Institut québécois d'intelligence artificielle Mila

Block 2
Common Presentation

Arsene Fansi-Tchango, PhD

Announcements

- The code/report/model from each team for block 1 + our baseline code/model are in the folder:

  /rap/jvb-000-aa/COURS2019/etudiants/submissions

- The baseline is in the `b1p{project_code}t_baseline` folder.

Mila

Institut
québécois
d'intelligence
artificielle

Mila

Visualization in Deep Learning: Tensorboard(X)

Institut
québécois
d'intelligence
artificielle

Mila

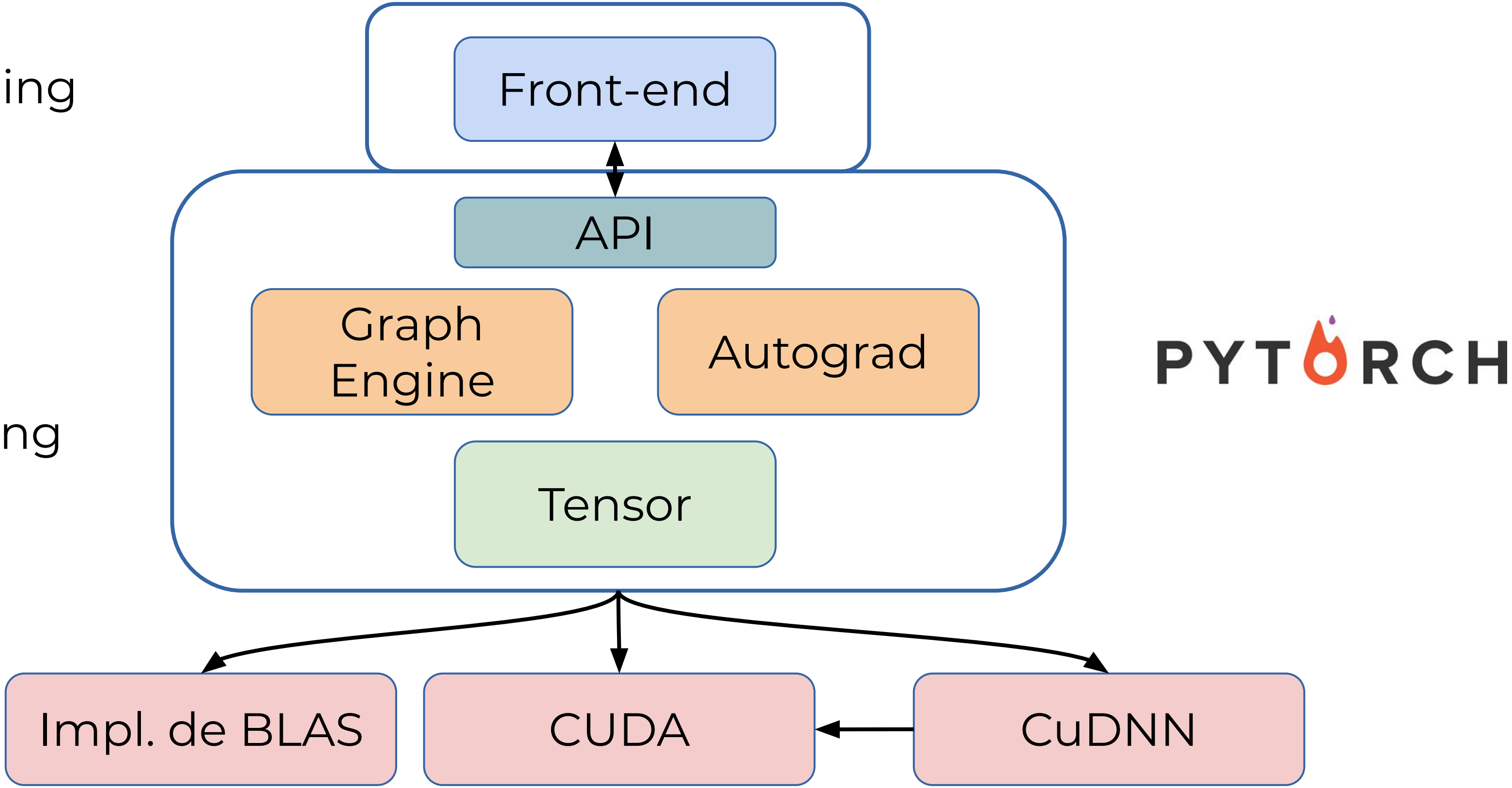
Tensorboard(X)

PyTorch

Deep Learning

Differential programming

Matrix calculation

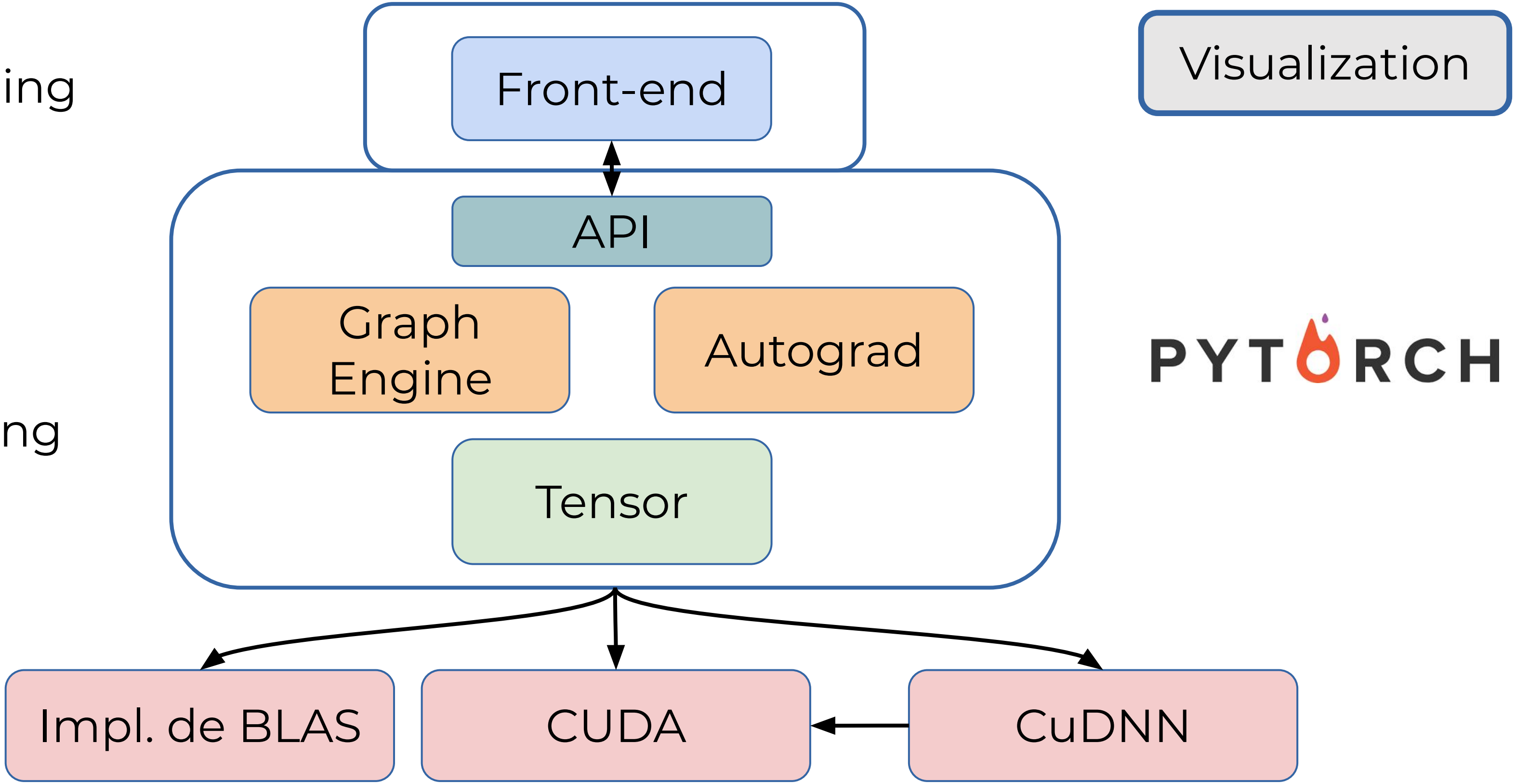


PyTorch

Deep Learning

Differential programming

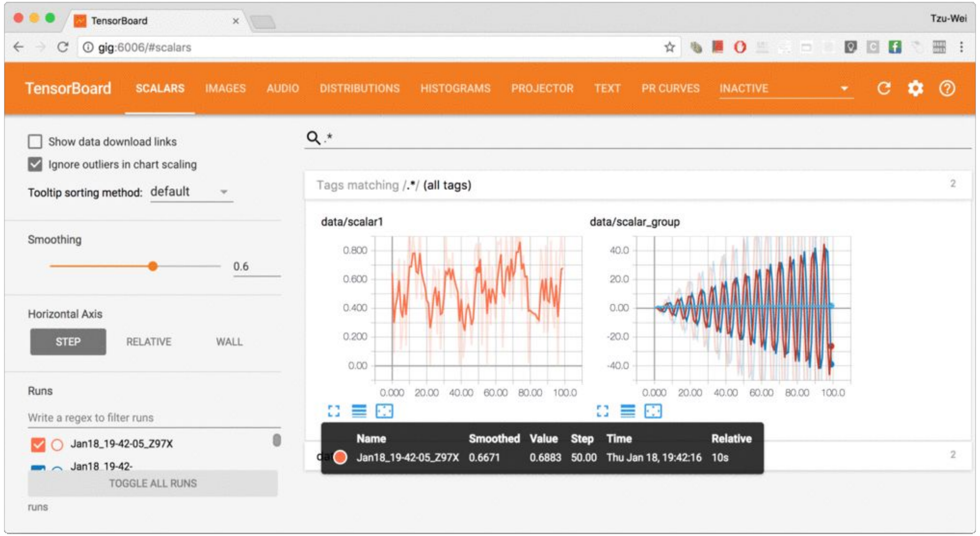
Matrix calculation



What is Tensorboard

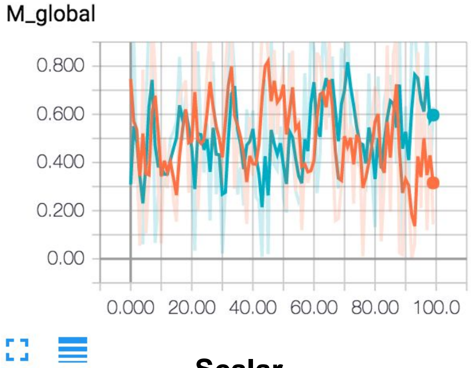
- A **TensorFlow** graphical interface useful for:
  - Graph visualization
  - Metric monitoring
  - Model Analysis
  - Debugging

TensorFlow

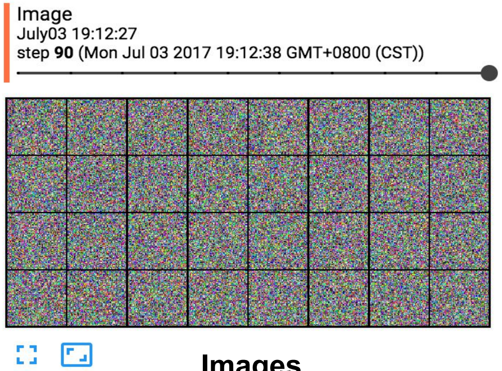


Mila

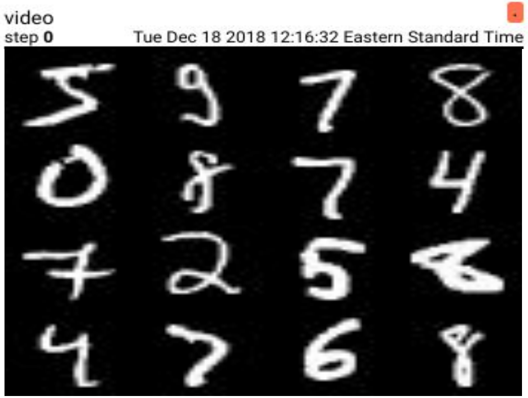
Levels of Information



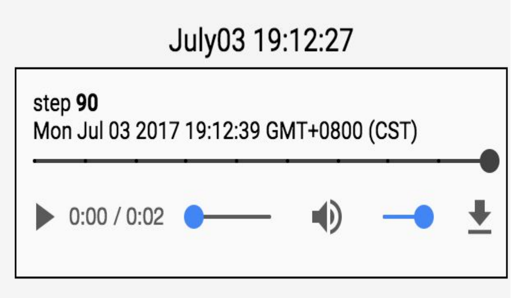
Scalar



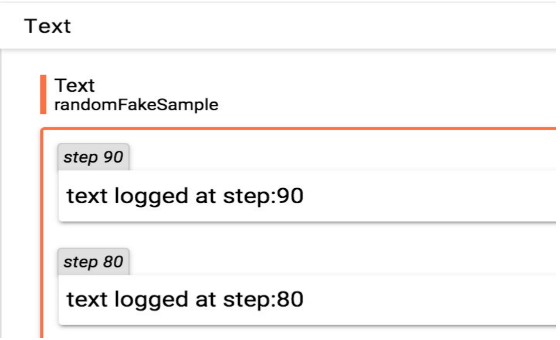
Images



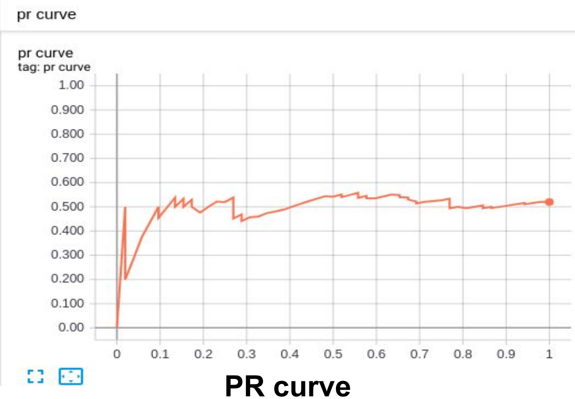
Video



Audio



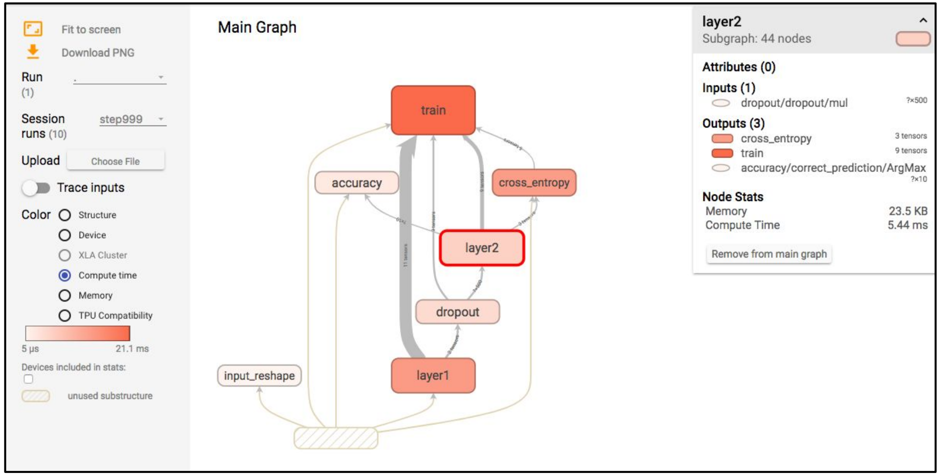
Text



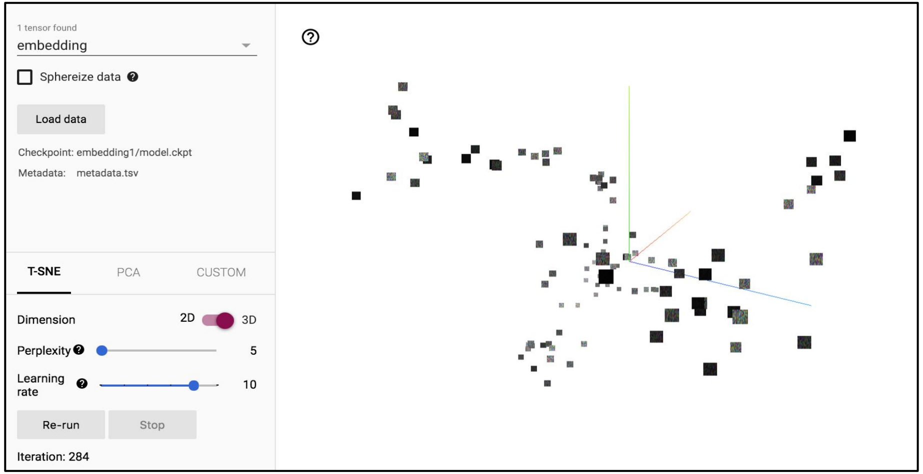
PR curve

Mila

Levels of Information

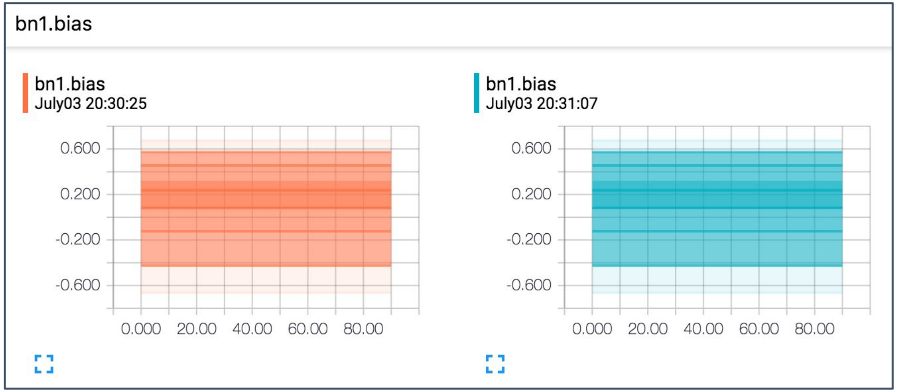


Computational Graphs

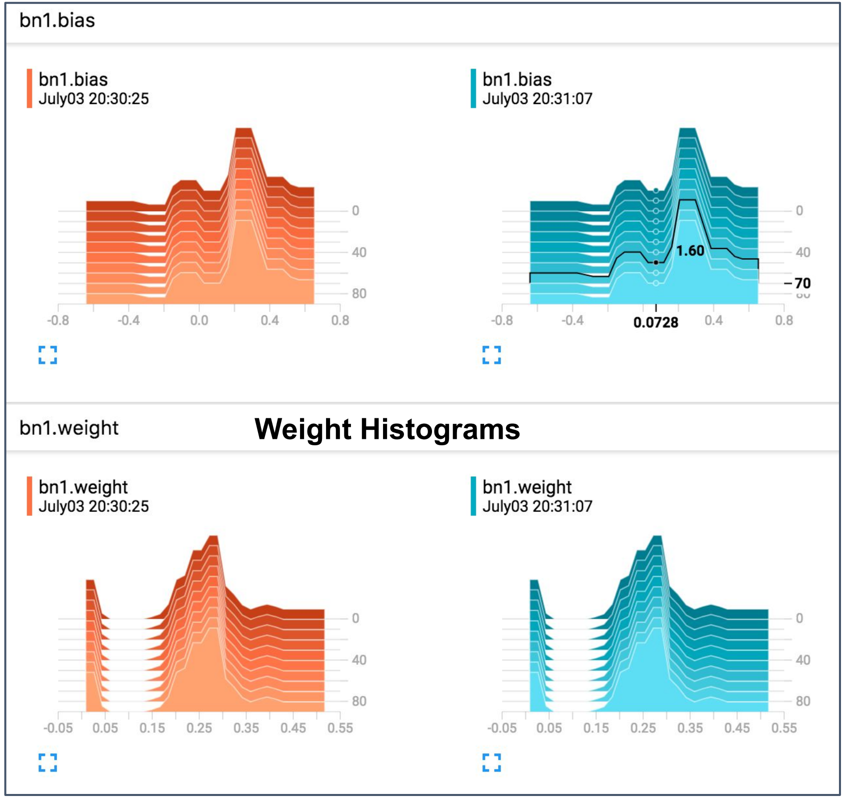


Embeddings

Levels of Information



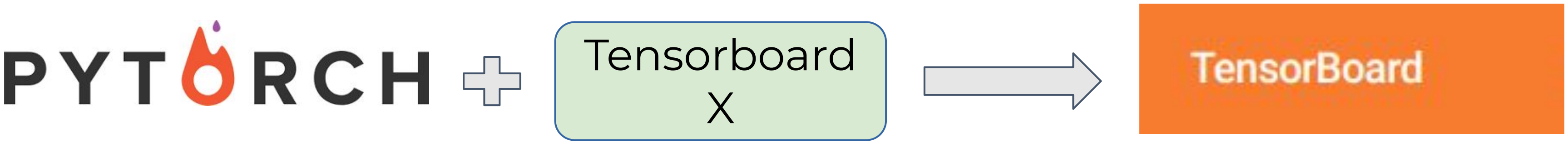
Weight Distributions



Weight Histograms

What is TensorboardX

- A module for visualization in tensorboard that works from **PyTorch**.



Interface for writing TensorBoard events with simple function calls

https://tensorboardx.readthedocs.io/en/latest/tensorboard.html

SummaryWriter

`class tensorboardX.SummaryWriter(log_dir=None, comment='', **kwargs)` [source]

Writes *Summary* directly to event files. The *SummaryWriter* class provides a high-level api to create an event file in a given directory and add summaries and events to it. The class updates the file contents asynchronously. This allows a training program to call methods to add data to the file directly from the training loop, without slowing down training.

- Class to directly write event files.
- General API format:

```
add_something(tag name, object, iteration number)
```

- Examples:
  - `add_scalar`(*tag*, *scalar_value*, *global_step=None*, *walltime=None*)
  - `add_image`(*tag*, *img_tensor*, *global_step=None*, *walltime=None*)
  - `add_video`(*tag*, *vid_tensor*, *global_step=None*, *fps=4*, *walltime=None*)
  - `add_audio`(*tag*, *snd_tensor*, *global_step=None*, *sample_rate=44100*, *walltime=None*)
  - `add_text`(*tag*, *text_string*, *global_step=None*, *walltime=None*)
  - `add_pr_curve`(*tag*, *labels*, *predictions*, *global_step=None*, *num_thresholds=127*, *weights=None*, *walltime=None*)
  - `add_graph`(*model*, *input_to_model=None*, *verbose=False*, ***kwargs*)
  - `add_embedding`(*mat*, *metadata=None*, *label_img=None*, *global_step=None*, *tag='default'*, *metadata_header=None*)
  - `add_histogram`(*tag*, *values*, *global_step=None*, *bins='tensorflow'*, *walltime=None*)

https://tensorboardx.readthedocs.io/en/latest/tensorboard.html

Example

```python
import torch
import torch.optim as optim
from tensorboardX import SummaryWriter

use_gpu = torch.cuda.is_available()
device = torch.device("cuda:0" if use_gpu else "cpu")
log_interval = 5


writer = SummaryWriter('runs')

model = ...

model.to(device)
optimizer = optim.SGD(model.parameters())

def train(epoch):
    model.train()
    torch.set_grad_enabled(True)
    for batch_idx, (data, target) in enumerate(train_loader):
        data, target = data.to(device), target.to(device)
        optimizer.zero_grad()
        output = model(data)
        loss = F.nll_loss(output, target)
        loss.backward()
        optimizer.step()
        if batch_idx % log_interval == 0:
            print('Train Epoch: {} [{}/{} ({:.0f}%)]\tLoss: {:.6f}'.format(
                epoch, batch_idx * len(data), len(train_loader.dataset),
                100. * batch_idx / len(train_loader), loss.data[0]))
            niter = epoch*len(train_loader)+batch_idx
            writer.add_scalar('Train/Loss', loss.data[0], niter)
```

https://tensorboardx.readthedocs.io/en/latest/tensorboard.html

Useful Commands

- **Installation**
    - conda install tensorboardX tensorboard
- **Execution**
    - tensorboard --logdir=<your_log_dir> [--port nPort]
- **Visualization** (on Browser)
    - http://localhost:nPort
- **Multiple run comparison**
    - tensorboard runs

https://tensorboardx.readthedocs.io/en/latest/tensorboard.html

Particularities on Helios

- No control of the computational nodes
  - You **can't run** tensorboard on those nodes
  - No **online monitoring** of your experiments

- But you can still do **offline monitoring**
  - Log your data as usual using **SummaryWriter**
  - **Copy the log files** at the end of the experiments on your local machines
  - Run tensorboard **locally**

Institut
québécois
d'intelligence
artificielle

Mila

Checkpointing
Save and Resume your experiments

Checkpointing

- A way to save the current state of an experiment
  - Possibility to **pick up** from the saved point

- A way to keep track of the best weights of a model
  - Possibility to save the weights with the best validation performance

- A way to save multiple models
  - Possibility to average the weights of those models to generate an ensemble

When to set checkpoints

- Every n_batches of mini batches

- n_batches shouldn't be too small because this might lead to:
  - Training slowdown if the validation set is large
  - Large disk space usage if the model weights at all checkpoints are saved

- n_batches shouldn't be too large because this might lead to:
  - Missing the best model(s)

Checkpointing - What to save

- The in-memory representation of the model
  - Possibility to re-create the model
- The weights of the model
- The training configuration
  - loss, epochs, and other meta-information (seed, hyperparameters, ...)
- The state of the optimizer
  - Possibility to resume training exactly where it left off

Checkpointing in PyTorch

- It is **recommended** to save only the model weights, not the model class

- Make use of the following functions:
    - torch.save
    - torch.load

Save a checkpoint in PyTorch

```
torch.save({
            'epoch': epoch,
            'model_state_dict': model.state_dict(),
            'optimizer_state_dict': optimizer.state_dict(),
            'loss': loss,
            ...
            }, PATH)
```

Load a checkpoint in PyTorch

```
model = TheModelClass(*args, **kwargs)
optimizer = TheOptimizerClass(*args, **kwargs)

checkpoint = torch.load(PATH)
model.load_state_dict(checkpoint['model_state_dict'])
optimizer.load_state_dict(checkpoint['optimizer_state_dict'])
epoch = checkpoint['epoch']
loss = checkpoint['loss']

model.eval()
# - or -
model.train()
```

Institut
québécois
d'intelligence
artificielle

Mila

Hyperparameter Tuning

What are Hyperparameters?

- In Machine Learning, a hyperparameter is a parameter whose value is set **before** the learning process begins.
- Hyperparameters **heavily affect the behaviour** of the underlying model.

Examples of Hyperparameters in DL

- What is the network depth? How wide is it?
- Every layer: Feedforward or Convolutional?
- How do layers connect to each other?
- What type of activation functions to use?
- Which optimization algorithm to use?
- What's the learning rate?
- How does the learning rate drop?
- Which initialization function to use?
- Is momentum necessary? What's the rate?

Examples of Hyperparameters in DL

- Is bias term needed in convolutional layers?
- Is dropout needed?
- Is batch norm needed?
- Is weight decay needed?
- What’s the weight decay speed?
- What’s the batch size?
- ...

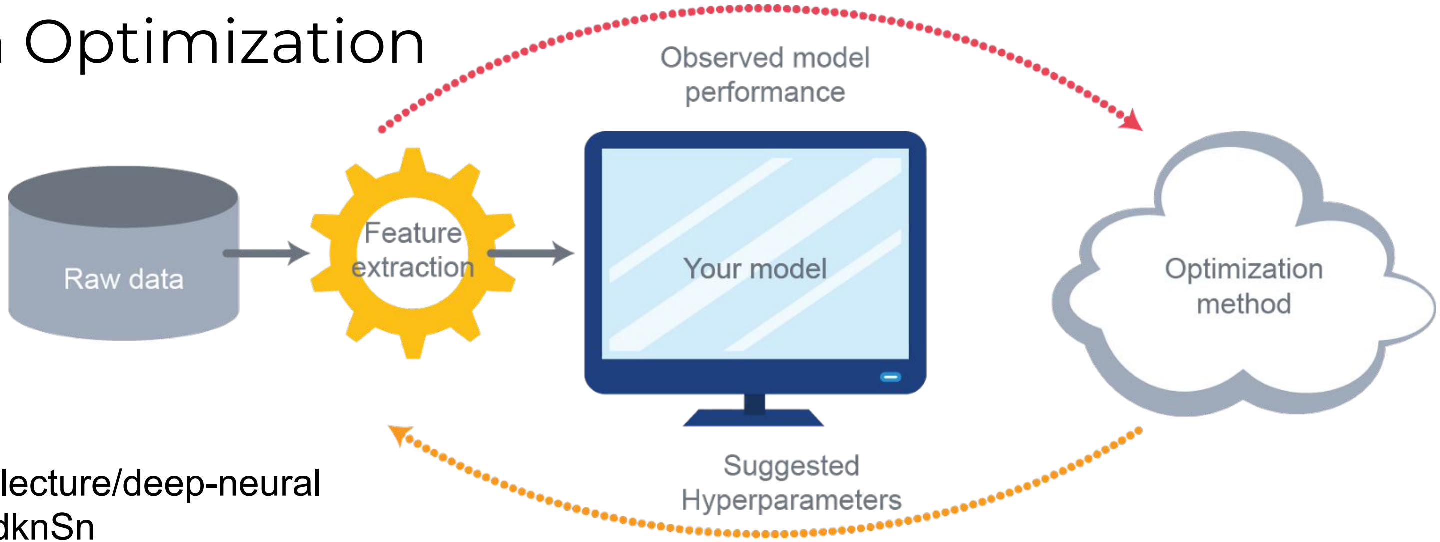
For each of these questions (hyperparameters), you have a **set of possible values** or a **range of values** associated. The combination of these values forms the **hyperparameter space**.

Hyperparameter Tuning

- From the hyperparameter space, find **a set of values** that lead to **optimal performances**.
  - E.g., Accuracy, MSE, ...

Some Existing approaches

- Grid Search
- Random Search
- Bayesian Optimization



https://www.coursera.org/lecture/deep-neural-network/tuning-process-dknSn

Manual Search

Manually do trial-and-error tuning:

- **Step 1**: Select a configuration of hyperparameters
- **Step 2**: Run it - See the results - Goto Step 1.

Keep the configuration with the highest performance.

Grid Search

Explore **all** the possible hyperparameter configurations

- For each configuration, compute the related performance metric
  - Cross-validation for robustness

Keep the configuration with the highest performance.

Random Search

- Avoid exploration of the whole hyperparameter space
- Proceed by **randomly sampling a limited number** of configurations
  - For each sampled configuration, compute the related performance metric
    - Cross-validation for robustness

Keep the configuration with the highest performance.

Random Search: Strategies

- Set of discrete values:
  - Uniform sampling

- Range Values
  - Linear scale
  - Log-scale

- For more information:

https://www.coursera.org/lecture/deep-neural-network/using-an-appropriate-scale-to-pick-hyperparameters-3rdqN

Bayesian Optimization

- Set a **prior** over hyperparameter distribution

- Sequentially **update** it while observing different experiments using Bayes rule
    - Allows us to fit hyperparameter space better and, thus, find the configuration with highest performance.

Frameworks for Hyperparameter Tuning

scikit learn

tune

HyperSearch

SKORCH

Hyperopt

Institut
québécois
d'intelligence
artificielle

Mila

PEP8

PEP8

- Style guide for Python code

- Check if your code is compliant with PEP8

    - Use of a linter:
        - tool that analyzes source code to flag programming errors, bugs, stylistic errors
        - E.g,: Flake8

    - flake8 --ignore W,F, E501