DeepPhysX_Torch

Robin ENJALBERT, Alban ODOT, Stephane COTIN

Jun 08, 2022

CONTENTS

Fully Connected	*Network*	*UNet*
FC *FCConfig*	*TorchDataTransformation* *TorchNetwork* *TorchNetworkConfig* *TorchOptimization*	*UNetDataTransformation* *UNet* *UNetConfig*

CHAPTER

ONE

FULLY CONNECTED

1.1 FC

Base: *TorchNetwork.TorchNetwork*

class FC.**FC**(*config: namedtuple*)

Create a Fully Connected layers Neural Network Architecture.

Parameters
config (*namedtuple*) – Namedtuple containing FC parameters

forward(*input_data: Tensor*) → Tensor

Gives input_data as raw input to the neural network.

Parameters
input_data (*torch.Tensor*) – Input tensor

Returns
Network prediction

1.2 FCConfig

Base: *TorchNetworkConfig.TorchNetworkConfig*

class FCConfig.**FCConfig**(*optimization_class: ~typing.Type[~DeepPhysX.Torch.Network.TorchOptimization.TorchOptimization] = <class 'DeepPhysX.Torch.Network.TorchOptimization.TorchOptimization'>, data_transformation_class: ~typing.Type[~DeepPhysX.Torch.Network.TorchDataTransformation.TorchDataTransformation] = <class 'DeepPhysX.Torch.Network.TorchDataTransformation.TorchDataTransformation'>, network_dir: ~typing.Optional[str] = None, network_name: str = 'FCNetwork', which_network: int = 0, save_each_epoch: bool = False, data_type: str = 'float32', lr: ~typing.Optional[float] = None, require_training_stuff: bool = True, loss: ~typing.Optional[~typing.Any] = None, optimizer: ~typing.Optional[~typing.Any] = None, dim_output: int = 0, dim_layers: ~typing.Optional[list] = None, biases: ~typing.Union[~typing.List[bool], bool] = True*)

FCConfig is a configuration class to parameterize and create FC, TorchOptimization and TorchDataTransformation for the NetworkManager.

Parameters

- **optimization_class** (*Type[TorchOptimization]*) – BaseOptimization class from which an instance will be created
- **data_transformation_class** (*Type[TorchDataTransformation]*) – DataTransformation class from which an instance will be created
- **network_dir** (*Optional[str]*) – Name of an existing network repository
- **network_name** (*str*) – Name of the network
- **which_network** (*int*) – If several networks in network_dir, load the specified one
- **save_each_epoch** (*bool*) – If True, network state will be saved at each epoch end; if False, network state will be saved at the end of the training
- **data_type** (*str*) – Type of the training data
- **lr** (*Optional[float]*) – Learning rate
- **require_training_stuff** (*bool*) – If specified, loss and optimizer class can be not necessary for training
- **loss** (*Optional[Any]*) – Loss class
- **optimizer** (*Optional[Any]*) – Network's parameters optimizer class
- **dim_output** (*int*) – Dimension of the output
- **dim_layers** (*Optional[List[int]]*) – Size of each layer of the network
- **biases** (*Union[List[bool], bool]*) – Layers should have biases or not. This value can either be given as a bool for all layers or as a list to detail each layer.

CHAPTER

TWO

NETWORK

2.1 TorchDataTransformation

Base: DataTransformation.DataTransformation

class TorchDataTransformation.**TorchDataTransformation**(*config: namedtuple*)

TorchDataTransformation is dedicated to data operations before and after network predictions.

Parameters
config (*namedtuple*) – Namedtuple containing the parameters of the network manager

transform_before_apply(**args*)

Apply data operations between loss computation and prediction apply in environment.

Parameters
data_out (*Any*) – Prediction data

Returns
Transformed prediction data

transform_before_loss(**args*)

Apply data operations between network's prediction and loss computation.

Parameters

- **data_out** (*Any*) – Prediction data
- **data_gt** (*Optional[Any]*) – Ground truth data

Returns
Transformed prediction data, transformed ground truth data

transform_before_prediction(**args*)

Apply data operations before network's prediction.

Parameters
data_in (*Any*) – Input data

Returns
Transformed input data

2.2 TorchNetwork

Base: `BaseNetwork.BaseNetwork`

`class TorchNetwork.TorchNetwork(config: namedtuple)`

TorchNetwork is a network class to compute predictions from input data according to actual state.

Parameters
config (*namedtuple*) – Namedtuple containing BaseNetwork parameters

`forward(input_data: Tensor) → Tensor`

Gives input_data as raw input to the neural network.

Parameters
input_data (*torch.Tensor*) – Input tensor

Returns
Network prediction

`get_parameters() → Dict[str, Tensor]`

Return the current state of Network parameters.

Returns
Network parameters

`load_parameters(path: str) → None`

Load network parameter from path.

Parameters
path (*str*) – Path to Network parameters to load

`nb_parameters() → int`

Return the number of parameters of the network.

Returns
Number of parameters

`static print_architecture(architecture) → str`

Format the network architecture string description.

Returns
String containing the network architecture description

save_parameters(*path: str*) → None

Saves the network parameters to the path location.

Parameters
path (*str*) – Path where to save the parameters.

set_device() → None

Set computer device on which Network's parameters will be stored and tensors will be computed.

set_eval() → None

Set the Network in eval mode (does not compute gradient).

set_train() → None

Set the Network in train mode (compute gradient).

transform_from_numpy(*data: ndarray*, *grad: bool = True*) → Tensor

Transform and cast data from numpy to the desired tensor type.

Parameters

- **data** (*ndarray*) – Array data to convert
- **grad** (*bool*) – If True, gradient will record operations on this tensor

Returns
Converted tensor

transform_to_numpy(*data: Tensor*) → ndarray

Transform and cast data from tensor type to numpy.

Parameters
data (*torch.Tensor*) – Any to convert

Returns
Converted array

2.3 TorchNetworkConfig

Base: `BaseNetworkConfig.BaseNetworkConfig`

class TorchNetworkConfig.**TorchNetworkConfig**(*network_class: ~typing.Type[~DeepPhysX.Torch.Network.TorchNetwork.TorchNetwork] = <class 'DeepPhysX.Torch.Network.TorchNetwork.TorchNetwork'>*, *optimization_class: ~typing.Type[~DeepPhysX.Torch.Network.TorchOptimization.TorchOptimization] = <class 'DeepPhysX.Torch.Network.TorchOptimization.TorchOptimization'>*, *data_transformation_class: ~typing.Type[~DeepPhysX.Torch.Network.TorchDataTransformation.TorchDataTr = <class 'DeepPhysX.Torch.Network.TorchDataTransformation.TorchDataTransformation'>*, *network_dir: ~typing.Optional[str] = None*, *network_name: str = 'Network'*, *network_type: str = 'TorchNetwork'*, *which_network: int = 0*, *save_each_epoch: bool = False*, *data_type: str = 'float32'*, *lr: ~typing.Optional[float] = None*, *require_training_stuff: bool = True*, *loss: ~typing.Optional[~typing.Any] = None*, *optimizer: ~typing.Optional[~typing.Any] = None*)

TorchNetworkConfig is a configuration class to parameterize and create TorchNetwork, TorchOptimization and TorchDataTransformation for the NetworkManager.

Parameters

- **network_class** (*Type[TorchNetwork]*) – BaseNetwork class from which an instance will be created
- **optimization_class** (*Type[TorchOptimization]*) – BaseOptimization class from which an instance will be created
- **data_transformation_class** (*Type[TorchDataTransformation]*) – DataTransformation class from which an instance will be created
- **network_dir** (*Optional[str]*) – Name of an existing network repository
- **network_name** (*str*) – Name of the network
- **network_type** (*str*) – Type of the network
- **which_network** (*int*) – If several networks in network_dir, load the specified one
- **save_each_epoch** (*bool*) – If True, network state will be saved at each epoch end; if False, network state will be saved at the end of the training
- **data_type** (*str*) – Type of the training data
- **lr** (*Optional[float]*) – Learning rate
- **require_training_stuff** (*bool*) – If specified, loss and optimizer class can be not necessary for training
- **loss** (*Optional[Any]*) – Loss class
- **optimizer** (*Optional[Any]*) – Network's parameters optimizer class

create_data_transformation() → Union[DataTransformation, TorchDataTransformation]

Create an instance of data_transformation_class with given parameters.

Returns
DataTransformation object from data_transformation_class and its parameters.

`create_network()` → Union[BaseNetwork, TorchNetwork]

Create an instance of network_class with given parameters.

Returns
BaseNetwork object from network_class and its parameters.

`create_optimization()` → Union[BaseOptimization, TorchOptimization]

Create an instance of optimization_class with given parameters.

Returns
BaseOptimization object from optimization_class and its parameters.

2.4 TorchOptimization

Base: `BaseOptimization.BaseOptimization`

`class TorchOptimization.TorchOptimization(`*config: namedtuple*`)`

TorchOptimization is dedicated to network optimization: compute loss between prediction and target, update network parameters.

Parameters
config (*namedtuple*) – Namedtuple containing TorchOptimization parameters

`compute_loss(`*prediction: Tensor*, *ground_truth: Tensor*, *data: Dict[str, Any]*`)` → Dict[str, float]
Compute loss from prediction / ground truth.

Parameters

- **prediction** (*Tensor*) – Tensor produced by the forward pass of the Network
- **ground_truth** (*Tensor*) – Ground truth tensor to be compared with prediction
- **data** (*Dict[str, Any]*) – Additional data sent as dict to compute loss value

Returns
Loss value

`optimize()` → None

Run an optimization step.

`set_loss()` → None

Initialize the loss function.

set_optimizer(*net*) → None

Define an optimization process.

Parameters
net (*BaseNetwork*) – Network whose parameters will be optimized.

transform_loss(*data: Dict[str, Any]*) → Dict[str, float]

Apply a transformation on the loss value using the potential additional data.

Parameters
data (*Dict[str, Any]*) – Additional data sent as dict to compute loss value

Returns
Transformed loss value

CHAPTER

THREE

UNET

3.1 UNet

Base: *TorchNetwork.TorchNetwork*

class UNet.**UNet**(*config: namedtuple*)

Create a UNet Neural Network Architecture.

Parameters
config (*namedtuple*) – Namedtuple containing UNet parameters

forward(*input_data: Tensor*) → Tensor

Gives input_data as raw input to the neural network.

Parameters
input_data (*torch.Tensor*) – Input tensor

Returns
Network prediction

3.2 UNetConfig

Base: *TorchNetworkConfig.TorchNetworkConfig*

class UNetConfig.**UNetConfig**(*optimization_class: ~typing.Type[~DeepPhysX.Torch.Network.TorchOptimization.TorchOptimization] = <class 'DeepPhysX.Torch.Network.TorchOptimization.TorchOptimization'>, data_transformation_class: ~typing.Type[~DeepPhysX.Torch.UNet.UnetDataTransformation.UnetDataTransformation] = <class 'DeepPhysX.Torch.UNet.UnetDataTransformation.UnetDataTransformation'>, network_dir: ~typing.Optional[str] = None, network_name: str = 'UNetNetwork', which_network: int = 0, save_each_epoch: bool = False, data_type: str = 'float32', lr: ~typing.Optional[float] = None, require_training_stuff: bool = True, loss: ~typing.Optional[~typing.Any] = None, optimizer: ~typing.Optional[~typing.Any] = None, input_size: ~typing.Optional[~typing.List[int]] = None, nb_dims: int = 3, nb_input_channels: int = 1, nb_first_layer_channels: int = 64, nb_output_channels: int = 3, nb_steps: int = 3, two_sublayers: bool = True, border_mode: str = 'valid', skip_merge: bool = False, data_scale: float = 1.0*)

UNetConfig is a configuration class to parameterize and create UNet, TorchOptimization and UNetDataTransformation for the NetworkManager.

Parameters

- **optimization_class** (*Type[TorchOptimization]*) – BaseOptimization class from which an instance will be created
- **data_transformation_class** (*Type[TorchDataTransformation]*) – DataTransformation class from which an instance will be created
- **network_dir** (*Optional[str]*) – Name of an existing network repository
- **network_name** (*str*) – Name of the network
- **which_network** (*int*) – If several networks in network_dir, load the specified one
- **save_each_epoch** (*bool*) – If True, network state will be saved at each epoch end; if False, network state will be saved at the end of the training
- **data_type** (*str*) – Type of the training data
- **lr** (*Optional[float]*) – Learning rate
- **require_training_stuff** (*bool*) – If specified, loss and optimizer class can be not necessary for training
- **loss** (*Optional[Any]*) – Loss class
- **optimizer** (*Optional[Any]*) – Network's parameters optimizer class
- **input_size** (*List[int]*) – Size of the input
- **nb_dims** (*int*) – Number of dimension of data
- **nb_input_channels** (*int*) – Number of channels of the input layer
- **nb_first_layer_channels** (*int*) – Number of channels of the first layer
- **nb_output_channels** (*int*) – Number of channels of the output layer
- **nb_steps** (*int*) – Number of steps of down layers / up layers
- **two_sublayers** (*bool*) – Duplicate each layer or not
- **border_mode** (*str*) – Zero-padding mode

- **skip_merge** (*bool*) – Skip the crop step at each up layer or not
- **data_scale** (*float*) – Scale to apply to data

3.3 UNetDataTransformation

Base: *TorchDataTransformation.TorchDataTransformation*

class UnetDataTransformation.**UnetDataTransformation**(*config: namedtuple*)

UNetDataTransformation is dedicated to data operations before and after UNet predictions.

Parameters
config (*namedtuple*) – Namedtuple containing the parameters of the network manager

compute_pad_widths(*desired_shape: List[int]*) → None

Define padding to apply on data given the data shape and the network architecture.

Parameters
desired_shape (*List[int]*) – Data shape without padding

transform_before_apply(**args*)

Apply data operations between loss computation and prediction apply in environment.

Parameters
data_out (*Any*) – Prediction data

Returns
Transformed prediction data

transform_before_loss(**args*)

Apply data operations between network's prediction and loss computation.

Parameters

- **data_out** (*Any*) – Prediction data
- **data_gt** (*Optional[Any]*) – Ground truth data

Returns
Transformed prediction data, transformed ground truth data

transform_before_prediction(**args*)

Apply data operations before network's prediction.

Parameters
data_in (*Any*) – Input data

Returns
Transformed input data

Return to DeepPhysX main page.

INDEX