A Quarto Page Layout Example

Inspired by Tufte Handout, Using Quarto

2024-01-15

This is how to write some text including some citations as sidenotes

Quarto (abbreviated Qto, 4to or 4º) is the format of a book or pamphlet produced from full sheets printed with eight pages of text, four to a side, then folded twice to produce four leaves. The leaves are then trimmed along the folds to produce eight book pages. Each printed page presents as one-fourth size of the full sheet.[1]

[1] If this is working that is really nice.

The earliest known European printed book is a quarto, the Sibyllenbuch, believed to have been printed by Johannes Gutenberg in 1452–53, before the Gutenberg Bible, surviving only as a fragment. Quarto is also used as a general description of size of books that are about 12 inches (30 cm) tall, and as such does not necessarily indicate the actual printing format of the books, which may even be unknown, as is the case for many modern books. These terms are discussed in greater detail in book sizes.

```
# this is a sample code block to reverse a list
def reverse_list(list):                                   (1)
    if something_happens:
        print('this happened')
    else:
        print('this happened becuase the other did not')
```

(1) will this code annotation work ? hardly

This is how to embed an image

Pro Tip: This dataset was cleaned using the Pandas library before ingestion.

What can we do with lisette lib

Overview

Feature	Category	Description
Layered API	Architecture	A hierarchy of High (ready-to-use), Mid (core loop), and Low (PyTorch wrappers) level APIs, allowing both beginners and experts to work efficiently.

Feature	Category	Description
DataBlock API	Data Loading	A flexible, “Lego-like” system for defining how to get items, label them, split them, and augment them without writing custom PyTorch datasets.
Learning Rate Finder	Training	The iconic `learn.lr_find()` tool that plots loss against learning rate, helping you pick the perfect hyperparameter before training starts.
One-Cycle Policy	Optimization	Implements Leslie Smith’s 1cycle policy (`fit_one_cycle`), allowing models to train much faster and with higher accuracy than standard SGD.
Discriminative LRs	Optimization	The ability to apply different learning rates to different layers of a neural network (e.g., training the “head” faster than the pre-trained body).

Feature	Category	Description
Transfer Learning	Modeling	First-class support for Transfer Learning via `learn.fine_tune()`, which automatically freezes and unfreezes layers correctly during training.
Tabular Embeddings	Tabular	Automatically creates trainable embedding matrices for high-cardinality categorical variables in tabular datasets.
ULMFiT	NLP	Universal Language Model Fine-tuning. A transfer learning method for NLP that performs state-of-the-art text classification.
Mixed Precision	Performance	Native support for fp16 training (`to_fp16()`), which speeds up training on modern GPUs and reduces memory usage.

Feature	Category	Description
TTA (Test Time Augmentation)	Inference	Automatically creates multiple augmented versions of test images and averages their predictions for higher accuracy (`learn.tta()`).
Transforms Pipeline	Augmentation	A dual-pipeline system (item transforms vs. batch transforms) that runs augmentations on the GPU for speed.
Callbacks System	Customization	A rich event system allowing you to inject code at any point in the training loop (e.g., `SaveModelCallback`, `EarlyStoppingCallback`).