Multi-GPU data parallel#

Derived from create_supervised_evaluator in Ignite. Factory function for creating an evaluator for supervised models. net (Module) – the model to train. metrics (Optional[Dict[str, Metric]]) – a map of metric names to Metrics. devices (Optional[Sequence[Union[device, str]]]) – device(s) type specification (default: None). Applies to both model and batches. None is all devices used, empty list is CPU only. non_blocking (bool) – if True and this copy is between CPU and GPU, the copy may occur asynchronously with respect to the host. For other cases, this argument has no effect. prepare_batch (Callable) – function that receives batch, device, non_blocking and outputs tuple of tensors (batch_x, batch_y). output_transform (Callable) – function that receives ‘x’, ‘y’, ‘y_pred’ and returns value to be assigned to engine’s state.output after each iteration. Default is returning (y_pred, y,) which fits output expected by metrics. If you change it you should use output_transform in metrics. distributed (bool) – whether convert model to DistributedDataParallel, if True, devices must contain only 1 GPU or CPU for current distributed rank. Note engine.state.output for this engine is defined by output_transform parameter and is a tuple of (batch_pred, batch_y) by default. an evaluator engine with supervised inference function. Engine Derived from create_supervised_trainer in Ignite. Factory function for creating a trainer for supervised models. net (Module) – the network to train. optimizer (Optimizer) – the optimizer to use. loss_fn (Callable) – the loss function to use. devices (Optional[Sequence[Union[device, str]]]) – device(s) type specification (default: None). Applies to both model and batches. None is all devices used, empty list is CPU only. non_blocking (bool) – if True and this copy is between CPU and GPU, the copy may occur asynchronously with respect to the host. For other cases, this argument has no effect. prepare_batch (Callable) – function that receives batch, device, non_blocking and outputs tuple of tensors (batch_x, batch_y). output_transform (Callable) – function that receives ‘x’, ‘y’, ‘y_pred’, ‘loss’ and returns value to be assigned to engine’s state.output after each iteration. Default is returning loss.item(). distributed (bool) – whether convert model to DistributedDataParallel, if True, devices must contain only 1 GPU or CPU for current distributed rank. a trainer engine with supervised update function. Engine Note engine.state.output for this engine is defined by output_transform parameter and is the loss of the processed batch by default.

Workflow#

Workflow defines the core work process inheriting from Ignite engine. All trainer, validator and evaluator share this same workflow as base class, because they all can be treated as same Ignite engine loops. It initializes all the sharable data in Ignite engine.state. And attach additional processing logics to Ignite engine based on Event-Handler mechanism. Users should consider inheriting from trainer or evaluator to develop more trainers or evaluators. device (Union[device, str]) – an object representing the device on which to run. max_epochs (int) – the total epoch number for engine to run, validator and evaluator have only 1 epoch. data_loader (Union[Iterable, DataLoader]) – Ignite engine use data_loader to run, must be Iterable or torch.DataLoader. epoch_length (Optional[int]) – number of iterations for one epoch, default to len(data_loader). non_blocking (bool) – if True and this copy is between CPU and GPU, the copy may occur asynchronously with respect to the host. For other cases, this argument has no effect. prepare_batch (Callable) – function to parse expected data (usually image, label and other network args) from engine.state.batch for every iteration, for more details please refer to: https://pytorch.org/ignite/generated/ignite.engine.create_supervised_trainer.html.

iteration_update (Optional[Callable[[Engine, Any], Any]]) – the callable function for every iteration, expect to accept engine and engine.state.batch as inputs, return data will be stored in engine.state.output. if not provided, use self._iteration() instead. for more details please refer to: https://pytorch.org/ignite/generated/ignite.engine.engine.Engine.html. postprocessing (Optional[Callable]) – execute additional transformation for the model output data. Typically, several Tensor based transforms composed by Compose. key_metric (Optional[Dict[str, Metric]]) – compute metric when every iteration completed, and save average value to engine.state.metrics when epoch completed. key_metric is the main metric to compare and save the checkpoint into files. additional_metrics (Optional[Dict[str, Metric]]) – more Ignite metrics that also attach to Ignite Engine. metric_cmp_fn (Callable) – function to compare current key metric with previous best key metric value, it must accept 2 args (current_metric, previous_best) and return a bool result: if True, will update best_metric and best_metric_epoch with current metric and epoch, default to greater than. handlers (Optional[Sequence]) – every handler is a set of Ignite Event-Handlers, must have attach function, like: CheckpointHandler, StatsHandler, etc. amp (bool) – whether to enable auto-mixed-precision training or inference, default is False. event_names (Optional[List[Union[str, EventEnum]]]) – additional custom ignite events that will register to the engine. new events can be a list of str or ignite.engine.events.EventEnum. event_to_attr (Optional[dict]) – a dictionary to map an event to a state attribute, then add to engine.state. for more details, check: https://pytorch.org/ignite/generated/ignite.engine.engine.Engine.html #ignite.engine.engine.Engine.register_events. decollate (bool) – whether to decollate the batch-first data to a list of data after model computation, recommend decollate=True when postprocessing uses components from monai.transforms. default to True. to_kwargs (Optional[Dict]) – dict of other args for prepare_batch API when converting the input data, except for device, non_blocking. amp_kwargs (Optional[Dict]) – dict of the args for torch.cuda.amp.autocast() API, for more details: https://pytorch.org/docs/stable/amp.html#torch.cuda.amp.autocast. TypeError – When data_loader is not a torch.utils.data.DataLoader. TypeError – When key_metric is not a Optional[dict]. TypeError – When additional_metrics is not a Optional[dict]. Get the statistics information of the workflow process. vars – variables name in the self.state, will use the variable name as the key and the state content as the value. if the variable doesn’t exist, default value is None. Execute training, validation or evaluation based on Ignite Engine. None

Trainer#

Base class for all kinds of trainers, inherits from Workflow. Get the statistics information of the training process. Default to return the rank, current_epoch, current_iteration, total_epochs, total_iterations. vars – except for the default stats, other variables name in the self.state to return, will use the variable name as the key and the state content as the value. if the variable doesn’t exist, default value is None. Execute training based on Ignite Engine. If call this function multiple times, it will continuously run from the previous state. None

SupervisedTrainer#

Standard supervised training method with image and label, inherits from Trainer and Workflow. device – an object representing the device on which to run. max_epochs – the total epoch number for trainer to run. train_data_loader – Ignite engine use

data_loader to run, must be Iterable or torch.DataLoader. network – network to train in the trainer, should be regular PyTorch torch.nn.Module. optimizer – the optimizer associated to the network, should be regular PyTorch optimizer from torch.optim or its subclass. loss_function – the loss function associated to the optimizer, should be regular PyTorch loss, which inherit from torch.nn.modules.loss. epoch_length – number of iterations for one epoch, default to len(train_data_loader). non_blocking – if True and this copy is between CPU and GPU, the copy may occur asynchronously with respect to the host. For other cases, this argument has no effect. prepare_batch – function to parse expected data (usually image, label and other network args) from engine.state.batch for every iteration, for more details please refer to: https://pytorch.org/ignite/generated/ignite.engine.create_supervised_trainer.html. iteration_update – the callable function for every iteration, expect to accept engine and engine.state.batch as inputs, return data will be stored in engine.state.output. if not provided, use self._iteration() instead. for more details please refer to: https://pytorch.org/ignite/generated/ignite.engine.engine.Engine.html. inferer – inference method that execute model forward on input data, like: SlidingWindow, etc. postprocessing – execute additional transformation for the model output data. Typically, several Tensor based transforms composed by Compose. key_train_metric – compute metric when every iteration completed, and save average value to engine.state.metrics when epoch completed. key_train_metric is the main metric to compare and save the checkpoint into files. additional_metrics – more Ignite metrics that also attach to Ignite Engine. metric_cmp_fn – function to compare current key metric with previous best key metric value, it must accept 2 args (current_metric, previous_best) and return a bool result: if True, will update best_metric and best_metric_epoch with current metric and epoch, default to greater than. train_handlers – every handler is a set of Ignite Event-Handlers, must have attach function, like: CheckpointHandler, StatsHandler, etc. amp – whether to enable auto-mixed-precision training, default is False. event_names – additional custom ignite events that will register to the engine. new events can be a list of str or ignite.engine.events.EventEnum. event_to_attr – a dictionary to map an event to a state attribute, then add to engine.state. for more details, check: https://pytorch.org/ignite/generated/ignite.engine.engine.Engine.html #ignite.engine.engine.Engine.register_events. decollate – whether to decollate the batch-first data to a list of data after model computation, recommend decollate=True when postprocessing uses components from monai.transforms. default to True. optim_set_to_none – when calling optimizer.zero_grad(), instead of setting to zero, set the grads to None. more details: https://pytorch.org/docs/stable/generated/torch.optim.Optimizer.zero_grad.html. to_kwargs – dict of other args for prepare_batch API when converting the input data, except for device, non_blocking. amp_kwargs – dict of the args for torch.cuda.amp.autocast() API, for more details: https://pytorch.org/docs/stable/amp.html#torch.cuda.amp.autocast.

GanTrainer#

Generative adversarial network training based on Goodfellow et al. 2014 https://arxiv.org/abs/1406.266, inherits from Trainer and Workflow. Generate m fakes from random latent codes. Update discriminator with these fakes and current batch reals, repeated d_train_steps times. If g_update_latents, generate m fakes from new random latent codes. Update generator with these fakes using discriminator feedback. device – an object representing the device on which to run. max_epochs – the total epoch number for engine to run. train_data_loader – Core ignite engines uses

DataLoader for training loop batchdata. g_network – generator (G) network architecture. g_optimizer – G optimizer function. g_loss_function – G loss function for optimizer. d_network – discriminator (D) network architecture. d_optimizer – D optimizer function. d_loss_function – D loss function for optimizer. epoch_length – number of iterations for one epoch, default to len(train_data_loader). g_inferer – inference method to execute G model forward. Defaults to SimpleInferer(). d_inferer – inference method to execute D model forward. Defaults to SimpleInferer(). d_train_steps – number of times to update D with real data minibatch. Defaults to 1. latent_shape – size of G input latent code. Defaults to 64. non_blocking – if True and this copy is between CPU and GPU, the copy may occur asynchronously with respect to the host. For other cases, this argument has no effect. d_prepare_batch – callback function to prepare batchdata for D inferer. Defaults to return GanKeys.REALS in batchdata dict. for more details please refer to: https://pytorch.org/ignite/generated/ignite.engine.create_supervised_trainer.html. g_prepare_batch – callback function to create batch of latent input for G inferer. Defaults to return random latents. for more details please refer to: https://pytorch.org/ignite/generated/ignite.engine.create_supervised_trainer.html. g_update_latents – Calculate G loss with new latent codes. Defaults to True. iteration_update – the callable function for every iteration, expect to accept engine and engine.state.batch as inputs, return data will be stored in engine.state.output. if not provided, use self._iteration() instead. for more details please refer to: https://pytorch.org/ignite/generated/ignite.engine.engine.Engine.html. postprocessing – execute additional transformation for the model output data. Typically, several Tensor based transforms composed by Compose. key_train_metric – compute metric when every iteration completed, and save average value to engine.state.metrics when epoch completed. key_train_metric is the main metric to compare and save the checkpoint into files. additional_metrics – more Ignite metrics that also attach to Ignite Engine. metric_cmp_fn – function to compare current key metric with previous best key metric value, it must accept 2 args (current_metric, previous_best) and return a bool result: if True, will update best_metric and best_metric_epoch with current metric and epoch, default to greater than. train_handlers – every handler is a set of Ignite Event-Handlers, must have attach function, like: CheckpointHandler, StatsHandler, etc. decollate – whether to decollate the batch-first data to a list of data after model computation, recommend decollate=True when postprocessing uses components from monai.transforms. default to True. optim_set_to_none – when calling optimizer.zero_grad(), instead of setting to zero, set the grads to None. more details: https://pytorch.org/docs/stable/generated/torch.optim.Optimizer.zero_grad.html. to_kwargs – dict of other args for prepare_batch API when converting the input data, except for device, non_blocking. amp_kwargs – dict of the args for torch.cuda.amp.autocast() API, for more details: https://pytorch.org/docs/stable/amp.html#torch.cuda.amp.autocast.

Evaluator#

Base class for all kinds of evaluators, inherits from Workflow. device – an object representing the device on which to run. val_data_loader – Ignite engine use data_loader to run, must be Iterable or torch.DataLoader. epoch_length – number of iterations for one epoch, default to len(val_data_loader). non_blocking – if True and this copy is between CPU and GPU, the copy may occur asynchronously with respect to the host. For other cases, this argument has no effect. prepare_batch – function to parse expected data (usually image, label and other network args) from engine.state.batch for every iteration, for more details please refer to:

https://pytorch.org/ignite/generated/ignite.engine.create_supervised_trainer.html. iteration_update – the callable function for every iteration, expect to accept engine and engine.state.batch as inputs, return data will be stored in engine.state.output. if not provided, use self._iteration() instead. for more details please refer to: https://pytorch.org/ignite/generated/ignite.engine.engine.Engine.html. postprocessing – execute additional transformation for the model output data. Typically, several Tensor based transforms composed by Compose. key_val_metric – compute metric when every iteration completed, and save average value to engine.state.metrics when epoch completed. key_val_metric is the main metric to compare and save the checkpoint into files. additional_metrics – more Ignite metrics that also attach to Ignite Engine. metric_cmp_fn – function to compare current key metric with previous best key metric value, it must accept 2 args (current_metric, previous_best) and return a bool result: if True, will update best_metric and best_metric_epoch with current metric and epoch, default to greater than. val_handlers – every handler is a set of Ignite Event-Handlers, must have attach function, like: CheckpointHandler, StatsHandler, etc. amp – whether to enable auto-mixed-precision evaluation, default is False. mode – model forward mode during evaluation, should be 'eval' or 'train', which maps to model.eval() or model.train(), default to 'eval'. event_names – additional custom ignite events that will register to the engine. new events can be a list of str or ignite.engine.events.EventEnum. event_to_attr – a dictionary to map an event to a state attribute, then add to engine.state. for more details, check: https://pytorch.org/ignite/generated/ignite.engine.engine.Engine.html #ignite.engine.engine.Engine.register_events. decollate – whether to decollate the batch-first data to a list of data after model computation, recommend decollate=True when postprocessing uses components from monai.transforms. default to True. to_kwargs – dict of other args for prepare_batch API when converting the input data, except for device, non_blocking. amp_kwargs – dict of the args for torch.cuda.amp.autocast() API, for more details: https://pytorch.org/docs/stable/amp.html#torch.cuda.amp.autocast. Get the statistics information of the validation process. Default to return the rank, best_validation_epoch and best_validation_metric. vars – except for the default stats, other variables name in the self.state to return, will use the variable name as the key and the state content as the value. if the variable doesn't exist, default value is None. Execute validation/evaluation based on Ignite Engine. global_epoch (int) – the overall epoch if during a training. evaluator engine can get it from trainer. None

SupervisedEvaluator#

Standard supervised evaluation method with image and label(optional), inherits from evaluator and Workflow. device – an object representing the device on which to run. val_data_loader – Ignite engine use data_loader to run, must be Iterable, typically be torch.DataLoader. network – network to evaluate in the evaluator, should be regular PyTorch torch.nn.Module. epoch_length – number of iterations for one epoch, default to len(val_data_loader). non_blocking – if True and this copy is between CPU and GPU, the copy may occur asynchronously with respect to the host. For other cases, this argument has no effect. prepare_batch – function to parse expected data (usually image, label and other network args) from engine.state.batch for every iteration, for more details please refer to: https://pytorch.org/ignite/generated/ignite.engine.create_supervised_trainer.html. iteration_update – the callable function for every iteration, expect to accept engine and engine.state.batch as inputs, return data will be stored in engine.state.output. if not provided, use self._iteration() instead. for more details please refer to:

https://pytorch.org/ignite/generated/ignite.engine.engine.Engine.html. inferer – inference method that execute model forward on input data, like: SlidingWindow, etc. postprocessing – execute additional transformation for the model output data. Typically, several Tensor based transforms composed by Compose. key_val_metric – compute metric when every iteration completed, and save average value to engine.state.metrics when epoch completed. key_val_metric is the main metric to compare and save the checkpoint into files. additional_metrics – more Ignite metrics that also attach to Ignite Engine. metric_cmp_fn – function to compare current key metric with previous best key metric value, it must accept 2 args (current_metric, previous_best) and return a bool result: if True, will update best_metric and best_metric_epoch with current metric and epoch, default to greater than. val_handlers – every handler is a set of Ignite Event-Handlers, must have attach function, like: CheckpointHandler, StatsHandler, etc. amp – whether to enable auto-mixed-precision evaluation, default is False. mode – model forward mode during evaluation, should be 'eval' or 'train', which maps to model.eval() or model.train(), default to 'eval'. event_names – additional custom ignite events that will register to the engine. new events can be a list of str or ignite.engine.events.EventEnum. event_to_attr – a dictionary to map an event to a state attribute, then add to engine.state. for more details, check: https://pytorch.org/ignite/generated/ignite.engine.engine.Engine.html #ignite.engine.engine.Engine.register_events. decollate – whether to decollate the batch-first data to a list of data after model computation, recommend decollate=True when postprocessing uses components from monai.transforms. default to True. to_kwargs – dict of other args for prepare_batch API when converting the input data, except for device, non_blocking. amp_kwargs – dict of the args for torch.cuda.amp.autocast() API, for more details: https://pytorch.org/docs/stable/amp.html#torch.cuda.amp.autocast.

EnsembleEvaluator#

Ensemble evaluation for multiple models, inherits from evaluator and Workflow. It accepts a list of models for inference and outputs a list of predictions for further operations. device – an object representing the device on which to run. val_data_loader – Ignite engine use data_loader to run, must be Iterable, typically be torch.DataLoader. epoch_length – number of iterations for one epoch, default to len(val_data_loader). networks – networks to evaluate in order in the evaluator, should be regular PyTorch torch.nn.Module. pred_keys – the keys to store every prediction data. the length must exactly match the number of networks. if None, use "pred_{index}" as key corresponding to N networks, index from 0 to N-1. non_blocking – if True and this copy is between CPU and GPU, the copy may occur asynchronously with respect to the host. For other cases, this argument has no effect. prepare_batch – function to parse expected data (usually image, label and other network args) from engine.state.batch for every iteration, for more details please refer to: https://pytorch.org/ignite/generated/ignite.engine.create_supervised_trainer.html. iteration_update – the callable function for every iteration, expect to accept engine and engine.state.batch as inputs, return data will be stored in engine.state.output. if not provided, use self._iteration() instead. for more details please refer to: https://pytorch.org/ignite/generated/ignite.engine.engine.Engine.html. inferer – inference method that execute model forward on input data, like: SlidingWindow, etc. postprocessing – execute additional transformation for the model output data. Typically, several Tensor based transforms composed by Compose. key_val_metric – compute metric when every iteration completed, and save average value to

engine.state.metrics when epoch completed. key_val_metric is the main metric to compare and save the checkpoint into files. additional_metrics – more Ignite metrics that also attach to Ignite Engine. metric_cmp_fn – function to compare current key metric with previous best key metric value, it must accept 2 args (current_metric, previous_best) and return a bool result: if True, will update best_metric and best_metric_epoch with current metric and epoch, default to greater than. val_handlers – every handler is a set of Ignite Event-Handlers, must have attach function, like: CheckpointHandler, StatsHandler, etc. amp – whether to enable auto-mixed-precision evaluation, default is False. mode – model forward mode during evaluation, should be 'eval' or 'train', which maps to model.eval() or model.train(), default to 'eval'. event_names – additional custom ignite events that will register to the engine. new events can be a list of str or ignite.engine.events.EventEnum. event_to_attr – a dictionary to map an event to a state attribute, then add to engine.state. for more details, check: https://pytorch.org/ignite/generated/ignite.engine.engine.Engine.html #ignite.engine.engine.Engine.register_events. decollate – whether to decollate the batch-first data to a list of data after model computation, recommend decollate=True when postprocessing uses components from monai.transforms. default to True. to_kwargs – dict of other args for prepare_batch API when converting the input data, except for device, non_blocking. amp_kwargs – dict of the args for torch.cuda.amp.autocast() API, for more details: https://pytorch.org/docs/stable/amp.html#torch.cuda.amp.autocast.

Utilities#

Additional Events engine can register and trigger in the iteration process. Refer to the example in ignite: https://pytorch.org/ignite/generated/ignite.engine.events.EventEnum.html. These Events can be triggered during training iteration: FORWARD_COMPLETED is the Event when network(image, label) completed. LOSS_COMPLETED is the Event when loss(pred, label) completed. BACKWARD_COMPLETED is the Event when loss.backward() completed. MODEL_COMPLETED is the Event when all the model related operations completed. INNER_ITERATION_STARTED is the Event when the iteration has an inner loop and the loop is started. INNER_ITERATION_COMPLETED is the Event when the iteration has an inner loop and the loop is completed. Interface of customized prepare_batch in the trainer or evaluator workflows. It takes the data of current batch, target device and non_blocking flag as input. Args batchdata, device, non_blocking refer to the ignite API: https://pytorch.org/ignite/v0.4.8/generated/ignite. engine.create_supervised_trainer.html. kwargs supports other args for Tensor.to() API. This wraps default_prepare_batch to return image and label only, so is consistent with its API. Customized prepare batch callable for trainers or evaluators which support extra input data for the network. Extra items are specified by the extra_keys parameter and are extracted from the input dictionary (ie. the batch). This uses default_prepare_batch but requires dictionary inputs. extra_keys (Union[str, Sequence[str], Dict[str, str]]) – If a string or sequence of strings is provided, values from the input dictionary are extracted from those keys and passed to the network as extra positional arguments. If a dictionary is provided, every pair (k, v) in that dictionary will become a new keyword argument assigning to k the value in the input dictionary keyed to v. The default function to compare metric values between current metric and previous best metric. current_metric (float) – metric value of current round computation. prev_best (float) – the best metric value of previous rounds to compare with. bool Default function to prepare the data for current iteration. The input

batchdata is either a single tensor, a pair of tensors, or a dictionary of data. In the first case the return value is the tensor and None, in the second case the return value is the two tensors, and in the dictionary case the return value depends on what keys are present. if CommonKeys.IMAGE and CommonKeys.LABEL are present then the tensors they key to are returned, if only CommonKeys.IMAGE is present that tensor and None is returned. If CommonKeys.REALS is present this is returned with None. All returned tensors are moved to the given device using the given non-blocking argument before being returned. This function implements the expected API for a prepare_batch callable in Ignite: https://pytorch.org/ignite/v0.4.8/generated/ignite.engine.create_supervised_trainer.html batchdata (Union[Dict[str, Tensor], Tensor, Sequence[Tensor]]) – input batch data which is either a single tensor, a pair, or a dictionary device (Union[str, device, None]) – device to move every returned tensor to non_blocking (bool) – equivalent argument for Tensor.to kwargs – further arguments for Tensor.to Union[Tuple[Tensor, Optional[Tensor]], Tensor] image, label(optional). Apply transform on batch and output. If batch and output are dictionaries, temporarily combine them for the transform, otherwise, apply the transform for output data only. Get a valid specification for one or more devices. If devices is None get devices for all CUDA devices available. If devices is and zero-length structure a single CPU compute device is returned. In any other cases devices is returned unchanged. devices (Optional[Sequence[Union[device, str]]]) – list of devices to request, None for all GPU devices, [] for CPU. RuntimeError – When all GPUs are selected (devices=None) but no GPUs are available. list of devices. list of torch.device previous Data next Inference methods