CN24 Documentation

Release 4.0.0-alpha

Clemens-Alexander Brust

Sep 01, 2021

Contents:

CHAPTER 1

Introduction

Welcome to CN24!

CHAPTER 2

CN24 as a Library

CHAPTER 3

Networks

Network architectures in CN24 are defined using a JSON file. The basic layout looks like the following example:

```
{
  "hyperparameters": { /* See section on hyperparameters */ },
  "net": {
    "task":         "detection",
    "input":        "conv1",
    "output":       "fc7",
    "nodes":        { /* See section on layer types */ },
    "error_layer": "square"
  },
  "data_input": { /* See section on data input */ }
}
```

3.1 Hyperparameters

This section controls the optimization process. The following hyperparameters can (and *should*) be set:

- **batch_size_parallel**: Sets the fourth dimension of the network's input. This directly affects VRAM usage if you are using a GPU.
- **batch_size_sequential**: If you want to use a larger minibatch size than your memory would allow using **batch_size_parallel**, you can change **batch_size_sequential**. The effective minibatch size is the product of both.
- **epoch_iterations**: The number of iterations (gradient steps) per epoch. This is an arbitrary setting. If it is not set, an epoch will have one iteration per training sample.
- **optimization_method**: Choose the optimizer you want to use for your network. Currently, the following optimization methods are supported:
  - *adam*: The Adam optimizer. It can be configured using the following hyperparameter keys:
    - **ad_step_size**, **ad_beta1** and **ad_beta2**: Matches the α, β_1 and β_2 parameters from the Adam paper.

        * **ad_epsilon**: Mathces the ϵ parameter from the Adam paper.
        * **ad_sqrt_step_size**: If set to 1, the effective step size will be α divided by the square root of the number of iterations already processed.
    - *gd*: Standard stochastic gradient descent with momentum. Using the number of iterations t, the effective learning rate is $\eta(1 + \gamma t)^q$. SGD supports the following hyperparameter keys:
        * **learning_rate**: Sets the learning rate η for gradient descent.
        * **learning_rate_exponent**: Sets the exponent q for the effective learning rate.
        * **learning_rate_gamma**: Sets the coefficient γ for the effective learning rate.
        * **gd_momentum**: Sets the momentum coefficient.
- **l1**: The coefficient for L_1 regularization of weights.
- **l2**: The coefficient for L_2 regularization of weights.

An example block might look like this:

```
"hyperparameters": {
  "testing_ratio": 1,
  "batch_size_parallel": 2,
  "batch_size_sequential": 32,
  "epoch_iterations": 100,
  "l1": 0,
  "l2": 0.0005,
  "optimization_method": "adam",
  "ad_step_size": 0.000001
}
```

3.2 Data Input

This section specifies the input size into the network. It is required because the node list does not contain any information on input or output shapes of the nodes.

3.3 Layer Types

3.3.1 Convolution Layer

3.3.2 Maximum Pooling Layer

CHAPTER 4

CN24 Shell

The CN24 shell should cover most experimental settings for supervised learning. To get an overview of all possible commands, enter the `help` command:

```
./cn24-shell
  [Version number etc...]
cn24> help
```

4.1 Usage

`cn24-shell [-v] [-q] [<SCRIPT>]`

- `-v`, `--verbose`: *Verbose mode*, extra information useful for debugging.
- `-q`, `--quiet`: *Quiet mode*, suppresses unimportant output, useful for scripting.
- `<SCRIPT>`: *Script file*, runs the specified script at startup commands.

4.2 Commands

The following section gives an overview of important `cn24-shell` commands, grouped by topics.

4.2.1 Networks

4.2.2 Models

4.2.3 Datasets

CHAPTER 5

Datasets

5.1 Hierarchy

Data in CN24 is managed in a three-level hierachy:

(1) **Areas** designate the data's experimental purpose. There are 3 default areas: *training*, *staging* and *testing*.

(2) **Bundles** are the default unit of dataset serialization. They can be moved freely between areas. Bundles in the training Area can be assigned a weight that influences the likelihood of selecting training samples from them.

(3) **Segments** contain the samples themselves. They can be moved freely between Bundles. They exist to group samples, e.g., training and validation samples or samples of different classes.

CN24 will create two empty default Bundles: *Default_Training* and *Default_Testing*

```
    Area                          Bundle                          Segment   Samples
Training
       |..............Default_Training                                          95
       |..................Weight:    1
                                      |.......................UM_road         95

 Staging
       |.............KITTIRoadTraining                                        193
                                      |.......................UM_lane         95
                                      |.......................UU_road         98

 Testing
       |..............Default_Testing                                         96
                                      |......................UMM_road         96
```

5.2 Data Format

Data is provided to CN24 in the form of serialized Bundles. The serialization method of choice is JSON, provided by nlohnmann's JSON library.

The Bundle format is best explained by an example:

```
{
  "name": "SampleBundle",
  "segments:" [
    {
      "name": "SampleSegmentA",
      "samples": [ ]
    },
    {
      "name": "SampleSegmentB",
      "samples": [ ]
    }
  ]
}
```

The samples themselves are JSON objects as well. Their exact schema depends on the task.

5.2.1 Detection

CN24 supports detection using the YOLO method. Samples need to specify the following:

- **image_filename**: Input image file
- **boxes**: JSON array of bounding boxes

Bounding boxes have the following properties:

- **x**, **y**: Coordinates of the *center* of the bounding box (pixels)
- **w**, **h**: Width and height of the bounding box (pixels)
- **class**: Class of the object inside the bounding box

Optionally, you can specificy these:

- **difficult**: If set to 1, the box is ignored during testing
- **dont_scale**: Instead of pixels, the coordinates and dimensions of the box are specified as normalized fractions of the image dimensions

The following is an example from the PASCAL VOC dataset:

```
{
  "boxes": [
    {
      "class": "bird",
      "difficult": 0,
      "h": 286,
      "w": 156,
      "x": 338,
      "y": 190
    }
  ],
  "image_filename": "2011_003213.jpg"
}
```

5.2.2 Classification

5.2.3 Binary Segmentation

Samples for binary segmentation consist of two image files with equal dimensions. One is the actual input image and the other the label image. At the moment, only binary segmentation is supported. Grayscale label images are preferred. However, CN24 will also accept RGB images as labels. In this case, the value of the third channel will be used as a label.

The following properties need to be specified:

- **image_filename**: Input image file
- **label_filename**: Label file

Optionally, you can supply a value for **localized_error_function**. Currently, the only supported values are *default* and *kitti*.

The following is an example from the KITTI-Vision Road Dataset:

```
{
  "label_filename": "gt_image_2/umm_road_000049.png",
  "localized_error_function": "kitti",
  "image_filename": "image_2/umm_000049.png"
}
```

To get started,...

What is CN24?

What is it not?