evaluation_iccv

October 25, 2021

1 ICCV Evaluation and Plots

1.1 Imports

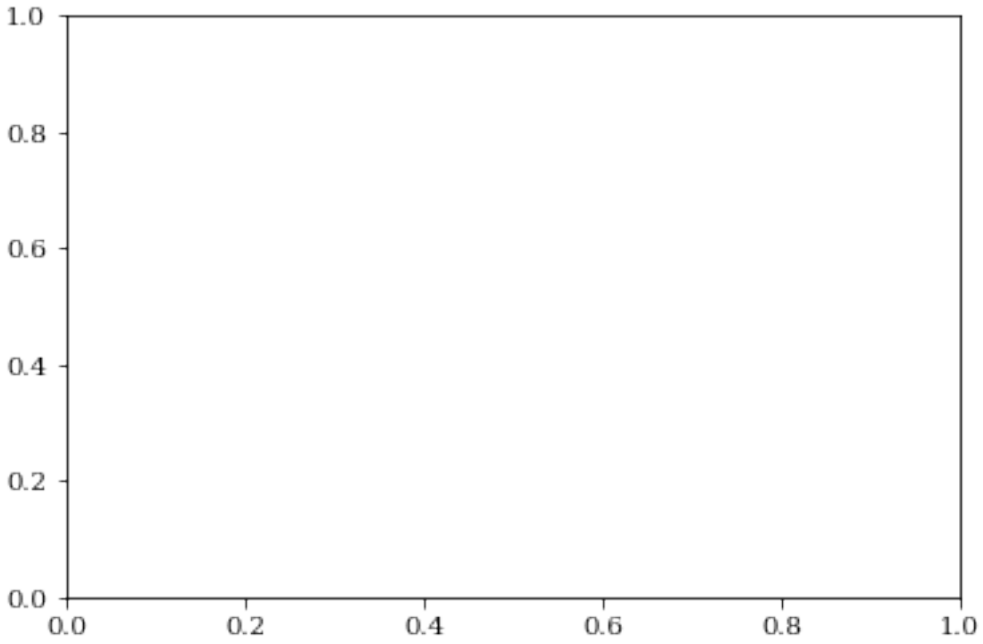
[1]:
```
import os
import sys
%matplotlib inline
sys.path.insert(1, os.path.dirname(os.path.abspath('')) + '/../../')
from IPython.display import display, Markdown
```

[2]:
```
import common.experiments.eval as ev
import experiments.iccv.cifar10_noaa as config
```

```
[251021195113|0/0MiB] [Warning] running in notebook
[251021195117|0/0MiB] read /BS/dstutz/work/data/Cifar10/train_images.h5
[251021195117|0/0MiB] read /BS/dstutz/work/data/Cifar10/train_labels.h5
[251021195118|0/0MiB] read /BS/dstutz/work/data/Cifar10/test_images.h5
[251021195118|0/0MiB] read /BS/dstutz/work/data/Cifar10/test_labels.h5
[251021195120|0/0MiB] read /BS/dstutz/work/data/Cifar10/train_images.h5
[251021195120|0/0MiB] read /BS/dstutz/work/data/Cifar10/train_labels.h5
[251021195120|0/0MiB] read /BS/dstutz/work/data/Cifar10/test_images.h5
[251021195120|0/0MiB] read /BS/dstutz/work/data/Cifar10/test_labels.h5
[251021195120|0/0MiB] read /BS/dstutz/work/data/Cifar10/test_images.h5
[251021195120|0/0MiB] read /BS/dstutz/work/data/Cifar10/test_labels.h5
[251021195121|0/0MiB] read /BS/dstutz/work/data/Cifar10/test_images.h5
[251021195121|0/0MiB] read /BS/dstutz/work/data/Cifar10/test_labels.h5
[251021195123|0/0MiB] read /BS/dstutz/work/data/Cifar10/train_images.h5
[251021195123|0/0MiB] read /BS/dstutz/work/data/Cifar10/train_labels.h5
[251021195125|0/0MiB] read /BS/dstutz/work/data/Cifar10/train_images.h5
[251021195125|0/0MiB] read /BS/dstutz/work/data/Cifar10/train_labels.h5
[251021195127|0/0MiB] read /BS/dstutz/work/data/Cifar10/train_images.h5
[251021195127|0/0MiB] read /BS/dstutz/work/data/Cifar10/train_labels.h5
[251021195127|0/0MiB] loading regular augmentation
[251021195133|0/0MiB] set up attacks …
```

```
/BS/dstutz/work/dev-box/anaconda3/envs/default/lib/python3.8/site-
packages/torch/nn/_reduction.py:44: UserWarning: size_average and reduce args
will be deprecated, please use reduction='sum' instead.
  warnings.warn(warning.format(ret))
```

```
[251021195134|0/0MiB] set up models …
```



[3]:
```
import numpy
import datetime
import terminaltables
import common.summary
import common.numpy
import common.plot
import common.utils
from common.log import log, LogLevel
import matplotlib
from matplotlib import pyplot as plt
from scipy.signal import savgol_filter
from scipy import stats
```

1.2 Style

[4]:
```
plt.style.use('seaborn-bright')
matplotlib.rcParams['figure.dpi'] = 125
matplotlib.rcParams['axes.grid'] = True
matplotlib.rcParams['axes.titlesize'] = 15
matplotlib.rcParams['legend.fontsize'] = 12
matplotlib.rcParams['xtick.labelsize'] = 11
matplotlib.rcParams['ytick.labelsize'] = 11
matplotlib.rcParams['axes.labelsize'] = 14
```

```
matplotlib.rcParams['legend.framealpha'] = 0.5
matplotlib.rcParams['legend.edgecolor'] = 'inherit'
matplotlib.rcParams['legend.facecolor'] = 'white'
matplotlib.rcParams['legend.frameon'] = True
matplotlib.rcParams['legend.fancybox'] = False
matplotlib.rcParams['legend.borderpad'] = 0.2
matplotlib.rcParams['legend.labelspacing'] = 0.4
matplotlib.rcParams['legend.handlelength'] = 1.5
matplotlib.rcParams['legend.handleheight'] = 0.5
matplotlib.rcParams['lines.linewidth'] = 3
matplotlib.rcParams['lines.markersize'] = 6.5
matplotlib.rcParams['mathtext.fontset'] = 'stix'
matplotlib.rcParams['font.family'] = 'STIXGeneral'
matplotlib.rc('text', usetex=True)
```

1.3 Models

[42]:
```
def get_models(subset='', short=True):
    """
    Contains and provides a list of all models or models used for specific␣
↪experiments.

    :param subset: experiment subset
    :type subset: str
    :param short: whether to return short names of models
    :type short: bool
    :return: model config variable names, training suffixes and model labels/
↪names
    :rtype: [str], [str], [str]
    """
    models = [
        [['corr-method-main', 'corr'], '_resnet18_rebn_whiten_64',␣
↪'at_linf_gd_normalized_lr0007_mom0_i7_e00314_f100', 'AT (baseline)'], #
        [['corr-other'], '_resnet18_rebn_whiten_64',␣
↪'at_linf_gd_normalized_lr0007_mom0_i14_e00314_f100', 'PGD-14'], #
        #
        [['corr'], '_resnet18_rebn_whiten_64',␣
↪'at_linf_gd_normalized_lr0007_mom0_i7_e00352_f100', r"Larger $\epsilon{=}9/
↪255$"], #
        # ii
        [['corr-other'], '_resnet18_rebn_whiten_64',␣
↪'at_ii_linf_gd_normalized_lr0007_mom0_i7_e00314_f100', 'Ignore incorrect'], #
        # pll
        [['corr-other'], '_resnet18_rebn_whiten_64',␣
↪'at_pll_linf_gd_normalized_lr0007_mom0_i7_e00314_f100', 'Prevent label␣
↪leaking'],
```

```
        # weight clipping
        [['corr'], '_resnet18_rebn_whiten_64',␣
↪'0005p_at_linf_gd_normalized_lr0007_mom0_i7_e00314_f100', 'Weight clipping␣
↪$w_{max}{=}0.005$', 'Weight clipping'], #
        # label smoothing
        [['corr-other'], '_resnet18_rebn_whiten_64',␣
↪'at_linf_gd_normalized_lr0007_mom0_i7_e00314_f100_ls01', r"Label smoothing␣
↪$\tau{=}0.1$", 'Label smoothing'], #
        [['corr-other'], '_resnet18_rebn_whiten_64',␣
↪'at_linf_gd_normalized_lr0007_mom0_i7_e00314_f100_ls02', r"Label smoothing␣
↪$\tau{=}0.2$", 'Label smoothing'], #
        [['corr'], '_resnet18_rebn_whiten_64',␣
↪'at_linf_gd_normalized_lr0007_mom0_i7_e00314_f100_ls03', r"Label smoothing␣
↪$\tau{=}0.3$", 'Label smoothing'], #
        [['corr-other'], '_resnet18_rebn_whiten_64',␣
↪'at_linf_gd_normalized_lr0007_mom0_i7_e00314_f100_ls04', r"Label smoothing␣
↪$\tau{=}0.4$", 'Label smoothing'], #
        [['corr-other'], '_resnet18_rebn_whiten_64',␣
↪'at_linf_gd_normalized_lr0007_mom0_i7_e00314_f100_ls05', r"Label smoothing␣
↪$\tau{=}0.5$", 'Label smoothing'], #
        # label noise
        [['corr-method-main', 'corr-other'], '_resnet18_rebn_whiten_64',␣
↪'at_linf_gd_normalized_lr0007_mom0_i7_e00314_f100_ln01', r"Label noise␣
↪$\tau{=}0.1$", 'Label noise'], #
        [['corr-method-main', 'corr-other'], '_resnet18_rebn_whiten_64',␣
↪'at_linf_gd_normalized_lr0007_mom0_i7_e00314_f100_ln02', r"Label noise␣
↪$\tau{=}0.2$", 'Label noise'], #
        [['corr-method-main', 'corr-other'], '_resnet18_rebn_whiten_64',␣
↪'at_linf_gd_normalized_lr0007_mom0_i7_e00314_f100_ln03', r"Label noise␣
↪$\tau{=}0.3$", 'Label noise'], #
        [['corr-method-main', 'corr'], '_resnet18_rebn_whiten_64',␣
↪'at_linf_gd_normalized_lr0007_mom0_i7_e00314_f100_ln04', r"Label noise␣
↪$\tau{=}0.4$", 'Label noise'], #
        [['corr-method-main', 'corr-other'], '_resnet18_rebn_whiten_64',␣
↪'at_linf_gd_normalized_lr0007_mom0_i7_e00314_f100_ln05', r"Label noise␣
↪$\tau{=}0.5$", 'Label noise'], #
        # cyc
        [['corr-other'], '_resnet18_rebn_whiten_64',␣
↪'at_linf_gd_normalized_lr0007_mom0_i7_e00314_f100_cyc', 'Cyclic', 'Cyclic'],
        # weight decay
        [['corr-other'], '_resnet18_rebn_whiten_64',␣
↪'at_linf_gd_normalized_lr0007_mom0_i7_e00314_f100_wd0001', 'Weight decay $0.
↪001$', 'Weight decay'], #
        [['corr-other'], '_resnet18_rebn_whiten_64',␣
↪'at_linf_gd_normalized_lr0007_mom0_i7_e00314_f100_wd001', 'Weight decay $0.
↪01$', 'Weight decay'], #
```

```
        [['corr'], '_resnet18_rebn_whiten_64',␣
↪'at_linf_gd_normalized_lr0007_mom0_i7_e00314_f100_wd005', 'Weight decay $0.
↪05$', 'Weight decay'], #
        # ssl
        [['corr-other'], '_resnet18_rebn_whiten_64',␣
↪'at_ssl05_linf_gd_normalized_lr0007_mom0_i7_e00314_f100', r"Self-supervision␣
↪$\lambda{=}0.5$", 'Self-supervision'], #
        [['corr-method-main', 'corr-other'], '_resnet18_rebn_whiten_64',␣
↪'at_ssl1_linf_gd_normalized_lr0007_mom0_i7_e00314_f100', r"Self-supervision␣
↪$\lambda{=}1$", 'Self-supervision'], #
        [['corr-method-main', 'corr-other'], '_resnet18_rebn_whiten_64',␣
↪'at_ssl2_linf_gd_normalized_lr0007_mom0_i7_e00314_f100', r"Self-supervision␣
↪$\lambda{=}2$", 'Self-supervision'], #
        [['corr-method-main', 'corr'], '_resnet18_rebn_whiten_64',␣
↪'at_ssl4_linf_gd_normalized_lr0007_mom0_i7_e00314_f100', r"Self-supervision␣
↪$\lambda{=}4$", 'Self-supervision'], #
        [['corr-method-main', 'corr-other'], '_resnet18_rebn_whiten_64',␣
↪'at_ssl8_linf_gd_normalized_lr0007_mom0_i7_e00314_f100', r"Self-supervision␣
↪$\lambda{=}8$", 'Self-supervision'], #
        # trades
        [['corr-method-main', 'corr-other'], '_resnet18_rebn_whiten_64',␣
↪'trades1_linf_gd_normalized_lr0007_mom0_i7_e00314_f100', r"TRADES␣
↪$\lambda{=}1$", 'TRADES'],
        [['corr-method-main', 'corr-other'], '_resnet18_rebn_whiten_64',␣
↪'trades3_linf_gd_normalized_lr0007_mom0_i7_e00314_f100', r"TRADES␣
↪$\lambda{=}3$", 'TRADES'],
        [['corr-method-main', 'corr-other'], '_resnet18_rebn_whiten_64',␣
↪'trades6_linf_gd_normalized_lr0007_mom0_i7_e00314_f100', r"TRADES␣
↪$\lambda{=}6$", 'TRADES'],
        [['corr-method-main', 'corr'], '_resnet18_rebn_whiten_64',␣
↪'trades9_linf_gd_normalized_lr0007_mom0_i7_e00314_f100', r"TRADES␣
↪$\lambda{=}9$", 'TRADES'],
    ]

    training_config_vars = []
    training_suffixes = []
    training_labels = []
    for model in models:
        if subset == '' or subset in model[0]:
            training_label = model[4] if short and len(model) > 4 else model[3]

            training_suffixes.append(model[1])
            training_config_vars.append(model[2])
            training_labels.append(training_label)

    assert len(training_labels) == len(training_config_vars)
```

```
    assert len(training_suffixes) == len(training_config_vars)
    return training_config_vars, training_suffixes, training_labels
```

1.4 introduction

[6]:
```
def iccv_plot_training_single(
    config, training_config, tags, factors=None, labels=None, index=-1,␣
↪reference=None,
    xmin=None, xmax=None, ymin=None, ymax=None, w=None, h=None, **kwargs):
    """
    Plot training (robust) loss throughout training from logs. Assumes that for␣
↪training a PickleSummaryWriter was used.
    """

    log_dir = ev.get_log_directory(config, training_config)
    log_dir += '/logs'
    logs = os.listdir(log_dir)
    logs.sort(key=lambda date: datetime.datetime.strptime(date, "%d%m%y%H%M%S"))

    # Comment out if you started training a couple of times and have multiple␣
↪logs:
    # for log in logs:
    #     print('%s: %s.%s.%s %s:%s:%s' % (
    #         log,
    #         log[0:2],
    #         log[2:4],
    #         log[4:6],
    #         log[6:8],
    #         log[8:10],
    #         log[10:12],
    #     ))

    log_sub_dir = log_dir + '/' + logs[index]
    summary_reader = common.summary.SummaryPickleReader(log_sub_dir)

    for tag in tags:
        assert tag in summary_reader.tags()

    if factors is None:
        factors = [1]*len(tags)
    else:
        assert len(factors) == len(tags)

    if labels is None:
        labels = tags
    else:
        assert len(labels) == len(tags)
```

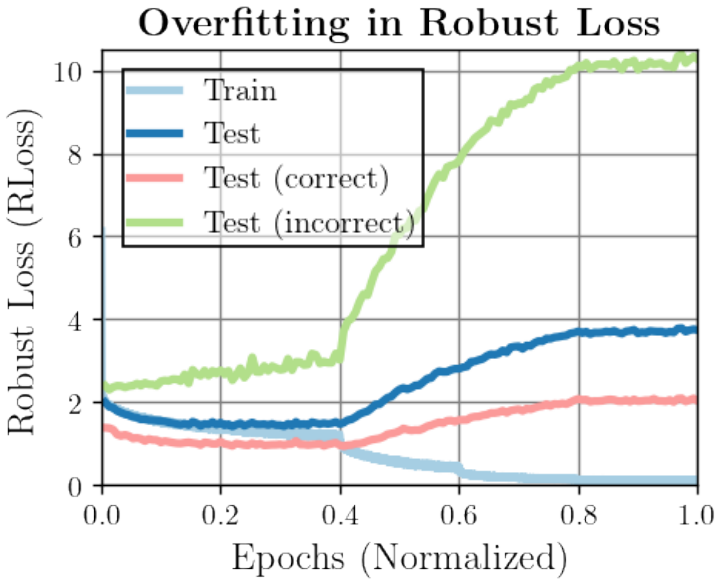
```
    plt.clf()
    ax = plt.gca()
    max_x = 0
    for t in range(len(tags)):
        tag = tags[t]
        factor = factors[t]
        label = labels[t]
        data = numpy.array(summary_reader.get_scalar(tag))
        max_x = max(max_x, numpy.max(factor*data[:, 0]))

        x = data[:, 0]/numpy.max(data[:, 0])
        y = data[:, 1]

        if reference:
            reference_value = numpy.min(y)

        if tag.find('train') >= 0:
            y = savgol_filter(y, 51, 1) # window size 51, polynomial order 3
        ax.plot(x, y,
                label=label, color=common.plot.color_brewer[t])

    if reference:
        ax.plot(
            numpy.array([0, 1]), numpy.array([reference_value,␣
↪reference_value]),
            color=common.plot.color_brewer[len(tags)], label='Early Stopping␣
↪(ES)', linewidth=2)

    common.plot.label(
        ax, label=True, xmin=xmin, xmax=xmax, ymin=ymin, ymax=ymax, w=w, h=h,␣
↪legend=True, **kwargs)
    plt.show()
```

[7]:
```
training_configs, weight_attack_configs = ev.load(
    config,
    training_config_vars=[
        'at_linf_gd_normalized_lr0007_mom0_i7_e00314_f100',
    ],
    training_suffixes='_resnet18_rebn_whiten_64',
    attack_config_vars=[],
)
index = -1
color_brewer_ = numpy.array([
    [166, 206, 227],
    [31, 120, 180],
    [251, 154, 153],
```

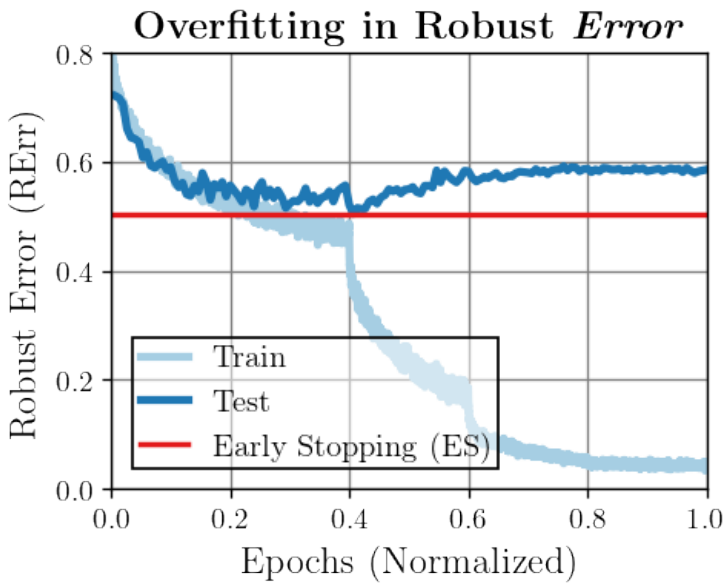
```
    [178, 223, 138],
])
common.plot.color_brewer = color_brewer_/255.
iccv_plot_training_single(
    config, training_configs[0], tags=[
        'train/adversarial_loss',
        'test/adversarial_loss',
        'test/adversarial_correct_loss',
        'test/adversarial_incorrect_loss',
    ], factors=[
        1,
        782,
        782,
        782,
    ], labels=[
        'Train',
        'Test',
        'Test (correct)',
        'Test (incorrect)',
    ], index=index, ymin=0, xmin=0, ymax=10.5, xmax=1, w=4, h=3,
    legend_loc='upper left', legend_anchor=(0.01, 0.99),
    xlabel='Epochs (Normalized)', ylabel='Robust Loss (RLoss)',
    title=r"\textbf{Overfitting in Robust Loss}")

color_brewer_ = numpy.array([
    [166, 206, 227],
    [31, 120, 180],
    [227, 26, 28],
])
common.plot.color_brewer = color_brewer_/255.
iccv_plot_training_single(
    config, training_configs[0], tags=[
        'train/adversarial_error',
        'test/adversarial_error',
    ], factors=[
        1,
        782,
    ], labels=[
        'Train',
        'Test',
    ], index=index, reference=True,
    ymin=0, xmin=0, ymax=0.8, xmax=1, w=4, h=3,
    legend_loc='lower left', legend_anchor=(0.01, 0.01),
    xlabel='Epochs (Normalized)', ylabel='Robust Error (RErr)',
    title=r"\textbf{Overfitting in Robust \emph{Error}}")
```

```
<ipython-input-6-9468bcbb63d3>:26: ResourceWarning: unclosed file
```

```
<_io.BufferedReader name='/BS/dstutz4/nobackup/experiments/ICCV/Cifar10NoAA//at_
linf_gd_normalized_lr0007_mom0_i7_e00314_f100_resnet18_rebn_whiten_64/logs/14102
1200329/events.pkl'>
  summary_reader = common.summary.SummaryPickleReader(log_sub_dir)
ResourceWarning: Enable tracemalloc to get the object allocation traceback
```




```
<ipython-input-6-9468bcbb63d3>:26: ResourceWarning: unclosed file
<_io.BufferedReader name='/BS/dstutz4/nobackup/experiments/ICCV/Cifar10NoAA//at_
linf_gd_normalized_lr0007_mom0_i7_e00314_f100_resnet18_rebn_whiten_64/logs/14102
1200329/events.pkl'>
  summary_reader = common.summary.SummaryPickleReader(log_sub_dir)
ResourceWarning: Enable tracemalloc to get the object allocation traceback
```



1.5 Visualization and Hessian Eigenvalues

1.5.1 Random Directions

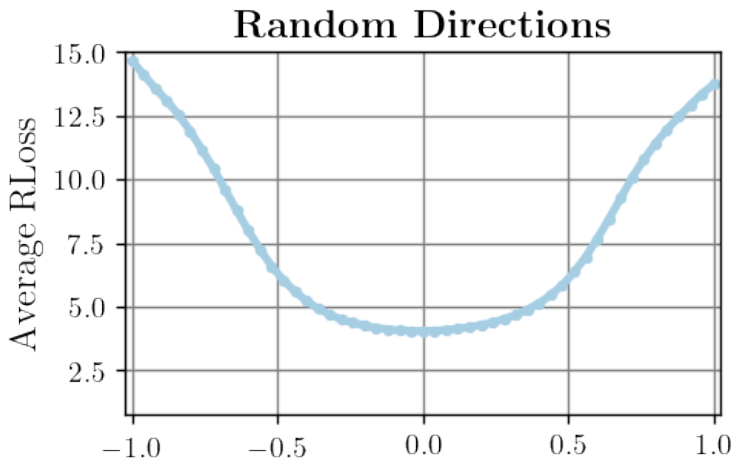
[8]:
```
def iccv_plot_weight_1d_compare(
    config, training_configs, attack_config, loss_attack=None, normalization='',
    statistic='mean', clip=2.5, plot_log=False, epoch=None, adversarial=False,
    labels=None, legend=True, errors=False, flip=False, **kwargs):
    """
    Plot robust loss along a random or adversarial direction according to the␣
↪provided attack_config.
    """

    if labels is None:
        labels =  [training_config.directory for training_config in␣
↪training_configs]

    xs = None
    ys = None

    for training_config in training_configs:
        visualization_directory = common.paths.experiment_dir('%s/
↪%s_%s_%svisualization' % (
            training_config.directory, attack_config.directory,
            normalization, 'adversarial_' if adversarial else '',
        ))
        if loss_attack is not None:
```

```
            visualization_directory += '/%s' % loss_attack.directory
        visualization_file = os.path.join(visualization_directory,␣
↪'visualization%s' % common.paths.HDF5_EXT)
        if epoch is not None:
            visualization_file += '.%d' % epoch

        if not os.path.exists(visualization_file):
            log('file %s not found' % visualization_file, LogLevel.ERROR)
            continue;

        steps = common.utils.read_hdf5(visualization_file, 'steps')
        losses = common.utils.read_hdf5(visualization_file, 'losses')
        factors = common.utils.read_hdf5(visualization_file, 'factors')

        if errors:
            losses = common.utils.read_hdf5(visualization_file, 'errors')
        else:
            if plot_log:
                losses = numpy.log(1 + losses)
            else:
                losses = numpy.clip(losses, 0, clip)

        x = getattr(numpy, statistic)(steps, axis=0)
        y = getattr(numpy, statistic)(losses, axis=0)

        xs = common.numpy.concatenate(xs, numpy.expand_dims(x, axis=0))
        ys = common.numpy.concatenate(ys, numpy.expand_dims(y, axis=0))

    if flip:
        ys = numpy.flip(ys, axis=1)

    plt.clf()
    if legend:
        common.plot.line(
            xs, ys, labels=labels, ax=plt.gca(), markers=['.']*xs.shape[0],
            markersize=matplotlib.rcParams['lines.markersize'],
            linewidth=matplotlib.rcParams['lines.linewidth'], **kwargs)
    else:
        common.plot.line(
            xs, ys, labels=None, ax=plt.gca(), markers=['.']*xs.shape[0],
            markersize=matplotlib.rcParams['lines.markersize'],
            linewidth=matplotlib.rcParams['lines.linewidth'], **kwargs)
        plt.gca().get_legend().remove()
    plt.show()
```

[9]:
```
matplotlib.rcParams['lines.markersize'] = 6.5
color_brewer_ = numpy.array([
```

```
    [166, 206, 227],
], dtype=float)
common.plot.color_brewer = color_brewer_/255.
common.plot.marker_brewer = ['.']*1
training_configs, weight_attack_configs = ev.load(config,
    training_config_vars=[
        'at_linf_gd_normalized_lr0007_mom0_i14_e00314_f100',
    ],
    training_suffixes=[
        '_resnet18_rebn_whiten_64'
    ],
    attack_config_vars=[
        'weight_l2_random_nonorm2_e01_at10'
    ],
)
input_attacks_configs = ev.load_input(config, attack_config_vars=[
    'input_linf_gd_normalized_lr0007_mom0_i10_e00314_at10',
])
normalization = 'layer_l2_05'
iccv_plot_weight_1d_compare(
    config, training_configs, weight_attack_configs[0],
    ylabel='Average RLoss',
    loss_attack=input_attacks_configs[0],
    normalization=normalization, plot_log=False, clip=100, h=2.5, w=4,
    legend_loc='upper right', legend_anchor=(-0.125, 1.015),
    adversarial=True, xmin=-1.025, xmax=1.025, ymin=0.75, ymax=15,
    title=r"\textbf{Random Directions}", save='main_random', legend=False)
```

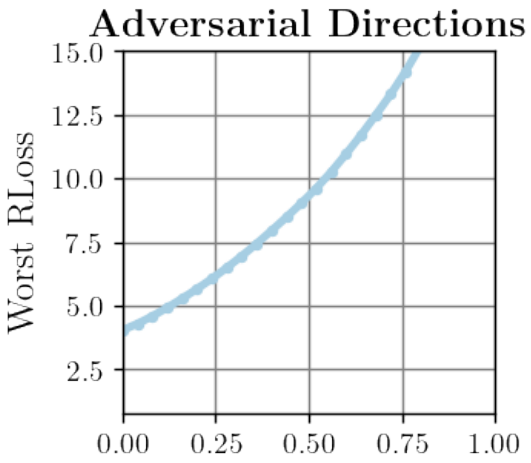
```
No handles with labels found to put in legend.
```



1.5.2 Adversarial Directions

[10]:
```
color_brewer_ = numpy.array([
    [166, 206, 227],
], dtype=float)
common.plot.color_brewer = color_brewer_/255.
common.plot.marker_brewer = ['.']*1
training_configs, weight_attack_configs = ev.load(config,
    training_config_vars=[
        'at_linf_gd_normalized_lr0007_mom0_i14_e00314_f100',
    ],
    training_suffixes=[
        '_resnet18_rebn_whiten_64'
    ],
    attack_config_vars=[
        'weight_l2_gd_nonorm2_lwrl2normalized_i7_lr001_mom0_e0005_at10_test'
    ],
)
input_attacks_configs = ev.load_input(config, attack_config_vars=[
    'input_linf_gd_normalized_lr0007_mom0_i10_e00314_at10',
])
normalization = 'layer_l2_001'
iccv_plot_weight_1d_compare(
    config, training_configs, weight_attack_configs[0],
    ylabel='Worst RLoss',
    loss_attack=input_attacks_configs[0],
    normalization=normalization, plot_log=False, clip=100, h=2.5, w=2.5,
    legend_loc='upper left', legend_anchor=(0.01, 0.99),
    adversarial=True, xmin=0, xmax=1, ymin=0.75, ymax=15,
    title=r"\textbf{Adversarial Directions}",
    save='main_adversarial', legend=False)
```

```
No handles with labels found to put in legend.
```



1.5.3 Hessian Eigenvalues

[11]:
```
def iccv_get_hessian_table(
    config, training_configs, k, test_input_evaluations,
    train_input_evaluations, training_labels=None):
    """
    Print Table of test and train robustness and Hessian eigenvalues.
    """

    table_data = []
    if training_labels is not None:
        assert len(training_labels) == len(training_configs)

    table_data.append([
        '**Model**',
        '**Test**',
        '**Train**',
        '**MAX**',
        '**abs(MIN)/MAX**',
    ])

    if training_labels is None:
        training_labels = [training_configs[t].directory for t in␣
↪range(len(training_configs))]

    for t in range(len(training_configs)):
        training_config = training_configs[t]
        eigs_file = common.paths.experiment_file(
            training_config.directory, 'eigenvalues_%d' % k, common.paths.
↪PICKLE_EXT)
        eigs = common.utils.read_pickle(eigs_file)

        test_rte = 0
        train_rte = 0
        if test_input_evaluations[t][0] is not None:
            test_rte = round(test_input_evaluations[t][0].robust_test_error(),␣
↪4)*100
        if train_input_evaluations[t][0] is not None:
            train_rte = round(train_input_evaluations[t][0].
↪robust_test_error(), 4)*100
        diff_rte = test_rte - train_rte
        table_data.append([
            training_labels[t],
            '%.1f' % test_rte,
```

```
            '%g (%g)' % (train_rte, diff_rte),
            '%g' % eigs[-1],
            '%.3f' % (abs(eigs[0])/eigs[-1]),
        ])

    table = terminaltables.GithubFlavoredMarkdownTable(table_data)

    return table.table
```

[12]:
```
training_configs, weight_attack_configs = ev.load(
    config,
    training_config_vars=[
        'at_linf_gd_normalized_lr0007_mom0_i14_e00314_f100',
    ],
    training_suffixes=[
        '_resnet18_rebn_whiten_64'
    ],
    attack_config_vars=[],
)
input_attacks_configs = ev.load_input(config, attack_config_vars=[
    'input_linf_aa_standard_e00314',
])
input_evaluations, input_epochs = ev.get_input_attack_evaluations(
    config, training_configs, input_attacks_configs, limit=10000)
train_input_attacks_configs = ev.load_input(config, attack_config_vars=[
    'input_linf_aa_standard_e00314_train',
])
train_input_evaluations, train_input_epochs = ev.get_input_attack_evaluations(
    config, training_configs, train_input_attacks_configs, limit=10000,␣
 ↪train=True)
display(Markdown(iccv_get_hessian_table(
    config, training_configs, 4, input_evaluations, train_input_evaluations)))
```

Model	Test	Train	MAX abs(MIN)/MAX
ICCV/Cifar10NoAA//at_linf_gd_normalized_lr0007_mom0_i14_e00314_f100_resnet18_rebn_whiten_64	60.50 (50.7)	49.8	2067.110

1.6 Robust Loss vs. Error

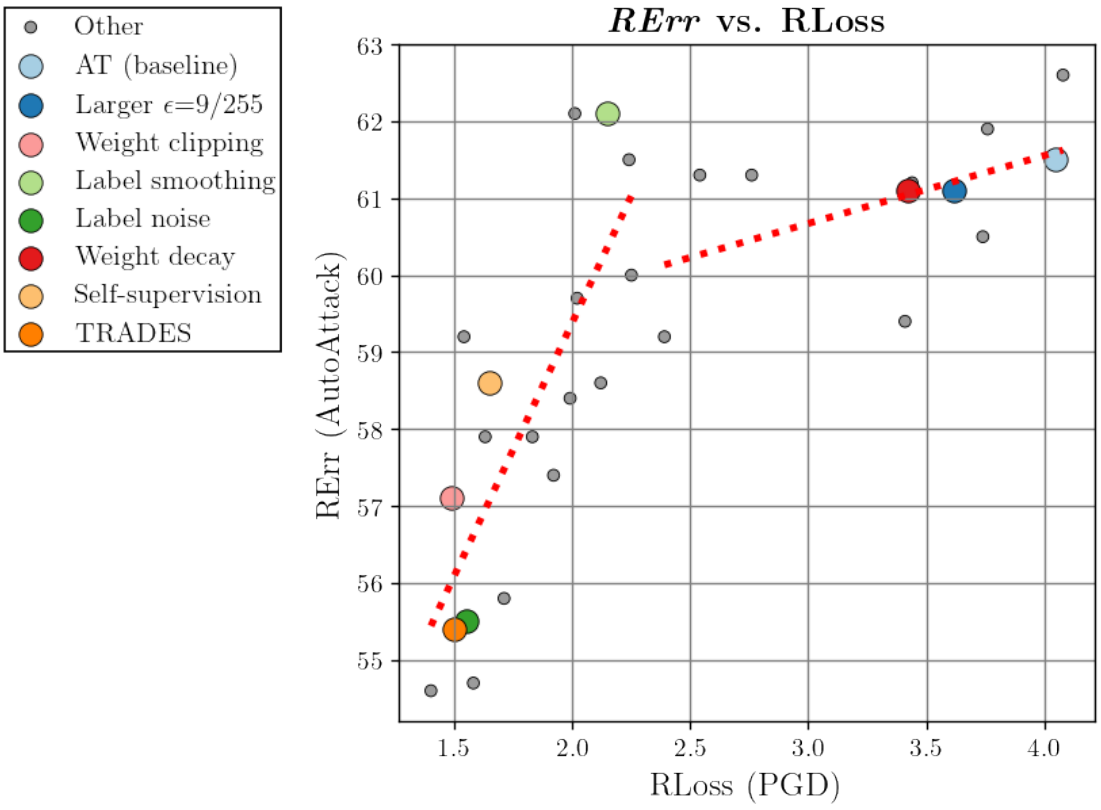
[13]:
```
def iccv_loss_err(
    config, training_configs, input_evaluations, aa_input_evaluations,
    legend_ncol=1, labels=None, legend=True, **kwargs):
    """
    Plot robust error against robust loss.
    """
```

```
    assert len(aa_input_evaluations) == len(input_evaluations)
    assert len(input_evaluations[0]) == 1
    assert len(aa_input_evaluations[0]) == 1

    x = []
    y = []
    c = []
    clean_labels = []

    if labels is not None:
        set_labels = False
    else:
        set_labels = True

    c_ = 0
    mapping = dict()
    for t in range(len(training_configs)):
        if input_evaluations[t][0] is not None and aa_input_evaluations[t][0]␣
↪is not None:
            y.append(round(getattr(aa_input_evaluations[t][0],␣
↪'robust_test_error')(), 4)*100)
            x.append(round(getattr(input_evaluations[t][0], 'robust_loss')(),␣
↪2))

            if labels is None:
                clean_labels.append(training_configs[t].directory)
                c.append(c_)
                c_ += 1
            else:
                if labels[t] not in mapping.keys():
                    c.append(c_)
                    mapping[labels[t]] = c_
                    c_ += 1
                    clean_labels.append(labels[t])
                else:
                    c.append(mapping[labels[t]])

    x = numpy.array(x)
    y = numpy.array(y)
    c = numpy.array(c)

    plt.clf()
    common.plot.scatter(
        x, y, c=c, labels=clean_labels,
        s=matplotlib.rcParams['lines.markersize'], **kwargs, ax=plt.gca())
    legend_ = plt.gca().legend(
        ncol=legend_ncol, loc=kwargs.get('legend_loc', None),
```

```
        bbox_to_anchor=kwargs.get('legend_anchor', None))
    legend_.get_frame().set_alpha(None)
    legend_.get_frame().set_facecolor((1, 1, 1, 0.5))

    x1 = x[x < 2.3]
    y1 = y[x < 2.3]
    x2 = x[x >= 2.3]
    y2 = y[x >= 2.3]

    res1 = stats.linregress(x1, y1)
    res2 = stats.linregress(x2, y2)
    xs1 = numpy.array([numpy.min(x1), numpy.max(x1)])
    plt.gca().plot(
        xs1, res1.intercept + res1.slope*xs1,
        color='red', marker=None, linewidth=3, linestyle=':')
    xs2 = numpy.array([numpy.min(x2), numpy.max(x2)])
    plt.gca().plot(
        xs2, res2.intercept + res2.slope*xs2,
        color='red', marker=None, linewidth=3, linestyle=':')

    if not legend:
        plt.gca().get_legend().remove()
    plt.show()
```

[14]:
```
matplotlib.rcParams['lines.markersize'] = 100
color_brewer_ = numpy.array([
    [155, 155, 155],
    #
    [166, 206, 227],
    [31, 120, 180],
    [251, 154, 153],
    [178, 223, 138],
    [51, 160, 44],
    [227, 26, 28],
    [253, 191, 111],
    [255, 127, 0],

], dtype=float)
common.plot.color_brewer = color_brewer_/255.
common.plot.marker_brewer = ['.']
common.plot.marker_brewer += ['o']*8

training_config_vars1, training_suffixes1, _ = get_models('corr-other')
training_config_vars2, training_suffixes2, training_labels2 = get_models('corr')
assert common.plot.color_brewer.shape[0] == len(training_config_vars2) + 1,␣
 ↪common.plot.color_brewer.shape[0]
```

```
assert len(common.plot.marker_brewer) == len(training_config_vars2) + 1,␣
 ↪len(common.plot.marker_brewer)

training_config_vars = training_config_vars1 + training_config_vars2
training_suffixes = training_suffixes1 + training_suffixes2
training_labels = ['Other']*len(training_config_vars1) + training_labels2

training_configs, weight_attack_configs = ev.load(
    config,
    training_config_vars=training_config_vars,
    training_suffixes=training_suffixes,
    attack_config_vars=[],
)
input_attack_configs = ev.load_input(
    config, attack_config_vars=[
        [
            'input_linf_gd_normalized_lr0007_mom0_i20_e00314_at10',
        ],
        [
            'input_linf_aa_standard_e00314',
        ],
    ])
input_evaluations, _ = ev.get_input_attack_evaluations(
    config, training_configs, [input_attack_configs[0]], validation=0,␣
 ↪train=False)
aa_input_evaluations, _ = ev.get_input_attack_evaluations(
    config, training_configs, [input_attack_configs[1]], validation=0,␣
 ↪train=False)
iccv_loss_err(
    config, training_configs, input_evaluations, aa_input_evaluations,
    save='main_loss_error', legend_ncol=1, labels=training_labels, legend=True,
    legend_loc='upper right', legend_anchor=(-0.15, 1.075), h=5.3, w=5.3,
    title=r"\textbf{\emph{RErr} vs. RLoss}",
    ylabel='RErr (AutoAttack)', xlabel='RLoss (PGD)')
```



1.7 Robust Flatness

1.7.1 Robust Flatness throughout Training

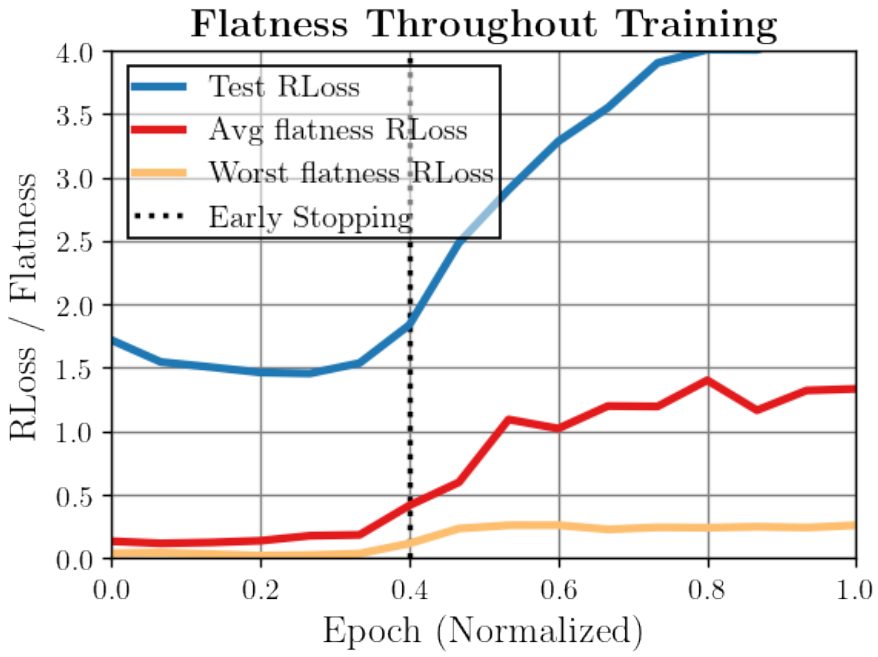
[25]:
```
def iccv_plot_flatness_epochs(
    config, training_config, weight_input_evaluations1,␣
↪weight_input_evaluations2,
    input_evaluations, train_input_evaluations, k=10, epochs=list(range(0, 150,␣
↪10)) + [None], **kwargs):
    """
    Plot robust loss and flatness over epochs.
    """

    assert len(weight_input_evaluations1) == len(epochs)
    assert len(weight_input_evaluations2) == len(epochs)
    assert len(input_evaluations) == len(epochs)

    eigs = None
    x = []
    y4 = []
    y5 = []
```

```
    y7 = []
    for e in range(len(epochs)):
        epoch = epochs[e]
        factor = 1
        if weight_input_evaluations1[e][0] is not None:
            y4.append(round(weight_input_evaluations1[e][0]('robust_loss',␣
↪'mean')[0], 4)*factor -\
                      round(input_evaluations[e][0].robust_loss(), 4)*factor)
        if weight_input_evaluations2[e][0] is not None:
            y5.append(round(weight_input_evaluations2[e][0]('robust_loss',␣
↪'max')[0], 4)*factor -\
                      round(input_evaluations[e][0].robust_loss(), 4)*factor)
        if input_evaluations[e][0] is not None:
            y7.append(round(input_evaluations[e][0].robust_loss(), 4)*factor)
        x.append((epoch if epoch is not None else 150)/150.)

    y4 = numpy.array(y4)
    y5 = numpy.array(y5)
    y7 = numpy.array(y7)
    x = numpy.array(x)

    plt.clf()
    ax = plt.gca()
    ax.plot(x, y7, label='Test RLoss', color=common.plot.color_brewer[1])
    ax.plot(x, y4, label=r"Avg flatness RLoss",
            color=common.plot.color_brewer[5])
    ax.plot(x, y5, label=r"Worst flatness RLoss",
            color=common.plot.color_brewer[6])

    ax.vlines([0.4], 0, 4, color='black', linewidth=2, linestyle=':',␣
↪label='Early Stopping')

    legend_ = ax.legend(loc=kwargs.get('legend_loc', None),␣
↪bbox_to_anchor=kwargs.get('legend_anchor', None))
    legend_.get_frame().set_alpha(None)
    legend_.get_frame().set_facecolor((1, 1, 1, 0.5))
    common.plot.label(ax, **kwargs)
    plt.show()
```

[26]:
```
color_brewer_ = numpy.array([
    [166, 206, 227],
    [31, 120, 180],
    [251, 154, 153],
    [178, 223, 138],
    [51, 160, 44],
    [227, 26, 28],
    [253, 191, 111],
```

```
], dtype=float)
common.plot.color_brewer = color_brewer_/255.
common.plot.marker_brewer = ['.']*6

training_configs, weight_attack_configs = ev.load(
    config,
    training_config_vars=[
        'at_linf_gd_normalized_lr0007_mom0_i7_e00314_f100',
    ],
    training_suffixes='_resnet18_rebn_whiten_64',
    attack_config_vars=[],
)
input_attack_configs = ev.load_input(
    config, attack_config_vars=[
        [
            'input_linf_gd_normalized_lr0007_mom0_i20_e00314_at10',
        ],
    ])
input_weight_attacks_configs = ev.load_weight_input(
    config, attack_config_vars=[
        ␣
 ↪['sequential2_weight_input_l2_random_nonorm2_e05_linf_gd_normalized_lr0007_e00314_i20_at10_
        ␣
 ↪['joint_weight_input_l2_gd_nonorm2_lwrl2normalized_lr001_e000075_linf_gd_normalized_lr0007_
    ])

input_evaluations = []
weight_input_evaluations1 = []
weight_input_evaluations2 = []
weight_input_evaluations3 = []
epochs = list(range(5, 150, 10)) + [None]
for epoch in epochs:
    input_evaluation, _ = ev.get_input_attack_evaluations(
        config, training_configs, [input_attack_configs[0]], validation=0,␣
 ↪train=False, epoch=epoch)
    weight_input_evaluation1, _ = ev.get_weight_input_attack_evaluations(
        config, training_configs, [input_weight_attacks_configs[0]],␣
 ↪train=False, epoch=epoch)
    weight_input_evaluation2, _ = ev.get_weight_input_attack_evaluations(
        config, training_configs, [input_weight_attacks_configs[1]],␣
 ↪train=False, epoch=epoch)
    input_evaluations.append(input_evaluation[0])
    weight_input_evaluations1.append(weight_input_evaluation1[0])
    weight_input_evaluations2.append(weight_input_evaluation2[0])

iccv_plot_flatness_epochs(
```

```
    config, training_configs[0], weight_input_evaluations1,␣
↪weight_input_evaluations2,
    input_evaluations, train_input_evaluations, k=2,␣
↪save='main_flatness_epochs',
    xmin=0, xmax=1, ymin=0, ymax=4, h=3.5, w=5,
    legend_loc='upper left', legend_anchor=(0, 1),
    ylabel='RLoss / Flatness', xlabel='Epoch (Normalized)',
    title=r"\textbf{Flatness Throughout Training}")
```



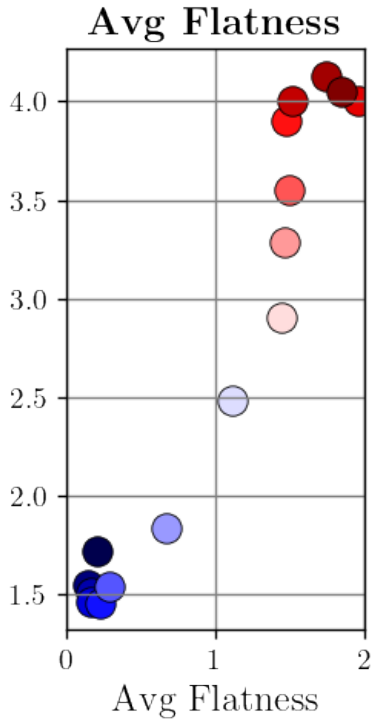
[27]:
```
def iccv_plot_weight_input_correlation2(
    config, training_configs, weight_evaluations, input_evaluations,
    weight_metric='robust_test_error',  weight_statistic='max',␣
↪weight_factor=100,
    input_metric='robust_test_error',␣
↪reference_input_metric='robust_test_error',
    input_factor=100, legend_ncol=1, labels=None, legend=True, zero_x=False,␣
↪regression=False, **kwargs):
    """
    Plot correlation between robust loss and flatness.
    """

    assert len(weight_evaluations) > 0
    assert len(weight_evaluations) == len(input_evaluations)
    assert len(weight_evaluations[0]) == 1
    assert len(input_evaluations[0]) == 1
```

```
    x = []
    y = []
    c = []
    clean_labels = []

    if labels is not None:
        set_labels = False
    else:
        set_labels = True

    c_ = 0
    mapping = dict()
    for t in range(len(training_configs)):
        if weight_evaluations[t][0] is not None and input_evaluations[t][0] is␣
↪not None:
            x.append(round(weight_evaluations[t][0](weight_metric,␣
↪weight_statistic)[0], 4)*weight_factor -\
                     round(getattr(input_evaluations[t][0],␣
↪reference_input_metric)(), 4)*input_factor)
            y.append(round(getattr(input_evaluations[t][0], input_metric)(),␣
↪4)*input_factor)

            if labels is None:
                clean_labels.append(training_configs[t].directory)
                c.append(c_)
                c_ += 1
            else:
                if labels[t] not in mapping.keys():
                    c.append(c_)
                    mapping[labels[t]] = c_
                    c_ += 1
                    clean_labels.append(labels[t])
                else:
                    c.append(mapping[labels[t]])

    x = numpy.array(x)
    if zero_x:
        x = numpy.maximum(numpy.zeros(x.shape), x)
    y = numpy.array(y)
    c = numpy.array(c)

    plt.clf()
    if regression:
        res = stats.linregress(x, y)
        xs1 = numpy.array([numpy.min(x), numpy.max(x)])
        plt.gca().plot(xs1, res.intercept + res.slope*xs1,
```

```
                        color='red', marker=None, linewidth=3, linestyle=':')

    common.plot.scatter(x, y, c=c, labels=clean_labels,
                        s=matplotlib.rcParams['lines.markersize'], **kwargs,␣
 ↪ax=plt.gca())
    legend_ = plt.gca().legend(ncol=legend_ncol, loc=kwargs.get('legend_loc',␣
 ↪None), bbox_to_anchor=kwargs.get('legend_anchor', None))
    legend_.get_frame().set_alpha(None)
    legend_.get_frame().set_facecolor((1, 1, 1, 0.5))
    if not legend:
        plt.gca().get_legend().remove()
    plt.show()
```

[28]:
```
cmap = matplotlib.cm.get_cmap('seismic')
rgba = cmap(0.5)
color_brewer_ = numpy.array([cmap(t)[:3] for t in numpy.linspace(0, 1, 16)])
common.plot.color_brewer = color_brewer_
common.plot.marker_brewer = ['o']*len(common.plot.color_brewer)
matplotlib.rcParams['lines.markersize'] = 125

training_configs, weight_attack_configs = ev.load(
    config,
    training_config_vars=[
        'at_linf_gd_normalized_lr0007_mom0_i7_e00314_f100',
    ],
    training_suffixes=[
        '_resnet18_rebn_whiten_64',
    ],
    attack_config_vars=[],
)
input_attack_configs = ev.load_input(
    config, attack_config_vars=[
        [
            'input_linf_gd_normalized_lr0007_mom0_i20_e00314_at10',
        ],
    ])
input_weight_attacks_configs = ev.load_weight_input(
    config, attack_config_vars=[
        ␣
 ↪['sequential2_weight_input_l2_random_nonorm2_e05_linf_gd_normalized_lr0007_e00314_i20_at10_
        ␣
 ↪#['joint_weight_input_l2_gd_nonorm2_lwrl2normalized_lr001_e000075_linf_gd_normalized_lr0007
    ])

epochs = list(range(5, 150, 10)) + [None]
training_labels = ['%s' % epoch for epoch in epochs]
input_evaluations = []
```

```
weight_input_evaluations = []
for epoch in epochs:
    input_evaluation, _ = ev.get_input_attack_evaluations(
        config, training_configs, input_attack_configs, validation=0,␣
 ↪train=False, epoch=epoch)
    weight_input_evaluation, _ = ev.get_weight_input_attack_evaluations(
        config, training_configs, input_weight_attacks_configs, train=False,␣
 ↪epoch=epoch)
    input_evaluations.append(input_evaluation[0])
    weight_input_evaluations.append(weight_input_evaluation[0])

iccv_plot_weight_input_correlation2(
    config, [training_configs[0]]*len(epochs), weight_input_evaluations,␣
 ↪input_evaluations,
    weight_metric='robust_loss', weight_factor=1, weight_statistic='max',
    input_metric='robust_loss', input_factor=1,␣
 ↪reference_input_metric='robust_loss',
    legend_loc='upper left', labels=training_labels,
    legend_anchor=(1.05, 1), h=4, w=2, xmin=0, xmax=2,
    xlabel='Avg Flatness', legend=False, title=r"\textbf{Avg Flatness}")
```



1.7.2 Robust Flatness and Hyper-Parameters

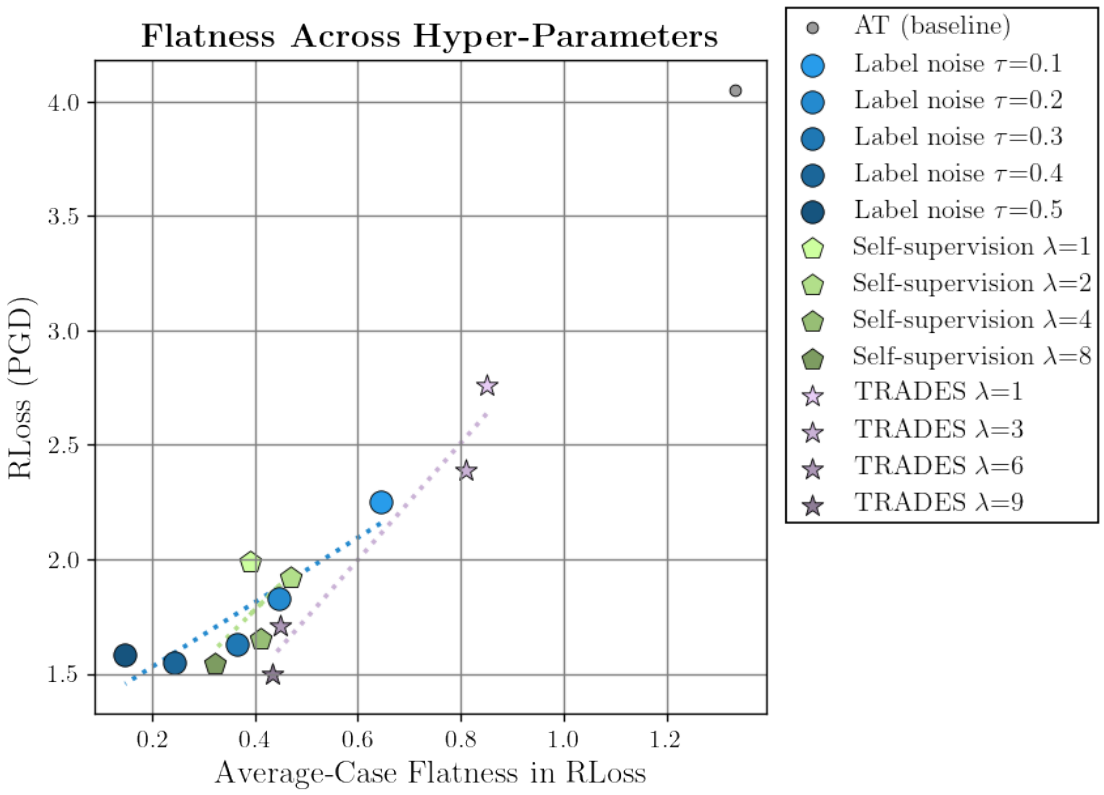
[43]:
```
def iccv_plot_weight_input_correlation2_methods(
    config, training_configs, weight_evaluations, input_evaluations,
    weight_metric='robust_test_error',  weight_statistic='max',␣
↪weight_factor=100,
    input_metric='robust_test_error',␣
↪reference_input_metric='robust_test_error',
    input_factor=100, legend_ncol=1, labels=None, legend=True, zero_x=False,␣
↪groups=None, **kwargs):
    """
    Plot robust loss and flatness correlation for a set of methods with␣
↪different hyper-parameters.
    """

    assert groups is not None
    assert len(weight_evaluations) > 0
    assert len(weight_evaluations) == len(input_evaluations)
    assert len(weight_evaluations[0]) == 1
    assert len(input_evaluations[0]) == 1

    x = []
    y = []
    c = []
    clean_labels = []

    if labels is not None:
        set_labels = False
    else:
        set_labels = True

    c_ = 0
    mapping = dict()
    line_xs = dict()
    line_ys = dict()
    for t in range(len(training_configs)):
        if weight_evaluations[t][0] is not None and input_evaluations[t][0] is␣
↪not None:
            x_val = round(weight_evaluations[t][0](weight_metric,␣
↪weight_statistic)[0], 4)*weight_factor -\
                    round(getattr(input_evaluations[t][0],␣
↪reference_input_metric)(), 4)*input_factor
            y_val = round(getattr(input_evaluations[t][0], input_metric)(),␣
↪4)*input_factor
            x.append(x_val)
            y.append(y_val)
```

```
            if groups[t] not in line_xs.keys():
                line_xs[groups[t]] = []
                line_ys[groups[t]] = []
            line_xs[groups[t]].append(x_val)
            line_ys[groups[t]].append(y_val)

            if labels is None:
                clean_labels.append(training_configs[t].directory)
                c.append(c_)
                c_ += 1
            else:
                if labels[t] not in mapping.keys():
                    c.append(c_)
                    mapping[labels[t]] = c_
                    c_ += 1
                    clean_labels.append(labels[t])
                else:
                    c.append(mapping[labels[t]])

    x = numpy.array(x)
    if zero_x:
        x = numpy.maximum(numpy.zeros(x.shape), x)
    y = numpy.array(y)
    c = numpy.array(c)

    plt.clf()
    common.plot.scatter(x, y, c=c, labels=clean_labels,
                        s=matplotlib.rcParams['lines.markersize'], **kwargs,␣
↪ax=plt.gca())

    for i in range(numpy.max(groups) + 1):
        res = stats.linregress(line_xs[i], line_ys[i])

        color_i = groups.index(i) + 1
        line_x = numpy.array(line_xs[i])
        plt.gca().plot(
            line_x, res.intercept + res.slope*line_x, color=common.plot.
↪color_brewer[color_i],
            marker=None, linewidth=2, linestyle=':', zorder=-1)

    legend_ = plt.gca().legend(
        ncol=legend_ncol, loc=kwargs.get('legend_loc', None),
        bbox_to_anchor=kwargs.get('legend_anchor', None))
    legend_.get_frame().set_alpha(None)
    legend_.get_frame().set_facecolor((1, 1, 1, 0.5))
    if not legend:
        plt.gca().get_legend().remove()
```

```
    plt.show()
```

[44]:
```
matplotlib.rcParams['lines.markersize'] = 100
matplotlib.rcParams['legend.fontsize'] = 12
color_brewer_ = numpy.array([
    [155, 155, 155],
    #
    common.plot.darken([31, 120, 180], 1.3),
    common.plot.darken([31, 120, 180], 1.15),
    [31, 120, 180],
    common.plot.lighten([31, 120, 180], 0.85),
    common.plot.lighten([31, 120, 180], 0.7),
    #
    #common.plot.darken([178, 223, 138], 1.3),
    common.plot.darken([178, 223, 138], 1.15),
    [178, 223, 138],
    common.plot.lighten([178, 223, 138], 0.85),
    common.plot.lighten([178, 223, 138], 0.7),
    #
    common.plot.darken([202, 178, 214], 1.15),
    [202, 178, 214],
    common.plot.lighten([202, 178, 214], 0.85),
    common.plot.lighten([202, 178, 214], 0.7),

], dtype=float)
color_brewer_ = numpy.minimum(numpy.ones(color_brewer_.shape)*255,␣
 ↪color_brewer_)
common.plot.color_brewer = color_brewer_/255.
common.plot.marker_brewer = [
    '.',
    #
    'o', 'o', 'o', 'o', 'o',
    #
    'p', 'p', 'p', 'p',
    #
    '*', '*', '*', '*',
]
groups = [
    -1,
    0, 0, 0, 0, 0,
    1, 1, 1, 1,
    2, 2, 2, 2,
]
training_config_vars, training_suffixes, training_labels =␣
 ↪get_models('corr-method-main', short=False)
assert common.plot.color_brewer.shape[0] == len(training_config_vars), common.
 ↪plot.color_brewer.shape[0]
```

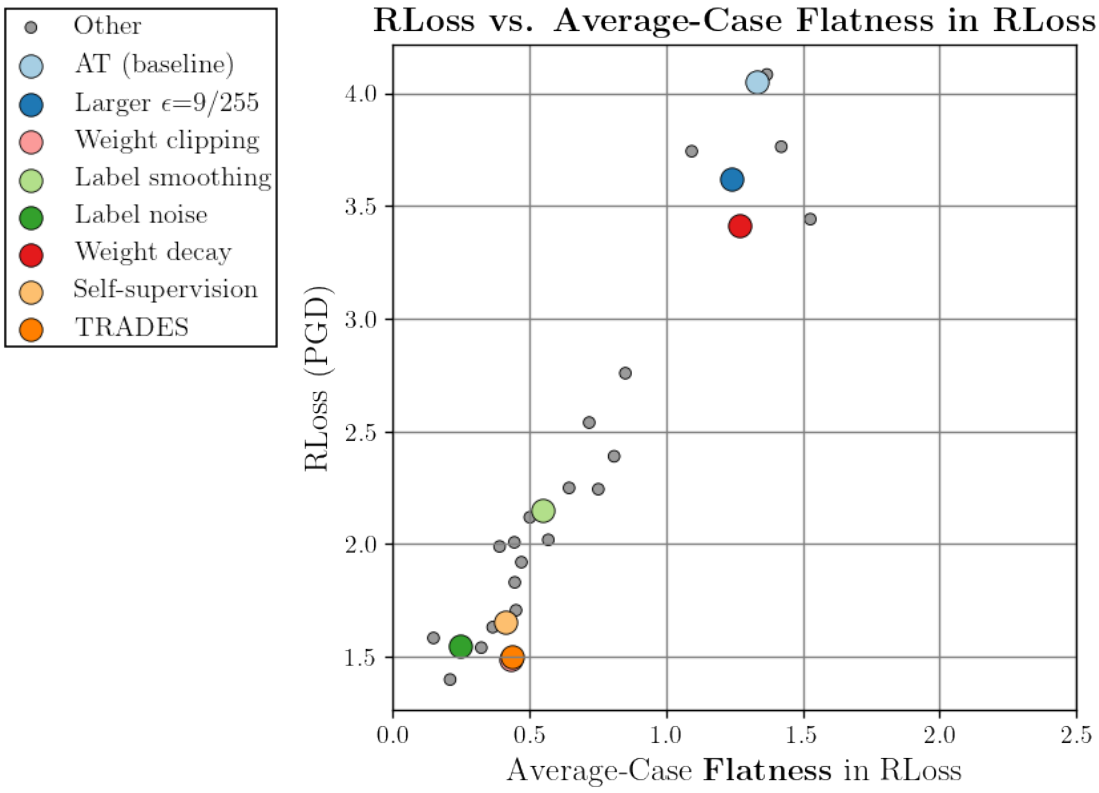
```
assert len(common.plot.marker_brewer) == len(training_config_vars) , len(common.
 ↪plot.marker_brewer)

training_configs, weight_attack_configs = ev.load(
    config,
    training_config_vars=training_config_vars,
    training_suffixes=training_suffixes,
    attack_config_vars=[],
)
input_weight_attacks_configs = ev.load_weight_input(
    config, attack_config_vars=[
['sequential2_weight_input_l2_random_nonorm2_e05_linf_gd_normalized_lr0007_e00314_i20_at10_tes
])
for i in range(len(input_weight_attacks_configs)):
    weight_evaluations, _ = ev.get_weight_input_attack_evaluations(
        config, training_configs, [input_weight_attacks_configs[i]],␣
 ↪train=False)
    input_attack_configs = ev.load_input(config, attack_config_vars=[
        [
            'input_linf_gd_normalized_lr0007_mom0_i20_e00314_at10',
        ],
    ])
    input_evaluations, _ = ev.get_input_attack_evaluations(
        config, training_configs, [input_attack_configs[0]], validation=0,␣
 ↪train=False)
    legend = (i == 0)

    iccv_plot_weight_input_correlation2_methods(
        config, training_configs, weight_evaluations, input_evaluations,
        weight_metric='robust_loss', weight_factor=1, weight_statistic='mean',
        input_metric='robust_loss', reference_input_metric='robust_loss',␣
 ↪input_factor=1,
        legend_loc='upper left', labels=training_labels, groups=groups,
        legend_anchor=(1.01, 1.1), h=5.3, w=5.3,  ymax=4, legend=legend,␣
 ↪legend_ncol=1,
        xlabel=r"Average-Case Flatness in RLoss", ylabel=r"RLoss (PGD)",
        title=r"\textbf{Flatness Across Hyper-Parameters}")
matplotlib.rcParams['legend.fontsize'] = 12
```



1.7.3 Robust Loss vs. Average-Case Flatness

[31]:
```
matplotlib.rcParams['lines.markersize'] = 100
color_brewer_ = numpy.array([
    [155, 155, 155],
    #
    [166, 206, 227],
    [31, 120, 180],
    [251, 154, 153],
    [178, 223, 138],
    [51, 160, 44],
    [227, 26, 28],
    [253, 191, 111],
    [255, 127, 0],
], dtype=float)
common.plot.color_brewer = color_brewer_/255.
common.plot.marker_brewer = ['.']
common.plot.marker_brewer += ['o']*8

training_config_vars1, training_suffixes1, _ = get_models('corr-other')
training_config_vars2, training_suffixes2, training_labels2 = get_models('corr')
```

```
assert common.plot.color_brewer.shape[0] == len(training_config_vars2) + 1,␣
↪common.plot.color_brewer.shape[0]
assert len(common.plot.marker_brewer) == len(training_config_vars2) + 1,␣
↪len(common.plot.marker_brewer)

training_config_vars = training_config_vars1 + training_config_vars2
training_suffixes = training_suffixes1 + training_suffixes2
training_labels = ['Other']*len(training_config_vars1) + training_labels2

training_configs, weight_attack_configs = ev.load(
    config,
    training_config_vars=training_config_vars,
    training_suffixes=training_suffixes,
    attack_config_vars=[],
)
input_weight_attacks_configs = ev.load_weight_input(
    config, attack_config_vars=[
        ␣
↪['sequential2_weight_input_l2_random_nonorm2_e05_linf_gd_normalized_lr0007_e00314_i20_at10_
    ])
for i in range(len(input_weight_attacks_configs)):
    weight_evaluations, _ = ev.get_weight_input_attack_evaluations(
        config, training_configs, [input_weight_attacks_configs[i]],␣
↪train=False)
    input_attack_configs = ev.load_input(
        config, attack_config_vars=[
            [
                'input_linf_gd_normalized_lr0007_mom0_i20_e00314_at10',
            ],
        ])
    input_evaluations, _ = ev.get_input_attack_evaluations(
        config, training_configs, [input_attack_configs[0]], validation=0,␣
↪train=False)

    iccv_plot_weight_input_correlation2(
        config, training_configs,  weight_evaluations, input_evaluations,
        weight_metric='robust_loss', weight_factor=1, weight_statistic='mean',␣
↪input_metric='robust_loss',
        reference_input_metric='robust_loss', input_factor=1,
        legend_loc='upper right', labels=training_labels, legend_anchor=(-0.15,␣
↪1.075), h=5.3, w=5.3,
        xmin=0, xmax=2.5, legend=True,
        xlabel=r"Average-Case \textbf{Flatness} in RLoss", ylabel=r"RLoss␣
↪(PGD)",
        title=r"\textbf{RLoss vs. Average-Case Flatness in RLoss")
```

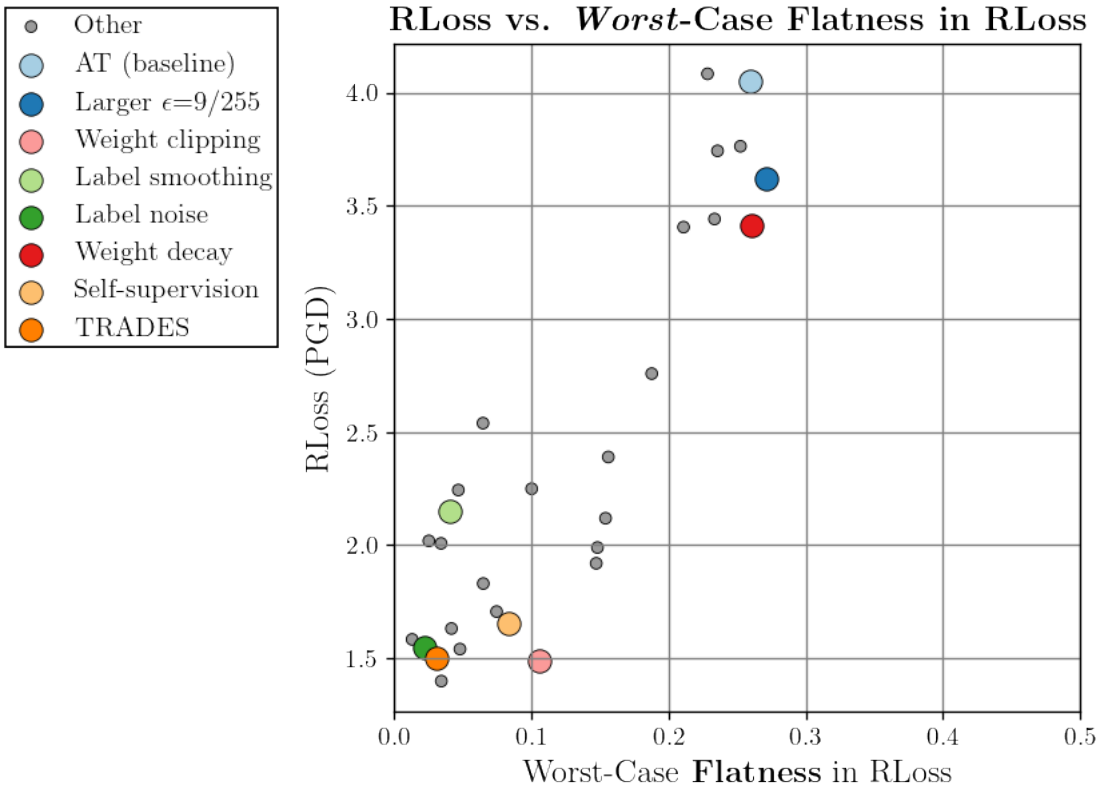


1.7.4 Robust Loss vs. Worst-Case Flatness

[36]:
```
matplotlib.rcParams['lines.markersize'] = 100
color_brewer_ = numpy.array([
    [155, 155, 155],
    #
    [166, 206, 227],
    [31, 120, 180],
    [251, 154, 153],
    [178, 223, 138],
    [51, 160, 44],
    [227, 26, 28],
    [253, 191, 111],
    [255, 127, 0],

], dtype=float)
common.plot.color_brewer = color_brewer_/255.
common.plot.marker_brewer = ['.']
common.plot.marker_brewer += ['o']*8

training_config_vars1, training_suffixes1, _ = get_models('corr-other')
```

```
training_config_vars2, training_suffixes2, training_labels2 = get_models('corr')
assert common.plot.color_brewer.shape[0] == len(training_config_vars2) + 1,␣
↪common.plot.color_brewer.shape[0]
assert len(common.plot.marker_brewer) == len(training_config_vars2) + 1,␣
↪len(common.plot.marker_brewer)

training_config_vars = training_config_vars1 + training_config_vars2
training_suffixes = training_suffixes1 + training_suffixes2
training_labels = ['Other']*len(training_config_vars1) + training_labels2

training_configs, weight_attack_configs = ev.load(
    config,
    training_config_vars=training_config_vars,
    training_suffixes=training_suffixes,
    attack_config_vars=[],
)
input_weight_attacks_configs = ev.load_weight_input(
    config, attack_config_vars=[
        ␣
↪['joint_weight_input_l2_gd_nonorm2_lwrl2normalized_lr001_e000075_linf_gd_normalized_lr0007_
    ])
for i in range(len(input_weight_attacks_configs)):
    weight_evaluations, _ = ev.get_weight_input_attack_evaluations(
        config, training_configs, [input_weight_attacks_configs[i]],␣
↪train=False)
    input_attack_configs = ev.load_input(
        config, attack_config_vars=[
            [
                'input_linf_gd_normalized_lr0007_mom0_i20_e00314_at10',
            ],
            [
                'input_linf_aa_standard_e00314',
            ],
        ])
    input_evaluations, _ = ev.get_input_attack_evaluations(
        config, training_configs, [input_attack_configs[0]], validation=0,␣
↪train=False)

    iccv_plot_weight_input_correlation2(
        config, training_configs, weight_evaluations, input_evaluations,
        weight_metric='robust_loss', weight_factor=1, weight_statistic='max',␣
↪input_metric='robust_loss',
        reference_input_metric='robust_loss', input_factor=1,
        legend_loc='upper right', labels=training_labels,
        legend_anchor=(-0.15, 1.075), h=5.3, w=5.3, xmin=0, xmax=0.5,
        xlabel=r"Worst-Case \textbf{Flatness} in RLoss", ylabel=r"RLoss (PGD)",
```

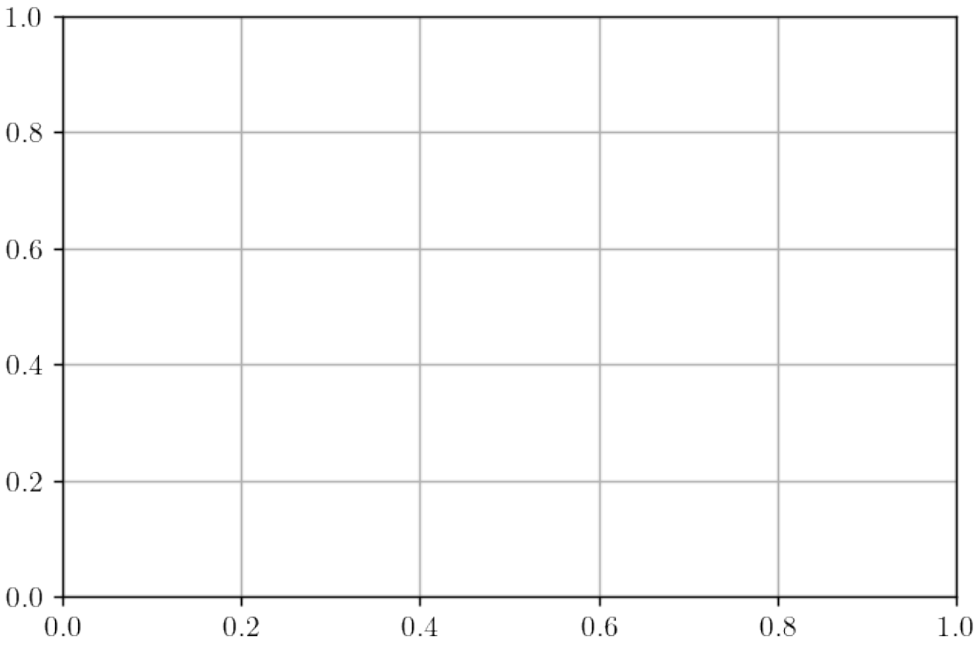
```
        legend=True, title=r"\textbf{RLoss vs. \emph{Worst}-Case Flatness in␣
↪RLoss")
```



1.7.5 Robust Generalization Gap vs. Average-Case Flatness

[37]:
```
def scatter(x, y, c=None, labels=None, use_labels=None, ax=plt.gca(), **kwargs):
    """
    Scatter plot or 2D data.

    :param x: x data
    :type x: numpy.ndarray
    :param y: y data
    :type y: numpy.ndarray
    :param c: labels as N x 1
    :type c: numpy.ndarray
    :param labels: label names
    :type labels: [str]
    """
    assert len(x.shape) == len(y.shape), 'only one dimensional data arrays␣
↪supported'
```

```
    assert x.shape[0] == y.shape[0], 'only two-dimensional data can be␣
↪scatter-plotted'
    assert c is None or x.shape[0] == c.shape[0], 'data and labels need to have␣
↪same number of rows'
    if c is not None:
        assert labels is not None, 'if classes are given, labels need also to␣
↪be given'

    if c is not None:
        if len(c.shape) > 1:
            c = numpy.squeeze(c)
    elif c is None:
        c = numpy.zeros((x.shape[0]))
        labels = [0]
    c = c.astype(int)  # Important for indexing

    if use_labels is None:
        use_labels = [False]*c.shape[0]

    unique_labels = numpy.unique(c)
    assert unique_labels.shape[0] <= len(common.plot.color_brewer), 'currently␣
↪a maxmimum of 12 different labels are supported'
    # assert unique_labels.shape[0] == len(labels), 'labels do not match given␣
↪classes'
    assert numpy.min(unique_labels) >= 0 and numpy.max(unique_labels) <␣
↪len(labels), 'classes contain elements not in labels'

    for i in range(unique_labels.shape[0]):
        marker = kwargs.get('marker', common.plot.marker_brewer[i])
        label = label=labels[unique_labels[i]] if use_labels[i] else None
        ax.scatter(x[c == unique_labels[i]], y[c == unique_labels[i]],
                   c=numpy.repeat(numpy.expand_dims(common.plot.
↪color_brewer[i], 0), x[c == unique_labels[i]].shape[0], axis=0),
                   marker=marker, s=kwargs.get('s', 45),
                   edgecolor='black', linewidth=kwargs.get('linewidth', 0.
↪5), label=label)
    has_colors = (c is not None)
    common.plot.label(ax, legend=has_colors, **kwargs)
```



[38]:
```
def iccv_plot_weight_input_diff_correlation2(
    config, training_configs, weight_evaluations, input_evaluations_a,␣
↪input_evaluations_b,
    weight_metric='robust_test_error',␣
↪reference_input_metric='robust_test_error',
    weight_statistic='max', weight_factor=100, labels=None,␣
↪input_metric='robust_test_error',
    input_factor=100, min_zero=False, legend_ncol=1, legend=True,␣
↪use_labels=None, regression=False, **kwargs):
    """
    Plot robust loss difference against flatness.
    """

    assert len(weight_evaluations) > 0
    assert len(weight_evaluations) == len(input_evaluations_a)
    assert len(weight_evaluations) == len(input_evaluations_b)
    assert len(weight_evaluations[0]) == 1
    assert len(input_evaluations_a[0]) == 1
    assert len(input_evaluations_b[0]) == 1

    x = []
    y = []
    c = []
    clean_labels = []
```

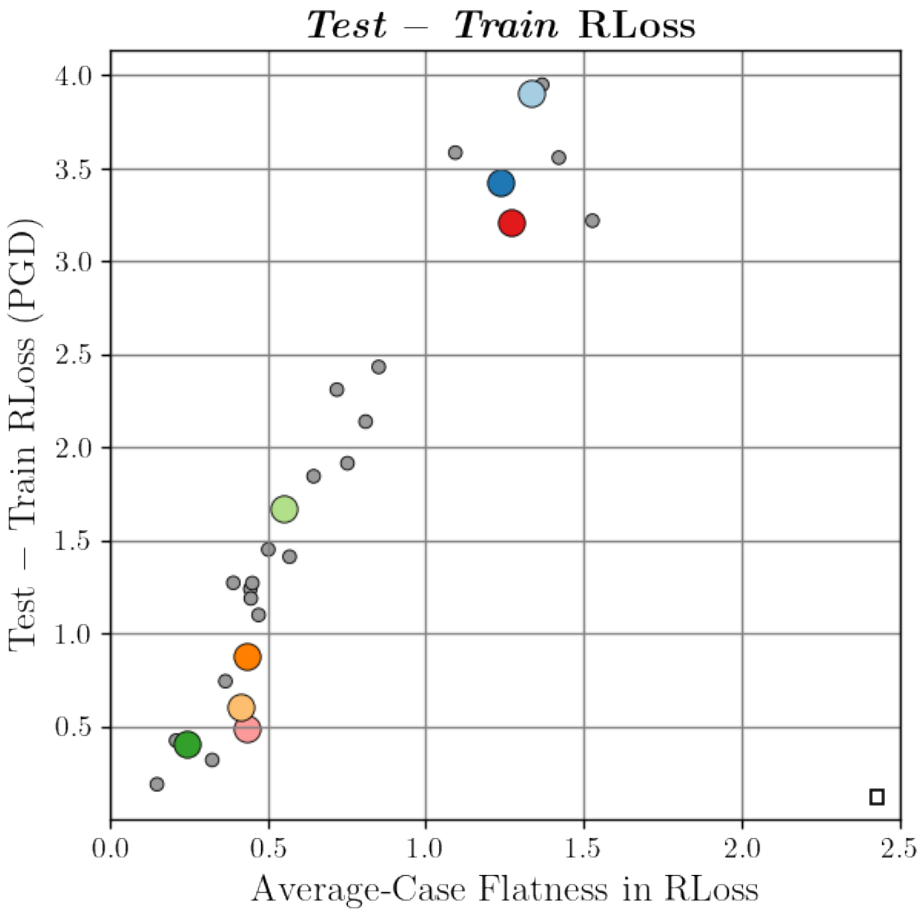
```
    if labels is not None:
        set_labels = False
    else:
        labels = []
        set_labels = True

    c_ = 0
    mapping = dict()
    for t in range(len(training_configs)):
        if weight_evaluations[t][0] is not None and input_evaluations_a[t][0]␣
↪is not None and input_evaluations_b[t][0] is not None:
            flatness = round(weight_evaluations[t][0](weight_metric,␣
↪weight_statistic)[0], 4)*weight_factor -\
                    round(getattr(input_evaluations_a[t][0],␣
↪reference_input_metric)(), 4)*input_factor
            if min_zero:
                flatness = max(0, flatness)
            x.append(flatness)
            y.append(round(getattr(input_evaluations_a[t][0], input_metric)(),␣
↪4)*input_factor -
                    round(getattr(input_evaluations_b[t][0], input_metric)(),␣
↪4)*input_factor)

            if labels is None:
                clean_labels.append(training_configs[t].directory)
                c.append(c_)
                c_ += 1
            else:
                if labels[t] not in mapping.keys():
                    c.append(c_)
                    mapping[labels[t]] = c_
                    c_ += 1
                    clean_labels.append(labels[t])
                else:
                    c.append(mapping[labels[t]])

    x = numpy.array(x)
    y = numpy.array(y)
    c = numpy.array(c)

    plt.clf()
    if regression:
        res = stats.linregress(x, y)
        xs1 = numpy.array([numpy.min(x), numpy.max(x)])
        plt.gca().plot(
            xs1, res.intercept + res.slope*xs1, color='red', marker=None,␣
↪linewidth=3, linestyle=':')
```

```
    scatter(
        x, y, c=c, labels=clean_labels, use_labels=use_labels,
        s=matplotlib.rcParams['lines.markersize'], **kwargs, ax=plt.gca())
    legend_ = plt.gca().legend(
        ncol=legend_ncol, loc=kwargs.get('legend_loc', None),
        bbox_to_anchor=kwargs.get('legend_anchor', None))
    legend_.get_frame().set_alpha(None)
    legend_.get_frame().set_facecolor((1, 1, 1, 0.5))
    if not legend:
        plt.gca().get_legend().remove()
    plt.show()
```

[39]:
```
matplotlib.rcParams['lines.markersize'] = 100
color_brewer_ = numpy.array([
    [155, 155, 155],
    #
    [166, 206, 227],
    [31, 120, 180],
    [251, 154, 153],
    [178, 223, 138],
    [51, 160, 44],
    [227, 26, 28],
    [253, 191, 111],
    [255, 127, 0],

], dtype=float)
common.plot.color_brewer = color_brewer_/255.
common.plot.marker_brewer = ['.']
common.plot.marker_brewer += ['o']*8

training_config_vars1, training_suffixes1, _ = get_models('corr-other')
training_config_vars2, training_suffixes2, training_labels2 = get_models('corr')
assert common.plot.color_brewer.shape[0] == len(training_config_vars2) + 1,␣
 ↪common.plot.color_brewer.shape[0]
assert len(common.plot.marker_brewer) == len(training_config_vars2) + 1,␣
 ↪len(common.plot.marker_brewer)

training_config_vars = training_config_vars1 + training_config_vars2
training_suffixes = training_suffixes1 + training_suffixes2
training_labels = ['Other']*len(training_config_vars1) + training_labels2

training_configs, weight_attack_configs = ev.load(config,
    training_config_vars=training_config_vars,
    training_suffixes=training_suffixes,
    attack_config_vars=[],
)
```

```
input_weight_attacks_configs = ev.load_weight_input(
    config, attack_config_vars=[
        ␣
↪['sequential2_weight_input_l2_random_nonorm2_e05_linf_gd_normalized_lr0007_e00314_i20_at10_
    ])
for i in range(len(input_weight_attacks_configs)):
    weight_evaluations, _ = ev.get_weight_input_attack_evaluations(
        config, training_configs, [input_weight_attacks_configs[i]],␣
↪train=False)
    input_attacks_configs = ev.load_input(
        config, attack_config_vars=[
            [
                'input_linf_gd_normalized_lr0007_mom0_i20_e00314_at10',
            ],
            [
                'input_linf_gd_normalized_lr0007_mom0_i20_e00314_at10_train',
            ],
        ])

    test_input_evaluations, _ = ev.get_input_attack_evaluations(
        config, training_configs, [input_attacks_configs[0]], validation=0,␣
↪train=False)
    train_input_evaluations, _ = ev.get_input_attack_evaluations(
        config, training_configs, [input_attacks_configs[1]], validation=0,␣
↪train=True)

    iccv_plot_weight_input_diff_correlation2(
        config, training_configs, weight_evaluations, test_input_evaluations,␣
↪train_input_evaluations,
        weight_metric='robust_loss', input_metric='robust_loss',␣
↪reference_input_metric='robust_loss',
        weight_factor=1, weight_statistic='mean', input_factor=1,
        legend_loc='lower right', labels=training_labels, legend_anchor=(1, 0),␣
↪h=5.3, w=5.3,
        xmin=0, xmax=2.5, legend=True,
        xlabel=r"Average-Case Flatness in RLoss", ylabel='Test $-$ Train RLoss␣
↪(PGD)',
        title=r"\textbf{\emph{Test $-$ Train} RLoss}")
```

```
No handles with labels found to put in legend.
No handles with labels found to put in legend.
```


Test − *Train* RLoss
Test − Train RLoss (PGD)
Average-Case Flatness in RLoss
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0.5
1.0
1.5
2.0
2.5