BaseCls

发行版本 *0.4*

basedet team, Megvii

2022 年 12 月 15 日

Contents

对应版本

0.4

更新时间

2022 年 12 月 15 日

BaseCls 是一个基于 MegEngine 的预训练模型库，帮助大家挑选或训练出更适合自己科研或者业务的模型结构。

CHAPTER 1

目录

1.1 Model Zoo

此目录为 BaseCls 的模型库根目录:

1.1.1 public（公开模型）

EfficientNet Series

effnet_b0

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import EffNetConfig

_cfg = dict(
    batch_size=64,
    model=dict(
        name="effnet_b0",
```

(续下页)

(接上页)

```
    ),
    solver=dict(
        basic_lr=0.1,
    ),
    model_ema=dict(
        enabled=True,
        momentum=0.99996,
    ),
)


class Cfg(EffNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

effnet_b0_lite

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import EffNetLiteConfig

_cfg = dict(
    batch_size=64,
    model=dict(
        name="effnet_b0_lite",
    ),
    solver=dict(
        basic_lr=0.1,
    ),
    model_ema=dict(
        enabled=True,
        momentum=0.99996,
    ),
)
```

(续下页)

(接上页)

```
class Cfg(EffNetLiteConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

effnet_b1

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import EffNetConfig

_cfg = dict(
    preprocess=dict(
        img_size=240,
    ),
    test=dict(
        img_size=240,
        crop_pct=240 / 272,
    ),
    model=dict(
        name="effnet_b1",
    ),
)


class Cfg(EffNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

effnet_b1_lite

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import EffNetLiteConfig

_cfg = dict(
    preprocess=dict(
        img_size=240,
    ),
    test=dict(
        img_size=240,
        crop_pct=240 / 272,
    ),
    model=dict(
        name="effnet_b1_lite",
    ),
)


class Cfg(EffNetLiteConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

effnet_b2

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import EffNetConfig

_cfg = dict(
    preprocess=dict(
        img_size=260,
```

(续下页)

(接上页)

```
    ),
    test=dict(
        img_size=260,
        crop_pct=260 / 292,
    ),
    model=dict(
        name="effnet_b2",
    ),
)


class Cfg(EffNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

effnet_b2_lite

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import EffNetLiteConfig

_cfg = dict(
    preprocess=dict(
        img_size=260,
    ),
    test=dict(
        img_size=260,
        crop_pct=260 / 292,
    ),
    model=dict(
        name="effnet_b2_lite",
    ),
)


class Cfg(EffNetLiteConfig):
```

(续下页)

(接上页)

```
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

effnet_b3

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import EffNetConfig

_cfg = dict(
    preprocess=dict(
        img_size=300,
    ),
    test=dict(
        img_size=300,
        crop_pct=300 / 332,
    ),
    model=dict(
        name="effnet_b3",
    ),
    augments=dict(
        rand_aug=dict(
            magnitude=11,
        ),
    ),
)


class Cfg(EffNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

effnet_b3_lite

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import EffNetLiteConfig

_cfg = dict(
    preprocess=dict(
        img_size=280,
    ),
    test=dict(
        img_size=280,
        crop_pct=280 / 312,
    ),
    model=dict(
        name="effnet_b3_lite",
    ),
    augments=dict(
        rand_aug=dict(
            magnitude=11,
        ),
    ),
)


class Cfg(EffNetLiteConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

effnet_b4

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
```

(续下页)

(接上页)

```
from basecls.configs import EffNetConfig

_cfg = dict(
    preprocess=dict(
        img_size=380,
    ),
    test=dict(
        img_size=380,
        crop_pct=380 / 412,
    ),
    model=dict(
        name="effnet_b4",
    ),
    augments=dict(
        rand_aug=dict(
            magnitude=13,
        ),
    ),
)


class Cfg(EffNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

effnet_b4_lite

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import EffNetLiteConfig

_cfg = dict(
    preprocess=dict(
        img_size=300,
    ),
    test=dict(
```

(续下页)

(接上页)

```
        img_size=300,
        crop_pct=300 / 332,
    ),
    model=dict(
        name="effnet_b4_lite",
    ),
    augments=dict(
        rand_aug=dict(
            magnitude=13,
        ),
    ),
)


class Cfg(EffNetLiteConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

effnet_b5

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import EffNetConfig

_cfg = dict(
    preprocess=dict(
        img_size=456,
    ),
    test=dict(
        img_size=456,
        crop_pct=456 / 488,
    ),
    model=dict(
        name="effnet_b5",
    ),
    augments=dict(
```

(续下页)

(接上页)

```
        rand_aug=dict(
            magnitude=15,
        ),
    ),
)


class Cfg(EffNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

effnet_b6

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import EffNetConfig

_cfg = dict(
    preprocess=dict(
        img_size=528,
    ),
    test=dict(
        img_size=528,
        crop_pct=528 / 560,
    ),
    model=dict(
        name="effnet_b6",
    ),
    augments=dict(
        rand_aug=dict(
            magnitude=17,
        ),
    ),
)
```

(续下页)

(接上页)

```
class Cfg(EffNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

effnet_b7

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import EffNetConfig

_cfg = dict(
    preprocess=dict(
        img_size=600,
    ),
    test=dict(
        img_size=600,
        crop_pct=600 / 632,
    ),
    model=dict(
        name="effnet_b7",
    ),
)


class Cfg(EffNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

effnet_b8

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import EffNetConfig

_cfg = dict(
    preprocess=dict(
        img_size=672,
    ),
    test=dict(
        img_size=672,
        crop_pct=672 / 704,
    ),
    model=dict(
        name="effnet_b8",
    ),
)


class Cfg(EffNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

effnet_l2

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import EffNetConfig

_cfg = dict(
    preprocess=dict(
        img_size=800,
```

(续下页)

(接上页)

```
    ),
    test=dict(
        img_size=800,
        crop_pct=800 / 832,
    ),
    model=dict(
        name="effnet_l2",
    ),
)


class Cfg(EffNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

effnetv2_b0

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import EffNetConfig

_cfg = dict(
    batch_size=64,
    preprocess=dict(
        img_size=192,
    ),
    model=dict(
        name="effnetv2_b0",
    ),
    solver=dict(
        basic_lr=0.1,
    ),
    model_ema=dict(
        enabled=True,
        momentum=0.99996,
    ),
```

(续下页)

(接上页)

```
)


class Cfg(EffNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

effnetv2_b1

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import EffNetConfig

_cfg = dict(
    batch_size=64,
    preprocess=dict(
        img_size=192,
    ),
    test=dict(
        img_size=240,
        crop_pct=240 / 272,
    ),
    model=dict(
        name="effnetv2_b1",
    ),
    solver=dict(
        basic_lr=0.1,
    ),
    model_ema=dict(
        enabled=True,
        momentum=0.99996,
    ),
)


class Cfg(EffNetConfig):
```

(续下页)

(接上页)

```
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

effnetv2_b2

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import EffNetConfig

_cfg = dict(
    batch_size=64,
    preprocess=dict(
        img_size=208,
    ),
    test=dict(
        img_size=260,
        crop_pct=260 / 292,
    ),
    model=dict(
        name="effnetv2_b2",
    ),
    solver=dict(
        basic_lr=0.1,
    ),
    model_ema=dict(
        enabled=True,
        momentum=0.99996,
    ),
)


class Cfg(EffNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

effnetv2_b3

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import EffNetConfig

_cfg = dict(
    preprocess=dict(
        img_size=240,
    ),
    test=dict(
        img_size=300,
        crop_pct=300 / 332,
    ),
    model=dict(
        name="effnetv2_b3",
    ),
)


class Cfg(EffNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

effnetv2_l

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import EffNetConfig

_cfg = dict(
    preprocess=dict(
        img_size=384,
```

(续下页)

(接上页)

```
    ),
    test=dict(
        img_size=480,
        crop_pct=1.0,
    ),
    model=dict(
        name="effnetv2_l",
    ),
    augments=dict(
        rand_aug=dict(
            magnitude=21,
        ),
        mixup=dict(
            mixup_alpha=0.5,
            cutmix_alpha=0.5,
        ),
    ),
)


class Cfg(EffNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

effnetv2_m

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import EffNetConfig

_cfg = dict(
    preprocess=dict(
        img_size=384,
    ),
    test=dict(
        img_size=480,
```

(续下页)

(接上页)

```
        crop_pct=1.0,
    ),
    model=dict(
        name="effnetv2_m",
    ),
    augments=dict(
        rand_aug=dict(
            magnitude=15,
        ),
        mixup=dict(
            mixup_alpha=0.2,
            cutmix_alpha=0.2,
        ),
    ),
)


class Cfg(EffNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

effnetv2_s

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import EffNetConfig

_cfg = dict(
    preprocess=dict(
        img_size=300,
    ),
    test=dict(
        img_size=384,
        crop_pct=1.0,
    ),
    model=dict(
```

(续下页)

(接上页)

```
        name="effnetv2_s",
    ),
)


class Cfg(EffNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

HRNet Series

hrnet_w18

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import HRNetConfig

_cfg = dict(
    model=dict(
        name="hrnet_w18",
    ),
)


class Cfg(HRNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

hrnet_w18_small_v1

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import HRNetConfig

_cfg = dict(
    batch_size=64,
    model=dict(
        name="hrnet_w18_small_v1",
    ),
    solver=dict(
        basic_lr=0.05,
    ),
)


class Cfg(HRNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

hrnet_w18_small_v2

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import HRNetConfig

_cfg = dict(
    batch_size=64,
    model=dict(
        name="hrnet_w18_small_v2",
    ),
    solver=dict(
```

(续下页)

(接上页)

```
        basic_lr=0.05,
    ),
)


class Cfg(HRNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

hrnet_w30

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import HRNetConfig

_cfg = dict(
    model=dict(
        name="hrnet_w30",
    ),
)


class Cfg(HRNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

hrnet_w32

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
```

(续下页)

(接上页)

```
from basecls.configs import HRNetConfig

_cfg = dict(
    model=dict(
        name="hrnet_w32",
    ),
)


class Cfg(HRNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

hrnet_w40

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import HRNetConfig

_cfg = dict(
    model=dict(
        name="hrnet_w40",
    ),
)


class Cfg(HRNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

hrnet_w44

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import HRNetConfig

_cfg = dict(
    model=dict(
        name="hrnet_w44",
    ),
)


class Cfg(HRNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

hrnet_w48

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import HRNetConfig

_cfg = dict(
    model=dict(
        name="hrnet_w48",
    ),
)


class Cfg(HRNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
```

(续下页)

(接上页)

```
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

hrnet_w64

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import HRNetConfig

_cfg = dict(
    model=dict(
        name="hrnet_w64",
    ),
)


class Cfg(HRNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

MobileNet Series

mbnetv1_x025

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import MBConfig

_cfg = dict(
    model=dict(
        name="mbnetv1_x025",
```

(续下页)

(接上页)

```
    ),
)


class Cfg(MBConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

mbnetv1_x050

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import MBConfig

_cfg = dict(
    model=dict(
        name="mbnetv1_x050",
    ),
)


class Cfg(MBConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

mbnetv1_x075

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
```

(续下页)

(接上页)

```
from basecls.configs import MBConfig

_cfg = dict(
    model=dict(
        name="mbnetv1_x075",
    ),
)


class Cfg(MBConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

mbnetv1_x100

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import MBConfig

_cfg = dict(
    batch_size=64,
    model=dict(
        name="mbnetv1_x100",
    ),
    solver=dict(
        basic_lr=0.025,
    ),
)


class Cfg(MBConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

mbnetv2_x035

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import MBConfig

_cfg = dict(
    model=dict(
        name="mbnetv2_x035",
    ),
)


class Cfg(MBConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

mbnetv2_x050

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import MBConfig

_cfg = dict(
    model=dict(
        name="mbnetv2_x050",
    ),
)


class Cfg(MBConfig):
    def __init__(self, values_or_file=None, **kwargs):
```

(续下页)

(接上页)

```
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

mbnetv2_x075

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import MBConfig

_cfg = dict(
    batch_size=64,
    model=dict(
        name="mbnetv2_x075",
    ),
    solver=dict(
        basic_lr=0.025,
    ),
)


class Cfg(MBConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

mbnetv2_x100

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import MBConfig
```

(续下页)

(接上页)

```
_cfg = dict(
    batch_size=64,
    model=dict(
        name="mbnetv2_x100",
    ),
    solver=dict(
        basic_lr=0.025,
    ),
)


class Cfg(MBConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

mbnetv2_x140

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import MBConfig

_cfg = dict(
    batch_size=64,
    model=dict(
        name="mbnetv2_x140",
    ),
    solver=dict(
        basic_lr=0.025,
    ),
)


class Cfg(MBConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

mbnetv3_large_x075

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import MBConfig

_cfg = dict(
    model=dict(
        name="mbnetv3_large_x075",
    ),
)


class Cfg(MBConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

mbnetv3_large_x100

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import MBConfig

_cfg = dict(
    model=dict(
        name="mbnetv3_large_x100",
    ),
)


class Cfg(MBConfig):
    def __init__(self, values_or_file=None, **kwargs):
```

(续下页)

(接上页)

```
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

mbnetv3_small_x075

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import MBConfig

_cfg = dict(
    model=dict(
        name="mbnetv3_small_x075",
        head=dict(
            dropout_prob=0.1,
        ),
    ),
    loss=dict(
        label_smooth=0.0,
    ),
)


class Cfg(MBConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

mbnetv3_small_x100

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
```

(续下页)

(接上页)

```
from basecls.configs import MBConfig

_cfg = dict(
    model=dict(
        name="mbnetv3_small_x100",
        head=dict(
            dropout_prob=0.1,
        ),
    ),
    loss=dict(
        label_smooth=0.0,
    ),
)


class Cfg(MBConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

RegNet Series

regnetx_002

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import RegNetConfig

_cfg = dict(
    batch_size=128,
    model=dict(
        name="regnetx_002",
    ),
    augments=dict(
        rand_aug=dict(
            magnitude=3,
        ),
```

(续下页)

(接上页)

```
    ),
    solver=dict(
        basic_lr=0.1,
    ),
    model_ema=dict(
        enabled=True,
        momentum=0.99992,
    ),
)


class Cfg(RegNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

regnetx_004

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import RegNetConfig

_cfg = dict(
    batch_size=128,
    model=dict(
        name="regnetx_004",
    ),
    augments=dict(
        rand_aug=dict(
            magnitude=5,
        ),
    ),
    solver=dict(
        basic_lr=0.1,
    ),
    model_ema=dict(
        enabled=True,
```

(续下页)

(接上页)

```
        momentum=0.99992,
    ),
)


class Cfg(RegNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

regnetx_006

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import RegNetConfig

_cfg = dict(
    batch_size=128,
    model=dict(
        name="regnetx_006",
    ),
    solver=dict(
        basic_lr=0.1,
    ),
    model_ema=dict(
        enabled=True,
        momentum=0.99992,
    ),
)


class Cfg(RegNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

regnetx_008

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import RegNetConfig

_cfg = dict(
    batch_size=128,
    model=dict(
        name="regnetx_008",
    ),
    solver=dict(
        basic_lr=0.1,
    ),
    model_ema=dict(
        enabled=True,
        momentum=0.99992,
    ),
)


class Cfg(RegNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

regnetx_016

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import RegNetConfig

_cfg = dict(
    batch_size=128,
```

(续下页)

(接上页)

```
    model=dict(
        name="regnetx_016",
    ),
    solver=dict(
        basic_lr=0.1,
    ),
    model_ema=dict(
        enabled=True,
        momentum=0.99992,
    ),
)


class Cfg(RegNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

regnetx_032

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import RegNetConfig

_cfg = dict(
    batch_size=64,
    model=dict(
        name="regnetx_032",
    ),
    solver=dict(
        basic_lr=0.05,
    ),
    model_ema=dict(
        enabled=True,
        momentum=0.99996,
    ),
)
```

(续下页)

(接上页)

```
class Cfg(RegNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

regnetx_040

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import RegNetConfig

_cfg = dict(
    batch_size=64,
    model=dict(
        name="regnetx_040",
    ),
    solver=dict(
        basic_lr=0.05,
    ),
    model_ema=dict(
        enabled=True,
        momentum=0.99996,
    ),
)


class Cfg(RegNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

regnetx_064

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import RegNetConfig

_cfg = dict(
    batch_size=64,
    model=dict(
        name="regnetx_064",
    ),
    solver=dict(
        basic_lr=0.05,
    ),
    model_ema=dict(
        enabled=True,
        momentum=0.99996,
    ),
)


class Cfg(RegNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

regnetx_080

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import RegNetConfig

_cfg = dict(
    batch_size=64,
```

(续下页)

(接上页)

```
    model=dict(
        name="regnetx_080",
    ),
    solver=dict(
        basic_lr=0.05,
    ),
    model_ema=dict(
        enabled=True,
        momentum=0.99996,
    ),
)


class Cfg(RegNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

regnetx_120

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import RegNetConfig

_cfg = dict(
    model=dict(
        name="regnetx_120",
    ),
)


class Cfg(RegNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

regnetx_160

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import RegNetConfig

_cfg = dict(
    model=dict(
        name="regnetx_160",
    ),
)


class Cfg(RegNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

regnetx_320

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import RegNetConfig

_cfg = dict(
    model=dict(
        name="regnetx_320",
    ),
)


class Cfg(RegNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
```

(续下页)

(接上页)

```
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

regnety_002

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import RegNetConfig

_cfg = dict(
    batch_size=128,
    model=dict(
        name="regnety_002",
    ),
    augments=dict(
        rand_aug=dict(
            magnitude=3,
        ),
    ),
    solver=dict(
        basic_lr=0.1,
    ),
    model_ema=dict(
        enabled=True,
        momentum=0.99992,
    ),
)


class Cfg(RegNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

regnety_004

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import RegNetConfig

_cfg = dict(
    batch_size=128,
    model=dict(
        name="regnety_004",
    ),
    augments=dict(
        rand_aug=dict(
            magnitude=5,
        ),
    ),
    solver=dict(
        basic_lr=0.1,
    ),
    model_ema=dict(
        enabled=True,
        momentum=0.99992,
    ),
)


class Cfg(RegNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

regnety_006

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import RegNetConfig

_cfg = dict(
    batch_size=128,
    model=dict(
        name="regnety_006",
    ),
    solver=dict(
        basic_lr=0.1,
    ),
    model_ema=dict(
        enabled=True,
        momentum=0.99992,
    ),
)


class Cfg(RegNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

regnety_008

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import RegNetConfig

_cfg = dict(
    batch_size=128,
```

(续下页)

(接上页)

```
    model=dict(
        name="regnety_008",
    ),
    solver=dict(
        basic_lr=0.1,
    ),
    model_ema=dict(
        enabled=True,
        momentum=0.99992,
    ),
)


class Cfg(RegNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

regnety_016

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import RegNetConfig

_cfg = dict(
    batch_size=128,
    model=dict(
        name="regnety_016",
    ),
    solver=dict(
        basic_lr=0.1,
    ),
    model_ema=dict(
        enabled=True,
        momentum=0.99992,
    ),
)
```

(续下页)

(接上页)

```
class Cfg(RegNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

regnety_032

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import RegNetConfig

_cfg = dict(
    batch_size=64,
    model=dict(
        name="regnety_032",
    ),
    solver=dict(
        basic_lr=0.05,
    ),
    model_ema=dict(
        enabled=True,
        momentum=0.99996,
    ),
)


class Cfg(RegNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

regnety_040

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import RegNetConfig

_cfg = dict(
    batch_size=64,
    model=dict(
        name="regnety_040",
    ),
    solver=dict(
        basic_lr=0.05,
    ),
    model_ema=dict(
        enabled=True,
        momentum=0.99996,
    ),
)


class Cfg(RegNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

regnety_064

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import RegNetConfig

_cfg = dict(
    batch_size=64,
```

(续下页)

(接上页)

```
    model=dict(
        name="regnety_064",
    ),
    solver=dict(
        basic_lr=0.05,
    ),
    model_ema=dict(
        enabled=True,
        momentum=0.99996,
    ),
)


class Cfg(RegNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

regnety_080

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import RegNetConfig

_cfg = dict(
    model=dict(
        name="regnety_080",
    ),
)


class Cfg(RegNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

regnety_120

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import RegNetConfig

_cfg = dict(
    model=dict(
        name="regnety_120",
    ),
)


class Cfg(RegNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

regnety_160

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import RegNetConfig

_cfg = dict(
    model=dict(
        name="regnety_160",
    ),
)


class Cfg(RegNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
```

(续下页)

(接上页)

```
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

regnety_320

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import RegNetConfig

_cfg = dict(
    model=dict(
        name="regnety_320",
    ),
)


class Cfg(RegNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

RepVGG Series

Use `RepVGG.convert_to_deploy(model)` to convert a training RepVGG to deploy:

```
model = RepVGG(..., deploy=False)
model.load_state_dict(...)
_ = RepVGG.convert_to_deploy(model)
```

repvgg_a0

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import RepVGGConfig

_cfg = dict(
    model=dict(
        name="repvgg_a0",
    ),
)


class Cfg(RepVGGConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

repvgg_a1

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import RepVGGConfig

_cfg = dict(
    model=dict(
        name="repvgg_a1",
    ),
)


class Cfg(RepVGGConfig):
    def __init__(self, values_or_file=None, **kwargs):
```

(续下页)

(接上页)

```
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

repvgg_a2

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import RepVGGConfig

_cfg = dict(
    model=dict(
        name="repvgg_a2",
    ),
)


class Cfg(RepVGGConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

repvgg_b0

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import RepVGGConfig

_cfg = dict(
    model=dict(
        name="repvgg_b0",
    ),
```

(续下页)

(接上页)

```
)


class Cfg(RepVGGConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

repvgg_b1

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import RepVGGConfig

_cfg = dict(
    model=dict(
        name="repvgg_b1",
    ),
)


class Cfg(RepVGGConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

repvgg_b1g2

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import RepVGGConfig
```

(续下页)

(接上页)

```
_cfg = dict(
    model=dict(
        name="repvgg_b1g2",
    ),
)


class Cfg(RepVGGConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

repvgg_b1g4

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import RepVGGConfig

_cfg = dict(
    model=dict(
        name="repvgg_b1g4",
    ),
)


class Cfg(RepVGGConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

repvgg_b2

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import RepVGGConfig

_cfg = dict(
    model=dict(
        name="repvgg_b2",
    ),
)


class Cfg(RepVGGConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

repvgg_b2g4

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import RepVGGConfig

_cfg = dict(
    model=dict(
        name="repvgg_b2g4",
    ),
    loss=dict(
        label_smooth=0.1,
    ),
    augments=dict(
        name="AutoAugment",
        mixup=dict(
```

(续下页)

(接上页)

```
            mixup_alpha=0.2,
            permute=1,
        ),
    ),
    solver=dict(
        max_epoch=200,
    ),
)


class Cfg(RepVGGConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

repvgg_b3

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import RepVGGConfig

_cfg = dict(
    model=dict(
        name="repvgg_b3",
    ),
    loss=dict(
        label_smooth=0.1,
    ),
    augments=dict(
        name="AutoAugment",
        mixup=dict(
            mixup_alpha=0.2,
            permute=1,
        ),
    ),
    solver=dict(
        max_epoch=200,
```

(续下页)

(接上页)

```
    ),
)


class Cfg(RepVGGConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

repvgg_b3g4

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import RepVGGConfig

_cfg = dict(
    model=dict(
        name="repvgg_b3g4",
    ),
    loss=dict(
        label_smooth=0.1,
    ),
    augments=dict(
        name="AutoAugment",
        mixup=dict(
            mixup_alpha=0.2,
            permute=1,
        ),
    ),
    solver=dict(
        max_epoch=200,
    ),
)


class Cfg(RepVGGConfig):
    def __init__(self, values_or_file=None, **kwargs):
```

(续下页)

(接上页)

```
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

repvgg_d2

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import RepVGGConfig

_cfg = dict(
    model=dict(
        name="repvgg_d2",
    ),
    test=dict(
        img_size=320,
    ),
    loss=dict(
        label_smooth=0.1,
    ),
    augments=dict(
        name="AutoAugment",
        mixup=dict(
            mixup_alpha=0.2,
            permute=1,
        ),
    ),
    solver=dict(
        max_epoch=200,
    ),
)


class Cfg(RepVGGConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

ResMLP Series

resmlp_b24

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import ResMLPConfig

_cfg = dict(
    model=dict(
        name="resmlp_b24",
    ),
)


class Cfg(ResMLPConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

resmlp_s12

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import ResMLPConfig

_cfg = dict(
    model=dict(
        name="resmlp_s12",
    ),
)


class Cfg(ResMLPConfig):
```

(续下页)

(接上页)

```
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

resmlp_s24

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import ResMLPConfig

_cfg = dict(
    model=dict(
        name="resmlp_s24",
    ),
)


class Cfg(ResMLPConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

resmlp_s36

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import ResMLPConfig

_cfg = dict(
    model=dict(
        name="resmlp_s36",
```

(续下页)

(接上页)

```
    ),
)


class Cfg(ResMLPConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

ResNet Series

resnet101

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import ResNetConfig

_cfg = dict(
    batch_size=64,
    model=dict(
        name="resnet101",
    ),
    solver=dict(
        basic_lr=0.05,
    ),
    model_ema=dict(
        enabled=True,
        momentum=0.99996,
    ),
)


class Cfg(ResNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

resnet101d

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import ResNetConfig

_cfg = dict(
    batch_size=64,
    model=dict(
        name="resnet101d",
    ),
    solver=dict(
        basic_lr=0.05,
    ),
    model_ema=dict(
        enabled=True,
        momentum=0.99996,
    ),
)


class Cfg(ResNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

resnet152

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import ResNetConfig

_cfg = dict(
    model=dict(
```

(续下页)

(接上页)

```
        name="resnet152",
    ),
)


class Cfg(ResNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

resnet152d

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import ResNetConfig

_cfg = dict(
    model=dict(
        name="resnet152d",
    ),
)


class Cfg(ResNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

resnet18

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
```

(续下页)

(接上页)

```
from basecls.configs import ResNetConfig

_cfg = dict(
    batch_size=128,
    model=dict(
        name="resnet18",
    ),
    solver=dict(
        basic_lr=0.1,
    ),
    model_ema=dict(
        enabled=True,
        momentum=0.99992,
    ),
)


class Cfg(ResNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

resnet18d

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import ResNetConfig

_cfg = dict(
    batch_size=128,
    model=dict(
        name="resnet18d",
    ),
    solver=dict(
        basic_lr=0.1,
    ),
    model_ema=dict(
```

(续下页)

(接上页)

```
        enabled=True,
        momentum=0.99992,
    ),
)


class Cfg(ResNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

resnet34

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import ResNetConfig

_cfg = dict(
    batch_size=64,
    model=dict(
        name="resnet34",
    ),
    solver=dict(
        basic_lr=0.05,
    ),
    model_ema=dict(
        enabled=True,
        momentum=0.99996,
    ),
)


class Cfg(ResNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

resnet34d

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import ResNetConfig

_cfg = dict(
    batch_size=64,
    model=dict(
        name="resnet34d",
    ),
    solver=dict(
        basic_lr=0.05,
    ),
    model_ema=dict(
        enabled=True,
        momentum=0.99996,
    ),
)


class Cfg(ResNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

resnet50

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import ResNetConfig

_cfg = dict(
    batch_size=64,
```

(续下页)

(接上页)

```
    model=dict(
        name="resnet50",
    ),
    solver=dict(
        basic_lr=0.05,
    ),
    model_ema=dict(
        enabled=True,
        momentum=0.99996,
    ),
)


class Cfg(ResNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

resnet50d

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import ResNetConfig

_cfg = dict(
    batch_size=64,
    model=dict(
        name="resnet50d",
    ),
    solver=dict(
        basic_lr=0.05,
    ),
    model_ema=dict(
        enabled=True,
        momentum=0.99996,
    ),
)
```

(续下页)

(接上页)

```
class Cfg(ResNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

resnext101_32x4d

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import ResNetConfig

_cfg = dict(
    model=dict(
        name="resnext101_32x4d",
    ),
)


class Cfg(ResNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

resnext101_32x8d

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import ResNetConfig
```

(续下页)

(接上页)

```
_cfg = dict(
    model=dict(
        name="resnext101_32x8d",
    ),
)


class Cfg(ResNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

resnext101_64x4d

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import ResNetConfig

_cfg = dict(
    model=dict(
        name="resnext101_64x4d",
    ),
)


class Cfg(ResNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

resnext152_32x4d

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import ResNetConfig

_cfg = dict(
    model=dict(
        name="resnext152_32x4d",
    ),
)


class Cfg(ResNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

resnext152_32x8d

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import ResNetConfig

_cfg = dict(
    model=dict(
        name="resnext152_32x8d",
    ),
)


class Cfg(ResNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
```

(续下页)

(接上页)

```
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

resnext152_64x4d

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import ResNetConfig

_cfg = dict(
    model=dict(
        name="resnext152_64x4d",
    ),
)


class Cfg(ResNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

resnext50_32x4d

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import ResNetConfig

_cfg = dict(
    batch_size=64,
    model=dict(
        name="resnext50_32x4d",
```

(续下页)

(接上页)

```
    ),
    solver=dict(
        basic_lr=0.05,
    ),
    model_ema=dict(
        enabled=True,
        momentum=0.99996,
    ),
)


class Cfg(ResNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

se_resnet101

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import ResNetConfig

_cfg = dict(
    model=dict(
        name="se_resnet101",
    ),
)


class Cfg(ResNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

se_resnet152

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import ResNetConfig

_cfg = dict(
    model=dict(
        name="se_resnet152",
    ),
)


class Cfg(ResNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

se_resnet18

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import ResNetConfig

_cfg = dict(
    batch_size=128,
    model=dict(
        name="se_resnet18",
    ),
    solver=dict(
        basic_lr=0.1,
    ),
    model_ema=dict(
        enabled=True,
```

(续下页)

(接上页)

```
        momentum=0.99992,
    ),
)


class Cfg(ResNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

se_resnet34

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import ResNetConfig

_cfg = dict(
    batch_size=64,
    model=dict(
        name="se_resnet34",
    ),
    solver=dict(
        basic_lr=0.05,
    ),
    model_ema=dict(
        enabled=True,
        momentum=0.99996,
    ),
)


class Cfg(ResNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

se_resnet50

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import ResNetConfig

_cfg = dict(
    batch_size=64,
    model=dict(
        name="se_resnet50",
    ),
    solver=dict(
        basic_lr=0.05,
    ),
    model_ema=dict(
        enabled=True,
        momentum=0.99996,
    ),
)


class Cfg(ResNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

se_resnext101_32x4d

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import ResNetConfig

_cfg = dict(
    model=dict(
```

(续下页)

(接上页)

```
        name="se_resnext101_32x4d",
    ),
)


class Cfg(ResNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

se_resnext101_32x8d

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import ResNetConfig

_cfg = dict(
    model=dict(
        name="se_resnext101_32x8d",
    ),
)


class Cfg(ResNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

se_resnext101_64x4d

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
```

(续下页)

(接上页)

```
from basecls.configs import ResNetConfig

_cfg = dict(
    model=dict(
        name="se_resnext101_64x4d",
    ),
)


class Cfg(ResNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

se_resnext152_32x4d

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import ResNetConfig

_cfg = dict(
    model=dict(
        name="se_resnext152_32x4d",
    ),
)


class Cfg(ResNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

se_resnext152_32x8d

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import ResNetConfig

_cfg = dict(
    model=dict(
        name="se_resnext152_32x8d",
    ),
)


class Cfg(ResNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

se_resnext152_64x4d

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import ResNetConfig

_cfg = dict(
    model=dict(
        name="se_resnext152_64x4d",
    ),
)


class Cfg(ResNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
```

(续下页)

(接上页)

```
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

se_resnext50_32x4d

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import ResNetConfig

_cfg = dict(
    batch_size=64,
    model=dict(
        name="se_resnext50_32x4d",
    ),
    solver=dict(
        basic_lr=0.05,
    ),
    model_ema=dict(
        enabled=True,
        momentum=0.99996,
    ),
)


class Cfg(ResNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

wide_resnet101_2

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import ResNetConfig

_cfg = dict(
    model=dict(
        name="wide_resnet101_2",
    ),
)


class Cfg(ResNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

wide_resnet50_2

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import ResNetConfig

_cfg = dict(
    batch_size=64,
    model=dict(
        name="wide_resnet50_2",
    ),
    solver=dict(
        basic_lr=0.05,
    ),
    model_ema=dict(
        enabled=True,
        momentum=0.99996,
    ),
)
```

(续下页)

(接上页)

```
class Cfg(ResNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

ShuffleNet V2 Series

snetv2_x050

Model info

Python source

```
from basecls.configs.snet_cfg import SNetConfig

_cfg = dict(
    model=dict(
        name="snetv2_x050",
    ),
)


class Cfg(SNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

snetv2_x100

Model info

Python source

```
from basecls.configs.snet_cfg import SNetConfig

_cfg = dict(
    model=dict(
        name="snetv2_x100",
    ),
```

(续下页)

(接上页)

```
)


class Cfg(SNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

snetv2_x150

Model info

Python source

```
from basecls.configs.snet_cfg import SNetConfig

_cfg = dict(
    model=dict(
        name="snetv2_x150",
    ),
)


class Cfg(SNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

snetv2_x200

Model info

Python source

```
from basecls.configs.snet_cfg import SNetConfig

_cfg = dict(
    model=dict(
        name="snetv2_x200",
```

(续下页)

(接上页)

```
    ),
)


class Cfg(SNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

Swin Transformer Series

swin_base_patch4_window12_384

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import SwinConfig

_cfg = dict(
    model=dict(
        name="swin_base_patch4_window12_384",
    ),
)


class Cfg(SwinConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

swin_base_patch4_window7_224

Model info

Python source

```python
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import SwinConfig

_cfg = dict(
    model=dict(
        name="swin_base_patch4_window7_224",
    ),
)


class Cfg(SwinConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

swin_large_patch4_window12_384

Model info

Python source

```python
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import SwinConfig

_cfg = dict(
    model=dict(
        name="swin_large_patch4_window12_384",
    ),
)


class Cfg(SwinConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

swin_large_patch4_window7_224

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import SwinConfig

_cfg = dict(
    model=dict(
        name="swin_large_patch4_window7_224",
    ),
)


class Cfg(SwinConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

swin_small_patch4_window7_224

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import SwinConfig

_cfg = dict(
    batch_size=128,
    model=dict(
        name="swin_small_patch4_window7_224",
    ),
    solver=dict(
        accumulation_steps=1,
    ),
)
```

(续下页)

(接上页)

```
class Cfg(SwinConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

swin_tiny_patch4_window7_224

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import SwinConfig

_cfg = dict(
    batch_size=128,
    model=dict(
        name="swin_tiny_patch4_window7_224",
    ),
    solver=dict(
        accumulation_steps=1,
    ),
)


class Cfg(SwinConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

VGG Series

vgg11_bn

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import VGGConfig

_cfg = dict(
    model=dict(
        name="vgg11_bn",
    ),
)


class Cfg(VGGConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

vgg13_bn

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import VGGConfig

_cfg = dict(
    model=dict(
        name="vgg13_bn",
    ),
)


class Cfg(VGGConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

vgg16_bn

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import VGGConfig

_cfg = dict(
    model=dict(
        name="vgg16_bn",
    ),
)


class Cfg(VGGConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

vgg19_bn

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import VGGConfig

_cfg = dict(
    model=dict(
        name="vgg19_bn",
    ),
)


class Cfg(VGGConfig):
    def __init__(self, values_or_file=None, **kwargs):
```

(续下页)

(接上页)

```
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

ViT Series

vit_base_patch16_224

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import ViTConfig

_cfg = dict(
    model=dict(
        name="vit_base_patch16_224",
    ),
)


class Cfg(ViTConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

vit_base_patch16_384

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import ViTConfig

_cfg = dict(
    model=dict(
        name="vit_base_patch16_384",
```

(续下页)

(接上页)

```
    ),
    preprocess=dict(
        img_size=384,
    ),
    test=dict(
        img_size=384,
        crop_pct=1.0,
    ),
)


class Cfg(ViTConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

vit_base_patch32_224

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import ViTConfig

_cfg = dict(
    model=dict(
        name="vit_base_patch32_224",
    ),
)


class Cfg(ViTConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

vit_base_patch32_384

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import ViTConfig

_cfg = dict(
    model=dict(
        name="vit_base_patch32_384",
    ),
    preprocess=dict(
        img_size=384,
    ),
    test=dict(
        img_size=384,
        crop_pct=1.0,
    ),
)


class Cfg(ViTConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

vit_small_patch16_224

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import ViTConfig

_cfg = dict(
    model=dict(
        name="vit_small_patch16_224",
```

(续下页)

(接上页)

```
    ),
)


class Cfg(ViTConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

vit_small_patch16_384

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import ViTConfig

_cfg = dict(
    model=dict(
        name="vit_small_patch16_384",
    ),
    preprocess=dict(
        img_size=384,
    ),
    test=dict(
        img_size=384,
        crop_pct=1.0,
    ),
)


class Cfg(ViTConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

vit_small_patch32_224

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import ViTConfig

_cfg = dict(
    batch_size=128,
    model=dict(
        name="vit_small_patch32_224",
    ),
    solver=dict(
        basic_lr=1.25e-4,
    ),
    model_ema=dict(
        momentum=0.99992,
    ),
)


class Cfg(ViTConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

vit_small_patch32_384

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import ViTConfig

_cfg = dict(
    model=dict(
        name="vit_small_patch32_384",
```

(续下页)

(接上页)

```
    ),
    preprocess=dict(
        img_size=384,
    ),
    test=dict(
        img_size=384,
        crop_pct=1.0,
    ),
)


class Cfg(ViTConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

vit_tiny_patch16_224

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import ViTConfig

_cfg = dict(
    batch_size=128,
    model=dict(
        name="vit_tiny_patch16_224",
    ),
    solver=dict(
        basic_lr=1.25e-4,
    ),
    model_ema=dict(
        momentum=0.99992,
    ),
)


class Cfg(ViTConfig):
```

(续下页)

(接上页)

```
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

vit_tiny_patch16_384

Model info

Python source

```
#!/usr/bin/env python3
# Copyright (c) 2014-2021 Megvii Inc. All rights reserved.
from basecls.configs import ViTConfig

_cfg = dict(
    model=dict(
        name="vit_tiny_patch16_384",
    ),
    preprocess=dict(
        img_size=384,
    ),
    test=dict(
        img_size=384,
        crop_pct=1.0,
    ),
)


class Cfg(ViTConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)
```

1.2 Tutorial

本用户指南旨在帮助用户快速安装与使用 BaseCls 。我们会根据社区的反馈，不断丰富指南内容，改进指南质量。

1.2.1 安装 BaseCls

安装环境

BaseCls 需要 Python >= 3.6。

BaseCls 依赖 MegEngine >= 1.6.0。

通过包管理器安装

通过 `pip` 包管理器安装 BaseCls 的命令如下：

```
pip3 install basecls --user
```

默认不会安装包括 MegEngine 在内的部分依赖，可以通过以下命令进行完整安装：

```
pip3 install basecls[all] --user
```

备注： 对于 `conda` 用户, 可以选择通过在环境中先安装 `pip` ，再按照上述方式进行 BaseCls 的安装。

通过源代码安装

为保证模型性能的可追溯性，避免实验碎片化，建议通过包管理器安装。如果包管理器安装的方式无法满足你的需求，则可以尝试自行通过源码安装。

请先将 源码 clone 到本地。

安装依赖

```
pip3 install -r requirements.txt --user
```

安装 BaseCls

```
python3 setup.py develop --user
```

验证安装

在 Python 中导入 BaseCls 验证安装成功并查看安装版本：

```
import basecls
print(basecls.__version__)
```

1.2.2 配置实验环境

实验目录

用户可以任意新建一个目录作为实验目录，BaseCls 鼓励的实验范式是 library 与 playground 分离，每个人（组）自行维护一个 playground 目录。

模型配置文件

一个模型通常对应一个 `.py` 模型配置文件，一个标准的模型配置文件如下所示：

```
from basecls.configs import ResNetConfig

_cfg = dict(
    batch_size=64,
    model=dict(
        name="resnet50",
    ),
    solver=dict(
        basic_lr=0.05,
    ),
)


class ResNet50Config(ResNetConfig):
    def __init__(self, values_or_file=None, **kwargs):
        super().__init__(_cfg)
        self.merge(values_or_file, **kwargs)


Cfg = ResNet50Config
```

BaseCls 要求模型配置文件必须包含一个名为 `Cfg` 的类，该类必须为 `ConfigDict` 的子类。用户可以继承已有的模型配置文件并修改。

BaseCls 基于注册机制，支持接入用户自定义的网络。详情见自定义网络 。

数据源

- 倘若数据集在本地，请使用*FolderLoader* 并填写 data.train_path 和 data.val_path 字段。
- 倘若只需要随机数据（如性能评测等），请使用*FakeData* 。

BaseCls 支持接入第三方数据源。详情见自定义数据源 。

数据增强

BaseCls 标准数据处理流程为 RandomResizedCrop -> RandomHorizontalFlip -> [augments] -> *RandomErasing* -> *MixupCutmixCollator* 。其中 [augments] 为可修改的数据增强部分，默认为*ColorAugment* 。

BaseCls 支持自定义数据增强。详情见自定义数据增强 。

1.2.3 训练

本节例子中假设你已经处于你的实验目录中，你的模型配置文件名为 resnet50.py 。

单机训练

目前单机训练会占用全部 GPU：

```
cls_train -f resnet50.py
```

若中途训练中断，可追加 --resume 参数，BaseCls 会从上一次保存的 checkpoint 开始继续训练。

训练结束后，模型将自动使用最后一个 checkpoint ，测试其在 ImageNet 1k validation set 上的 Top-1 Accuracy 和 Top-5 Accuracy。

可以在终端中追加参数覆盖模型配置文件中的字段，例如你需要在显存更大的机器上训练模型：

```
cls_train -f resnet50.py batch_size 128 solver.basic_lr 0.1
```

模型配置文件中的 batch_size 和 solver.basic_lr 字段将被覆盖。

> **警告：** 并不推荐通过传参修改配置项，因为这可能会影响“模型配置文件—测试结果”的可复现性。直接修改模型配置文件是更推荐的做法。

1.2.4 测试

本节例子中假设你已经处于你的实验目录中，你的模型配置文件名为 `resnet50.py` 。

单机测试

测试脚本默认在模型配置文件的 `output_dir` 目录中寻找最后一个 checkpoint ，并测试在 ImageNet 1k validation set 上的 Top-1 Accuracy 和 Top-5 Accuracy 。

```
cls_test -f resnet50.py
```

如果你需要测试指定的 checkpoint 或模型权重，请加 `-w` 参数，既可以是本地路径，也可以是 OSS 路径。

1.2.5 调试

本节例子中假设你已经处于你的实验目录中，你的模型配置文件名为 `resnet50.py` 。

在终端中调试

如果只希望构造出模型在终端中调试，或在其他地方引用，BaseCls 提供了简单的工具函数，帮助用户直接从模型配置文件中构造网络：

```
# 创建配置
from resnet50 import Cfg
cfg = Cfg()

# 创建模型
from basecls.models import build_model
model = build_model(cfg)

# 如果不需要分类头，可以直接删除
del model.head

# 载入权重
from basecls.models import load_model
if getattr(cfg, "weights", None) is not None:
    load_model(model, cfg.weights, strict=False)

# 开始调试
from megengine.utils.module_stats import module_stats
module_stats(model, input_shapes=(1, 3, 224, 224))
```

BaseCls 同样支持在 `hub` 中调用官方已有的模型并载入初始化权重（初次载入需要联网）：

```python
# 创建模型
import megengine.hub as hub
model = hub.load(
    "megvii-research/basecls:main",
    "resnet50",  # 模型名称
    use_cache=False,
    pretrained=True,  # 载入初始化权重
)

# 开始调试
from megengine.utils.module_stats import module_stats
module_stats(model, input_shapes=(1, 3, 224, 224))
```

1.2.6 进阶内容

本节例子中假设你已经处于你的实验目录中，你的模型配置文件名为 `resnet50.py` 。

DTR

BaseCls 支持利用 DTR 技术进行训练。相关知识内容请参考文档。

通过修改模型配置文件中的 `dtr` 相关字段开启：

```
cfg.dtr = True
```

也可以在终端中追加参数覆盖模型配置文件中的字段：

```
cls_train -f resnet50.py dtr True
```

AMP

BaseCls 支持利用 AMP 自动混合精度进行训练，可以提升训练吞吐。相关知识内容请参考文档。

通过修改模型配置文件中的 `amp` 相关字段开启：

```
cfg.amp.enabled = True
```

也可以在终端中追加参数覆盖模型配置文件中的字段：

```
cls_train -f resnet50.py amp.enabled True
```

混合精度训练中使用 fp16 代替 fp32 进行梯度计算，因此会涉及浮点数表达能力的问题。为了防止小量级的梯度在 fp16 表示中损失精度，实践中往往将损失函数扩大适当的倍率，称为 `loss_scale` ，在 `GradScaler` 中实现了相关逻辑。

混合精度的默认行为是固定 `loss_scale = 128.0`，另外一种动态调整 `loss_scale` 的方法可以更好地利用 fp16 的动态范围，可以使用 `dynamic_scale` 字段开启。开启时初始 `loss_scale = 65536.0`，并每 `2000` 轮迭代进行一次翻倍，当遇到梯度为 `inf` 时会立即将其减半：

```
cfg.amp.dynamic_scale = True
```

Model EMA

Model EMA 是一种加快模型收敛的方法，常用于较新的卷积模型和 Transformer 类模型。通过统计模型参数的滑动平均，能够获得更好的泛化能力。

通过修改模型配置文件中的 `model_ema` 相关字段开启，默认每步迭代都会更新 Model EMA：

```
cfg.model_ema.enabled = True
cfg.model_ema.momentum = 0.9999
cfg.model_ema.update_period = 1
```

也可以设置 `update_period` 隔一定步数进行更新，一个建议是当提升更新间隔时，要相应减小 `momentum` 的值：

```
cfg.model_ema.enabled = True
cfg.model_ema.momentum = 0.9992
cfg.model_ema.update_period = 8
```

另外如果对于设置正确的 `momentum` 没有经验，也可以设置 `alpha` 的默认配置，`momentum` 会根据更新间隔和更新量 `alpha` 自动适配：

```
cfg.model_ema.enabled = True
cfg.model_ema.alpha = 1e-5
cfg.model_ema.update_period = 32
```

相关更新公式为：

$$\text{momentum} = 1 - \alpha \times \frac{\text{update\_period} \times \text{total\_batch\_size}}{\text{max\_epochs}}$$

1.2.7 自定义网络

BaseCls 支持接入用户自定义的网络。

实现范式

- 网络必须继承自 `Module` 。
- 自定义参数通过模型配置文件 `model` 字段传入。
- 以下字段为保留字段不可使用：
  - `model.name` ， BaseCls 用此字段构造网络。
  - `model.head.name` ， BaseCls 用此字段构造分类头。

具体步骤

实现网络并注册

```
from typing import Any, Mapping

import megengine as mge
import megengine.functional as F
import megengine.module as M
from basecls.layers import build_head
from basecls.utils import registers

class AlexNetHead(M.Module):

    def __init__(self, w_in: int, w_out: int, width: int = 4096):
        super().__init__()
        self.avg_pool = M.AdaptiveAvgPool2d((6, 6))
        self.classifier = M.Sequential(
            M.Dropout(),
            M.Linear(w_in * 6 * 6, width),
            M.ReLU(),
            M.Dropout(),
            M.Linear(width, width),
            M.ReLU(),
            M.Linear(width, w_out),
        )

    def forward(self, x: mge.Tensor) -> mge.Tensor:
        x = self.avg_pool(x)
        x = F.flatten(x, 1)
        x = self.classifier(x)
        return x

```

(续下页)

(接上页)

```python
@registers.models.register()
class AlexNet(M.Module):

    def __init__(self, head: Mapping[str, Any] = None):
        super().__init__()
        self.features = M.Sequential(
            M.Conv2d(3, 64, kernel_size=11, stride=4, padding=2),
            M.ReLU(),
            M.MaxPool2d(kernel_size=3, stride=2),
            M.Conv2d(64, 192, kernel_size=5, padding=2),
            M.ReLU(),
            M.MaxPool2d(kernel_size=3, stride=2),
            M.Conv2d(192, 384, kernel_size=3, padding=1),
            M.ReLU(),
            M.Conv2d(384, 256, kernel_size=3, padding=1),
            M.ReLU(),
            M.Conv2d(256, 256, kernel_size=3, padding=1),
            M.ReLU(),
            M.MaxPool2d(kernel_size=3, stride=2),
        )
        self.head = build_head(256, head)

    def forward(self, x: mge.Tensor) -> mge.Tensor:
        x = self.features(x)
        if getattr(self, "head", None) is not None:
            x = self.head(x)
        return x
```

修改模型配置文件

```python
_cfg = dict(
    ...
    num_classes=1000,
    model=dict(
        name="AlexNet",
        ...  # 你想传入的自定义参数
        head=dict(
            name=AlexNetHead,  # 也可以直接传入一个类
            # w_out=1000,  # 若该字段未定义，会自动传入cfg.num_classes
        ),
    )
    ...
```

(续下页)

(接上页)

```
)
```

1.2.8 自定义数据源

BaseCls 支持接入第三方数据源。

实现范式

- 数据源类必须实现 `build` 类方法，返回一个 `Iterable` 对象（实现了 `__iter__` 方法）。
- 自定义参数通过模型配置文件 `data` 字段传入。
- 以下字段为保留字段不可使用：
  - `data.name`，BaseCls 用此字段构造数据源。

具体步骤

实现数据源并注册

```python
from basecls.utils import registers
from basecore.config import ConfigDict

@registers.dataloaders.register()
class YourDataSourceBuilder:

    @classmethod
    def build(cls, cfg: ConfigDict, augments):
        return YourDataSource(cfg, augments)

class YourDataSource:

    def __init__(self, cfg: ConfigDict, augments):
        pass

    def __iter__(self):
        pass
```

修改模型配置文件

```
_cfg = dict(
    ...
    data=dict(
        name="YourDataSourceBuilder",
        ...  # 你想传入的自定义参数
    )
    ...
)
```

1.2.9 自定义数据增强

BaseCls 支持自定义数据增强。

实现范式

- 数据增强类必须实现 `build` 类方法，返回一个 `VisionTransform` 对象。
- 自定义参数通过模型配置文件 `augments` 字段传入。
- 以下字段为保留字段不可使用：
  - `augments.name` ，BaseCls 用此字段构造数据增强类。

具体步骤

实现网络并注册

```
import megengine.data.transform as T
import numpy as np
from basecls.utils import registers
from basecore.config import ConfigDict

@registers.augments.register()
class YourAugmentBuilder:

    @classmethod
    def build(cls, cfg: ConfigDict) -> T.Transform:
        return YourAugment(cfg)

class YourAugment(T.VisionTransform):

```

(续下页)

(接上页)

```
    def __init__(self, cfg: ConfigDict):
        pass

    def _apply_image(self, image: np.ndarray) -> np.ndarray:
        pass
```

修改模型配置文件

```
_cfg = dict(
    ...
    argments=dict(
        name="YourAugmentBuilder",
        ...  # 你想传入的自定义参数
    )
    ...
)
```

1.2.10 贡献指南

开发环境

```
# 安装依赖
pip3 install -r requirements-dev.txt --user

# 配置 pre-commit
pre-commit install
```

开发流程

提交者需补充相应修改的单元测试。

```
# （外部开发者）fork repo，或（内部开发者）建立 new-feature 分支
git checkout -b new-feature

# 进行修改

# 代码风格检查与格式化
make lint
make format
```

(续下页)

(接上页)

```
# 单元测试与覆盖率检查
make unittest

# 提交修改
git commit

# 提交 MR/PR
```

1.3 basecls

1.3.1 basecls.configs

class basecls.configs.**BaseConfig**(*values_or_file=None*, ***kwargs*)

基类: ConfigDict

link_log_dir(*link_name='log'*)

class basecls.configs.**EffNetConfig**(*values_or_file=None*, ***kwargs*)

基类: *BaseConfig*

class basecls.configs.**EffNetLiteConfig**(*values_or_file=None*, ***kwargs*)

基类: *EffNetConfig*

class basecls.configs.**HRNetConfig**(*values_or_file=None*, ***kwargs*)

基类: *BaseConfig*

class basecls.configs.**MBConfig**(*values_or_file=None*, ***kwargs*)

基类: *BaseConfig*

class basecls.configs.**RegNetConfig**(*values_or_file=None*, ***kwargs*)

基类: *BaseConfig*

class basecls.configs.**RepVGGConfig**(*values_or_file=None*, ***kwargs*)

基类: *BaseConfig*

class basecls.configs.**ResMLPConfig**(*values_or_file=None*, ***kwargs*)

基类: *BaseConfig*

class basecls.configs.**ResNetConfig**(*values_or_file=None*, ***kwargs*)

基类: *BaseConfig*

class basecls.configs.**SNetConfig**(*values_or_file=None*, ***kwargs*)

基类: *BaseConfig*

class basecls.configs.**SwinConfig**(*values_or_file=None*, ***kwargs*)

基类: *BaseConfig*

class basecls.configs.**VGGConfig**(*values_or_file=None*, ***kwargs*)

基类: *BaseConfig*

class basecls.configs.**ViTConfig**(*values_or_file=None*, ***kwargs*)

基类: *BaseConfig*

basecls.configs.base_cfg

class basecls.configs.base_cfg.**BaseConfig**(*values_or_file=None*, ***kwargs*)

基类: ConfigDict

link_log_dir(*link_name='log'*)

basecls.configs.effnet_cfg

class basecls.configs.effnet_cfg.**EffNetConfig**(*values_or_file=None*, ***kwargs*)

基类: *BaseConfig*

class basecls.configs.effnet_cfg.**EffNetLiteConfig**(*values_or_file=None*, ***kwargs*)

基类: *EffNetConfig*

basecls.configs.hrnet_cfg

class basecls.configs.hrnet_cfg.**HRNetConfig**(*values_or_file=None*, ***kwargs*)

基类: *BaseConfig*

basecls.configs.mbnet_cfg

class basecls.configs.mbnet_cfg.**MBConfig**(*values_or_file=None*, ***kwargs*)

基类: *BaseConfig*

basecls.configs.regnet_cfg

class basecls.configs.regnet_cfg.**RegNetConfig**(*values_or_file=None*, ***kwargs*)

基类: *BaseConfig*

basecls.configs.repvgg_cfg

class basecls.configs.repvgg_cfg.**RepVGGConfig**(*values_or_file=None*, ***kwargs*)

基类: *BaseConfig*

basecls.configs.resmlp_cfg

class basecls.configs.resmlp_cfg.**ResMLPConfig**(*values_or_file=None*, ***kwargs*)

基类: *BaseConfig*

basecls.configs.resnet_cfg

class basecls.configs.resnet_cfg.**ResNetConfig**(*values_or_file=None*, ***kwargs*)

基类: *BaseConfig*

basecls.configs.snet_cfg

class basecls.configs.snet_cfg.**SNetConfig**(*values_or_file=None*, ***kwargs*)

基类: *BaseConfig*

basecls.configs.swin_cfg

class basecls.configs.swin_cfg.**SwinConfig**(*values_or_file=None*, ***kwargs*)

基类: *BaseConfig*

basecls.configs.vgg_cfg

class basecls.configs.vgg_cfg.**VGGConfig**(*values_or_file=None*, ***kwargs*)

基类: *BaseConfig*

basecls.configs.vit_cfg

class basecls.configs.vit_cfg.**ViTConfig**(*values_or_file=None*, ***kwargs*)

基类: *BaseConfig*

1.3.2 basecls.data

class basecls.data.**FakeData**

基类：object

Fake data useful for benchmark.

classmethod build(*cfg*, *train=True*, *augments=None*)

返回类型

DataLoader

class basecls.data.**FolderLoader**

基类：object

Local dataloader factory.

The source is the local folder.

classmethod build(*cfg*, *train=True*, *augments=None*)

返回类型

DataLoader

basecls.data.**build_dataloader**(*cfg*, *train*, *augments=None*, *mode='folder'*, *infinite=False*, *rank=None*)

Build function for MegEngine dataloader.

参数

- **cfg** (ConfigDict) –config for building dataloader.
- **train** (bool) –train set or test set.
- **augments** (Optional[Transform]) –augments for building dataloder. Default: None
- **infinite** (bool) –make dataloader infinite or not. default: False
- **rank** (Optional[int]) –machine rank, only useful for infinite dataloader. Default: None

返回类型

DataLoader

返回

A dataloader.

basecls.data.**build_dataset**(*cfg*, *train=True*, *mode='folder'*)

Build function for dataset.

参数

- **cfg** (ConfigDict) –config for building dataset.
- **train** (bool) –train set or test set. Default: True

返回类型

VisionDataset

返回

A dataset which loads data from Nori on OSS.

class basecls.data.**AutoAugment**

基类：object

AutoAugment.

classmethod build(*cfg*)

返回类型

Transform

class basecls.data.**ColorAugment**

基类：object

Color augmentation.

classmethod build(*cfg*)

返回类型

Transform

class basecls.data.**RandAugment**

基类：object

Random augmentation.

classmethod build(*cfg*)

返回类型

Transform

class basecls.data.**SimpleAugment**

基类：object

Simple augmentation.

classmethod build(*cfg*)

返回类型

Transform

basecls.data.**build_mixup**(*cfg*, *train=True*)

Build (optionally) Mixup/CutMix augment.

参数

- **cfg** (ConfigDict) –config for building Mixup/CutMix collator.

- **train** (bool) –train set or test set. Default: True

返回类型

Optional[Collator]

返回

MixupCutmixCollator or None

basecls.data.**build_transform**(*cfg*, *train=True*, *augments=None*)

Build function for MegEngine transform.

参数

- **cfg** (ConfigDict) –config for building transform.
- **train** (bool) –train set or test set. Default: True
- **augments** (Optional[Transform]) –augments for building transform.

返回类型

Transform

返回

A transform.

basecls.data.augment

AutoAugment and RandAugment

AutoAugment: "AutoAugment: Learning Augmentation Policies from Data"

RandAugment: "RandAugment: Practical automated data augmentation with a reduced search space"

引用

https://github.com/facebookresearch/pycls/blob/main/pycls/datasets/augment.py https://github.com/rwightman/pytorch-image-models/blob/master/timm/data/auto_augment.py

class basecls.data.augment.**TorchAutoAugment**

基类: object

class basecls.data.augment.**TorchRandAugment**(*magnitude*, *magnitude_std=0.0*, *prob=0.5*, *n_ops=2*)

基类: object

basecls.data.build

class basecls.data.build.**FakeData**

基类：object

Fake data useful for benchmark.

classmethod build(*cfg*, *train=True*, *augments=None*)

返回类型

DataLoader

class basecls.data.build.**FolderLoader**

基类：object

Local dataloader factory.

The source is the local folder.

classmethod build(*cfg*, *train=True*, *augments=None*)

返回类型

DataLoader

basecls.data.const

basecls.data.dataloader

basecls.data.dataloader.**build_dataloader**(*cfg*, *train*, *augments=None*, *mode='folder'*, *infinite=False*, *rank=None*)

Build function for MegEngine dataloader.

参数

- **cfg** (ConfigDict) –config for building dataloader.
- **train** (bool) –train set or test set.
- **augments** (Optional[Transform]) –augments for building dataloder. Default: None
- **infinite** (bool) –make dataloader infinite or not. default: False
- **rank** (Optional[int]) –machine rank, only useful for infinite dataloader. Default: None

返回类型

DataLoader

返回

A dataloader.

basecls.data.dataset

basecls.data.dataset.**build_dataset**(*cfg*, *train=True*, *mode='folder'*)

Build function for dataset.

参数

- **cfg** (ConfigDict) –config for building dataset.
- **train** (bool) –train set or test set. Default: True

返回类型

VisionDataset

返回

A dataset which loads data from Nori on OSS.

basecls.data.fake_data

class basecls.data.fake_data.**FakeDataLoader**(*batch_size*, *img_size=224*, *channels=3*, *length=100*, *num_classes=1000*)

基类：object

参数

- **batch_size** (int) –batch size
- **img_size** (Union[int, Tuple[int, int]]) –height and width. Default: 224
- **channels** (int) –color channels. Default: 3
- **length** (int) –loader length. Default: 100
- **num_classes** (int) –number of classes. Default: 1000

basecls.data.mixup

Mixup and CutMix

Mixup: "Mixup: Beyond Empirical Risk Minimization"

CutMix: "CutMix: Regularization Strategy to Train Strong Classifiers with Localizable Features"

引用

https://github.com/rwightman/pytorch-image-models/blob/master/timm/data/mixup.py

class basecls.data.mixup.**MixupCutmixTransform**(*mixup_alpha=1.0*, *cutmix_alpha=0.0*, *cutmix_minmax=None*, *prob=1.0*, *switch_prob=0.5*, *mode='batch'*, *data_format='HWC'*, *num_classes=1000*, *calibrate_cutmix_lambda=True*, *calibrate_mixup_lambda=False*, *permute=False*, ***, *order=None*)

基类：VisionTransform

Implement Mixup and CutMix as VisionTransform.

备注： When composed in Compose , batch_compose must be set to True.

参数

- **mixup_alpha** (float) –mixup alpha value, mixup is active if > 0. Default: 1.0
- **cutmix_alpha** (float) –cutmix alpha value, cutmix is active if > 0. Default: 0.0
- **cutmix_minmax** (Optional[List[float]]) –cutmix min/max image ratio, cutmix is active and uses this vs alpha if not None. Default: None
- **prob** (float) –probability of applying mixup or cutmix per batch or element. Default: 1.0
- **switch_prob** (float) –probability of switching to cutmix instead of mixup when both are active. Default: 0.5
- **mode** (str) –how to apply mixup/cutmix params, supports "batch", "pair" (pair of elements) and "elem" (element). Default: "batch"
- **data_format** (str) –"CHW" or "HWC", use "HWC" if use this transform before T.ToMode(). Default: "HWC"
- **num_classes** (int) –number of classes for target. Default: 1000
- **calibrate_cutmix_lambda** (bool) –apply lambda correction when cutmix bbox clipped by image borders. Correction is based on clipped area for cutmix. Default: True
- **calibrate_mixup_lambda** (bool) –enforce mixup lambda to be greater than 0.5, only make difference in "elem" mode. Default: False
- **permute** (bool) –whether mixup with permuted samples instead of flipped samples. Default: False

apply_batch(*inputs*)

Apply transform on batch input data.

class basecls.data.mixup.**MixupCutmixCollator**(**args*, ***kwargs*)

基类：Collator

A faster version implemented as a collator.

apply(*inputs*)

basecls.data.rand_erase

Random Erasing (Cutout)

Random Erasing: "Random Erasing Data Augmentation"

引用

https://github.com/rwightman/pytorch-image-models/blob/master/timm/data/random_erasing.py

class basecls.data.rand_erase.**RandomErasing**(*prob=0.5*, *scale_range=(0.02, 1 / 3)*, *ratio=0.3*, *mode='const'*, *count=1*, *num_splits=0*, *pad_mean=0.0*, *pad_std=1.0*, ***, *order=None*)

基类：VisionTransform

Randomly selects a rectangle region in an image and erases its pixels.

This variant of RandomErasing is intended to be applied to either a batch or single image tensor after it has been normalized by dataset mean and std.

参数

- **prob** (float) –probability that Random Erasing operation will be performed. Default: 0.5
- **scale_range** (Tuple[float, float]) –percentage of erased area wrt input image area. Default: (0.02, 1.0 / 3)
- **ratio** (Union[float, Tuple[float, float]]) –aspect ratio of erased area. if a scalar, range will be (ratio, 1.0 / ratio). Default: 0.3
- **mode** (str) –pixel color mode, one of "const", "rand", or "pixel". Default: "const" "const" - erase block is constant color of 0 for all channels "rand" - erase block is same per-channel random (normal) color "pixel" - erase block is per-pixel random (normal) color
- **count** (Union[int, Tuple[int, int]]) –maximum number or range of erasing blocks per image, area per box is scaled by count. if a scalar, per-image count is randomly chosen between 1 and this value. if a range, per-image count is randomly chosen between this range. Default: 1

- **num_splits** (int) –augmentation splits. if > 1, the first split will not be erased. Default: 0
- **pad_mean** (Union[float, Tuple[float, float, float]]) –the mean of padding pixels. Default: 0.0
- **pad_std** (Union[float, Tuple[float, float, float]]) –the std of padding pixels. Default: 1.0

apply_batch(*inputs*)

Apply transform on batch input data.

basecls.data.transform

basecls.data.transform.**build_transform**(*cfg*, *train=True*, *augments=None*)

Build function for MegEngine transform.

参数

- **cfg** (ConfigDict) –config for building transform.
- **train** (bool) –train set or test set. Default: True
- **augments** (Optional[Transform]) –augments for building transform.

返回类型

Transform

返回

A transform.

class basecls.data.transform.**AutoAugment**

基类：object

AutoAugment.

classmethod build(*cfg*)

返回类型

Transform

class basecls.data.transform.**SimpleAugment**

基类：object

Simple augmentation.

classmethod build(*cfg*)

返回类型

Transform

class basecls.data.transform.**ColorAugment**

基类：object

Color augmentation.

classmethod build(*cfg*)

返回类型
Transform

class basecls.data.transform.**RandAugment**

基类：object

Random augmentation.

classmethod build(*cfg*)

返回类型
Transform

basecls.data.transform.**build_mixup**(*cfg*, *train=True*)

Build (optionally) Mixup/CutMix augment.

参数

- **cfg** (ConfigDict) –config for building Mixup/CutMix collator.
- **train** (bool) –train set or test set. Default: True

返回类型
Optional[Collator]

返回
MixupCutmixCollator or None

1.3.3 basecls.engine

class basecls.engine.**DefaultHooks**

基类：object

The default hooks factory.

It combines LRSchedulerHook -> PreciseBNHook -> ResumeHook -> TensorboardHook -> LoggerHook -> CheckpointHook -> EvalHook.

classmethod build(*cfg*)

Build function with a simple strategy.

参数
cfg (ConfigDict) –config for setting hooks.

返回类型

List[BaseHook]

返回

A hook list.

class basecls.engine.**AccEvaluator**

基类：BaseEvaluator

Classification evaluator with top-1 and top-5 accuracy.

ResultType

Tuple[int, float, float] 的别名

preprocess(*input_data*)

Preprocess input data per batch.

参数

input_data (Sequence[ndarray]) –input data.

返回类型

Tensor

返回

Preprocessed input data.

postprocess(*model_outputs*, *input_data*)

Postprocess model outputs with input data per batch.

参数

- **model_outputs** (Tensor) –model outputs.
- **input_data** (Sequence[ndarray]) –input data.

返回类型

Tuple[int, float, float]

返回

A tuple that (batch size, top-1 accuracy per batch, top-5 accuracy per batch).

evaluate(*results*)

Evaluation function.

参数

results (Iterable[Tuple[int, float, float]]) –all results.

class basecls.engine.**ClsTester**(*cfg*, *model*, *dataloader*)

基类：BaseTester

test(*warm_iters=5*, *log_seconds=5*)

class basecls.engine.**ClsTrainer**(*cfg*, *model*, *dataloader*, *solver*, *hooks=None*)

基类：BaseTrainer

Classification trainer.

参数

- **cfg** (ConfigDict) –config for training.
- **model** (Module) –model for training.
- **dataloader** (Union[DataLoader, *FakeDataLoader*]) –dataloader for training.
- **solver** (*Solver*) –solver for training.
- **hooks** (Optional[Iterable[BaseHook]]) –hooks for training.

cfg

config for training.

model

model for training.

ema

model exponential moving average.

dataloader

dataloader for training.

solver

solver for training.

progress

object for recording training process.

loss

loss function for training.

meter

object for recording metrics.

train()

参数

- **start_info** (*Iterable*) –[epoch, iter] for training start.
- **max_info** (*Iterable*) –[max_epoch, max_iter] for training.

before_train()

before_epoch()

after_epoch()

train_one_iter()

Basic logic of training one iteration.

model_step(*samples*, *targets*)

modify_grad()

model_ema_step()

Implement momentum based Exponential Moving Average (EMA) for model states https://github.com/rwightman/pytorch-image-models/blob/master/timm/utils/model_ema.py

Also inspired by Pycls https://github.com/facebookresearch/pycls/pull/138/, which is more flexible and efficient

Heuristically, one can use a momentum of 0.9999 as used by Tensorflow and 0.9998 as used by timm, which updates model ema every iter. To be more efficient, one can set `update_period` to e.g. 8 or 32 to speed up your training, and decrease your momentum at scale: set `momentum=0.9978` from 0.9999 (32 times) when you `update_period=32`.

Also, to make model EMA really work (improve generalization), one should carefully tune the momentum based on various factors, e.g. the learning rate scheduler, the total batch size, the training epochs, e.t.c.

To initialize a momentum in Pycls style, one set `model_ema.alpha = 1e-5` instead. Momentum will be calculated through `_calculate_pycls_momentum`.

basecls.engine.build

class basecls.engine.build.**DefaultHooks**

基类：object

The default hooks factory.

It combines `LRSchedulerHook` -> `PreciseBNHook` -> `ResumeHook` -> `TensorboardHook` -> `LoggerHook` -> `CheckpointHook` -> `EvalHook`.

classmethod build(*cfg*)

Build function with a simple strategy.

参数

cfg (ConfigDict) –config for setting hooks.

返回类型

List[BaseHook]

返回

A hook list.

basecls.engine.evaluator

class basecls.engine.evaluator.**AccEvaluator**

基类：BaseEvaluator

Classification evaluator with top-1 and top-5 accuracy.

ResultType

Tuple[int, float, float] 的别名

preprocess(*input_data*)

Preprocess input data per batch.

参数

input_data (Sequence[ndarray]) –input data.

返回类型

Tensor

返回

Preprocessed input data.

postprocess(*model_outputs*, *input_data*)

Postprocess model outputs with input data per batch.

参数

- **model_outputs** (Tensor) –model outputs.
- **input_data** (Sequence[ndarray]) –input data.

返回类型

Tuple[int, float, float]

返回

A tuple that (batch size, top-1 accuracy per batch, top-5 accuracy per batch).

evaluate(*results*)

Evaluation function.

参数

results (Iterable[Tuple[int, float, float]]) –all results.

basecls.engine.hooks

class basecls.engine.hooks.**CheckpointHook**(*save_dir=None*, *save_every_n_epoch=1*)

基类：BaseHook

Hook for managing checkpoints during training.

Effect during after_epoch and after_train procedure.

参数

- **save_dir** (Optional[str]) –checkpoint directory.
- **save_every_n_epoch** (int) –interval for saving checkpoint. Default: 1

after_epoch()

Called after each epoch.

after_train()

Called after training process.

class basecls.engine.hooks.**EvalHook**(*save_dir=None*, *eval_every_n_epoch=1*)

基类：BaseHook

Hook for evaluating during training.

Effect during after_epoch and after_train procedure.

参数

- **save_dir** (Optional[str]) –checkpoint directory.
- **eval_every_n_epoch** (int) –interval for evaluating. Default: 1

after_epoch()

Called after each epoch.

after_train()

Called after training process.

test(*cfg*, *model*, *ema=None*)

class basecls.engine.hooks.**LoggerHook**(*log_every_n_iter=20*)

基类：BaseHook

Hook for logging during training.

Effect during before_train, after_train, before_iter and after_iter procedure.

参数

log_every_n_iter (int) –interval for logging. Default: 20

before_train()

Called before training process.

after_train()

Called after training process.

before_iter()

Called before each iteration.

after_iter()

Called after each iteration.

get_loss_str(*meter*)

Get loss information during trainging process.

get_stat_str(*meter*)

Get stat information during trainging process.

get_memory_str(*meter*)

Get memory information during trainging process.

get_train_info_str()

Get training process related information such as learning rate.

get_time_str(*left_iters*)

Get time related information sucn as data_time, train_time, ETA and so on.

返回类型

str

class basecls.engine.hooks.**LRSchedulerHook**

基类：BaseHook

Hook for learning rate scheduling during training.

Effect during before_epoch procedure.

before_epoch()

Called before each epoch.

get_lr_factor(*cfg*, *epoch_id*)

Calculate learning rate factor.

It supports "step", "linear", "cosine", "exp", and "rel_exp" schedule.

参数

- **cfg** (ConfigDict) –config for training.
- **epoch_id** (int) –current epoch.

返回类型
float

返回
Learning rate factor.

total_lr

Total learning rate.

class basecls.engine.hooks.**PreciseBNHook**(*precise_every_n_epoch=1*)

基类：BaseHook

Hook for precising BN during training.

Effect during after_epoch procedure.

参数
precise_every_n_epoch (int) –interval for precising BN. Default: 1

before_train()

Called before training process.

after_epoch()

Called after each epoch.

class basecls.engine.hooks.**ResumeHook**(*save_dir=None*, *resume=False*)

基类：BaseHook

Hook for resuming training process.

Effect during before_train procedure.

参数
- **save_dir** (Optional[int]) –checkpoint directory.
- **resume** (bool) –enable resume or not. Default: False

before_train()

Called before training process.

class basecls.engine.hooks.**TensorboardHook**(*log_dir*, *log_every_n_iter=20*, *scalar_type='latest'*)

基类：BaseHook

Hook for tensorboard during training.

Effect during before_train, after_train and after_iter procedure.

参数
- **log_dir** (str) –tensorboard directory.
- **log_every_n_iter** (int) –interval for logging. Default: 20

- **scalar_type** (str) –statistic to record, supports "latest", "avg", "global_avg" and "median". Default: "latest"

before_train()

Called before training process.

after_train()

Called after training process.

after_iter()

Called after each iteration.

write(*context*)

classmethod calc_iter(*progress*)

basecls.engine.tester

class basecls.engine.tester.**ClsTester**(*cfg*, *model*, *dataloader*)

基类：BaseTester

test(*warm_iters=5*, *log_seconds=5*)

basecls.engine.trainer

class basecls.engine.trainer.**ClsTrainer**(*cfg*, *model*, *dataloader*, *solver*, *hooks=None*)

基类：BaseTrainer

Classification trainer.

参数

- **cfg** (ConfigDict) –config for training.
- **model** (Module) –model for training.
- **dataloader** (Union[DataLoader, *FakeDataLoader*]) –dataloader for training.
- **solver** (*Solver*) –solver for training.
- **hooks** (Optional[Iterable[BaseHook]]) –hooks for training.

cfg

config for training.

model

model for training.

ema

model exponential moving average.

dataloader

dataloader for training.

solver

solver for training.

progress

object for recording training process.

loss

loss function for training.

meter

object for recording metrics.

train()

参数

- **start_info** (*Iterable*) –[epoch, iter] for training start.
- **max_info** (*Iterable*) –[max_epoch, max_iter] for training.

before_train()

before_epoch()

after_epoch()

train_one_iter()

Basic logic of training one iteration.

model_step(*samples*, *targets*)

modify_grad()

model_ema_step()

Implement momentum based Exponential Moving Average (EMA) for model states https://github.com/rwightman/pytorch-image-models/blob/master/timm/utils/model_ema.py

Also inspired by Pycls https://github.com/facebookresearch/pycls/pull/138/, which is more flexible and efficient

Heuristically, one can use a momentum of 0.9999 as used by Tensorflow and 0.9998 as used by timm, which updates model ema every iter. To be more efficient, one can set `update_period` to e.g. 8 or 32 to speed up your training, and decrease your momentum at scale: set `momentum=0.9978` from 0.9999 (32 times) when you `update_period=32`.

Also, to make model EMA really work (improve generalization), one should carefully tune the momentum based on various factors, e.g. the learning rate scheduler, the total batch size, the training epochs, e.t.c.

To initialize a momentum in Pycls style, one set `model_ema.alpha = 1e-5` instead. Momentum will be calculated through `_calculate_pycls_momentum`.

1.3.4 basecls.layers

class basecls.layers.**ELU**(*alpha=1.0*, *name=None*)

基类：Module

ELU activation function.

$$\mathrm{ELU}(x) = \begin{cases} x, & \text{if } x > 0, \\ \alpha\left(\exp(x) - 1\right), & \text{if } x \leq 0 \end{cases}$$

参数

alpha (float) –the α value for the ELU formulation. Default: 1.0

forward(x)

返回类型

Tensor

class basecls.layers.**HSigmoid**(*name=None*)

基类：Module

Hard sigmoid activation function.

$$\mathrm{HSigmoid}(x) = \begin{cases} 0 & \text{if } x \leq -3, \\ 1 & \text{if } x \geq 3, \\ x/6 + 1/2 & \text{otherwise} \end{cases}$$

forward(x)

返回类型

Tensor

class basecls.layers.**HSwish**(*name=None*)

基类：Module

Hard swish activation function.

$$\mathrm{HSwish}(x) = \begin{cases} 0 & \text{if } x \leq -3, \\ x & \text{if } x \geq 3, \\ x(x+3)/6 & \text{otherwise} \end{cases}$$

forward(*x*)

返回类型

Tensor

class basecls.layers.**ReLU6**(*name=None*)

基类：Module

ReLU6 activation function.

$$\text{ReLU6}(x) = \min\left(\max(0, x), 6\right)$$

forward(*x*)

返回类型

Tensor

class basecls.layers.**Tanh**(*name=None*)

基类：Module

Tanh activation function.

$$\text{Tanh}(x) = \tanh(x) = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)}$$

forward(*x*)

返回类型

Tensor

basecls.layers.**activation**(*name*, ***kwargs*)

Helper for building an activation layer.

参数

name (Union[str, Callable]) –activation name, supports "elu", "gelu", "hsigmoid", "hswish", "leaky_relu", "relu", "relu6", "prelu", "silu" and "tanh".

返回类型

Module

返回

An activation module.

class basecls.layers.**ClsHead**(*w_in*, *w_out=1000*, *width=0*, *dropout_prob=0.0*, *norm_name='BN'*, *act_name='relu'*, *bias=True*)

基类：Module

Cls head: Conv, BN, Act, AvgPool, FC.

参数

- **w_in** (int) –input width.
- **w_out** (int) –output width, normally the number of classes. Default: 1000
- **width** (int) –width for first conv in head, conv will be omitted if set to 0. Default: 0
- **dropout_prob** (float) –dropout probability. Default: 0.0
- **norm_name** (str) –normalization function. Default: "BN"
- **act_name** (str) –activation function. Default: "relu"
- **bias** (bool) –whether fc has bias. Default: True

forward(*x*)

返回类型

Tensor

class basecls.layers.**MBV3Head**(*w_in*, *w_out=1000*, *width=960*, *w_h=1280*, *dropout_prob=0.0*, *se_r=0.0*, *norm_name='BN'*, *act_name='hswish'*, *bias=True*)

基类：Module

MobileNet V3 head: Conv, BN, Act, AvgPool, SE, FC, Act, FC.

参数

- **w_in** (int) –input width.
- **w_out** (int) –output width, normally the number of classes.
- **width** (int) –width for first conv in head.
- **w_h** (int) –width for first linear in head.
- **dropout_prob** (float) –dropout probability. Default: 0.0
- **se_r** (float) –Squeeze-and-Excitation (SE) ratio. Default: 0.0
- **norm_name** (str) –normalization function. Default: "BN"
- **act_name** (str) –activation function. Default: "hswish"
- **bias** (bool) –whether fc has bias. Default: True

forward(*x*)

返回类型

Tensor

class basecls.layers.**VGGHead**(*w_in*, *w_out=1000*, *width=4096*, *dropout_prob=0.5*, *act_name='relu'*, ***kwargs*)

基类：Module

VGG head: AvgPool, [FC, Act, Dropout] x2, FC.

参数

- **w_in** (int) –input width.
- **w_out** (int) –output width, normally the number of classes. Default: 1000
- **width** (int) –width for linear in head. Default: 4096
- **dropout_prob** (float) –dropout probability. Default: 0.5
- **act_name** (str) –activation function. Default: "relu"

forward(*x*)

返回类型
Tensor

basecls.layers.**build_head**(*w_in*, *head_args=None*, *norm_name='BN'*, *act_name='relu'*)

The factory function to build head.

备注: if head_args is None or head_args["name"] is None, this function will do nothing and return None.

参数

- **w_in** (int) –input width.
- **head_args** (Optional[Mapping[str, Any]]) –head args. Default: None
- **norm_name** (str) –default normalization function, will be overridden by the same key in head_args. Default: "BN"
- **act_name** (str) –default activation function, will be overridden by the same key in head_args. Default: "relu"

返回类型
Module

返回
A head.

class basecls.layers.**BinaryCrossEntropy**(***kwargs*)

基类：Module

The module for binary cross entropy.

See binary_cross_entropy() for more details.

forward(*x*, *y*)

返回类型

Tensor

class basecls.layers.**CrossEntropy**(*axis=1*, *label_smooth=0.0*)

基类：Module

The module for cross entropy.

It supports both categorical labels and one-hot labels. See `cross_entropy()` for more details.

参数

- **axis** (int) –reduced axis. Default: 1
- **label_smooth** (float) –label smooth factor. Default: 0.0

forward(*x*, *y*)

返回类型

Tensor

basecls.layers.**build_loss**(*cfg*)

The factory function to build loss.

参数

cfg (ConfigDict) –config for building loss function.

返回类型

Module

返回

A loss function.

class basecls.layers.**SE**(*w_in*, *w_se*, *act_name*, *approx_sigmoid=False*)

基类：Module

Squeeze-and-Excitation (SE) block: AvgPool, FC, Act, FC, Sigmoid.

参数

- **w_in** (int) –input width.
- **w_se** (int) –se width.
- **act_name** (str) –activation name.
- **approx_sigmoid** (bool) –approximated sigmoid function.

avg_pool

gad2d layer.

f_ex

sequantial which conbines conv2d -> act -> conv2d -> sigmoid.

forward(*x*)

返回类型

Tensor

class basecls.layers.**DropPath**(*drop_prob=0.0*, ***kwargs*)

基类：Dropout

DropPath block.

参数

drop_prob –the probability to drop (set to zero) each path.

forward(*x*)

basecls.layers.**conv2d**(*w_in*, *w_out*, *k*, ***, *stride=1*, *dilation=1*, *groups=1*, *bias=False*)

Helper for building a conv2d layer.

It will calculate padding automatically.

参数

- **w_in** (int) –input width.
- **w_out** (int) –output width.
- **k** (int) –kernel size.
- **stride** (int) –stride. Default: 1
- **dilation** (int) –dilation. Default: 1
- **groups** (int) –groups. Default: 1
- **bias** (bool) –enable bias or not. Default: False

返回类型

Conv2d

返回

A conv2d module.

basecls.layers.**gap2d**(*shape=1*)

Helper for building a gap2d layer.

参数

shape –output shape. Default: 1

返回类型

AdaptiveAvgPool2d

返回

A gap2d module.

basecls.layers.**linear**(*w_in*, *w_out*, *, *bias=False*)

Helper for building a linear layer.

参数

- **w_in** (int) –input width.
- **w_out** (int) –output width.
- **bias** (bool) –enable bias or not. Default: False

返回类型

Linear

返回

A linear module.

basecls.layers.**norm2d**(*name*, *w_in*, ***kwargs*)

Helper for building a norm2d layer.

参数

- **norm_name** –normalization name, supports None, "BN", "GN", "IN", "LN" and "SyncBN".
- **w_in** (int) –input width.

返回类型

Module

返回

A norm2d module.

basecls.layers.**pool2d**(*k*, *, *stride=1*, *name='max'*)

Helper for building a pool2d layer.

参数

- **k** (int) –kernel size.
- **stride** (int) –stride. Default: 1
- **name** (str) –pooling name, supports "avg" and "max".

返回类型

Module

返回

A pool2d module.

class basecls.layers.**Preprocess**(*mean*, *std*)

基类：Module

forward(*inputs*)

返回类型

Tuple[Tensor, Tensor]

basecls.layers.**adjust_block_compatibility**(*ws*, *bs*, *gs*)

Adjusts the compatibility of widths, bottlenecks and groups.

参数

- **ws** (Sequence[int]) –widths.
- **bs** (Sequence[float]) –bottleneck multipliers.
- **gs** (Sequence[int]) –group widths.

返回类型

Tuple[List[int], ...]

返回

The adjusted widths, bottlenecks and groups.

basecls.layers.**compute_precise_bn_stats**(*cfg*, *model*, *dataloader*)

Computes precise BN stats on training data.

References: https://github.com/facebookresearch/pycls/blob/main/pycls/core/net.py

参数

- **cfg** (ConfigDict) –config for precising BN.
- **model** (Module) –model for precising BN.
- **dataloader** (Union[DataLoader, *FakeDataLoader*]) –dataloader for precising BN.

basecls.layers.**init_vit_weights**(*module*)

Initialization for Vision Transformer (ViT).

References: https://github.com/rwightman/pytorch-image-models/blob/master/timm/models/vision_transformer.py

参数

m –module to be initialized.

basecls.layers.**init_weights**(*m*, *pytorch_style=False*, *zero_init_final_gamma=False*)

Performs ResNet-style weight initialization.

About zero-initialize: Zero-initialize the last BN in each residual branch, so that the residual branch starts with zeros, and each residual block behaves like an identity. This improves the model by 0.2~0.3% according to https://arxiv.org/abs/1706.02677.

References: https://github.com/facebookresearch/pycls/blob/main/pycls/models/blocks.py

参数

- **m** (Module) –module to be initialized.
- **pytorch_style** (bool) –utilize pytorch style init for group conv. Default: False
- **zero_init_final_gamma** (bool) –enable zero-initialize or not. Default: False

basecls.layers.**lecun_normal_**(*tensor*)

basecls.layers.**make_divisible**(*value*, *divisor=8*, *min_value=None*, *round_limit=0.0*)

basecls.layers.**trunc_normal_**(*tensor*, *mean=0.0*, *std=1.0*, *a=-2.0*, *b=2.0*)

basecls.layers.activations

basecls.layers.activations.**activation**(*name*, ***kwargs*)

Helper for building an activation layer.

参数

name (Union[str, Callable]) –activation name, supports "elu", "gelu", "hsigmoid", "hswish", "leaky_relu", "relu", "relu6", "prelu", "silu" and "tanh".

返回类型

Module

返回

An activation module.

class basecls.layers.activations.**ELU**(*alpha=1.0*, *name=None*)

基类：Module

ELU activation function.

$$\mathrm{ELU}(x) = \begin{cases} x, & \text{if } x > 0, \\ \alpha\left(\exp(x) - 1\right), & \text{if } x \leq 0 \end{cases}$$

参数

alpha (float) –the α value for the ELU formulation. Default: 1.0

forward(*x*)

返回类型

Tensor

class basecls.layers.activations.**HSigmoid**(*name=None*)

基类：Module

Hard sigmoid activation function.

$$\text{HSigmoid}(x) = \begin{cases} 0 & \text{if } x \leq -3, \\ 1 & \text{if } x \geq 3, \\ x/6 + 1/2 & \text{otherwise} \end{cases}$$

forward(*x*)

返回类型

Tensor

class basecls.layers.activations.**HSwish**(*name=None*)

基类：Module

Hard swish activation function.

$$\text{HSwish}(x) = \begin{cases} 0 & \text{if } x \leq -3, \\ x & \text{if } x \geq 3, \\ x(x+3)/6 & \text{otherwise} \end{cases}$$

forward(*x*)

返回类型

Tensor

class basecls.layers.activations.**ReLU6**(*name=None*)

基类：Module

ReLU6 activation function.

$$\text{ReLU6}(x) = \min\left(\max(0, x), 6\right)$$

forward(*x*)

返回类型

Tensor

class basecls.layers.activations.**Tanh**(*name=None*)

基类：Module

Tanh activation function.

$$\text{Tanh}(x) = \tanh(x) = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)}$$

forward(*x*)

返回类型
Tensor

basecls.layers.heads

basecls.layers.heads.**build_head**(*w_in*, *head_args=None*, *norm_name='BN'*, *act_name='relu'*)

The factory function to build head.

备注： if `head_args` is `None` or `head_args["name"]` is `None`, this function will do nothing and return `None`.

参数

- **w_in** (`int`) –input width.
- **head_args** (`Optional[Mapping[str, Any]]`) –head args. Default: `None`
- **norm_name** (`str`) –default normalization function, will be overridden by the same key in `head_args`. Default: `"BN"`
- **act_name** (`str`) –default activation function, will be overridden by the same key in `head_args`. Default: `"relu"`

返回类型
Module

返回
A head.

class basecls.layers.heads.**ClsHead**(*w_in*, *w_out=1000*, *width=0*, *dropout_prob=0.0*, *norm_name='BN'*, *act_name='relu'*, *bias=True*)

基类：Module

Cls head: Conv, BN, Act, AvgPool, FC.

参数

- **w_in** (`int`) –input width.
- **w_out** (`int`) –output width, normally the number of classes. Default: `1000`
- **width** (`int`) –width for first conv in head, conv will be omitted if set to 0. Default: `0`
- **dropout_prob** (`float`) –dropout probability. Default: `0.0`
- **norm_name** (`str`) –normalization function. Default: `"BN"`

- **act_name** (str) –activation function. Default: "relu"
- **bias** (bool) –whether fc has bias. Default: True

forward(*x*)

返回类型
Tensor

class basecls.layers.heads.**MBV3Head**(*w_in*, *w_out=1000*, *width=960*, *w_h=1280*, *dropout_prob=0.0*, *se_r=0.0*, *norm_name='BN'*, *act_name='hswish'*, *bias=True*)

基类: Module

MobileNet V3 head: Conv, BN, Act, AvgPool, SE, FC, Act, FC.

参数

- **w_in** (int) –input width.
- **w_out** (int) –output width, normally the number of classes.
- **width** (int) –width for first conv in head.
- **w_h** (int) –width for first linear in head.
- **dropout_prob** (float) –dropout probability. Default: 0.0
- **se_r** (float) –Squeeze-and-Excitation (SE) ratio. Default: 0.0
- **norm_name** (str) –normalization function. Default: "BN"
- **act_name** (str) –activation function. Default: "hswish"
- **bias** (bool) –whether fc has bias. Default: True

forward(*x*)

返回类型
Tensor

class basecls.layers.heads.**VGGHead**(*w_in*, *w_out=1000*, *width=4096*, *dropout_prob=0.5*, *act_name='relu'*, ***kwargs*)

基类: Module

VGG head: AvgPool, [FC, Act, Dropout] x2, FC.

参数

- **w_in** (int) –input width.
- **w_out** (int) –output width, normally the number of classes. Default: 1000
- **width** (int) –width for linear in head. Default: 4096
- **dropout_prob** (float) –dropout probability. Default: 0.5

- **act_name** (str) –activation function. Default: "relu"

forward(*x*)

返回类型

Tensor

basecls.layers.losses

basecls.layers.losses.**build_loss**(*cfg*)

The factory function to build loss.

参数

cfg (ConfigDict) –config for building loss function.

返回类型

Module

返回

A loss function.

class basecls.layers.losses.**BinaryCrossEntropy**(***kwargs*)

基类：Module

The module for binary cross entropy.

See binary_cross_entropy() for more details.

forward(*x*, *y*)

返回类型

Tensor

class basecls.layers.losses.**CrossEntropy**(*axis=1*, *label_smooth=0.0*)

基类：Module

The module for cross entropy.

It supports both categorical labels and one-hot labels. See cross_entropy() for more details.

参数

- **axis** (int) –reduced axis. Default: 1
- **label_smooth** (float) –label smooth factor. Default: 0.0

forward(*x*, *y*)

返回类型

Tensor

basecls.layers.modules

basecls.layers.modules.**conv2d**(*w_in*, *w_out*, *k*, ***, *stride=1*, *dilation=1*, *groups=1*, *bias=False*)

Helper for building a conv2d layer.

It will calculate padding automatically.

参数

- **w_in** (int) –input width.
- **w_out** (int) –output width.
- **k** (int) –kernel size.
- **stride** (int) –stride. Default: 1
- **dilation** (int) –dilation. Default: 1
- **groups** (int) –groups. Default: 1
- **bias** (bool) –enable bias or not. Default: False

返回类型

Conv2d

返回

A conv2d module.

basecls.layers.modules.**norm2d**(*name*, *w_in*, ***kwargs*)

Helper for building a norm2d layer.

参数

- **norm_name** –normalization name, supports None, "BN", "GN", "IN", "LN" and "SyncBN".
- **w_in** (int) –input width.

返回类型

Module

返回

A norm2d module.

basecls.layers.modules.**pool2d**(*k*, ***, *stride=1*, *name='max'*)

Helper for building a pool2d layer.

参数

- **k** (int) –kernel size.
- **stride** (int) –stride. Default: 1
- **name** (str) –pooling name, supports "avg" and "max".

返回类型

Module

返回

A pool2d module.

basecls.layers.modules.**gap2d**(*shape=1*)

Helper for building a gap2d layer.

参数

shape –output shape. Default: 1

返回类型

AdaptiveAvgPool2d

返回

A gap2d module.

basecls.layers.modules.**linear**(*w_in*, *w_out*, ***, *bias=False*)

Helper for building a linear layer.

参数

- **w_in** (int) –input width.
- **w_out** (int) –output width.
- **bias** (bool) –enable bias or not. Default: False

返回类型

Linear

返回

A linear module.

class basecls.layers.modules.**SE**(*w_in*, *w_se*, *act_name*, *approx_sigmoid=False*)

基类：Module

Squeeze-and-Excitation (SE) block: AvgPool, FC, Act, FC, Sigmoid.

参数

- **w_in** (int) –input width.
- **w_se** (int) –se width.
- **act_name** (str) –activation name.
- **approx_sigmoid** (bool) –approximated sigmoid function.

avg_pool

gad2d layer.

f_ex

sequantial which conbines conv2d -> act -> conv2d -> sigmoid.

forward(*x*)

返回类型

Tensor

class basecls.layers.modules.**DropPath**(*drop_prob=0.0*, ***kwargs*)

基类：Dropout

DropPath block.

参数

drop_prob –the probability to drop (set to zero) each path.

forward(*x*)

basecls.layers.wrapper

class basecls.layers.wrapper.**Preprocess**(*mean*, *std*)

基类：Module

forward(*inputs*)

返回类型

Tuple[Tensor, Tensor]

basecls.layers.wrapper.**adjust_block_compatibility**(*ws*, *bs*, *gs*)

Adjusts the compatibility of widths, bottlenecks and groups.

参数

- **ws** (Sequence[int]) –widths.
- **bs** (Sequence[float]) –bottleneck multipliers.
- **gs** (Sequence[int]) –group widths.

返回类型

Tuple[List[int], ...]

返回

The adjusted widths, bottlenecks and groups.

basecls.layers.wrapper.**calculate_fan_in_and_fan_out**(*tensor*, *pytorch_style=False*)

Fixed megengine.module.init.calculate_fan_in_and_fan_out() for group conv2d.

备注： The group conv2d kernel shape in MegEngine is `(G, O/G, I/G, K, K)`. This function calculates `fan_out = O/G * K * K` as default, but PyTorch uses `fan_out = O * K * K`.

参数

- **tensor** (`Tensor`) –tensor to be initialized.
- **pytorch_style** (`bool`) –utilize pytorch style init for group conv. Default: `False`

basecls.layers.wrapper.**compute_precise_bn_stats**(*cfg*, *model*, *dataloader*)

Computes precise BN stats on training data.

References: https://github.com/facebookresearch/pycls/blob/main/pycls/core/net.py

参数

- **cfg** (`ConfigDict`) –config for precising BN.
- **model** (`Module`) –model for precising BN.
- **dataloader** (`Union[DataLoader, FakeDataLoader]`) –dataloader for precising BN.

basecls.layers.wrapper.**init_weights**(*m*, *pytorch_style=False*, *zero_init_final_gamma=False*)

Performs ResNet-style weight initialization.

About zero-initialize: Zero-initialize the last BN in each residual branch, so that the residual branch starts with zeros, and each residual block behaves like an identity. This improves the model by 0.2~0.3% according to https://arxiv.org/abs/1706.02677.

References: https://github.com/facebookresearch/pycls/blob/main/pycls/models/blocks.py

参数

- **m** (`Module`) –module to be initialized.
- **pytorch_style** (`bool`) –utilize pytorch style init for group conv. Default: `False`
- **zero_init_final_gamma** (`bool`) –enable zero-initialize or not. Default: `False`

basecls.layers.wrapper.**init_vit_weights**(*module*)

Initialization for Vision Transformer (ViT).

References: https://github.com/rwightman/pytorch-image-models/blob/master/timm/models/vision_transformer.py

参数

m –module to be initialized.

basecls.layers.wrapper.**trunc_normal_**(*tensor*, *mean=0.0*, *std=1.0*, *a=-2.0*, *b=2.0*)

basecls.layers.wrapper.**lecun_normal_**(*tensor*)

basecls.layers.wrapper.**make_divisible**(*value*, *divisor=8*, *min_value=None*, *round_limit=0.0*)

1.3.5 basecls.models

basecls.models.**build_model**(*cfg*)

The factory function to build model.

备注： if `cfg.model` does not have the attr `head`, this function will build model with the default head. Otherwise if `cfg.model.head` is `None`, this function will build model without any head.

备注： if `cfg.model.head` does not have the attr `w_out` and `cfg.num_classes` exists, `w_out` will be overridden by `cfg.num_classes`.

参数

cfg (`ConfigDict`) –config for building model.

返回类型

`Module`

返回

A model.

basecls.models.**load_model**(*model*, *weight_path*, *strict=True*)

Load model weights.

参数

- **model** (`Module`) –model for loading weights.
- **weight_path** (`str`) –weight path, both local path and OSS path are supported.
- **strict** (`bool`) –load weights in strict mode or not. Default: `True`

basecls.models.**sync_model**(*model*)

Sync parameters and buffers.

参数

model (`Module`) –model for syncing.

class basecls.models.**EffNet**(*stem_w*, *block_name*, *depths*, *widths*, *strides*, *kernels*, *exp_rs=1.0*, *se_rs=0.0*, *drop_path_prob=0.0*, *depth_mult=1.0*, *width_mult=1.0*, *omit_mult=False*, *norm_name='BN'*, *act_name='silu'*, *head=None*)

基类：Module

EfficientNet model.

参数

- **stem_w** (int) –stem width.
- **block_name** (Union[str, Callable, Sequence[Union[str, Callable]]]) –block name.
- **depths** (Sequence[int]) –depth for each stage (number of blocks in the stage).
- **widths** (Sequence[int]) –width for each stage (width of each block in the stage).
- **strides** (Sequence[int]) –strides for each stage (applies to the first block of each stage).
- **kernels** (Sequence[int]) –kernel sizes for each stage.
- **exp_rs** (Union[float, Sequence[Union[float, Sequence[float]]]]) –expansion ratios for MBConv blocks in each stage.
- **se_r** –Squeeze-and-Excitation (SE) ratio. Default: 0.25
- **drop_path_prob** (float) –drop path probability. Default: 0.0
- **depth_mult** (float) –depth multiplier. Default: 1.0
- **width_mult** (float) –width multiplier. Default: 1.0
- **omit_mult** (bool) –omit multiplier for stem width, head width, the first stage depth and the last stage depth, enabled in EfficientNet-Lite. Default: False
- **norm_name** (str) –normalization function. Default: "BN"
- **act_name** (str) –activation function. Default: "silu"
- **head** (Optional[Mapping[str, Any]]) –head args. Default: None

forward(*x*)

static get_block_func(*name*)

Retrieves the block function by name.

class basecls.models.**HRNet**(*stage_modules*, *stage_blocks*, *stage_block_names*, *stage_channels*, *w_stem=64*, *multi_scale_output=True*, *merge_block_name='bottleneck'*, *merge_channels=[32, 64, 128, 256]*, *norm_name='BN'*, *act_name='relu'*, *head=None*, ***kwargs*)

基类：Module

HRNet model.

参数

- **stage_modules** (List[int]) –Number of modules for each stage.
- **stage_blocks** (List[List[int]]) –Number of blocks for each module in stages.
- **stage_block_names** (List[str]) –Branch block types for each stage.
- **stage_channels** (List[List[int]]) –Number of channels for each stage.
- **w_stem** (int) –Stem width. Default: 64
- **multi_scale_output** (bool) –Whether output multi-scale features. Default: True
- **merge_block_name** (str) –Merge block type. Default: "bottleneck"
- **merge_channels** (List[int]) –Channels of each scale in merge block. Default: [32, 64, 128, 256]
- **norm_name** (str) –Normalization layer. Default: "BN"
- **act_name** (str) –Activation function. Default: "relu"
- **head** (Optional[Mapping[str, Any]]) –head args. Default: None

forward(*x*)

class basecls.models.**MBNet**(*stem_w*, *depths*, *widths*, *strides*, *kernels*, *exp_rs=1.0*, *se_rs=0.0*, *stage_act_names=None*, *has_proj_act=False*, *has_skip=True*, *drop_path_prob=0.0*, *width_mult=1.0*, *norm_name='BN'*, *act_name='relu6'*, *head=None*)

基类：Module

MobileNet model.

参数

- **stem_w** (int) –stem width.
- **depths** (Sequence[int]) –depth for each stage (number of blocks in the stage).
- **widths** (Sequence[int]) –width for each stage (width of each block in the stage).
- **strides** (Sequence[int]) –strides for each stage (applies to the first block of each stage).
- **kernels** (Sequence[int]) –kernel sizes for each stage.
- **exp_rs** (Union[float, Sequence[Union[float, Sequence[float]]]]) –expansion ratios for MobileNet basic blocks in each stage. Default: 1.0
- **se_rs** (Union[float, Sequence[Union[float, Sequence[float]]]]) –Squeeze-and-Excitation (SE) ratios. Default: 0.0
- **stage_act_names** (Optional[Sequence[str]]) –activation function for stages. Default: None
- **has_proj_act** (bool) –whether apply activation to output. Default: False

- **has_skip** (bool) –whether apply skip connection. Default: True
- **drop_path_prob** (float) –drop path probability. Default: 0.0
- **width_mult** (float) –width multiplier. Default: 1.0
- **norm_name** (str) –normalization function. Default: "BN"
- **act_name** (str) –activation function. Default: "relu6"
- **head** (Optional[Mapping[str, Any]]) –head args. Default: None

forward(*x*)

class basecls.models.**RegNet**(*stem_name*, *stem_w*, *block_name*, *depth*, *w0*, *wa*, *wm*, *group_w*, *stride=2*, *bot_mul=1.0*, *se_r=0.0*, *drop_path_prob=0.0*, *zero_init_final_gamma=False*, *norm_name='BN'*, *act_name='relu'*, *head=None*)

基类：*ResNet*

RegNet model.

参数

- **stem_name** (Union[str, Callable]) –stem name.
- **stem_w** (int) –stem width.
- **block_name** (Union[str, Callable]) –block name.
- **depth** (int) –depth.
- **w0** (int) –initial width.
- **wa** (float) –slope.
- **wm** (float) –quantization.
- **group_w** (int) –group width for each stage (applies to bottleneck block).
- **stride** (int) –stride for each stage (applies to the first block of each stage). Default: 2
- **bot_mul** (float) –bottleneck multiplier for each stage (applies to bottleneck block). Default: 1.0
- **se_r** (float) –Squeeze-and-Excitation (SE) ratio. Default: 0.0
- **drop_path_prob** (float) –drop path probability. Default: 0.0
- **zero_init_final_gamma** (bool) –enable zero-initialize or not. Default: False
- **norm_name** (str) –normalization function. Default: "BN"
- **act_name** (str) –activation function. Default: "relu"
- **head** (Optional[Mapping[str, Any]]) –head args. Default: None

class basecls.models.**RepVGG**(*num_blocks*, *width_multiplier*, *head=None*, *groups=1*, *se_r=0.0*, *act_name='relu'*, *deploy=False*)

基类：Module

RepVGG Model.

Use *RepVGG.convert_to_deploy()* to convert a training *RepVGG* to deploy:

```
model = RepVGG(..., deploy=False)
model.load_state_dict(...)
_ = RepVGG.convert_to_deploy(model)
```

参数

- **num_blocks** (Sequence[int]) –RepVGG depths.
- **width_multiplier** (Sequence[int]) –RepVGG widths, base_width is [64, 128, 256, 512].
- **head** (Optional[Mapping[str, Any]]) –head args. Default: None
- **groups** (Union[int, List[Union[int, List[int]]]]) –number of groups for blocks. Default: 1
- **se_r** (float) –Squeeze-and-Excitation (SE) ratio. Default: 0.0
- **act_name** (str) –activation function. Default: "relu"
- **deploy** (bool) –switch a reparamed RepVGG into deploy mode. Default: False

forward(*x*)

classmethod convert_to_deploy(*module*)

返回类型

Module

class basecls.models.**ResMLP**(*img_size=224*, *patch_size=16*, *in_chans=3*, *embed_dim=768*, *depth=12*, *drop_rate=0.0*, *drop_path_rate=0.0*, *embed_layer=PatchEmbed*, *init_scale=1e-4*, *ffn_ratio=4.0*, *act_name='gelu'*, *num_classes=1000*, ***kwargs*)

基类：Module

ResMLP model.

参数

- **img_size** (int) –Input image size. Default: 224
- **patch_size** (int) –Patch token size. Default: 16
- **in_chans** (int) –Number of input image channels. Default: 3

- **embed_dim** (int) –Number of linear projection output channels. Default: 768
- **depth** (int) –Depth of Transformer Encoder layer. Default: 12
- **drop_rate** (float) –Dropout rate. Default: 0.0
- **drop_path_rate** (float) –Stochastic depth rate. Default: 0.0
- **embed_layer** (Module) –Patch embedding layer. Default: PatchEmbed
- **init_scale** (float) –Initial value for LayerScale. Default: 1e-4
- **ffn_ratio** (float) –Ratio of ffn hidden dim to embedding dim. Default: 4.0
- **act_name** (str) –Activation function. Default: "gelu"
- **num_classes** (int) –Number of classes. Default: 1000

forward(*x*)

class basecls.models.**ResNet**(*stem_name*, *stem_w*, *block_name*, *depths*, *widths*, *strides*, *bot_muls=1.0*, *group_ws=None*, *se_r=0.0*, *avg_down=False*, *drop_path_prob=0.0*, *zero_init_final_gamma=False*, *norm_name='BN'*, *act_name='relu'*, *head=None*)

基类：Module

ResNet model.

参数

- **stem_name** (Union[str, Callable]) –stem name.
- **stem_w** (int) –stem width.
- **block_name** (Union[str, Callable]) –block name.
- **depths** (Sequence[int]) –depth for each stage (number of blocks in the stage).
- **widths** (Sequence[int]) –width for each stage (width of each block in the stage).
- **strides** (Sequence[int]) –strides for each stage (applies to the first block of each stage).
- **bot_muls** (Union[float, Sequence[float]]) –bottleneck multipliers for each stage (applies to bottleneck block). Default: 1.0
- **group_ws** (Optional[Sequence[int]]) –group widths for each stage (applies to bottleneck block). Default: None
- **se_r** (float) –Squeeze-and-Excitation (SE) ratio. Default: 0.0
- **drop_path_prob** (float) –drop path probability. Default: 0.0
- **zero_init_final_gamma** (bool) –enable zero-initialize or not. Default: False
- **norm_name** (str) –normalization function. Default: "BN"
- **act_name** (str) –activation function. Default: "relu"

- **head** (Optional[Mapping[str, Any]]) –head args. Default: None

forward(*x*)

static get_stem_func(*name*)

Retrieves the stem function by name.

static get_block_func(*name*)

Retrieves the block function by name.

class basecls.models.**SNetV2**(*block*, *stem_w*, *depths*, *widths*, *strides*, *kernels*, *use_maxpool=True*, *se_r=0.0*, *drop_path_prob=0.0*, *norm_name='BN'*, *act_name='relu'*, *head=None*)

基类：Module

ShufflenetV2 model.

参数

- **block** (Callable) –building block to use, SNV2XceptionBlock for v2+.
- **stem_w** (int) –width for stem layer.
- **depths** (Sequence[int]) –depth for each stage (number of blocks in the stage).
- **widths** (Sequence[int]) –width for each stage (width of each block in the stage).
- **strides** (Sequence[int]) –strides for each stage (applies to the first block of each stage).
- **kernels** (Sequence[int]) –kernel sizes for each stage.
- **use_maxpool** (bool) –whether use maxpool stride 2 after stem. Default: True
- **se_r** (float) –Squeeze-and-Excitation (SE) ratio. Default: 0.0
- **drop_path_prob** (float) –drop path probability. Default: 0.0
- **norm_name** (str) –normalization function. Default: "BN"
- **act_name** (str) –activation function. Default: "relu6"
- **head** (Optional[Mapping[str, Any]]) –head args. Default: None

forward(*x*)

class basecls.models.**SwinTransformer**(*img_size=224*, *patch_size=4*, *in_chans=3*, *embed_dim=96*, *depths=[2, 2, 6, 2]*, *num_heads=[3, 6, 12, 24]*, *window_size=7*, *ffn_ratio=4.0*, *qkv_bias=True*, *qk_scale=None*, *ape=False*, *patch_norm=True*, *drop_rate=0.0*, *attn_drop_rate=0.0*, *drop_path_rate=0.1*, *embed_layer=PatchEmbed*, *norm_name='LN'*, *act_name='gelu'*, *num_classes=1000*, ***kwargs*)

基类：Module

Swin Transformer

A PyTorch impl of :

Swin Transformer: Hierarchical Vision Transformer using Shifted Windows - https://arxiv.org/pdf/2103.14030

参数

- **img_size** (int) –Input image size. Default: 224
- **patch_size** (int) –Patch size. Default: 4
- **in_chans** (int) –Number of input image channels. Default: 3
- **embed_dim** (int) –Patch embedding dimension. Default: 96
- **depths** (Sequence[int]) –Depth of each Swin Transformer layer.
- **num_heads** (Sequence[int]) –Number of attention heads in different layers.
- **window_size** (int) –Window size. Default: 7
- **ffn_ratio** (float) –Ratio of ffn hidden dim to embedding dim. Default: 4.0
- **qkv_bias** (bool) –If True, add a learnable bias to query, key, value. Default: True
- **qk_scale** (Optional[float]) –Override default qk scale of head_dim ** -0.5 if set. Default: None
- **ape** (bool) –If True, add absolute position embedding to the patch embedding. Default: False
- **patch_norm** (bool) –If True, add normalization after patch embedding. Default: True
- **drop_rate** (float) –Dropout rate. Default: 0
- **attn_drop_rate** (float) –Attention dropout rate. Default: 0
- **drop_path_rate** (float) –Stochastic depth rate. Default: 0.1
- **embed_layer** (Module) –Patch embedding layer. Default: PatchEmbed
- **norm_name** (str) –Normalization layer. Default: "LN"
- **act_name** (str) –Activation layer. Default: "gelu"
- **num_classes** (int) –Number of classes for classification head. Default: 1000

forward(*x*)

class basecls.models.**VGG**(*depths*, *widths*, *norm_name=None*, *act_name='relu'*, *head=None*)

基类：Module

VGG model.

参数

- **depths** (Sequence[int]) –depth for each stage (number of blocks in the stage).
- **widths** (Sequence[int]) –width for each stage (width of each block in the stage).
- **norm_name** (Optional[str]) –normalization function. Default: None
- **act_name** (str) –activation function. Default: "relu"
- **head** (Optional[Mapping[str, Any]]) –head args. Default: None

forward(*x*)

返回类型

Tensor

class basecls.models.**ViT**(*img_size=224*, *patch_size=16*, *in_chans=3*, *embed_dim=768*, *depth=12*, *num_heads=12*, *ffn_ratio=4.0*, *qkv_bias=True*, *qk_scale=None*, *representation_size=None*, *distilled=False*, *drop_rate=0.0*, *attn_drop_rate=0.0*, *drop_path_rate=0.0*, *embed_layer=PatchEmbed*, *norm_name='LN'*, *act_name='gelu'*, *num_classes=1000*, ***kwargs*)

基类：Module

ViT model.

参数

- **img_size** (int) –Input image size. Default: 224
- **patch_size** (int) –Patch token size. Default: 16
- **in_chans** (int) –Number of input image channels. Default: 3
- **embed_dim** (int) –Number of linear projection output channels. Default: 768
- **depth** (int) –Depth of Transformer Encoder layer. Default: 12
- **num_heads** (int) –Number of attention heads. Default: 12
- **ffn_ratio** (float) –Ratio of ffn hidden dim to embedding dim. Default: 4.0
- **qkv_bias** (bool) –If True, add a learnable bias to query, key, value. Default: True
- **qk_scale** (Optional[float]) –Override default qk scale of head_dim ** -0.5 if set. Default: None
- **representation_size** (Optional[int]) –Size of representation layer (pre-logits). Default: None
- **distilled** (bool) –Includes a distillation token and head. Default: False
- **drop_rate** (float) –Dropout rate. Default: 0.0
- **attn_drop_rate** (float) –Attention dropout rate. Default: 0.0
- **drop_path_rate** (float) –Stochastic depth rate. Default: 0.0

- **embed_layer** (Module) –Patch embedding layer. Default: PatchEmbed
- **norm_name** (str) –Normalization layer. Default: "LN"
- **act_name** (str) –Activation function. Default: "gelu"
- **num_classes** (int) –Number of classes. Default: 1000

init_weights()

forward(*x*)

load_state_dict(*state_dict*, *strict=True*)

Loads a given dictionary created by state_dict() into this module. If strict is True, the keys of state_dict() must exactly match the keys returned by state_dict().

Users can also pass a closure: Function[key: str, var: Tensor] -> Optional[np.ndarray] as a *state_dict*, in order to handle complex situations. For example, load everything except for the final linear classifier:

```
state_dict = {...}  #  Dict[str, np.ndarray]
model.load_state_dict({
    k: None if k.startswith('fc') else v
    for k, v in state_dict.items()
}, strict=False)
```

Here returning None means skipping parameter k.

To prevent shape mismatch (e.g. load PyTorch weights), we can reshape before loading:

```
state_dict = {...}
def reshape_accordingly(k, v):
    return state_dict[k].reshape(v.shape)
model.load_state_dict(reshape_accordingly)
```

We can also perform inplace re-initialization or pruning:

```
def reinit_and_pruning(k, v):
    if 'bias' in k:
        M.init.zero_(v)
    if 'conv' in k:
```

basecls.models.build

basecls.models.build.**build_model**(*cfg*)

The factory function to build model.

备注： if `cfg.model` does not have the attr `head`, this function will build model with the default head. Otherwise if `cfg.model.head` is `None`, this function will build model without any head.

备注： if `cfg.model.head` does not have the attr `w_out` and `cfg.num_classes` exists, `w_out` will be overridden by `cfg.num_classes`.

参数

cfg (`ConfigDict`) –config for building model.

返回类型

`Module`

返回

A model.

basecls.models.build.**load_model**(*model*, *weight_path*, *strict=True*)

Load model weights.

参数

- **model** (`Module`) –model for loading weights.
- **weight_path** (`str`) –weight path, both local path and OSS path are supported.
- **strict** (`bool`) –load weights in strict mode or not. Default: `True`

basecls.models.build.**sync_model**(*model*)

Sync parameters and buffers.

参数

model (`Module`) –model for syncing.

basecls.models.effnet

EfficientNet Series

EfficientNet: "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks"

引用

https://github.com/facebookresearch/pycls/blob/main/pycls/models/effnet.py https://github.com/rwightman/pytorch-image-models/blob/master/timm/models/efficientnet.py https://github.com/rwightman/pytorch-image-models/blob/master/timm/models/mobilenetv3.py

class basecls.models.effnet.**FuseMBConv**(*w_in*, *w_out*, *stride*, *kernel*, *exp_r*, *se_r*, *has_skip*, *drop_path_prob*, *norm_name*, *act_name*, ***kwargs*)

基类：Module

Fusing the proj conv1x1 and depthwise conv into a conv2d.

参数

- **w_in** (int) –input width.
- **w_out** (int) –output width.
- **stride** (int) –stride of conv.
- **kernel** (int) –kernel of conv.
- **exp_r** (float) –expansion ratio.
- **se_r** (float) –SE ratio.
- **has_skip** (bool) –whether apply skip connection.
- **drop_path_prob** (float) –drop path probability.
- **norm_name** (str) –normalization function.
- **act_name** (str) –activation function.

forward(*x*)

class basecls.models.effnet.**EffNet**(*stem_w*, *block_name*, *depths*, *widths*, *strides*, *kernels*, *exp_rs=1.0*, *se_rs=0.0*, *drop_path_prob=0.0*, *depth_mult=1.0*, *width_mult=1.0*, *omit_mult=False*, *norm_name='BN'*, *act_name='silu'*, *head=None*)

基类：Module

EfficientNet model.

参数

- **stem_w** (int) –stem width.

- **block_name** (Union[str, Callable, Sequence[Union[str, Callable]]]) – block name.
- **depths** (Sequence[int]) –depth for each stage (number of blocks in the stage).
- **widths** (Sequence[int]) –width for each stage (width of each block in the stage).
- **strides** (Sequence[int]) –strides for each stage (applies to the first block of each stage).
- **kernels** (Sequence[int]) –kernel sizes for each stage.
- **exp_rs** (Union[float, Sequence[Union[float, Sequence[float]]]]) –expansion ratios for MBConv blocks in each stage.
- **se_r** –Squeeze-and-Excitation (SE) ratio. Default: 0.25
- **drop_path_prob** (float) –drop path probability. Default: 0.0
- **depth_mult** (float) –depth multiplier. Default: 1.0
- **width_mult** (float) –width multiplier. Default: 1.0
- **omit_mult** (bool) –omit multiplier for stem width, head width, the first stage depth and the last stage depth, enabled in EfficientNet-Lite. Default: False
- **norm_name** (str) –normalization function. Default: "BN"
- **act_name** (str) –activation function. Default: "silu"
- **head** (Optional[Mapping[str, Any]]) –head args. Default: None

forward(*x*)

static get_block_func(*name*)

Retrieves the block function by name.

basecls.models.hrnet

HRNet Series

HRNet: "Deep High-Resolution Representation Learning for Visual Recognition"

引用

https://github.com/HRNet/HRNet-Image-Classification/blob/master/lib/models/cls_hrnet.py

class basecls.models.hrnet.**UpsampleNearest**(*scale_factor*)

基类：Module

Nearest upsample block

参数

scale_factor (int) –Upsample scale factor.

forward(*x*)

class basecls.models.hrnet.**HRFusion**(*channels*, *multi_scale_output*, *norm_name*, *act_name*)

基类：Module

HRNet fusion block.

参数

- **channels** (List[int]) –Fusion channels.
- **multi_scale_output** (bool) –Whether output multi-scale features.
- **norm_name** (str) –Normalization layer.
- **act_name** (str) –Activation function.

forward(*x_list*)

class basecls.models.hrnet.**HRModule**(*block_name*, *num_blocks*, *in_channels*, *channels*, *multi_scale_output*, *norm_name*, *act_name*)

基类：Module

HRNet module.

参数

- **block_name** (str) –Branch block type.
- **num_blocks** (List[int]) –Number of blocks.
- **in_channels** (List[int]) –Input channels.
- **channels** (List[int]) –Output channels.
- **multi_scale_output** (bool) –Whether output multi-scale features.
- **norm_name** (str) –Normalization layer.
- **act_name** (str) –Activation function.

forward(*x_list*)

class basecls.models.hrnet.**HRTrans**(*in_chs*, *out_chs*, *norm_name*, *act_name*)

基类：Module

HRNet transition block.

参数

- **in_chs** (List[int]) –Input channels.
- **out_chs** (List[int]) –Output channels.
- **norm_name** (str) –Normalization layer.

- **act_name** (str) –Activation function.

forward(*x_list*)

class basecls.models.hrnet.**HRStage**(*num_modules*, *num_blocks*, *block_name*, *pre_channels*, *cur_channels*, *multi_scale_output*, *w_fst*, *norm_name*, *act_name*)

基类：Module

HRNet stage.

参数

- **num_modules** (int) –Number of modules.
- **num_blocks** (List[int]) –Number of blocks for each module.
- **block_name** (str) –Branch block type.
- **pre_channels** (List[int]) –Channels of previous stage (an empty list for the first stage).
- **cur_channels** (List[int]) –Channels of current stage.
- **multi_scale_output** (bool) –Whether output multi-scale features.
- **w_fst** (Optional[int]) –Width of stem for the first stage (None for other stages).
- **norm_name** (str) –Normalization layer.
- **act_name** (str) –Activation function.

forward(*x_list*)

class basecls.models.hrnet.**HRMerge**(*block_name*, *pre_channels*, *channels*, *norm_name*, *act_name*)

基类：Module

HRNet merge block.

参数

- **block_name** (str) –Head block type.
- **pre_channels** (List[int]) –Channels of the last stage.
- **channels** (List[int]) –Channels of each scale to merge.
- **norm_name** (str) –Normalization layer.
- **act_name** (str) –Activation function.

forward(*x_list*)

class basecls.models.hrnet.**HRNet**(*stage_modules*, *stage_blocks*, *stage_block_names*, *stage_channels*, *w_stem=64*, *multi_scale_output=True*, *merge_block_name='bottleneck'*, *merge_channels=[32, 64, 128, 256]*, *norm_name='BN'*, *act_name='relu'*, *head=None*, ***kwargs*)

基类：Module

HRNet model.

参数

- **stage_modules** (List[int]) –Number of modules for each stage.
- **stage_blocks** (List[List[int]]) –Number of blocks for each module in stages.
- **stage_block_names** (List[str]) –Branch block types for each stage.
- **stage_channels** (List[List[int]]) –Number of channels for each stage.
- **w_stem** (int) –Stem width. Default: 64
- **multi_scale_output** (bool) –Whether output multi-scale features. Default: True
- **merge_block_name** (str) –Merge block type. Default: "bottleneck"
- **merge_channels** (List[int]) –Channels of each scale in merge block. Default: [32, 64, 128, 256]
- **norm_name** (str) –Normalization layer. Default: "BN"
- **act_name** (str) –Activation function. Default: "relu"
- **head** (Optional[Mapping[str, Any]]) –head args. Default: None

forward(*x*)

basecls.models.mbnet

MobileNet Series

MobileNetV1: "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications"

MobileNetV2: "MobileNetV2: Inverted Residuals and Linear Bottlenecks"

MobileNetV3: "Searching for MobileNetV3"

引用

https://github.com/rwightman/pytorch-image-models/blob/master/timm/models/efficientnet.py https://github.com/rwightman/pytorch-image-models/blob/master/timm/models/mobilenetv3.py

class basecls.models.mbnet.**MBConv**(*w_in*, *w_out*, *stride*, *kernel*, *exp_r*, *se_r*, *se_from_exp*, *se_act_name*, *se_approx*, *se_rd_fn*, *has_proj_act*, *has_skip*, *drop_path_prob*, *norm_name*, *act_name*)

基类：Module

Mobile inverted bottleneck block with SE.

Version	Expansion	DWConv	SE	PWConv	OutAct	Skip
basic	[EXP, BN, AF]	[kxk_DW, BN, AF]	[SE]	[1x1_Conv, BN]	[AF]	[Skip]
V1		[3x3_DW, BN, ReLU6]		[1x1_Conv, BN]	[ReLU6]	
V2	[EXP, BN, ReLU6]	[3x3_DW, BN, ReLU6]		[1x1_Conv, BN]		[Skip]
V3	[EXP, BN, AF]	[kxk_DW, BN, AF]	[SE]	[1x1_Conv, BN]		[Skip]

参数

- **w_in** (int) –input width.
- **w_out** (int) –output width.
- **stride** (int) –stride of depthwise conv.
- **kernel** (int) –kernel of depthwise conv.
- **exp_r** (float) –expansion ratio.
- **se_r** (float) –SE ratio.
- **se_from_exp** (bool) –calculate SE channels from expanded (mid) channels.
- **se_act_name** (str) –activation function for SE.
- **se_approx** (bool) –whether approximated sigmoid function (HSigmoid).
- **se_rd_fn** (Callable) –SE round channel function.
- **has_proj_act** (bool) –whether apply activation to output.
- **has_skip** (bool) –whether apply skip connection.
- **drop_path_prob** (float) –drop path probability.
- **norm_name** (str) –normalization function.
- **act_name** (str) –activation function.

forward(*x*)

class basecls.models.mbnet.**MBStage**(*w_in*, *w_out*, *stride*, *depth*, *exp_r*, *drop_path_prob*, ***kwargs*)

基类：Module

MBNet stage (sequence of blocks w/ the same output shape).

forward(*x*)

class basecls.models.mbnet.**MBNet**(*stem_w*, *depths*, *widths*, *strides*, *kernels*, *exp_rs=1.0*, *se_rs=0.0*, *stage_act_names=None*, *has_proj_act=False*, *has_skip=True*, *drop_path_prob=0.0*, *width_mult=1.0*, *norm_name='BN'*, *act_name='relu6'*, *head=None*)

基类：Module

MobileNet model.

参数

- **stem_w** (int) –stem width.
- **depths** (Sequence[int]) –depth for each stage (number of blocks in the stage).
- **widths** (Sequence[int]) –width for each stage (width of each block in the stage).
- **strides** (Sequence[int]) –strides for each stage (applies to the first block of each stage).
- **kernels** (Sequence[int]) –kernel sizes for each stage.
- **exp_rs** (Union[float, Sequence[Union[float, Sequence[float]]]]) –expansion ratios for MobileNet basic blocks in each stage. Default: 1.0
- **se_rs** (Union[float, Sequence[Union[float, Sequence[float]]]]) –Squeeze-and-Excitation (SE) ratios. Default: 0.0
- **stage_act_names** (Optional[Sequence[str]]) –activation function for stages. Default: None
- **has_proj_act** (bool) –whether apply activation to output. Default: False
- **has_skip** (bool) –whether apply skip connection. Default: True
- **drop_path_prob** (float) –drop path probability. Default: 0.0
- **width_mult** (float) –width multiplier. Default: 1.0
- **norm_name** (str) –normalization function. Default: "BN"
- **act_name** (str) –activation function. Default: "relu6"
- **head** (Optional[Mapping[str, Any]]) –head args. Default: None

forward(*x*)

basecls.models.regnet

RegNet Series

RegNet X/Y: "Designing Network Design Spaces"

引用

https://github.com/facebookresearch/pycls/blob/main/pycls/models/anynet.py https://github.com/facebookresearch/pycls/blob/main/pycls/models/regnet.py

class basecls.models.regnet.**RegBottleneckBlock**(*w_in*, *w_out*, *stride*, *bot_mul*, *group_w*, *se_r*, *norm_name*, *act_name*, ***kwargs*)

基类：Module

Residual bottleneck block for RegNet: x + f(x), f = 1x1, 3x3 [+SE], 1x1.

forward(*x*)

class basecls.models.regnet.**RegNet**(*stem_name*, *stem_w*, *block_name*, *depth*, *w0*, *wa*, *wm*, *group_w*, *stride=2*, *bot_mul=1.0*, *se_r=0.0*, *drop_path_prob=0.0*, *zero_init_final_gamma=False*, *norm_name='BN'*, *act_name='relu'*, *head=None*)

基类：*ResNet*

RegNet model.

参数

- **stem_name** (Union[str, Callable]) –stem name.
- **stem_w** (int) –stem width.
- **block_name** (Union[str, Callable]) –block name.
- **depth** (int) –depth.
- **w0** (int) –initial width.
- **wa** (float) –slope.
- **wm** (float) –quantization.
- **group_w** (int) –group width for each stage (applies to bottleneck block).
- **stride** (int) –stride for each stage (applies to the first block of each stage). Default: 2
- **bot_mul** (float) –bottleneck multiplier for each stage (applies to bottleneck block). Default: 1.0
- **se_r** (float) –Squeeze-and-Excitation (SE) ratio. Default: 0.0
- **drop_path_prob** (float) –drop path probability. Default: 0.0
- **zero_init_final_gamma** (bool) –enable zero-initialize or not. Default: False
- **norm_name** (str) –normalization function. Default: "BN"
- **act_name** (str) –activation function. Default: "relu"
- **head** (Optional[Mapping[str, Any]]) –head args. Default: None

basecls.models.repvgg

RepVGG Series

RegVGG: "RepVGG: Making VGG-style ConvNets Great Again"

引用

https://github.com/DingXiaoH/RepVGG/blob/main/repvgg.py

class basecls.models.repvgg.**RepVGGBlock**(*w_in*, *w_out*, *stride=1*, *groups=1*, *se_r=0.0*, *act_name='relu'*, *deploy=False*)

基类：Module

RepVGG Reparamed Block.

参数

- **w_in** (int) –number of input channels.
- **w_out** (int) –number of output channels.
- **stride** (int) –stride of the 2D conv. Default: 1
- **groups** (int) –number of groups of the 2D conv. Default: 1
- **se_r** (float) –Squeeze-and-Excitation (SE) ratio. Default: 0.0
- **act_name** (str) –activation function. Default: "relu"
- **deploy** (bool) –fuse branches into a plain Conv2d layer. Default: False

forward(*inputs*)

classmethod convert_to_deploy(*module*)

Convert/fuse reparameterized RepvggBlock for deploy

返回类型

Module

class basecls.models.repvgg.**RepVGG**(*num_blocks*, *width_multiplier*, *head=None*, *groups=1*, *se_r=0.0*, *act_name='relu'*, *deploy=False*)

基类：Module

RepVGG Model.

Use *RepVGG.convert_to_deploy()* to convert a training *RepVGG* to deploy:

```
model = RepVGG(..., deploy=False)
model.load_state_dict(...)
_ = RepVGG.convert_to_deploy(model)
```

参数

- **num_blocks** (Sequence[int]) –RepVGG depths.
- **width_multiplier** (Sequence[int]) –RepVGG widths, base_width is [64, 128, 256, 512].
- **head** (Optional[Mapping[str, Any]]) –head args. Default: None
- **groups** (Union[int, List[Union[int, List[int]]]]) –number of groups for blocks. Default: 1
- **se_r** (float) –Squeeze-and-Excitation (SE) ratio. Default: 0.0
- **act_name** (str) –activation function. Default: "relu"
- **deploy** (bool) –switch a reparamed RepVGG into deploy mode. Default: False

forward(*x*)

classmethod convert_to_deploy(*module*)

返回类型
Module

basecls.models.resmlp

ResMLP Series

ResMLP: "ResMLP: Feedforward networks for image classification with data-efficient training"

引用

https://github.com/facebookresearch/deit/blob/main/resmlp_models.py

class basecls.models.resmlp.**Affine**(*dim*)

基类: Module

ResMLP Affine Layer.

forward(*x*)

class basecls.models.resmlp.**ResMLPBlock**(*dim*, *drop*, *drop_path*, *num_patches*, *init_scale*, *ffn_ratio*, *act_name*)

基类: Module

ResMLP block.

参数

- **dim** (int) –Number of input channels.

- **drop** (float) –Dropout ratio.
- **drop_path** (float) –Stochastic depth rate.
- **num_patches** (int) –Number of patches.
- **init_scale** (float) –Initial value for LayerScale.
- **ffn_ratio** (float) –Ratio of ffn hidden dim to embedding dim.
- **act_name** (str) –activation function.

forward(*x*)

class basecls.models.resmlp.**ResMLP**(*img_size=224*, *patch_size=16*, *in_chans=3*, *embed_dim=768*, *depth=12*, *drop_rate=0.0*, *drop_path_rate=0.0*, *embed_layer=PatchEmbed*, *init_scale=1e-4*, *ffn_ratio=4.0*, *act_name='gelu'*, *num_classes=1000*, ***kwargs*)

基类：Module

ResMLP model.

参数

- **img_size** (int) –Input image size. Default: 224
- **patch_size** (int) –Patch token size. Default: 16
- **in_chans** (int) –Number of input image channels. Default: 3
- **embed_dim** (int) –Number of linear projection output channels. Default: 768
- **depth** (int) –Depth of Transformer Encoder layer. Default: 12
- **drop_rate** (float) –Dropout rate. Default: 0.0
- **drop_path_rate** (float) –Stochastic depth rate. Default: 0.0
- **embed_layer** (Module) –Patch embedding layer. Default: PatchEmbed
- **init_scale** (float) –Initial value for LayerScale. Default: 1e-4
- **ffn_ratio** (float) –Ratio of ffn hidden dim to embedding dim. Default: 4.0
- **act_name** (str) –Activation function. Default: "gelu"
- **num_classes** (int) –Number of classes. Default: 1000

forward(*x*)

basecls.models.resnet

ResNet Series

ResNet: "Deep Residual Learning for Image Recognition"

ResNet-D: "Bag of Tricks for Image Classification with Convolutional Neural Networks"

ResNeXt: "Aggregated Residual Transformations for Deep Neural Networks"

Se-ResNet: "Squeeze-and-Excitation Networks"

Wide ResNet: "Wide Residual Networks"

引用

https://github.com/facebookresearch/pycls/blob/main/pycls/models/anynet.py https://github.com/facebookresearch/pycls/blob/main/pycls/models/resnet.py

class basecls.models.resnet.**ResBasicBlock**(*w_in*, *w_out*, *stride*, *bot_mul*, *se_r*, *avg_down*, *drop_path_prob*, *norm_name*, *act_name*, ***kwargs*)

基类: Module

Residual basic block: x + f(x), f = [3x3 conv, BN, Act] x2.

forward(*x*)

class basecls.models.resnet.**ResBottleneckBlock**(*w_in*, *w_out*, *stride*, *bot_mul*, *group_w*, *se_r*, *avg_down*, *drop_path_prob*, *norm_name*, *act_name*, ***kwargs*)

基类: Module

Residual bottleneck block: x + f(x), f = 1x1, 3x3, 1x1 [+SE].

forward(*x*)

class basecls.models.resnet.**ResDeepStem**(*w_in*, *w_out*, *norm_name*, *act_name*, ***kwargs*)

基类: Module

ResNet-D stem: [3x3, BN, Act] x3, MaxPool.

forward(*x*)

class basecls.models.resnet.**ResStem**(*w_in*, *w_out*, *norm_name*, *act_name*, ***kwargs*)

基类: Module

ResNet stem: 7x7, BN, Act, MaxPool.

forward(*x*)

class basecls.models.resnet.**SimpleStem**(*w_in*, *w_out*, *norm_name*, *act_name*, ***kwargs*)

基类：Module

Simple stem: 3x3, BN, Act.

forward(*x*)

class basecls.models.resnet.**AnyStage**(*w_in*, *w_out*, *stride*, *depth*, *block_func*, *drop_path_prob*, ***kwargs*)

基类：Module

AnyNet stage (sequence of blocks w/ the same output shape).

forward(*x*)

class basecls.models.resnet.**ResNet**(*stem_name*, *stem_w*, *block_name*, *depths*, *widths*, *strides*, *bot_muls=1.0*, *group_ws=None*, *se_r=0.0*, *avg_down=False*, *drop_path_prob=0.0*, *zero_init_final_gamma=False*, *norm_name='BN'*, *act_name='relu'*, *head=None*)

基类：Module

ResNet model.

参数

- **stem_name** (Union[str, Callable]) –stem name.
- **stem_w** (int) –stem width.
- **block_name** (Union[str, Callable]) –block name.
- **depths** (Sequence[int]) –depth for each stage (number of blocks in the stage).
- **widths** (Sequence[int]) –width for each stage (width of each block in the stage).
- **strides** (Sequence[int]) –strides for each stage (applies to the first block of each stage).
- **bot_muls** (Union[float, Sequence[float]]) –bottleneck multipliers for each stage (applies to bottleneck block). Default: 1.0
- **group_ws** (Optional[Sequence[int]]) –group widths for each stage (applies to bottleneck block). Default: None
- **se_r** (float) –Squeeze-and-Excitation (SE) ratio. Default: 0.0
- **drop_path_prob** (float) –drop path probability. Default: 0.0
- **zero_init_final_gamma** (bool) –enable zero-initialize or not. Default: False
- **norm_name** (str) –normalization function. Default: "BN"
- **act_name** (str) –activation function. Default: "relu"
- **head** (Optional[Mapping[str, Any]]) –head args. Default: None

forward(*x*)

static get_stem_func(*name*)

Retrieves the stem function by name.

static get_block_func(*name*)

Retrieves the block function by name.

basecls.models.snet

Shufflenet Series

ShufflenetV2: "ShuffleNet V2: Practical Guidelines for Efficient CNN Architecture Design"

引用

https://github.com/megvii-model/ShuffleNet-Series/blob/master/ShuffleNetV2/network.py

class basecls.models.snet.**SNV2Block**(*w_in*, *w_out*, *w_mid*, ***, *kernel*, *stride*, *norm_name*, *act_name*, *se_r*, *drop_path_prob*, ***kwargs*)

基类：Module

ShuffleNet V2 Block

forward(*old_x*)

class basecls.models.snet.**SNV2XceptionBlock**(*w_in*, *w_out*, *w_mid*, ***, *kernel*, *stride*, *norm_name*, *act_name*, *se_r*, *drop_path_prob*, ***kwargs*)

基类：*SNV2Block*

ShuffleNet V2 Xception stype block used in ShuffleNet V2+

class basecls.models.snet.**SNetV2**(*block*, *stem_w*, *depths*, *widths*, *strides*, *kernels*, *use_maxpool=True*, *se_r=0.0*, *drop_path_prob=0.0*, *norm_name='BN'*, *act_name='relu'*, *head=None*)

基类：Module

ShufflenetV2 model.

参数

- **block** (Callable) –building block to use, SNV2XceptionBlock for v2+.
- **stem_w** (int) –width for stem layer.
- **depths** (Sequence[int]) –depth for each stage (number of blocks in the stage).
- **widths** (Sequence[int]) –width for each stage (width of each block in the stage).
- **strides** (Sequence[int]) –strides for each stage (applies to the first block of each stage).

- **kernels** (Sequence[int]) –kernel sizes for each stage.
- **use_maxpool** (bool) –whether use maxpool stride 2 after stem. Default: True
- **se_r** (float) –Squeeze-and-Excitation (SE) ratio. Default: 0.0
- **drop_path_prob** (float) –drop path probability. Default: 0.0
- **norm_name** (str) –normalization function. Default: "BN"
- **act_name** (str) –activation function. Default: "relu6"
- **head** (Optional[Mapping[str, Any]]) –head args. Default: None

forward(*x*)

basecls.models.swin

Swin Transformer Series

Swin Transformer: "Hierarchical Vision Transformer using Shifted Windows"

引用

https://github.com/microsoft/Swin-Transformer/blob/main/models/swin_transformer.py

basecls.models.swin.**window_partition**(*x*, *window_size*)

参数

- **x** –(B, H, W, C)
- **window_size** (int) –window size

返回

(num_windows*B, window_size, window_size, C)

返回类型

windows

basecls.models.swin.**window_reverse**(*windows*, *window_size*, *H*, *W*)

参数

- **windows** –(num_windows*B, window_size, window_size, C)
- **window_size** (int) –Window size
- **H** (int) –Height of image
- **W** (int) –Width of image

返回

(B, H, W, C)

返回类型

x

class basecls.models.swin.**WindowAttention**(*dim*, *window_size*, *num_heads*, *qkv_bias=True*, *qk_scale=None*, *attn_drop=0.0*, *proj_drop=0.0*)

基类：Module

Window based multi-head self attention (W-MSA) module with relative position bias. It supports both of shifted and non-shifted window.

参数

- **dim** (int) –Number of input channels.
- **window_size** (int) –The height and width of the window.
- **num_heads** (int) –Number of attention heads.
- **qkv_bias** (bool) –If True, add a learnable bias to query, key, value. Default: True
- **qk_scale** (Optional[float]) –Override default qk scale of head_dim ** -0.5 if set.
- **attn_drop** (float) –Dropout ratio of attention weight. Default: 0.0
- **proj_drop** (float) –Dropout ratio of output. Default: 0.0

forward(*x*, *mask=None*)

参数

- **x** –input features with shape of (num_windows*B, N, C)
- **mask** –(0/-inf) mask with shape of (num_windows, Wh*Ww, Wh*Ww) or None

class basecls.models.swin.**PatchMerging**(*dim*, *input_resolution*, *norm_name='LN'*)

基类：Module

Patch Merging Layer.

参数

- **dim** (int) –Number of input channels.
- **input_resolution** (Tuple[int, int]) –Resolution of input feature.
- **norm_name** (str) –Normalization layer. Default: "LN"

forward(*x*)

x: B, H*W, C

class basecls.models.swin.**SwinBlock**(*dim*, *input_resolution*, *num_heads*, *window_size=7*, *shift_size=0*, *ffn_ratio=4.0*, *qkv_bias=True*, *qk_scale=None*, *drop=0.0*, *attn_drop=0.0*, *drop_path=0.0*, *norm_name='LN'*, *act_name='gelu'*)

基类：Module

Swin Transformer Block.

参数

- **dim** (int) –Number of input channels.
- **input_resolution** (Tuple[int, int]) –Input resulotion.
- **num_heads** (int) –Number of attention heads.
- **window_size** (int) –Window size. Default: 7
- **shift_size** (int) –Shift size for SW-MSA. Default: 0
- **ffn_ratio** (float) –Ratio of ffn hidden dim to embedding dim. Default: 4.0
- **qkv_bias** (bool) –If True, add a learnable bias to query, key, value. Default: True
- **qk_scale** (Optional[float]) –Override default qk scale of head_dim ** -0.5 if set.
- **drop** (float) –Dropout ratio of non-attention weight. Default: 0.0
- **attn_drop** (float) –Dropout ratio of attention weight. Default: 0.0
- **drop_path** (float) –Stochastic depth rate. Default: 0.0
- **norm_name** (str) –Normalization layer. Default: "LN"
- **act_name** (str) –Activation layer. Default: "gelu"

forward(*x*)

class basecls.models.swin.**SwinBasicLayer**(*dim*, *input_resolution*, *depth*, *num_heads*, *window_size*, *ffn_ratio=4.0*, *qkv_bias=True*, *qk_scale=None*, *drop=0.0*, *attn_drop=0.0*, *drop_path=0.0*, *downsample=None*, *norm_name='LN'*, *act_name='gelu'*)

基类：Module

A basic Swin Transformer layer for one stage.

参数

- **dim** (int) –Number of input channels.
- **input_resolution** (Tuple[int, int]) –Input resolution.
- **depth** (int) –Number of blocks.
- **num_heads** (int) –Number of attention heads.
- **window_size** (int) –Local window size.

- **ffn_ratio** (float) –Ratio of ffn hidden dim to embedding dim.
- **qkv_bias** (bool) –If True, add a learnable bias to query, key, value. Default: True
- **qk_scale** (Optional[float]) –Override default qk scale of head_dim ** -0.5 if set.
- **drop** (float) –Dropout rate. Default: 0.0
- **attn_drop** (float) –Attention dropout rate. Default: 0.0
- **drop_path** (float) –Stochastic depth rate. Default: 0.0
- **norm_name** (str) –Normalization layer. Default: "LN"
- **act_name** (str) –Activation layer. Default: "gelu"
- **downsample** (Optional[Module]) –Downsample layer at the end of the layer. Default: None

forward(*x*)

class basecls.models.swin.**SwinTransformer**(*img_size=224*, *patch_size=4*, *in_chans=3*, *embed_dim=96*, *depths=[2, 2, 6, 2]*, *num_heads=[3, 6, 12, 24]*, *window_size=7*, *ffn_ratio=4.0*, *qkv_bias=True*, *qk_scale=None*, *ape=False*, *patch_norm=True*, *drop_rate=0.0*, *attn_drop_rate=0.0*, *drop_path_rate=0.1*, *embed_layer=PatchEmbed*, *norm_name='LN'*, *act_name='gelu'*, *num_classes=1000*, ***kwargs*)

基类：Module

Swin Transformer

A PyTorch impl of :

Swin Transformer: Hierarchical Vision Transformer using Shifted Windows - https://arxiv.org/pdf/2103.14030

参数

- **img_size** (int) –Input image size. Default: 224
- **patch_size** (int) –Patch size. Default: 4
- **in_chans** (int) –Number of input image channels. Default: 3
- **embed_dim** (int) –Patch embedding dimension. Default: 96
- **depths** (Sequence[int]) –Depth of each Swin Transformer layer.
- **num_heads** (Sequence[int]) –Number of attention heads in different layers.
- **window_size** (int) –Window size. Default: 7

- **ffn_ratio** (float) –Ratio of ffn hidden dim to embedding dim. Default: 4.0
- **qkv_bias** (bool) –If True, add a learnable bias to query, key, value. Default: True
- **qk_scale** (Optional[float]) –Override default qk scale of head_dim ** -0.5 if set. Default: None
- **ape** (bool) –If True, add absolute position embedding to the patch embedding. Default: False
- **patch_norm** (bool) –If True, add normalization after patch embedding. Default: True
- **drop_rate** (float) –Dropout rate. Default: 0
- **attn_drop_rate** (float) –Attention dropout rate. Default: 0
- **drop_path_rate** (float) –Stochastic depth rate. Default: 0.1
- **embed_layer** (Module) –Patch embedding layer. Default: PatchEmbed
- **norm_name** (str) –Normalization layer. Default: "LN"
- **act_name** (str) –Activation layer. Default: "gelu"
- **num_classes** (int) –Number of classes for classification head. Default: 1000

forward(*x*)

basecls.models.vgg

VGG Series

VGG: "Very Deep Convolutional Networks for Large-Scale Image Recognition"

class basecls.models.vgg.**VGGStage**(*w_in*, *w_out*, *depth*, *norm_name*, *act_name*)

基类：Module

VGG stage (sequence of blocks w/ the same output shape).

forward(*x*)

返回类型

Tensor

class basecls.models.vgg.**VGG**(*depths*, *widths*, *norm_name=None*, *act_name='relu'*, *head=None*)

基类：Module

VGG model.

参数

- **depths** (Sequence[int]) –depth for each stage (number of blocks in the stage).
- **widths** (Sequence[int]) –width for each stage (width of each block in the stage).

- **norm_name** (Optional[str]) –normalization function. Default: None
- **act_name** (str) –activation function. Default: "relu"
- **head** (Optional[Mapping[str, Any]]) –head args. Default: None

forward(*x*)

返回类型

Tensor

basecls.models.vit

Vision Transformer (ViT)

ViT: "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale"

DeiT: "Training data-efficient image transformers & distillation through attention"

引用

https://github.com/rwightman/pytorch-image-models/blob/master/timm/models/vision_transformer.py

class basecls.models.vit.**PatchEmbed**(*img_size=224*, *patch_size=16*, *in_chans=3*, *embed_dim=768*, *flatten=True*, *norm_name=None*, ***kwargs*)

基类：Module

Image to Patch Embedding

参数

- **img_size** (int) –Image size. Default: 224
- **patch_size** (int) –Patch token size. Default: 16
- **in_chans** (int) –Number of input image channels. Default: 3
- **embed_dim** (int) –Number of linear projection output channels. Default: 768
- **flatten** (bool) –Flatten embedding. Default: True
- **norm_name** (Optional[str]) –Normalization layer. Default: None

forward(*x*)

class basecls.models.vit.**Attention**(*dim*, *num_heads=8*, *qkv_bias=False*, *qk_scale=None*, *attn_drop=0.0*, *proj_drop=0.0*)

基类：Module

Self-Attention block.

参数

- **dim** (int) –input Number of input channels.
- **num_heads** (int) –Number of attention heads. Default: 8
- **qkv_bias** (bool) –If True, add a learnable bias to query, key, value. Default: False
- **qk_scale** (Optional[float]) –Override default qk scale of head_dim ** -0.5 if set.
- **attn_drop** (float) –Dropout ratio of attention weight. Default: 0.0
- **proj_drop** (float) –Dropout ratio of output. Default: 0.0

forward(*x*)

class basecls.models.vit.**FFN**(*in_features*, *hidden_features=None*, *out_features=None*, *drop=0.0*, *act_name='gelu'*)

基类：Module

FFN for ViT

参数

- **in_features** (int) –Number of input features.
- **hidden_features** (Optional[int]) –Number of input features. Default: None
- **out_features** (Optional[int]) –Number of output features. Default: None
- **drop** (float) –Dropout ratio. Default: 0.0
- **act_name** (str) –activation function. Default: "gelu"

forward(*x*)

class basecls.models.vit.**EncoderBlock**(*dim*, *num_heads*, *ffn_ratio=4.0*, *qkv_bias=False*, *qk_scale=None*, *attn_drop=0.0*, *drop=0.0*, *drop_path=0.0*, *norm_name='LN'*, *act_name='gelu'*, ***kwargs*)

基类：Module

Transformer Encoder block.

参数

- **dim** (int) –Number of input channels.
- **num_heads** (int) –Number of attention heads.
- **ffn_ratio** (float) –Ratio of ffn hidden dim to embedding dim. Default: 4.0
- **qkv_bias** (bool) –If True, add a learnable bias to query, key, value. Default: False
- **qk_scale** (Optional[float]) –Override default qk scale of head_dim ** -0.5 if set.
- **drop** (float) –Dropout ratio of non-attention weight. Default: 0.0

- **attn_drop** (float) –Dropout ratio of attention weight. Default: 0.0
- **drop_path** (float) –Stochastic depth rate. Default: 0.0
- **norm_name** (str) –Normalization layer. Default: "LN"
- **act_name** (str) –Activation layer. Default: "gelu"

forward(*x*)

class basecls.models.vit.**ViT**(*img_size=224*, *patch_size=16*, *in_chans=3*, *embed_dim=768*, *depth=12*, *num_heads=12*, *ffn_ratio=4.0*, *qkv_bias=True*, *qk_scale=None*, *representation_size=None*, *distilled=False*, *drop_rate=0.0*, *attn_drop_rate=0.0*, *drop_path_rate=0.0*, *embed_layer=PatchEmbed*, *norm_name='LN'*, *act_name='gelu'*, *num_classes=1000*, ***kwargs*)

基类：Module

ViT model.

参数

- **img_size** (int) –Input image size. Default: 224
- **patch_size** (int) –Patch token size. Default: 16
- **in_chans** (int) –Number of input image channels. Default: 3
- **embed_dim** (int) –Number of linear projection output channels. Default: 768
- **depth** (int) –Depth of Transformer Encoder layer. Default: 12
- **num_heads** (int) –Number of attention heads. Default: 12
- **ffn_ratio** (float) –Ratio of ffn hidden dim to embedding dim. Default: 4.0
- **qkv_bias** (bool) –If True, add a learnable bias to query, key, value. Default: True
- **qk_scale** (Optional[float]) –Override default qk scale of head_dim ** -0.5 if set. Default: None
- **representation_size** (Optional[int]) –Size of representation layer (pre-logits). Default: None
- **distilled** (bool) –Includes a distillation token and head. Default: False
- **drop_rate** (float) –Dropout rate. Default: 0.0
- **attn_drop_rate** (float) –Attention dropout rate. Default: 0.0
- **drop_path_rate** (float) –Stochastic depth rate. Default: 0.0
- **embed_layer** (Module) –Patch embedding layer. Default: *PatchEmbed*
- **norm_name** (str) –Normalization layer. Default: "LN"
- **act_name** (str) –Activation function. Default: "gelu"

- **num_classes** (int) –Number of classes. Default: 1000

init_weights()

forward(*x*)

load_state_dict(*state_dict*, *strict=True*)

Loads a given dictionary created by state_dict() into this module. If strict is True, the keys of state_dict() must exactly match the keys returned by state_dict().

Users can also pass a closure: Function[key: str, var: Tensor] -> Optional[np.ndarray] as a *state_dict*, in order to handle complex situations. For example, load everything except for the final linear classifier:

```
state_dict = {...}  #  Dict[str, np.ndarray]
model.load_state_dict({
    k: None if k.startswith('fc') else v
    for k, v in state_dict.items()
}, strict=False)
```

Here returning None means skipping parameter k.

To prevent shape mismatch (e.g. load PyTorch weights), we can reshape before loading:

```
state_dict = {...}
def reshape_accordingly(k, v):
    return state_dict[k].reshape(v.shape)
model.load_state_dict(reshape_accordingly)
```

We can also perform inplace re-initialization or pruning:

```
def reinit_and_pruning(k, v):
    if 'bias' in k:
        M.init.zero_(v)
    if 'conv' in k:
```

1.3.6 basecls.solver

class basecls.solver.**BaseSolver**

基类: object

Base class for solver factory.

A solver factory should return a *Solver* object, which combines an Optimizer and a GradManager.

classmethod build(*cfg*, *model*)

Abstract build function

参数

- **cfg** (ConfigDict) –config for training.
- **model** (Module) –model for training.

返回类型

Solver

返回

A solver.

class basecls.solver.**DefaultSolver**

基类：*BaseSolver*

The default solver factory.

According to `cfg.reduce_mode`, learning rate and weight decay will be scaled automatically following the linear scaling rule, see "Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour" for more details.

It supports `"sgd"`, `"adam"` and `"adamw"`.

备注： This linear scaling rule can only work well with SGD. We are still looking for the applicable scaling rule for Adam and AdamW. Thus we recommend keeping default training settings (like learning rate and world size) when using Adam and AdamW.

---

classmethod build(*cfg*, *model*)

Build function with the linear scaling strategy.

参数

- **cfg** (ConfigDict) –config for training.
- **model** (Module) –model for training.

返回类型

Solver

返回

A solver.

classmethod build_optimizer(*cfg*, *params*, *lr*, *wd*)

Build optimizer according to training config.

参数

- **cfg** (ConfigDict) –config for training.
- **params** (Union[Iterable[Parameter], dict]) –iterable of parameters to optimize or dicts defining parameter groups.

- **lr** (float) –learning rate.
- **weight_decay** –weight decay (L2, penalty).

返回类型
Optimizer

返回
An optimizer.

class basecls.solver.**Solver**(*optimizer*, *grad_manager*, *grad_scaler*)

基类：tuple

grad_manager

Alias for field number 1

grad_scaler

Alias for field number 2

optimizer

Alias for field number 0

basecls.solver.optimizer

class basecls.solver.optimizer.**LAMB**(*params*, *lr*, *betas=(0.9, 0.999)*, *eps=1e-8*, *bias_correction=True*, *weight_decay=0.0*, *always_adapt=False*)

基类：Optimizer

Implements LAMB algorithm.

LAMB is proposed in "Large Batch Optimization for Deep Learning: Training BERT in 76 minutes".

参数

- **params** (Union[Iterable[Parameter], dict]) –iterable of parameters to optimize or dicts defining parameter groups.
- **lr** (float) –learning rate.
- **betas** (Tuple[float, float]) –coefficients used for computing running averages of gradient and its square. Default: (0.9, 0.999)
- **eps** (float) –term added to the denominator to improve numerical stability. Default: 1e-8
- **bias_correction** (bool) –enables bias correction by 1 - beta ** step. Default: True
- **weight_decay** (float) –weight decay (L2 penalty). Default: 0.0
- **always_adapt** (bool) –apply adaptive lr to 0.0 weight decay parameter. Default: False

class basecls.solver.optimizer.**LARS**(*params*, *lr*, *momentum=0.0*, *nesterov=False*, *weight_decay=0.0*, *always_adapt=False*)

基类：Optimizer

Implements LARS algorithm.

LARS is proposed in "Large Batch Optimization for Deep Learning: Training BERT in 76 minutes".

参数

- **params** (Union[Iterable[Parameter], dict]) –iterable of parameters to optimize or dicts defining parameter groups.
- **lr** (float) –learning rate.
- **momentum** (float) –momentum factor. Default: 0.0
- **nesterov** (bool) –enables Nesterov momentum. Default: False
- **weight_decay** (float) –weight decay (L2 penalty). Default: 0.0
- **always_adapt** (bool) –apply adaptive lr to 0.0 weight decay parameter. Default: False

class basecls.solver.optimizer.**SGD**(*params*, *lr*, *momentum=0.0*, *nesterov=False*, *weight_decay=0.0*)

基类：Optimizer

Implements stochastic gradient descent.

Nesterov momentum is based on the formula from "On the importance of initialization and momentum in deep learning".

参数

- **params** (Union[Iterable[Parameter], dict]) –iterable of parameters to optimize or dicts defining parameter groups.
- **lr** (float) –learning rate.
- **momentum** (float) –momentum factor. Default: 0.0
- **nesterov** (bool) –enables Nesterov momentum. Default: False
- **weight_decay** (float) –weight decay (L2 penalty). Default: 0.0

basecls.solver.optimizer.lamb

LAMB optimizer

References: https://github.com/rwightman/pytorch-image-models/blob/master/timm/optim/lamb.py

class basecls.solver.optimizer.lamb.**LAMB**(*params*, *lr*, *betas=(0.9, 0.999)*, *eps=1e-8*, *bias_correction=True*, *weight_decay=0.0*, *always_adapt=False*)

基类：Optimizer

Implements LAMB algorithm.

LAMB is proposed in "Large Batch Optimization for Deep Learning: Training BERT in 76 minutes".

参数

- **params** (Union[Iterable[Parameter], dict]) –iterable of parameters to optimize or dicts defining parameter groups.
- **lr** (float) –learning rate.
- **betas** (Tuple[float, float]) –coefficients used for computing running averages of gradient and its square. Default: (0.9, 0.999)
- **eps** (float) –term added to the denominator to improve numerical stability. Default: 1e-8
- **bias_correction** (bool) –enables bias correction by 1 - beta ** step. Default: True
- **weight_decay** (float) –weight decay (L2 penalty). Default: 0.0
- **always_adapt** (bool) –apply adaptive lr to 0.0 weight decay parameter. Default: False

basecls.solver.optimizer.lars

LARS optimizer

References: https://github.com/rwightman/pytorch-image-models/blob/master/timm/optim/lars.py

class basecls.solver.optimizer.lars.**LARS**(*params*, *lr*, *momentum=0.0*, *nesterov=False*, *weight_decay=0.0*, *always_adapt=False*)

基类：Optimizer

Implements LARS algorithm.

LARS is proposed in "Large Batch Optimization for Deep Learning: Training BERT in 76 minutes".

参数

- **params** (Union[Iterable[Parameter], dict]) –iterable of parameters to optimize or dicts defining parameter groups.
- **lr** (float) –learning rate.
- **momentum** (float) –momentum factor. Default: 0.0
- **nesterov** (bool) –enables Nesterov momentum. Default: False
- **weight_decay** (float) –weight decay (L2 penalty). Default: 0.0
- **always_adapt** (bool) –apply adaptive lr to 0.0 weight decay parameter. Default: False

basecls.solver.optimizer.sgd

class basecls.solver.optimizer.sgd.**SGD**(*params*, *lr*, *momentum=0.0*, *nesterov=False*, *weight_decay=0.0*)

基类：Optimizer

Implements stochastic gradient descent.

Nesterov momentum is based on the formula from "On the importance of initialization and momentum in deep learning".

参数

- **params** (Union[Iterable[Parameter], dict]) –iterable of parameters to optimize or dicts defining parameter groups.
- **lr** (float) –learning rate.
- **momentum** (float) –momentum factor. Default: 0.0
- **nesterov** (bool) –enables Nesterov momentum. Default: False
- **weight_decay** (float) –weight decay (L2 penalty). Default: 0.0

basecls.solver.build

class basecls.solver.build.**Solver**(*optimizer*, *grad_manager*, *grad_scaler*)

基类：tuple

grad_manager

Alias for field number 1

grad_scaler

Alias for field number 2

optimizer

Alias for field number 0

class basecls.solver.build.**BaseSolver**

基类：object

Base class for solver factory.

A solver factory should return a *Solver* object, which combines an Optimizer and a GradManager.

classmethod build(*cfg*, *model*)

Abstract build function

参数

- **cfg** (ConfigDict) –config for training.
- **model** (Module) –model for training.

返回类型

Solver

返回

A solver.

class basecls.solver.build.**DefaultSolver**

基类：*BaseSolver*

The default solver factory.

According to `cfg.reduce_mode`, learning rate and weight decay will be scaled automatically following the linear scaling rule, see "Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour" for more details.

It supports `"sgd"`, `"adam"` and `"adamw"`.

备注: This linear scaling rule can only work well with SGD. We are still looking for the applicable scaling rule for Adam and AdamW. Thus we recommend keeping default training settings (like learning rate and world size) when using Adam and AdamW.

---

classmethod build(*cfg*, *model*)

Build function with the linear scaling strategy.

参数

- **cfg** (ConfigDict) –config for training.
- **model** (Module) –model for training.

返回类型

Solver

返回

A solver.

classmethod build_optimizer(*cfg*, *params*, *lr*, *wd*)

Build optimizer according to training config.

参数

- **cfg** (ConfigDict) –config for training.
- **params** (Union[Iterable[Parameter], dict]) –iterable of parameters to optimize or dicts defining parameter groups.
- **lr** (float) –learning rate.
- **weight_decay** –weight decay (L2, penalty).

返回类型

Optimizer

返回

An optimizer.

basecls.solver.weight_decay

basecls.solver.weight_decay.**get_param_groups**(*module*, *weight_decay_policy*)

Directly get optimizer's param_groups with different weight decays given policy.

`cfg.solver.weight_decay` can be a float or a sequence of weight decay policies.

For example:

```
cfg.solver.weight_decay = 1e-5
```

is equivalent to

```
cfg.solver.weight_decay = [
    1e-5
]
```

Weight decay policy works in sequential order, i.e., for each parameter, we try patterns from the beginning to the end. If unmatched, default weight decay (-1) will be applied. For example:

```
from basecls.layers import NORM_TYPES
cfg.solver.weight_decay = [
    (1e-5, "bias"),
    (0, NORM_TYPES),
    1e-4,
]
```

The parameter will first match `(1e-5, "bias")` then `(0, NORM_TYPES)`, so any bias parameter, including the bias of normalization layers, will have weight decay `1e-5`.

For mobile models, e.g. mobilenet and shufflenet, you may want to disable weight decay for normalization layers and any bias. This can be achieved by the following:

```
from basecls.layers import NORM_TYPES
cfg.solver.weight_decay = [
    (0, "bias"),
    (0, NORM_TYPES),
    4e-5,
]
```

参数

- **module** (Module) –training model
- **weight_decay_policy** (Union[float, Sequence[Tuple[float, Union[str, Type, Sequence[Type]]]]]) –weight decay policy defined in `cfg.solver.weight_decay`

返回类型

Dict[float, Iterable[Tensor]]

1.3.7 basecls.tools

basecls.tools.cls_test

basecls.tools.cls_test.**make_parser**()

Build args parser for testing script.

返回类型

ArgumentParser

返回

The args parser.

basecls.tools.cls_test.**worker**(*args*)

Worker function for testing script.

参数

args (Namespace) –args for testing script.

basecls.tools.cls_test.**build**(*cfg*)

Build function for testing script.

参数

cfg (ConfigDict) –config for testing.

返回

A tester.

basecls.tools.cls_test.**main**()

Main function for testing script.

basecls.tools.cls_train

basecls.tools.cls_train.**default_parser**()

Build args parser for training script.

返回类型

ArgumentParser

返回

The args parser.

basecls.tools.cls_train.**worker**(*args*)

Worker function for training script.

参数

args (Namespace) –args for training script.

basecls.tools.cls_train.**build**(*cfg*)

Build function for training script.

参数

cfg (ConfigDict) –config for training.

返回

A trainer.

basecls.tools.cls_train.**main**()

Main function for training script.

1.3.8 basecls.utils

class basecls.utils.**Registry**(*name*)

基类：object

The registry that provides name -> object mapping, to support third-party users' custom modules.

To create a registry (e.g. a backbone registry):

```
OPTIMIZER = Registry('optimizer')
```

To register an object:

```
@OPTIMIZER.register()
class MyOptimizer():
    ...
```

Or:

```
OPTIMIZER.register(MyOptimizer)
```

Or:

```
@OPTIMIZER.register("Name for Registry")
class MyOptimizer():
    ...
```

register(*obj=None*, *name=None*)

Register the given object under the the name *obj.__name__*. Can be used as either a decorator or not. See docstring of this class for usage.

返回类型

Optional[object]

get(*name*)

返回类型

object

items()

keys()

values()

basecls.utils.**set_nccl_env**()

Set NCCL environments, which is essential to multi-node training.

basecls.utils.**set_num_threads**(*num=1*)

Set number of threads in OpenMP, OpenCV, MKL, OPENBLAS, VECLIB, NUMEXPR, etc.

参数

num (int) –number of threads. Default: 1

basecls.utils.**default_logging**(*cfg*, *model*)

basecls.utils.**setup_logger**(*log_path*, *log_file*, *to_loguru=False*)

class basecls.utils.**registers**

基类：object

All registried module could be found here.

```
augments = Registry of augments:
╒═══════════════╤═══════════════════════════════════════════════════╕ | Names
| Objects |
╞═══════════════╪═══════════════════════════════════════════════════╡ |
SimpleAugment | <class 'basecls.data.transform.SimpleAugment'> |
├───────────────┼───────────────────────────────────────────────────┤ |
ColorAugment | <class 'basecls.data.transform.ColorAugment'> |
├───────────────┼───────────────────────────────────────────────────┤ |
AutoAugment | <class 'basecls.data.transform.AutoAugment'> |
├───────────────┼───────────────────────────────────────────────────┤ |
RandAugment | <class 'basecls.data.transform.RandAugment'> |
╘═══════════════╧═══════════════════════════════════════════════════╛

dataloaders = Registry of dataloaders:
╒═══════════════╤═══════════════════════════════════════════════════╕ | Names |
Objects | ╞═══════════════╪═══════════════════════════════════════════════════╡ |
FakeData | <class 'basecls.data.build.FakeData'> |
├───────────────┼───────────────────────────────────────────────────┤ |
FolderLoader | <class 'basecls.data.build.FolderLoader'> |
╘═══════════════╧═══════════════════════════════════════════════════╛

hooks = Registry of hooks:
╒═══════════════╤═══════════════════════════════════════════════════╕ | Names |
Objects | ╞═══════════════╪═══════════════════════════════════════════════════╡ |
DefaultHooks | <class 'basecls.engine.build.DefaultHooks'> |
╘═══════════════╧═══════════════════════════════════════════════════╛
```

```
models = Registry of models:
╒═══════════════════════════════╤═══════════════════════════════════════════════════════════════════════════════╕
│ Names                         │ Objects                                                                       │
╞═══════════════════════════════╪═══════════════════════════════════════════════════════════════════════════════╡
│ ResNet                        │ <class 'basecls.models.resnet.ResNet'>                                        │
├───────────────────────────────┼───────────────────────────────────────────────────────────────────────────────┤
│ resnet18                      │ <function resnet18>                                                           │
├───────────────────────────────┼───────────────────────────────────────────────────────────────────────────────┤
│ resnet34                      │ <function resnet34>                                                           │
├───────────────────────────────┼───────────────────────────────────────────────────────────────────────────────┤
│ resnet50                      │ <function resnet50>                                                           │
├───────────────────────────────┼───────────────────────────────────────────────────────────────────────────────┤
│ resnet101                     │ <function resnet101>                                                          │
├───────────────────────────────┼───────────────────────────────────────────────────────────────────────────────┤
│ resnet152                     │ <function resnet152>                                                          │
├───────────────────────────────┼───────────────────────────────────────────────────────────────────────────────┤
│ resnet18d                     │ <function resnet18d>                                                          │
├───────────────────────────────┼───────────────────────────────────────────────────────────────────────────────┤
│ resnet34d                     │ <function resnet34d>                                                          │
├───────────────────────────────┼───────────────────────────────────────────────────────────────────────────────┤
│ resnet50d                     │ <function resnet50d>                                                          │
├───────────────────────────────┼───────────────────────────────────────────────────────────────────────────────┤
│ resnet101d                    │ <function resnet101d>                                                         │
├───────────────────────────────┼───────────────────────────────────────────────────────────────────────────────┤
│ resnet152d                    │ <function resnet152d>                                                         │
├───────────────────────────────┼───────────────────────────────────────────────────────────────────────────────┤
│ resnext50_32x4d               │ <function resnext50_32x4d>                                                    │
├───────────────────────────────┼───────────────────────────────────────────────────────────────────────────────┤
│ resnext101_32x4d              │ <function resnext101_32x4d>                                                   │
├───────────────────────────────┼───────────────────────────────────────────────────────────────────────────────┤
│ resnext101_32x8d              │ <function resnext101_32x8d>                                                   │
├───────────────────────────────┼───────────────────────────────────────────────────────────────────────────────┤
│ resnext101_64x4d              │ <function resnext101_64x4d>                                                   │
├───────────────────────────────┼───────────────────────────────────────────────────────────────────────────────┤
│ resnext152_32x4d              │ <function resnext152_32x4d>                                                   │
├───────────────────────────────┼───────────────────────────────────────────────────────────────────────────────┤
│ resnext152_32x8d              │ <function resnext152_32x8d>                                                   │
├───────────────────────────────┼───────────────────────────────────────────────────────────────────────────────┤
│ resnext152_64x4d              │ <function resnext152_64x4d>                                                   │
├───────────────────────────────┼───────────────────────────────────────────────────────────────────────────────┤
│ se_resnet18                   │ <function se_resnet18>                                                        │
├───────────────────────────────┼───────────────────────────────────────────────────────────────────────────────┤
│ se_resnet34                   │ <function se_resnet34>                                                        │
├───────────────────────────────┼───────────────────────────────────────────────────────────────────────────────┤
│ se_resnet50                   │ <function se_resnet50>                                                        │
├───────────────────────────────┼───────────────────────────────────────────────────────────────────────────────┤
```

```
solvers = Registry of solvers:
╒═══════════════╤═══════════════════════════════════════════════════════╕ | Names |
Objects | ╞═══════════════╪═══════════════════════════════════════════════════════╡
| DefaultSolver | <class 'basecls.solver.build.DefaultSolver'> |
╘═══════════════╧═══════════════════════════════════════════════════════╛

trainers = Registry of trainers:
╒═══════════╤═══════════════════════════════════════════════════╕ | Names |
Objects | ╞═══════════╪═══════════════════════════════════════════════════╡ |
ClsTrainer | <class 'basecls.engine.trainer.ClsTrainer'> |
╘═══════════╧═══════════════════════════════════════════════════╛
```

basecls.utils.**recursive_update**(*d*, *u*)

basecls.utils.env

basecls.utils.env.**set_nccl_env**()

Set NCCL environments, which is essential to multi-node training.

basecls.utils.env.**set_num_threads**(*num=1*)

Set number of threads in OpenMP, OpenCV, MKL, OPENBLAS, VECLIB, NUMEXPR, etc.

参数

num (int) –number of threads. Default: 1

basecls.utils.logger

basecls.utils.logger.**default_logging**(*cfg*, *model*)

basecls.utils.logger.**setup_logger**(*log_path*, *log_file*, *to_loguru=False*)

1.3.9 basecls.zoo

basecls.zoo.benchmark

basecls.zoo.benchmark.**main**()

basecls.zoo.benchmark.**worker**(*args*)

class basecls.zoo.benchmark.**ClsBench**(*model*, *dataloader*, *trace=False*)

基类: object

benchmark(*warm_iters=50*, *log_seconds=2*)

model_step(*samples*, *targets*)

class basecls.zoo.benchmark.**TrainBench**(*model*, *dataloader*, *trace=False*, *amp_version=0*)

基类：*ClsBench*

model_step(*samples*, *targets*)

class basecls.zoo.benchmark.**EvalBench**(*model*, *dataloader*, *trace=False*, *amp_version=0*)

基类：*ClsBench*

model_step(*samples*, *targets*)

basecls.zoo.benchmark_timm

basecls.zoo.testing_all

basecls.zoo.testing_all.**make_parser**()

Build args parser for testing script.

返回类型

ArgumentParser

返回

The args parser.

basecls.zoo.testing_all.**worker**(*args*)

Worker function for testing script.

参数

args (Namespace) –args for testing script.

basecls.zoo.testing_all.**build**(*cfg*)

Build function for testing script.

参数

cfg (ConfigDict) –config for testing.

返回

A tester.

basecls.zoo.testing_all.**main**()

Main function for testing script.

basecls.zoo.utils

class basecls.zoo.utils.**Meta**(*uid*, *series*, *name*, *create_time='2022/12/15,08:58:42'*, *basecls_version='0.4'*, *flops=None*, *params=None*, *activations=None*, *img_size=None*, *acc1=None*, *acc5=None*)

基类: object

uid: str

series: str

name: str

create_time: str = '2022/12/15,08:58:42'

basecls_version: str = '0.4'

flops: int = None

params: int = None

activations: int = None

img_size: int = None

acc1: float = None

acc5: float = None

basecls.zoo.utils.**get_series_and_name_and_id**(*p*)

返回类型

dict

basecls.zoo.utils.**purify_weight**(*src*, *dst=None*)

返回类型

str

备注: 文档还在施工中，如有任何问题通过查询文档难以解决，欢迎联系，也欢迎大家参与贡献 。

CHAPTER 2

致谢

- 感谢每一位使用 BaseCls 的用户，以及来自研究院各组的开发者/贡献者们。
- 感谢 Engine 组在 BaseCls 开发过程中的大力协助。
- 感谢 Sphinx 项目和 PyData 社区，BaseCls 使用了 Sphinx 生成文档，并使用了社区维护的 pydata-sphinx-theme 主题。

Python 模块索引

b

索引

C

D

E

F

G

H

I

K

L

M

N

O

P

R

S

T

U

V

W