STAT GR5242 Advanced Machine Learning

Improved Residual Network for Image Classification

Author: Jie Li, Xiaofan Zhang, Zhaoyang Wang

UNI: jl5246, xz2735, zw2551

In [1]:

```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
import skimage.transform
import time, os, datetime
from Model import Residual_Unit
from Model import Attention_Block
```

In [2]:

```
from tensorflow.keras.datasets import cifar10
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.optimizers import Adam, SGD
from tensorflow.keras.callbacks import ReduceLROnPlateau, EarlyStopping,TensorBoard,
from tensorflow.keras.models import load_model
from tensorflow.keras.layers import Input
from tensorflow.keras.regularizers import l2
from tensorflow.keras.layers import BatchNormalization
from tensorflow.keras.layers import Conv2D
from tensorflow.keras.layers import MaxPooling2D
from tensorflow.keras.layers import Activation
from tensorflow.keras.layers import AveragePooling2D
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dropout
from tensorflow.keras.layers import Dense
from tensorflow.keras.models import Model
```

In [3]:

```
print("TF version: ",tf.__version__)
print("Keras version:",tf.keras.__version__)
```

```
TF version:  2.0.0
Keras version: 2.2.4-tf
```

Load CIFAR-10 Data

In [4]:

```
# Load the CIFAR10 data.
(x_train, y_train), (x_test, y_test) = cifar10.load_data()
x_train = x_train[:30000, :, :, :]
y_train = y_train[:30000]
x_val = x_train[-5000:, :, :, :]
y_val = y_train[-5000:]

print('x_train shape:', x_train.shape)
print('y_train shape:', y_train.shape)
print('x_validation shape:', x_val.shape)
print('y_validation shape:', y_val.shape)
print('x_test shape:', x_test.shape)
print('y_test shape:', y_test.shape)

# Convert class vectors to binary class matrices.
y_train = to_categorical(y_train, 10)
y_val = to_categorical(y_val, 10)
y_test = to_categorical(y_test, 10)
```

```
x_train shape: (30000, 32, 32, 3)
y_train shape: (30000, 1)
x_validation shape: (5000, 32, 32, 3)
y_validation shape: (5000, 1)
x_test shape: (10000, 32, 32, 3)
y_test shape: (10000, 1)
```

Data Augmentation

In [12]:

```
# define generators for training and validation data
train_datagen = ImageDataGenerator(
    featurewise_center=True,
    featurewise_std_normalization=True,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    validation_split=0.2)

val_datagen = ImageDataGenerator(
    featurewise_center=True,
    featurewise_std_normalization=True)

test_datagen = ImageDataGenerator(
    featurewise_center=True,
    featurewise_std_normalization=True)

# compute quantities required for feature normalization
train_datagen.fit(x_train)
val_datagen.fit(x_val)
test_datagen.fit(x_test)
```

Construct Resdiual Attention Network

Stacking two Attention modules follow by Residual Network

In [27]:

```
# define learning rate scheduler
def lr_schedule(epoch):
    lr = 1e-4
    if epoch > 50:
        lr *= 1e-2
    elif epoch > 20:
        lr *= 1e-1
    print('Learning rate:', lr)
    return lr
lr_scheduler = LearningRateScheduler(lr_schedule)

# Resdiual Attention Network
def AttentionResNet56_mini(shape, in_channel, kernel_size, n_classes, dropout=None, r

    """
    :param shape: The tuple of input data.
    :param in_channel: The 4-th dimension (channel number) of input weight matrix. For
    :param kernel_size: Integer. the shape of the kernel. For example, default kernel_
    :param n_classes: Integer. The number of target classes. For example, n_classes =
    :param dropout: Float between 0 and 1. Fraction of the input units to drop.
    :param regularization: Float. Fraction of the input units to drop.
    """

    input_data = Input(shape=shape)  # 32x32x32
    x = Conv2D(in_channel, kernel_size=kernel_size, padding='same')(input_data)  # 32x
    x = BatchNormalization()(x)
    x = Activation('relu')(x)
    x = MaxPooling2D(pool_size=2, padding='same')(x)  # 16x16x32

    out_channel = in_channel * 4
    x = Residual_Unit(x, in_channel, out_channel)  # 16x16x128
    x = Attention_Block(x, skip=1)

    in_channel = out_channel // 2
    out_channel = in_channel * 4
    x = Residual_Unit(x, in_channel, out_channel, stride=2)  # 8x8x256
    x = Attention_Block(x, skip=1)

    in_channel = out_channel // 2
    out_channel = in_channel * 4
    x = Residual_Unit(x, in_channel, out_channel, stride=1)  # 4x4x1024
    x = Residual_Unit(x, in_channel, out_channel)
    x = Residual_Unit(x, in_channel, out_channel)

    x = AveragePooling2D(pool_size=4, strides=1)(x)  # 1x1x1024
    x = Flatten()(x)

    output = Dense(n_classes, activation='softmax')(x)
    model = Model(input_data, output)

    return model

def training(model, log_name, batch_size=128, epc=60):
    batch_size = batch_size
    epc = epc
    start = time.time()

    # define training generator
    train_generator = train_datagen.flow(x_train, y_train, batch_size=batch_size)
    step_size_train = train_generator.n // train_generator.batch_size

    # define validation generator
    val_generator = val_datagen.flow(x_val, y_val, batch_size=batch_size)
    step_size_val = val_generator.n // val_generator.batch_size

    # define test validation generator
    test_generator = test_datagen.flow(x_test, y_test, batch_size=batch_size)
    step_size_test = test_generator.n // test_generator.batch_size

    # usefull callbacks
    log_dir='Logs/' + log_name
    tensorboard_callback = TensorBoard(log_dir=log_dir, histogram_freq=1)
    lr_reducer = ReduceLROnPlateau(monitor='val_accuracy', factor=0.1, patience=5, ver
    early_stopper = EarlyStopping(monitor='val_accuracy', patience=15, verbose=1)

    model.fit_generator(train_generator,
                        steps_per_epoch = step_size_train,
                        epochs = epc,
                        validation_data = val_generator,
                        validation_steps = step_size_val,
                        callbacks=[tensorboard_callback, lr_reducer, lr_scheduler, ear

    end = time.time()
    print("Time taken by above cell is {}.".format((end-start)/60))

    # evaluation
    val_scores = model.evaluate_generator(val_generator, verbose=0)
    test_scores = model.evaluate_generator(test_generator, verbose=1)
    print('validation loss:', val_scores[0])
    print('validation accuracy:', val_scores[1])
    print('Test loss:', test_scores[0])
    print('Test accuracy:', test_scores[1])

    return model
```

In [24]:

```
# define model
model = AttentionResNet56_mini(shape=(32,32,3), in_channel=32, kernel_size=5, n_classe

# define loss, metrics, optimizer
optimizer = SGD(lr = lr_schedule(0), momentum=0.9, nesterov=True)
# optimizer = Adam(lr = lr_schedule(0) )
model.compile(optimizer, loss='categorical_crossentropy', metrics=['accuracy'])
model.summary()
```

```
                                                                       conv2d_189[0][0]
__________________________________________________________________________________
______________
max_pooling2d_9 (MaxPooling2D)  (None, 4, 4, 256)    0           add_43[0][0]
__________________________________________________________________________________
______________
batch_normalization_142 (BatchN (None, 4, 4, 256)    1024        max_pooling2d_9
[0][0]
__________________________________________________________________________________
______________
activation_147 (Activation)     (None, 4, 4, 256)    0           batch_normalizati
on_142[0][0]
__________________________________________________________________________________
______________
conv2d_198 (Conv2D)             (None, 4, 4, 256)    65792       activation_147[0]
[0]
__________________________________________________________________________________
______________
batch_normalization_143 (BatchN (None, 4, 4, 256)    1024        conv2d_198[0][0]
```

Training and Evaluation

Train our base model with **nesterov SGD** optimizer

In [28]:

```
# training
model = training(model, '56mini-SGD-Base')
```

```
Learning rate: 1e-05
Epoch 35/60
234/234 [==============================] - 75s 322ms/step - loss: 1.2155 - accurac
y: 0.5676 - val_loss: 1.0880 - val_accuracy: 0.6100
Learning rate: 1e-05
Epoch 36/60
234/234 [==============================] - 76s 325ms/step - loss: 1.2221 - accurac
y: 0.5666 - val_loss: 1.0958 - val_accuracy: 0.6118
Learning rate: 1e-05
Epoch 37/60
234/234 [==============================] - 76s 326ms/step - loss: 1.2130 - accurac
y: 0.5657 - val_loss: 1.0982 - val_accuracy: 0.6158
Learning rate: 1e-05
Epoch 38/60
234/234 [==============================] - 75s 321ms/step - loss: 1.2162 - accurac
y: 0.5680 - val_loss: 1.1053 - val_accuracy: 0.6142
Learning rate: 1e-05
Epoch 39/60
233/234 [============================>.] - ETA: 0s - loss: 1.2119 - accuracy: 0.56
91
```

Instead of using SGD, change to **Adam** optimizer

In [29]:

```
# define model
model = AttentionResNet56_mini(shape=(32,32,3), in_channel=32, kernel_size=5, n_classe

# define loss, metrics, optimizer
optimizer = Adam(lr = lr_schedule(0) )
model.compile(optimizer, loss='categorical_crossentropy', metrics=['accuracy'])
```

```
Learning rate: 0.0001
```

In [30]:

```
# training
model = training(model, '56mini-Adam-Base')
```

```
Epoch 58/60
234/234 [==============================] - 75s 321ms/step - loss: 0.4818 - accurac
y: 0.8298 - val_loss: 0.3698 - val_accuracy: 0.8686
Learning rate: 1.0000000000000002e-06
Epoch 59/60
234/234 [==============================] - 74s 318ms/step - loss: 0.4814 - accurac
y: 0.8320 - val_loss: 0.3681 - val_accuracy: 0.8694
Learning rate: 1.0000000000000002e-06
Epoch 60/60
234/234 [==============================] - 75s 319ms/step - loss: 0.4861 - accurac
y: 0.8299 - val_loss: 0.3607 - val_accuracy: 0.8694
Time taken by above cell is 76.77415573596954.
79/79 [==============================] - 9s 112ms/step - loss: 0.6643 - accuracy:
0.7854
validation loss: 0.35428522787988187
validation accuracy: 0.8696

Test loss: 0.6642644971231871
Test accuracy: 0.7854
```

Improvements

Since the overfiting issue occurs, we would like to add regularization methods to generalize our model as following:

- Batch normalization
- L2 norm regularization
- Dropout
- Early stop

In [30]:

```
def AttentionResNet56_mini(shape, in_channel, kernel_size, n_classes, dropout=None, r

    """
    :param shape: The tuple of input data.
    :param in_channel: The 4-th dimension (channel number) of input weight matrix. For
    :param kernel_size: Integer. the shape of the kernel. For example, default kernel_
    :param n_classes: Integer. The number of target classes. For example, n_classes =
    :param dropout: Float between 0 and 1. Fraction of the input units to drop.
    :param regularization: Float. Fraction of the input units to drop.
    """

    input_data = Input(shape=shape)  # 32x32x32
    x = Conv2D(in_channel, kernel_size=kernel_size, padding='same')(input_data)  # 32x
    x = BatchNormalization()(x)
    x = Activation('relu')(x)
    x = MaxPooling2D(pool_size=2, padding='same')(x)  # 16x16x32

    out_channel = in_channel * 4
    x = Residual_Unit(x, in_channel, out_channel)  # 16x16x128
    x = Attention_Block(x, skip=1)

    in_channel = out_channel // 2
    out_channel = in_channel * 4
    x = Residual_Unit(x, in_channel, out_channel, stride=2)  # 8x8x256
    x = Attention_Block(x, skip=1)

    in_channel = out_channel // 2
    out_channel = in_channel * 4
    x = Residual_Unit(x, in_channel, out_channel, stride=1)  # 4x4x1024
    x = Residual_Unit(x, in_channel, out_channel)
    x = Residual_Unit(x, in_channel, out_channel)

    # add BN and Activation
    x = BatchNormalization()(x)  # new
    x = Activation('relu')(x)  # new
    x = AveragePooling2D(pool_size=4, strides=1)(x)  # 1x1x1024
    x = Flatten()(x)

    if dropout:
        x = Dropout(dropout)(x)  # new
    output = Dense(n_classes, kernel_regularizer=l2(regularization), activation='softm
    model = Model(input_data, output)

    return model
```

In [33]:

```
# define model
model = AttentionResNet56_mini(shape=(32,32,3), in_channel=32, kernel_size=5, n_classe

# define loss, metrics, optimizer
optimizer = Adam(lr = lr_schedule(0) )
model.compile(optimizer, loss='categorical_crossentropy', metrics=['accuracy'])
```

```
Learning rate: 0.0001
```

In [34]:

```
# training
model = training(model, '56mini-regularization')
```

```
y: 0.8558 - val_loss: 0.3471 - val_accuracy: 0.8931
Learning rate: 1.0000000000000002e-06
Epoch 58/60
234/234 [==============================] - 77s 330ms/step - loss: 0.4481 - accurac
y: 0.8562 - val_loss: 0.3535 - val_accuracy: 0.8912
Learning rate: 1.0000000000000002e-06
Epoch 59/60
234/234 [==============================] - 76s 325ms/step - loss: 0.4526 - accurac
y: 0.8545 - val_loss: 0.3472 - val_accuracy: 0.8942
Learning rate: 1.0000000000000002e-06
Epoch 60/60
234/234 [==============================] - 77s 329ms/step - loss: 0.4478 - accurac
y: 0.8544 - val_loss: 0.3475 - val_accuracy: 0.8932
Time taken by above cell is 77.04772863785426.
79/79 [==============================] - 9s 114ms/step - loss: 0.6473 - accuracy:
0.8029
validation loss: 0.34524615183472634
validation accuracy: 0.893
Test loss: 0.6473370205752457
Test accuracy: 0.8029
```

Double sampling in Attention Module, reduce the batch size and increase the epoch size

In [33]:

```
def AttentionResNet56_mini(shape, in_channel, kernel_size, n_classes, dropout=None, r

    """
    :param shape: The tuple of input data.
    :param in_channel: The 4-th dimension (channel number) of input weight matrix. For
    :param kernel_size: Integer. the shape of the kernel. For example, default kernel_
    :param n_classes: Integer. The number of target classes. For example, n_classes =
    :param dropout: Float between 0 and 1. Fraction of the input units to drop.
    :param regularization: Float. Fraction of the input units to drop.
    """

    input_data = Input(shape=shape)  # 32x32x32
    x = Conv2D(in_channel, kernel_size=kernel_size, padding='same')(input_data)  # 32x
    x = MaxPooling2D(pool_size=2, padding='same')(x)  # 16x16x32

    out_channel = in_channel * 4
    x = Residual_Unit(x, in_channel, out_channel)  # 16x16x128
    x = Attention_Block(x, skip=2)

    in_channel = out_channel // 2
    out_channel = in_channel * 4
    x = Residual_Unit(x, in_channel, out_channel, stride=2)  # 8x8x256
    x = Attention_Block(x, skip=1)

    in_channel = out_channel // 2
    out_channel = in_channel * 4
    x = Residual_Unit(x, in_channel, out_channel, stride=1)  # 4x4x1024
    x = Residual_Unit(x, in_channel, out_channel)
    x = Residual_Unit(x, in_channel, out_channel)

    # add BN and Activation
    x = BatchNormalization()(x)  # new
    x = Activation('relu')(x)  # new
    x = AveragePooling2D(pool_size=4, strides=1)(x)  # 1x1x1024
    x = Flatten()(x)

    if dropout:
        x = Dropout(dropout)(x)  # new
    output = Dense(n_classes, kernel_regularizer=l2(regularization), activation='softm
    model = Model(input_data, output)

    return model
```

In [34]:

```
# define model
model = AttentionResNet56_mini(shape=(32,32,3), in_channel=32, kernel_size=5, n_classe

# define loss, metrics, optimizer
optimizer = Adam(lr = lr_schedule(0) )
model.compile(optimizer, loss='categorical_crossentropy', metrics=['accuracy'])
model.summary()
```

```
________________________________________________________________________________
__________________
conv2d_571 (Conv2D)             (None, 8, 8, 256)    16640       activation_426[0]
[0]
________________________________________________________________________________
__________________
conv2d_572 (Conv2D)             (None, 8, 8, 256)    33024       add_136[0][0]
________________________________________________________________________________
__________________
add_137 (Add)                   (None, 8, 8, 256)    0           conv2d_571[0][0]
                                                                 conv2d_572[0][0]
________________________________________________________________________________
__________________
batch_normalization_412 (BatchN (None, 8, 8, 256)    1024        add_137[0][0]
________________________________________________________________________________
__________________
activation_427 (Activation)     (None, 8, 8, 256)    0           batch_normalizati
on_412[0][0]
________________________________________________________________________________
```

In [36]:

```
# training
model = training(model, '56mini-final', batch_size=64, epc=80)
```

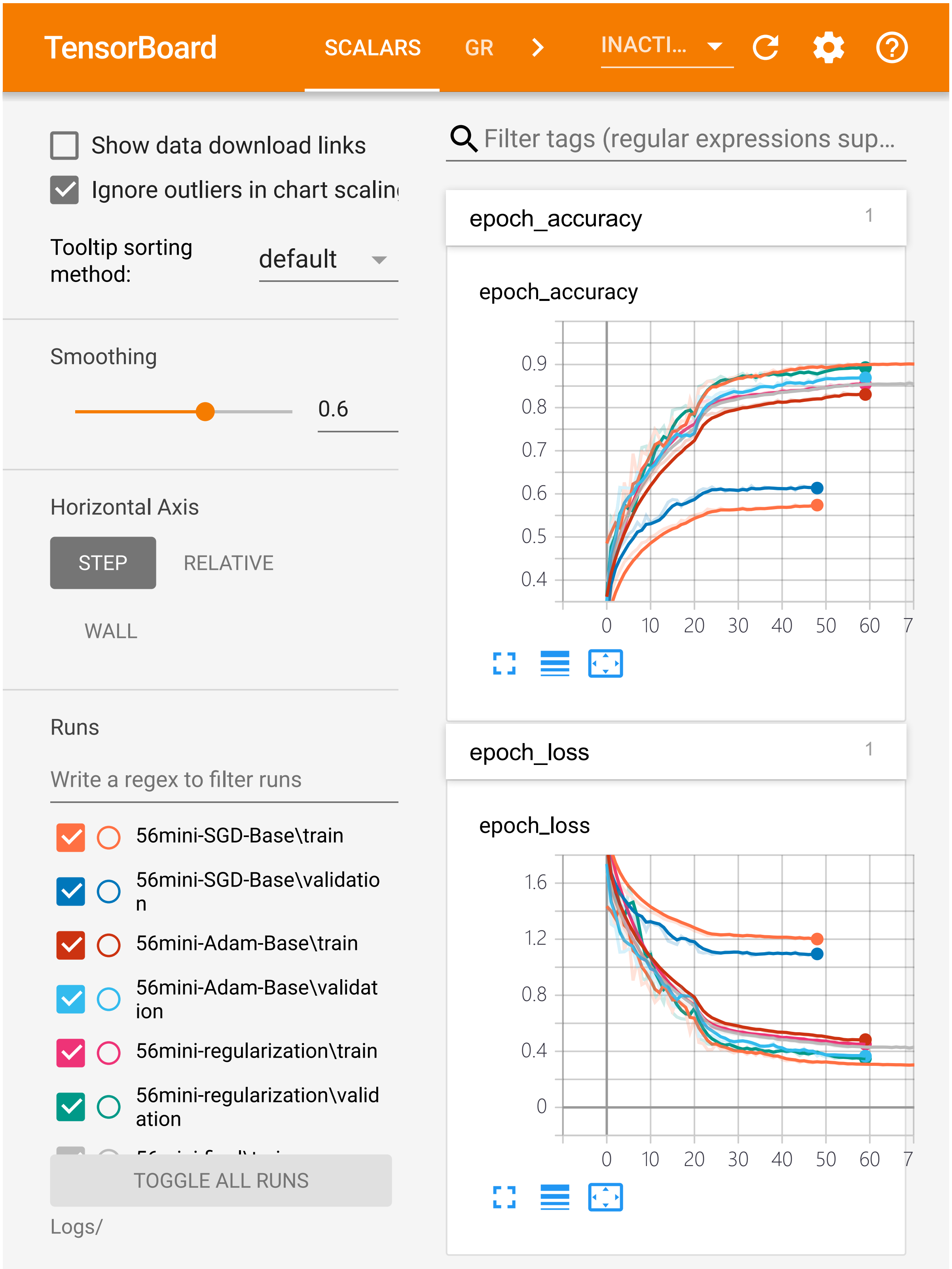
```
Epoch 57/80
468/468 [==============================] - 220s 470ms/step - loss: 0.4368 - accura
cy: 0.8500 - val_loss: 0.3108 - val_accuracy: 0.8992
Learning rate: 1.0000000000000002e-06
Epoch 58/80
468/468 [==============================] - 220s 471ms/step - loss: 0.4320 - accura
cy: 0.8539 - val_loss: 0.3067 - val_accuracy: 0.9008
Learning rate: 1.0000000000000002e-06
Epoch 59/80
468/468 [==============================] - 218s 467ms/step - loss: 0.4306 - accura
cy: 0.8537 - val_loss: 0.3079 - val_accuracy: 0.8986
Learning rate: 1.0000000000000002e-06
Epoch 60/80
468/468 [==============================] - 220s 470ms/step - loss: 0.4278 - accura
cy: 0.8553 - val_loss: 0.3070 - val_accuracy: 0.8994
Learning rate: 1.0000000000000002e-06
Epoch 61/80
467/468 [============================>.] - ETA: 0s - loss: 0.4282 - accuracy: 0.85
51
Epoch 00061: ReduceLROnPlateau reducing learning rate to 9.999999747524280e-08.
```

Performance Analysis

In [1]:
```
%load_ext tensorboard
```

In [3]:
```
%tensorboard --logdir Logs/
```

```
Reusing TensorBoard on port 6006 (pid 4216), started 0:22:26 ago. (Use '!kill 4216' t
o kill it.)
```



Filter tags (regular expressions sup...

□ Show data download links

☑ Ignore outliers in chart scalin

Tooltip sorting method: default

Smoothing 0.6

Horizontal Axis

STEP RELATIVE WALL

Runs

Write a regex to filter runs

☑ ○ 56mini-SGD-Base\train

☑ ○ 56mini-SGD-Base\validation

☑ ○ 56mini-Adam-Base\train

☑ ○ 56mini-Adam-Base\validation

☑ ○ 56mini-regularization\train

☑ ○ 56mini-regularization\validation

Logs/