch_size=bacth_size, shuffle=True)

        #Initialize network
        model = simple_CNN(in_channels=in_channels, num_classes=num_classes)
        model.to(device) #move model to device

        #Loss and optimizer
        criterion = nn.CrossEntropyLoss()
        optimizer = torch.optim.Adam(model.parameters(), lr=lr)

        #TENSORBOARD WRITER
        writer = SummaryWriter(f'runs/MNIST/batchSize_{batch_size}_lr_{lr}')

        for epoch in range(num_epochs):
            for batch_idx, (data, targets) in enumerate(train_loader):
                # Get data to cuda if possible
                data = data.to(device=device)
```

(import) datasets: Any

**Saving different models to compare them after**

# Example No.2

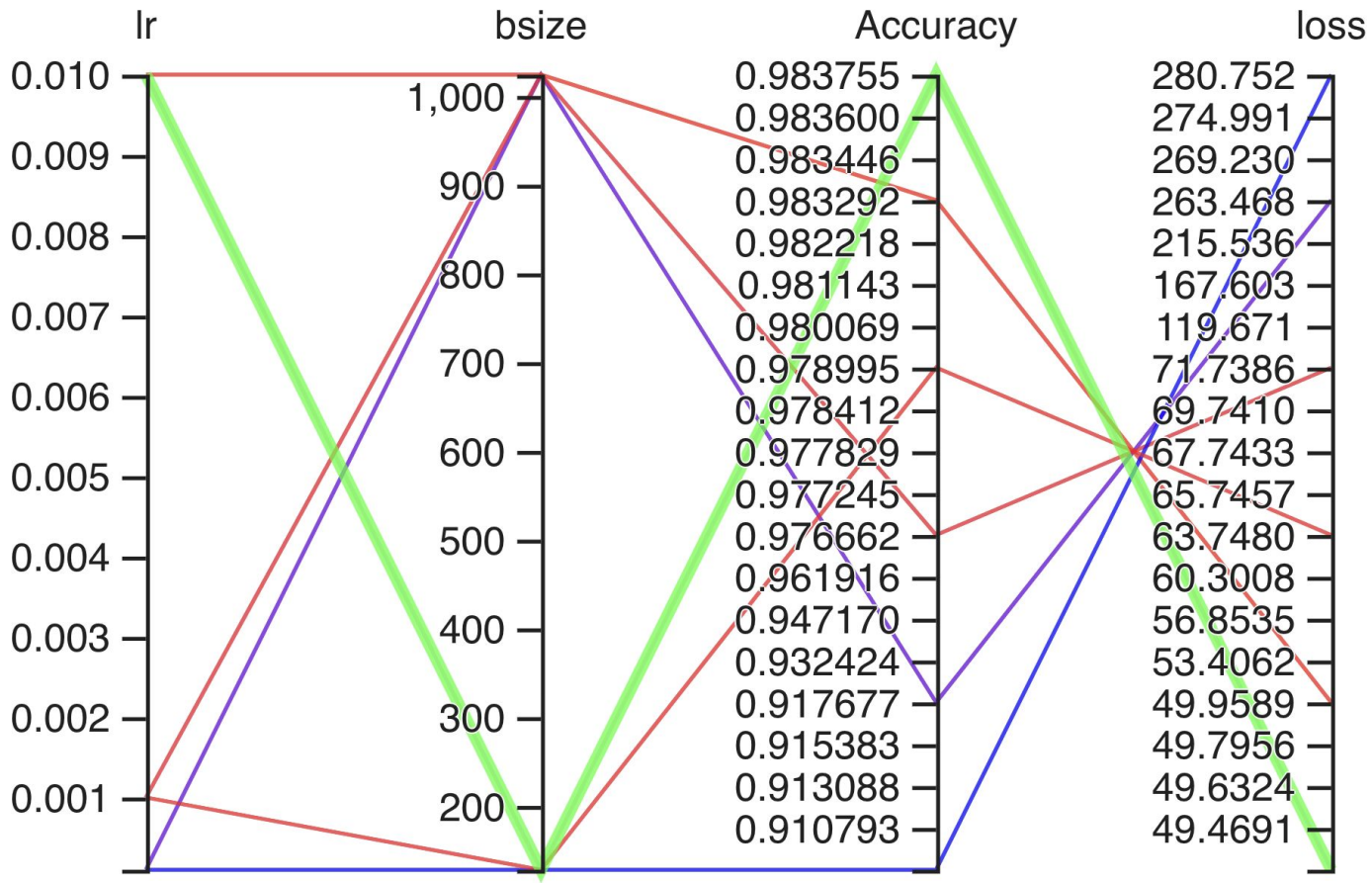

TensorBoard
TIME SERIES
SCALARS
INACTIVE
UPLOAD
Filter runs (regex)
Filter tags (regex)
All
Scalars
Image
Histogram
Settings
Run
MNIST/using_tensorboard
MNIST/batchSize_1_lr_0.1
MNIST/batchSize_1_lr_0.01
MNIST/batchSize_1_lr_0.001
MNIST/batchSize_1_lr_0.0001
MNIST/batchSize_64_lr_0.1
MNIST/batchSize_64_lr_0.01
MNIST/batchSize_64_lr_0.001
MNIST/batchSize_64_lr_0.0001
MNIST/batchSize_128_lr_0.1
MNIST/batchSize_128_lr_0.01
MNIST/batchSize_128_lr_0.001
MNIST/batchSize_128_lr_0.0001
MNIST/batchSize_1024_lr_0.1
Training Accurcy
Training Accurcy
1
0.8
0.6
0.4
0.2
0
500
1,000
1,500
2,000
2,500
3,000
3,500
4,000
4,500
Training loss
Training loss
2.5
2
1.5
1

# Example No.3

```
for epoch in range(num_epochs):
    accuracies, losses = [], []
    for batch_idx, (data, targets) in enumerate(train_loader):
        # Get data to cuda if possible
        data = data.to(device=device)
        targets = targets.to(device=device)

        # forward
        scores = model(data)
        loss = criterion(scores, targets)
        losses.append(loss.item())

        # backward
        optimizer.zero_grad()
        loss.backward()

        # gradient descent or adam step
        optimizer.step()

        #Calculate running trining accuracy
        _, predictions = scores.max(1)
        num_correct = (predictions == targets).sum()
        running_train_acc = float(num_correct)/float(data.shape[0])
        accuracies.append(running_train_acc)
        # tensorboard
        writer.add_scalar('Training loss', loss, global_step = step)
        writer.add_scalar('Training Accurcy', running_train_acc, global_step = step)
        step += 1 #step is the number of batches
writer.add_hparams(
    hparam_dict = {"lr": lr, "bsize": batch_size},
    metric_dict = {"Accuracy": sum(accuracies)/len(accuracies), "loss": sum(losses)}
)
```

**Adding a set of hyperparameters to be compared**

## HPARAMS/PARALLEL COORDINATES VIEW

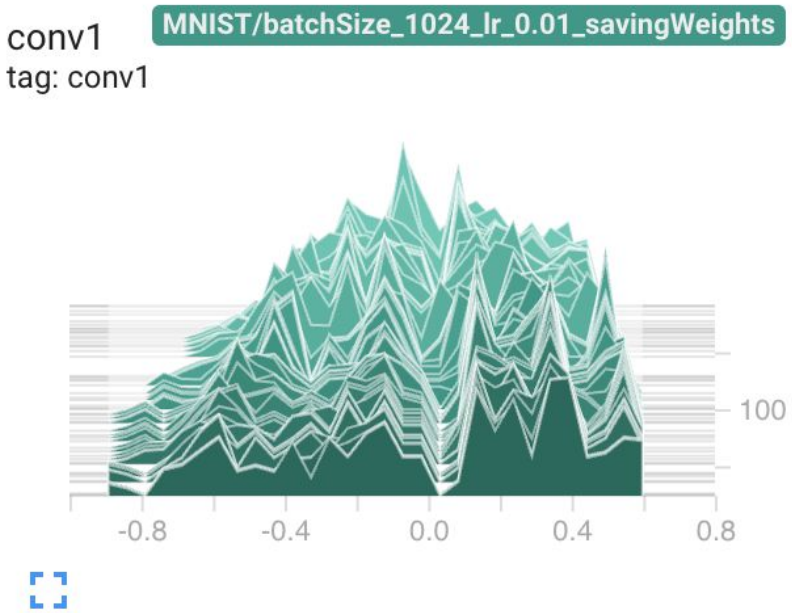


# Example No.4

```
# gradient descent or adam step
optimizer.step()

# histogram of weights
writer.add_histogram("fc1", model.fc1.weight, global_step = step)
writer.add_histogram("conv1", model.conv1.weight, global_step = step)
writer.add_histogram("conv2", model.conv1.weight, global_step = step)
#Calculate running trining accuracy
_, predictions = scores.max(1)
num_correct = (predictions == targets).sum()
running_train_acc = float(num_correct)/float(data.shape[0])
accuracies.append(running_train_acc)
# tensorboard
writer.add_scalar('Training loss', loss, global_step = step)
writer.add_scalar('Training Accurcy', running_train_acc, global_step = step)
step += 1 #step is the number of batches
```

**Visualizing weights hISTOGRAMS**

## HPARAMS/PARALLEL COORDINATES VIEW

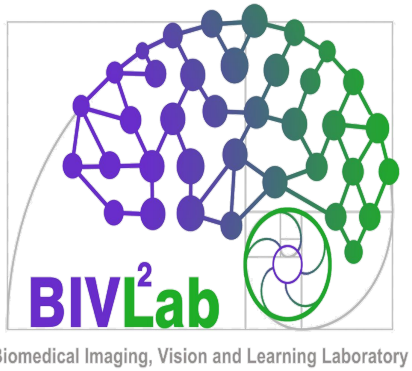


# Resources:

- Pytorch Tensorboard tutorial: https://pytorch.org/docs/stable/tensorboard.html
- Youtube tutorial: Pytorch TensorBoard Tutorial (Aladdin Persson YT))
- https://pytorch.org/tutorials/intermediate/tensorboard_tutorial.html

# Thank You!


Universidad
Industrial de
Santander
UIS


BIVL2Lab
Biomedical Imaging, Vision and Learning Laboratory