Neural Networks visualization: Tensorboard and Hiddenlayer

- Meetu Malhotra

Agenda

- Introduction to Tensorboard
- Type of visualizations in tensorboard
- Tensorboard libraries
- Hiddenlayer
- Torchviz

Introduction to Tensorboard

- Google’s tensorflow’s tensorboard is a web server to serve visualizations of the training progress of a neural network
- TensorBoard allows easy visualization of data. You can visualize images, you can visualize text and audio data.
- TensorBoard provides the visualization and tools needed for machine learning experimentation:
  - Tracking and visualizing metrics such as loss and accuracy
  - Visualizing the model graph (ops and layers)
  - Viewing histograms of weights and biases, and how they change over time
  - Displaying images, text, and audio data
  - Visualizing multiple models with different hyper parameters

Type of visualizations in tensorboard:

- Scalars show how the loss and metrics change with every epoch
- Graphs help you visualize your model, with all nodes, layers and weights and biases linkages and their respective information
- Histograms and Distributions show the distribution of weights and biases and verify that they are changing in an expected way
- Distribution view is a top view of the histogram view

Tensorboard libraries

- Loading the associated libraries –

```
from torch.utils.tensorboard import SummaryWriter
import tensorflow as tf
```

- SummaryWriter is the primary class used in tensorboard. It has multiple methods, which can be used to visualize different data.
  For example, add_scaler() to display scaler data, add_histogram() to display weights and biases distributions, add_audio() method to display audio data etc.

- Loading the tensorboard to notebook extension is required

```
%load_ext tensorboard
```

- To display tensorboard-

```
%tensorboard --logdir runs/
```

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

1. If port X is already in use, you can use below command where you specify port number Y, while running tensorboard

```
%tensorboard --logdir log/runs --port=8008
```

Or you can kill the pid using command !kill <PID>

2. Its always a good practice to save different models in different log directories

For example,

```
%tensorboard --logdir log/model1_HPset1
%tensorboard --logdir log/model2_HPset2
%tensorboard --logdir log/model3_HPset3
%tensorboard --logdir log/model4_HPset4
```

3. You can use keras or Pytorch for neuralNetwork, to visualize the network there are multiple libraries such as

ANN Visualizer
Visual Keras
Keras Model Plot
TensorBoard
hiddenlayer

Hiddenlayer

- A lightweight library for neural network graphs and training metrics for PyTorch, Tensorflow, and Keras.
- Hidden layers internal workings are not visible; network adjusts weights and biases via backpropagation.
- Hidden layers extract useful features from input data to improve prediction accuracy.
- Use HiddenLayer to render a graph of our neural network.
- HiddenLayer is simple, easy to extend, and works great with Jupyter Notebook.
- It's not intended to replace advanced tools, such as TensorBoard, but rather for cases where advanced tools are too big for the task.

- Install HiddenLayer using **`pip install hiddenlayer.`**
- Build a neural network model using PyTorch or TensorFlow.
- Use HiddenLayer to generate a graph of the model with **`build_graph().`**
- Save the graph as an image file with the **`save()`** method.
- Optionally, generate a heatmap of layer activations with **`build_activation_graph()`**.

Torchviz

- TorchViz is a Python package that allows us to visualize PyTorch neural network architectures in a simple and intuitive way.
- The package can generate graphs for both forward and backward computations of a PyTorch model, which makes it easy to understand how the network is functioning during training.
- First we need to intsall **torchviz** package by running **pip install torchviz.**
- The **make_dot** function from the **torchviz** package creates a visualization of the computational graph of a PyTorch model.
- The graph generated by `make_dot()` represents the computation graph of the PyTorch model.
- Each node in the graph represents a PyTorch operation, such as a convolution or a fully connected layer, while the edges represent the flow of data through the network.
- The graph also includes information about the dimensions of the tensors flowing through the network, which can be helpful in understanding the network's behavior.

References:

- https://www.datacamp.com/tutorial/tensorboard-tutorial
- https://github.com/mert-kurttutan/torchview
- https://debuggercafe.com/implementing-resnet18-in-pytorch-from-scratch/
- https://github.com/szagoruyko/pytorchviz
- https://github.com/waleedka/hiddenlayer