ML4Cyber — Lab 4 API Reference

PyTorch (`torch`)

Docs: `https://pytorch.org/docs/stable/`

- Use `torch.cuda.is_available()` (`https://pytorch.org/docs/stable/generated/torch.cuda.is_available.html`) to check if CUDA GPU is available.
- Use `torch.backends.mps.is_available()` (`https://pytorch.org/docs/stable/backends.html#mps`) to check if Apple Silicon GPU (MPS) is available.
- Use `torch.device(...)` (`https://pytorch.org/docs/stable/tensor_attributes.html#torch.device`) to create a device object (e.g., `"cuda"`, `"mps"`, `"cpu"`).
- Use `model.to(device)` (`https://pytorch.org/docs/stable/generated/torch.nn.Module.html#torch.nn.Module.to`) to move a model to a device.
- Use `torch.inference_mode()` (`https://pytorch.org/docs/stable/generated/torch.inference_mode.html`) when you only need forward passes (faster; no gradients).
- Use `tensor.cpu()` (`https://pytorch.org/docs/stable/generated/torch.Tensor.cpu.html`) to move a tensor to CPU before converting to NumPy.
- Use `tensor.numpy()` (`https://pytorch.org/docs/stable/generated/torch.Tensor.numpy.html`) to convert a CPU tensor to a NumPy array.

Torch Neural Networks (`torch.nn`)

Docs: `https://pytorch.org/docs/stable/nn.html`

- Use `nn.Sequential(...)` (`https://pytorch.org/docs/stable/generated/torch.nn.Sequential.html`) to build a model as a simple ordered stack of layers.
- Use `nn.Linear(in_features, out_features)` (`https://pytorch.org/docs/stable/generated/torch.nn.Linear.html`) for a fully-connected layer.
- Use `nn.Dropout(p)` (`https://pytorch.org/docs/stable/generated/torch.nn.Dropout.html`) to randomly drop activations during training (helps reduce over-fitting).
- Use `nn.GELU()` (`https://pytorch.org/docs/stable/generated/torch.nn.GELU.html`) as an activation function (smooth alternative to ReLU).
- Use `model.parameters()` (`https://pytorch.org/docs/stable/generated/torch.nn.Module.html#torch.nn.Module.parameters`) to pass model parameters into an optimizer.

Torch NN Functional (`torch.nn.functional`)

Docs: `https://pytorch.org/docs/stable/nn.functional.html`

- Use `torch.nn.functional.mse_loss(x, y)` (`https://pytorch.org/docs/stable/generated/torch.nn.functional.mse_loss.html`) for mean-squared error reconstruction loss.

Torch Optimizers (`torch.optim`)

Docs: `https://pytorch.org/docs/stable/optim.html`

- Use `torch.optim.AdamW(...)` (`https://pytorch.org/docs/stable/generated/torch.optim.AdamW.html`) to optimize model parameters (Adam with weight decay).

Torch Data Utilities (`torch.utils.data`)

Docs: `https://pytorch.org/docs/stable/data.html`

- Use `torch.utils.data.DataLoader(...)` (`https://pytorch.org/docs/stable/data.html#torch.utils.data.DataLoader`) to batch and shuffle training data.

NumPy (`numpy`)

Docs: `https://numpy.org/doc/stable/`

- Use `np.load(...)` (`https://numpy.org/doc/stable/reference/generated/numpy.load.html`) to load arrays from `.npz` files.
- Use `np.savez(...)` (`https://numpy.org/doc/stable/reference/generated/numpy.savez.html`) to save multiple arrays into one `.npz` file.

scikit-learn (`sklearn`)

Docs: `https://scikit-learn.org/stable/`

t-SNE (`sklearn.manifold`)

- Use `TSNE(...).fit_transform(X)` (`https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html`) to create a 2D embedding for visualization.

K-means (`sklearn.cluster`)

- Use `MiniBatchKMeans(...).fit_predict(X)` (`https://scikit-learn.org/stable/modules/generated/sklearn.cluster.MiniBatchKMeans.html`) to cluster samples and get cluster IDs.
- Use `kmeans.inertia_` (`https://scikit-learn.org/stable/modules/generated/sklearn.cluster.MiniBatchKMeans.html`) to access total within-cluster squared error (useful for MSE vs. k).

Metrics (`sklearn.metrics`)

- Use `confusion_matrix(y_true, y_pred)` (`https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html`) to compare labels vs. cluster assignments.

Matplotlib (`matplotlib`)

Docs: `https://matplotlib.org/stable/`

- Use `plt.figure(...)` (Pyplot tutorial: `https://matplotlib.org/stable/tutorials/pyplot.html`) to create a figure.
- Use `plt.plot(...)` (`https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.plot.html`) for line plots (e.g., MSE vs. k, entropy vs. k).
- Use `plt.xlabel(...)` / `plt.ylabel(...)` (`https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.xlabel.html`) to label axes.
- Use `plt.legend()` (`https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.legend.html`) to show a legend.
- Use `plt.show()` (`https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.show.html`) to display the figure.

Python Standard Library

- Use `itertools.chain(...)` (`https://docs.python.org/3/library/itertools.html#itertools.chain`) to combine parameters from encoder and decoder when creating an optimizer.
- If your lab provides a timing helper (e.g., `with timeit("..."): ...`), you may see `contextlib` used behind the scenes; docs: `https://docs.python.org/3/library/contextlib.html`.