

Configuration Manual: A Deep Learning Feature-Ranked Backpropagation Framework for Sustainability

MSc Research Project Artificial Intelligence

Gaurav Student ID: x23189487

School of Computing National College of Ireland

Supervisor:

Paul Stynes

National College of Ireland Project Submission Sheet School of Computing



Student Name:	Gaurav
Student ID:	x23189487
Programme:	Artificial Intelligence
Year:	2024
Module:	MSc Research Project
Supervisor:	Paul Stynes
Submission Due Date:	12/12/2024
Project Title:	Configuration Manual: A Deep Learning Feature-Ranked
	Backpropagation Framework for Sustainability
Word Count:	936
Page Count:	5

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

<u>ALL</u> internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	
Date:	11th December 2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	
Attach a Moodle submission receipt of the online project submission, to	
each project (including multiple copies).	
You must ensure that you retain a HARD COPY of the project, both for	
your own reference and in case a project is lost or mislaid. It is not sufficient to keep	
a copy on computer	

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual: A Deep Learning Feature-Ranked Backpropagation Framework for Sustainability

Gaurav x23189487

This configuration manual provides detailed information to ensure reproducibility and consistency across environments. It has all the specific required software tools, settings, and configurations required to replicate the experimental setup for training and evaluating the proposed framework.

1 System Requirements

The code was tested on AWS Deep Learning AMI: **g4dn.12xlarge** instance with **48vCPUs**, **196GB RAM** and **4x16GB** Nvidia Tesla T4 GPUs.

1.1 Hardware Requirements

- 1. CPU: 16-core or higher (multi-core processing preferred).
- 2. GPU: At least 1 NVIDIA GPU with 16GB VRAM (e.g., Tesla T4 or equivalent). For distributed training, a minimum of 2 GPUs is recommended.
- 3. RAM: Minimum 16GB.
- 4. Storage: At least 500GB of available disk space for dataset storage, logs and output model files.

1.2 Software Requirements

- 1. Operating System: Ubuntu 22.04 or Windows equivalent (tested on AWS Deep Learning AMI with Ubuntu 22.04).
- 2. GPU Driver: NVIDIA Driver version 525 or higher(tested on 550).
- 3. CUDA Toolkit: Version 11.8 or higher(tested on 12.4).
- 4. cuDNN: Version 8.7 or higher(tested on 9.6).

2 Conda Environment Setup

The code is ran and tested in a contained Anaconda environemt which can be downloaded from **Download Anaconda**¹.

All the required packages and libraries on which the framework is dependent upon can be found in **environment.yml**.

This environment file uses Anaconda as base. To create a conda environment simply run:

```
conda env create -n <environment_name> -f environment.yml
conda activate <environment_name>
```

3 Data Preparation

This section guides you on how to download and prepare the dataset from starting to final stage.

3.1 Dataset Source

Dataset used is ILSVRC 2012 from Imagenet which has over 1.2 million images corresponding to 1000 different classes. The dataset is downloaded from HuggingFace, which can be downloaded from ImageNet-1k dataset².

Data can be downloaded directly from the website or by using the following script. To use the script an AccessToken has to be created on the https://huggingface.co.

```
wget --header="Authorization: Bearer <access_token>" https://
huggingface.co/datasets/ILSVRC/imagenet-1k/resolve/main/data/<
file_name>.tar.gz -0 <file_name>.tar.gz
```

Note

We have only used training and validation data. The reason behind making this call was: the test data is completely unlabelled, due to this calculating models performance was difficult without manual validation.

3.2 File Extraction

To extract .tar.gz files:

```
tar -xzvf <file_name>.tar.gz
```

¹Downlaod Anaconda: https://docs.conda.io/projects/conda/en/latest/user-guide/ install/index.html

²ImageNet-1k dataset: https://huggingface.co/datasets/ILSVRC/imagenet-1k/tree/main/ data

3.3 Dataset Organization

After extracting the data you will find that all the images are in the same folder and distinguishing between the classes of images is impossible. To handle this, inside data_preperation folder, there are five scripts which servers their own purpose:

- 1. move_train_files_to_class_folder.py and move_val_files_to_class_folder.py to organize training and validation datasets into class-based folders.
- 2. Use combine_data.py to merge the datasets.
- 3. If needed, reduce the number of classes as per requirement using reduce_combined_classes.py. This step can be ignored if training and testing on complete dataset.

3.4 Dataset Split

Split the dataset into training, validation, and testing sets using **split_dataset.py** with a 60:20:20 ratio respectively.

4 Configuration Settings

The complete process of training, testing and evaluation is controlled from one file: config.yaml.

4.1 config.yaml File

The experimental parameters can be configured:

4.1.1 Data Settings

- train_dir: Path to training data folder (e.g., ../data/train).
- val_dir: Path to validation data folder (e.g., ../data/val).
- test_dir: Path to test data folder (e.g., ../data/test).

4.1.2 Model Settings

- model_type: Options are 'alexnet', 'resnet', 'vggnet'.
- num_classes: Total number of output classes (e.g., 50). This should match the number of classes selected during Step 3.
- architecture: 'standard' for full backpropagation or 'selective' for selective backpropagation.
- batch_size: Number of batches the data should be split in.

4.1.3 Training Parameters

- learning_rate: Initial learning rate (e.g., 0.0001).
- num_epochs: Number of training epochs (e.g., 50). However, the is equipped with Early Stopping to overcome the the problem of overfitting if it arises due to large number of epochs.
- high_percentage: Starting percentage of weights to update (e.g., 50). Used when 'architecture' set to 'selective'.
- low_percentage: Final percentage of weights to update (e.g., 20). Used when 'architecture' set to 'selective'.
- percentage_schedule: Schedule for percentage adjustment ('linear' or 'exponential'). From high to low. Used when 'architecture' set to 'selective'.

4.1.4 Logging

- log_dir: Directory for saving logs (e.g., logs). All the logs, metrics, results are saved in this directory based on the type of model. For example, alexnet standard model is training, log_dir:logs/alexnet_standard/.
- log_interval: Frequency of logging (e.g., 10 iterations).

4.1.5 Distributed Training Setup

PyTorch's Distributed Data Parallel (DDP) was used to handle multi-GPU training. It haas the following key configurations:

- world_size: Number of GPUs to use (e.g., 4).
- backend: Communication backend ('nccl' for NVIDIA GPUs).
- Environment variables:

```
export MASTER_ADDR=localhost
export MASTER_PORT=12355
```

5 Execution

The code uses a single 'main.py' file which is used for training, testing and evaluation purposes.

5.1 Training and Testing

After configuration settings, to 'train and test' a model of your choice, run the following command(without graphs generation for metrics):

python main.py --config config.yaml

If graphs for metrics are required as well, run the following command

python main_ --config config.yaml --generate_graphs

6 Logging and Results

- 1. Training and validation metrics (e.g., accuracy, loss) are logged in 'log_dir' as CSV files.
- 2. Testing reults are logged in the 'log_dir'.
- 3. Training and testing logs are saved in the 'log_dir' as '.log' files.
- 4. Generated graphs are saved in 'graphs_dir'.
- 5. Trained model weights are saved in the 'checkpoints' directory.

To ensure reproducibility of the results, random seeds are used in 'train.py' and 'test.py' files.

```
import random
import numpy as np
import torch
random.seed(42)
np.random.seed(42)
torch.manual_seed(42)
```

References