Speaker Classifier 실행 및 분석(DNN)

6조 강재훈 김민선 송영훈

Speaker_classifier_tflearn.py

```
import tensorflow as tf
print("You are using tensorflow version "+ tf.__version__) #+" tflearn version "+ tflearn.version)
if tf.__version__ >= '0.12' and os.name == 'nt':
    print("sorry, tflearn is not ported to tensorflow 0.12 on windows yet!(?)")
    quit() # why? works on Mac?

speakers = data.get_speakers()
number_classes=len(speakers)
print("speakers",speakers)

batch=data.wave_batch_generator(batch_size=1000, source=data.Source.DIGIT_WAVES, target=data.Target.speaker)
X,Y=next(batch)
```

Speech_data.py

```python
def wave_batch_generator(batch_size=10,source=Source.DIGIT_WAVES,target=Target.digits): #speaker
    maybe_download(source, DATA_DIR)
    if target == Target.speaker: speakers=get_speakers()
    batch_waves = []
    labels = []
    # input_width=CHUNK*6 # wow, big!!
    files = os.listdir(path)
    while True:
        shuffle(files)
        print("loaded batch of %d files" % len(files))
        for wav in files:
            if not wav.endswith(".wav"):continue
            if target==Target.digits: labels.append(dense_to_one_hot(int(wav[0])))
            elif target==Target.speaker: labels.append(one_hot_from_item(speaker(wav), speakers))
            elif target==Target.first_letter:  label=dense_to_one_hot((ord(wav[0]) - 48) % 32,32)
            else: raise Exception("todo : Target.word label!")
            chunk = load_wav_file(path+wav)
            batch_waves.append(chunk)
            # batch_waves.append(chunks[input_width])
            if len(batch_waves) >= batch_size:
                yield batch_waves, labels
                batch_waves = []  # Reset for next batch
                labels = []
```

Speaker_classifier_tflearn.py

```
import tensorflow as tf
print("You are using tensorflow version "+ tf.__version__) #+" tflearn version "+ tflearn.version)
if tf.__version__ >= '0.12' and os.name == 'nt':
    print("sorry, tflearn is not ported to tensorflow 0.12 on windows yet!(?)")
    quit() # why? works on Mac?

speakers = data.get_speakers()
number_classes=len(speakers)
print("speakers",speakers)

batch=data.wave_batch_generator(batch_size=1000, source=data.Source.DIGIT_WAVES, target=data.Target.speaker)
X,Y=next(batch)
```

Speech_data.py

```python
def wave_batch_generator(batch_size=10,source=Source.DIGIT_WAVES,target=Target.digits): #speaker
	maybe_download(source, DATA_DIR)
	if target == Target.speaker: speakers=get_speakers()
	batch_waves = []
	labels = []
	# input_width=CHUNK*6 # wow, big!!
	files = os.listdir(path)
	while True:
		shuffle(files)
		print("loaded batch of %d files" % len(files))
		for wav in files:
			if not wav.endswith(".wav"):continue
			if target==Target.digits: labels.append(dense_to_one_hot(int(wav[0])))
			elif target==Target.speaker: labels.append(one_hot_from_item(speaker(wav), speakers))
			elif target==Target.first_letter:  label=dense_to_one_hot((ord(wav[0]) - 48) % 32,32)
			else: raise Exception("todo : Target.word label!")
			chunk = load_wav_file(path+wav)
			batch_waves.append(chunk)
			# batch_waves.append(chunks[input_width])
			if len(batch_waves) >= batch_size:
				yield batch_waves, labels
				batch_waves = []  # Reset for next batch
				labels = []
```

Speech_data.py

```
class Source:  # labels
    DIGIT_WAVES = 'spoken_numbers_pcm.tar'
    DIGIT_SPECTROS = 'spoken_numbers_spectros_64x64.tar'  # 64x64  baby data set, works astonishingly well
    NUMBER_WAVES = 'spoken_numbers_wav.tar'
    NUMBER_IMAGES = 'spoken_numbers.tar'  # width=256 height=256
    WORD_SPECTROS = 'https://dl.dropboxusercontent.com/u/23615316/spoken_words.tar'  # width,height=512# todo: sliding window!
    WORD_WAVES = 'spoken_words_wav.tar'
    TEST_INDEX = 'test_index.txt'
    TRAIN_INDEX = 'train_index.txt'

from enum import Enum
class Target(Enum):  # labels
    digits=1
    speaker=2
    words_per_minute=3
    word_phonemes=4
    word = 5  # int vector as opposed to binary hotword
    sentence=6
    sentiment=7
    first_letter=8
    hotword = 9
    # test_word=9 # use 5 even for speaker etc
```

Library

1. tflearn : high level library built on top of tensorflow
    a. easier to read
    b. great for fast prototyping(시제품화)(?)

2. speach_data : Web에서 data 가져와서, 우릴 위해 format해줌.

Hyperparameters

≒ tuning options

1. learning rate (time vs. accuracy)
    a. 크면 속도 증가
    b. 작으면 정확성 증가
2. training rate: train을 몇 step 하고싶은가.

Batch Generator Function

training & testing data 만들기

python’s built-in next function으로 batch를 train, test data로 나눈다.

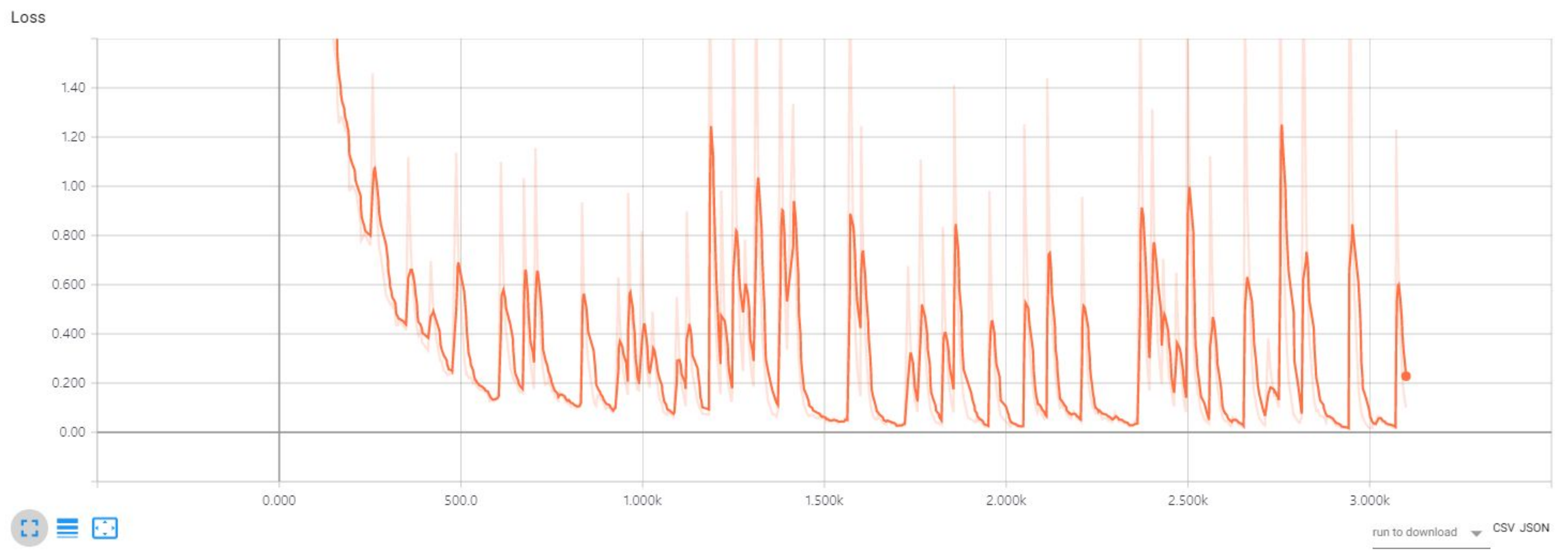
Why RNN?

spoken words = sequence of soundwaves이기 때문에 RNN을 사용하겠다.

RNN은 sequence를 처리하는 것이 가능하기 때문이다.

```
net = tflearn.input_data(shape=[None, 8192]) #Two wave chunks
net = tflearn.fully_connected(net, 64)
net = tflearn.dropout(net, 0.5)
net = tflearn.fully_connected(net, number_classes, activation='softmax')
net = tflearn.regression(net, optimizer='adam', loss='categorical_crossentropy')
```

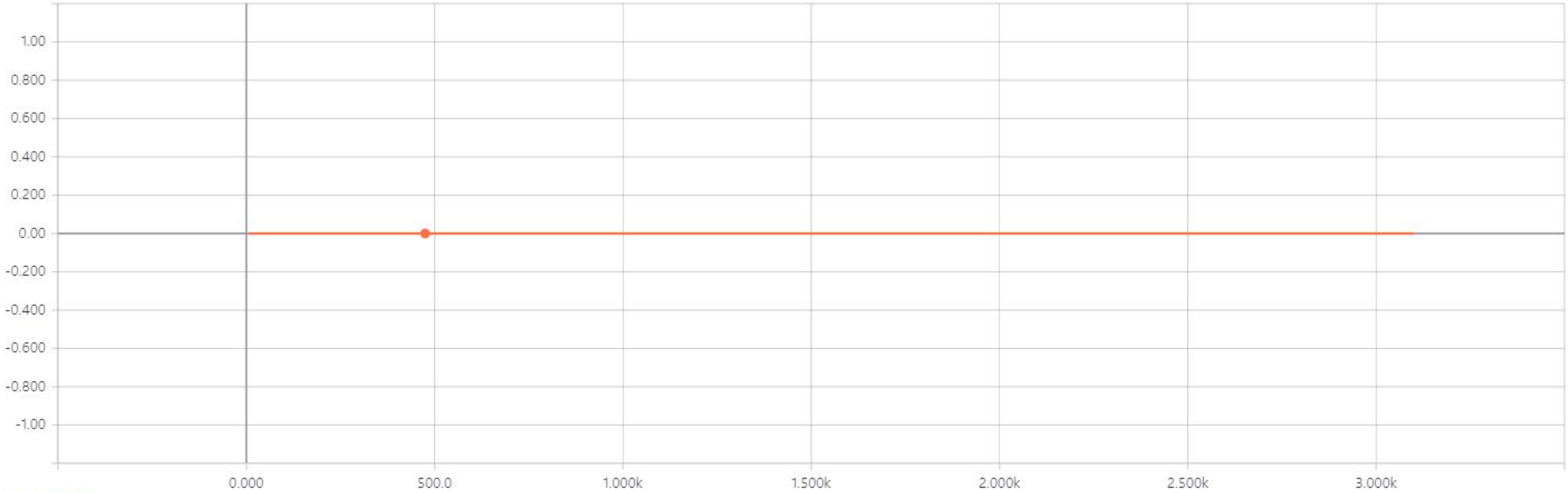
Loss



Loss
1.40
1.20
1.00
0.800
0.600
0.400
0.200
0.00
0.000
500.0
1.000k
1.500k
2.000k
2.500k
3.000k
run to download
CSV JSON

FullyConnected 1

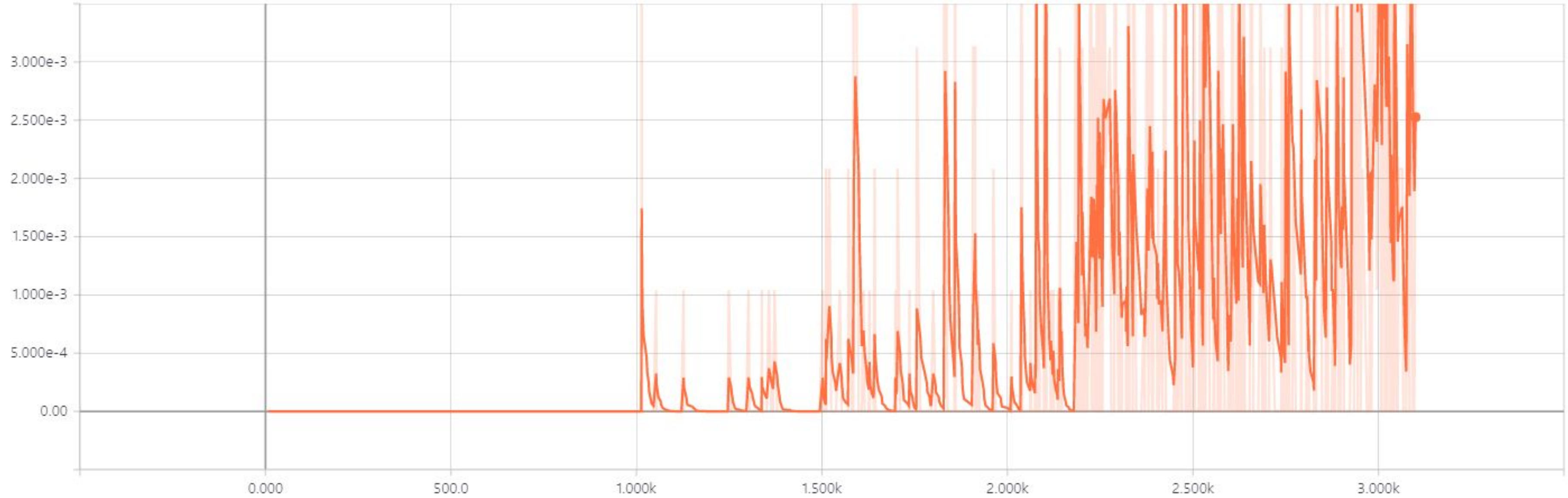
FullyConnected/BiasAdd/Sparsity



run to download CSV JSON

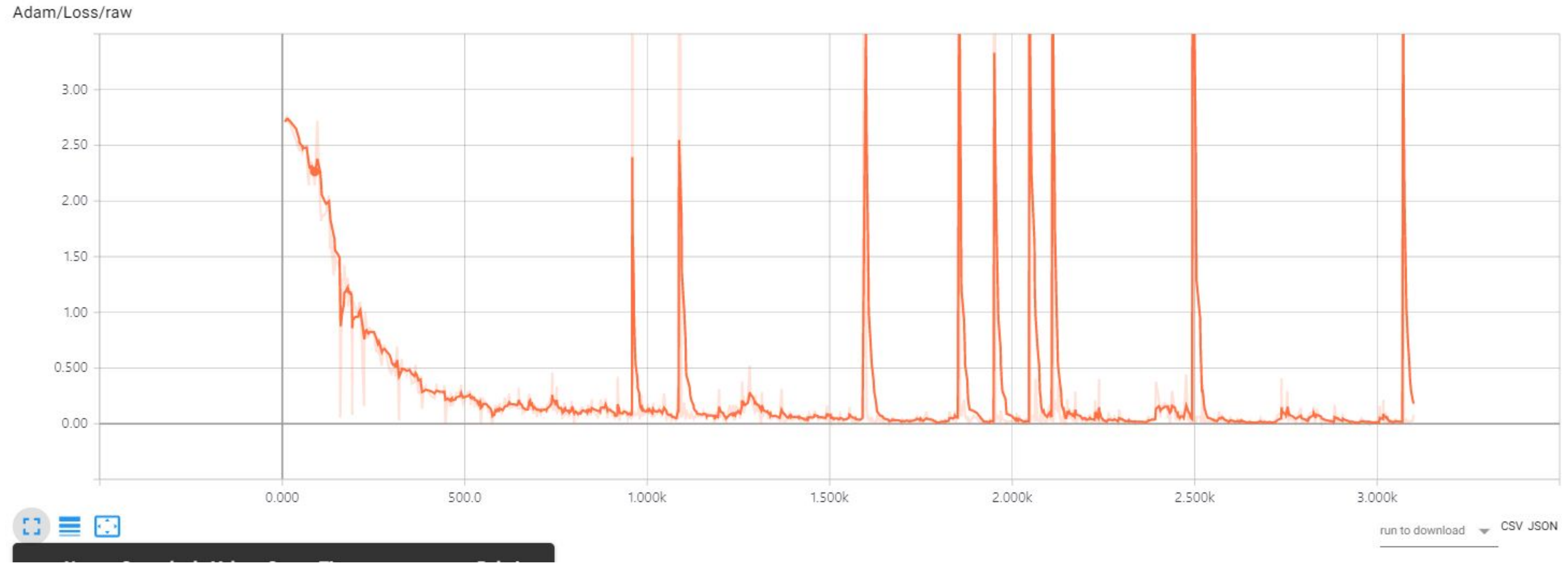
FullyConnected_1 1

FullyConnected_1/Softmax/Sparsity

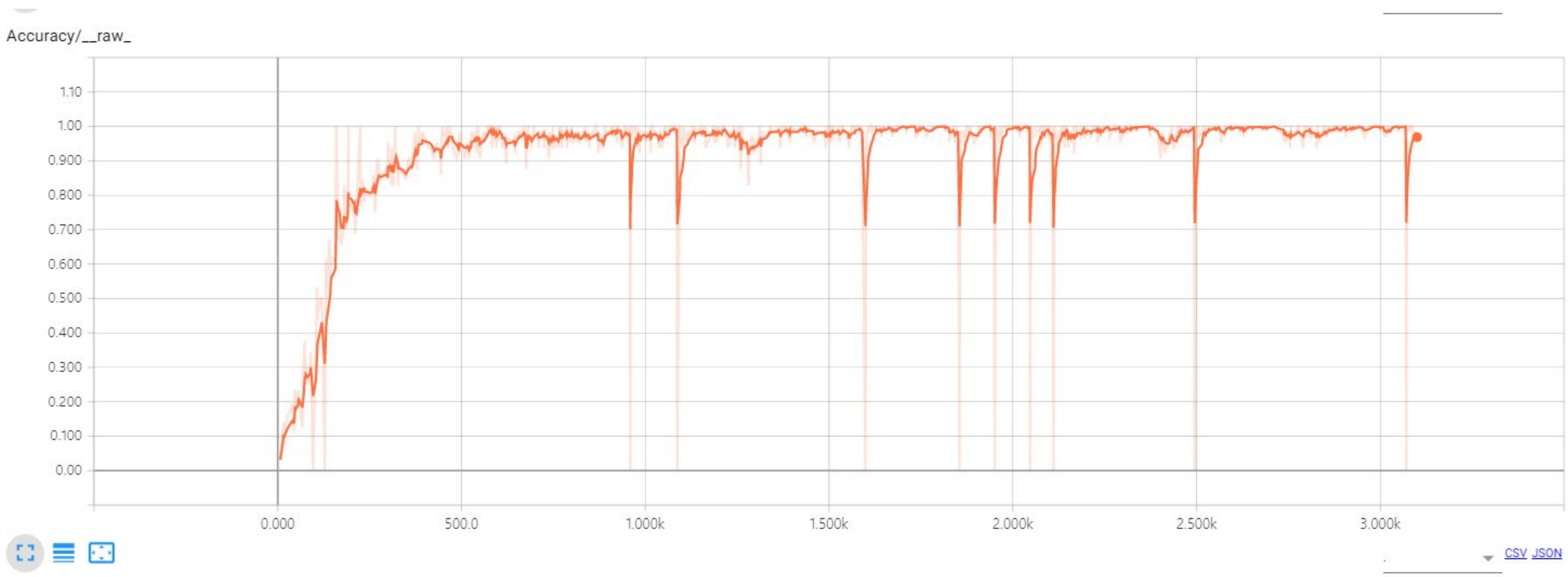


run to download CSV JSON

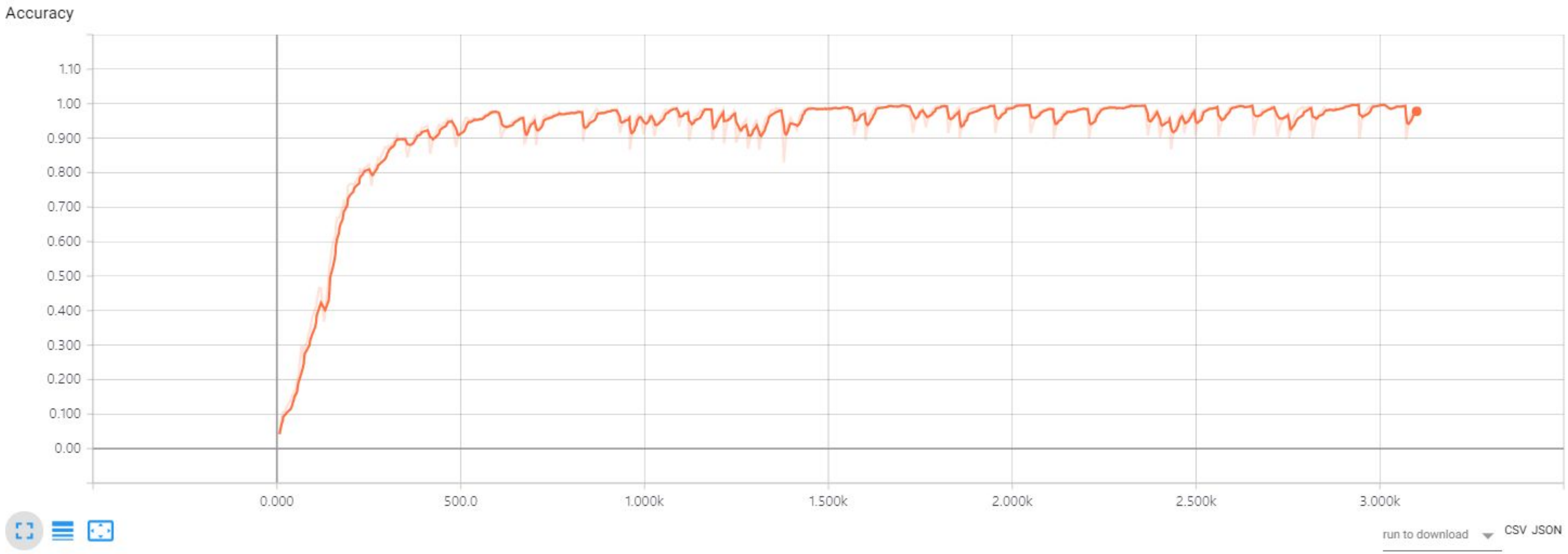
Adam 1



Adam/Loss/raw
3.00
2.50
2.00
1.50
1.00
0.500
0.00
0.000
500.0
1.000k
1.500k
2.000k
2.500k
3.000k
run to download
CSV JSON

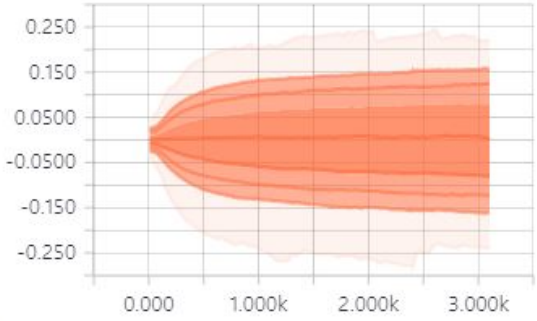


Accuracy/__raw_
1.10
1.00
0.900
0.800
0.700
0.600
0.500
0.400
0.300
0.200
0.100
0.00
0.000
500.0
1.000k
1.500k
2.000k
2.500k
3.000k
CSV JSON

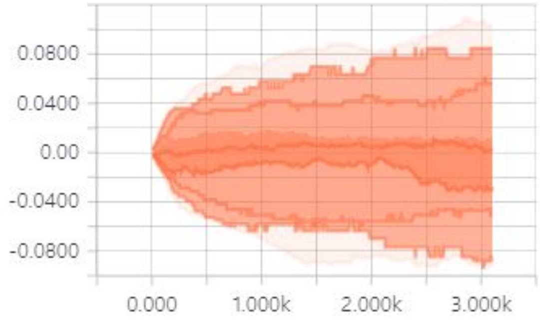

Accuracy
1.10
1.00
0.900
0.800
0.700
0.600
0.500
0.400
0.300
0.200
0.100
0.00
0.000
500.0
1.000k
1.500k
2.000k
2.500k
3.000k
run to download
CSV JSON

Adam

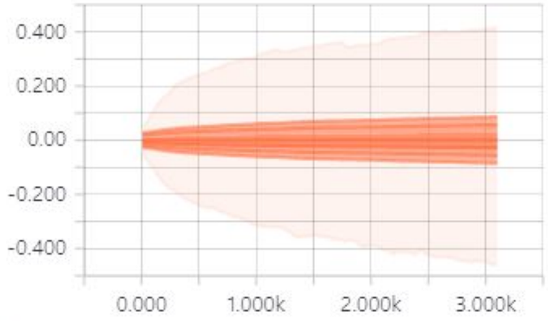
Adam/FullyConnected_1/W



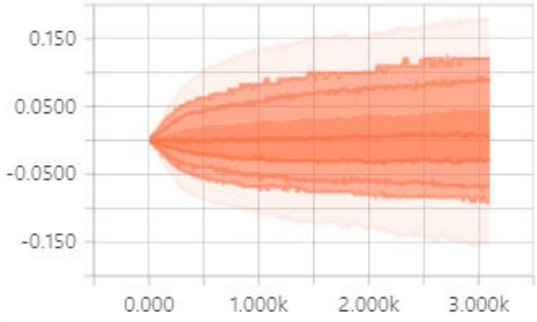
Adam/FullyConnected_1/b



Adam/FullyConnected/W

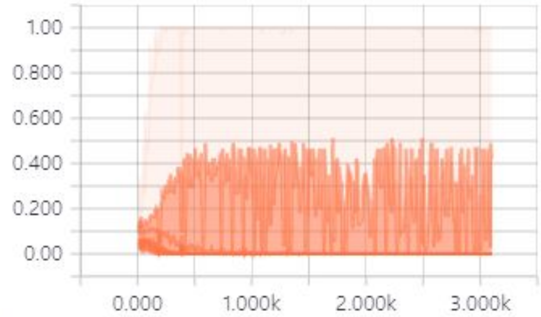


Adam/FullyConnected/b

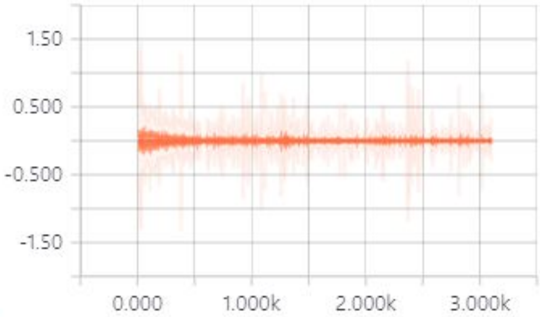


FullyConnected_1

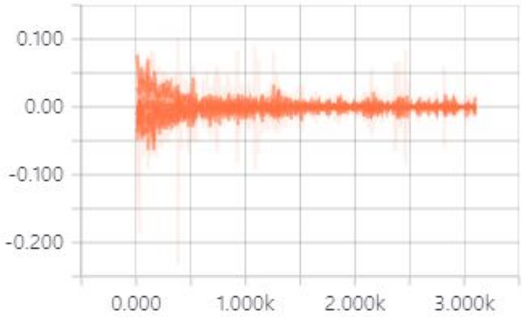
FullyConnected_1/Softmax/Activations



FullyConnected_1/W/Gradients

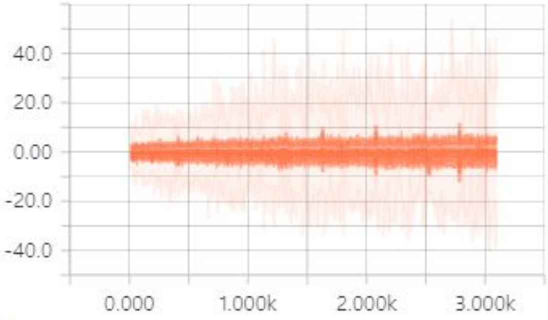


FullyConnected_1/b/Gradients

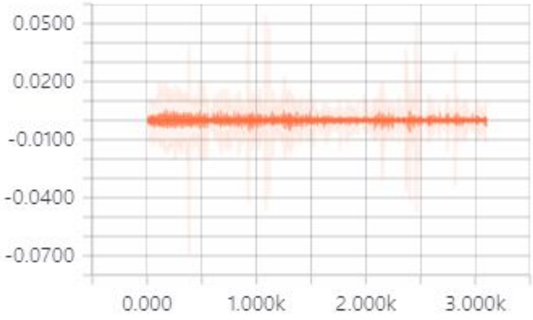


FullyConnected

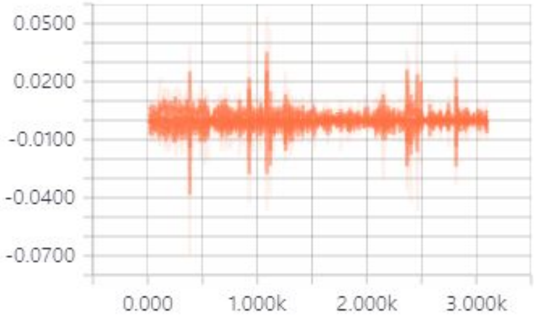
FullyConnected/BiasAdd/Activations

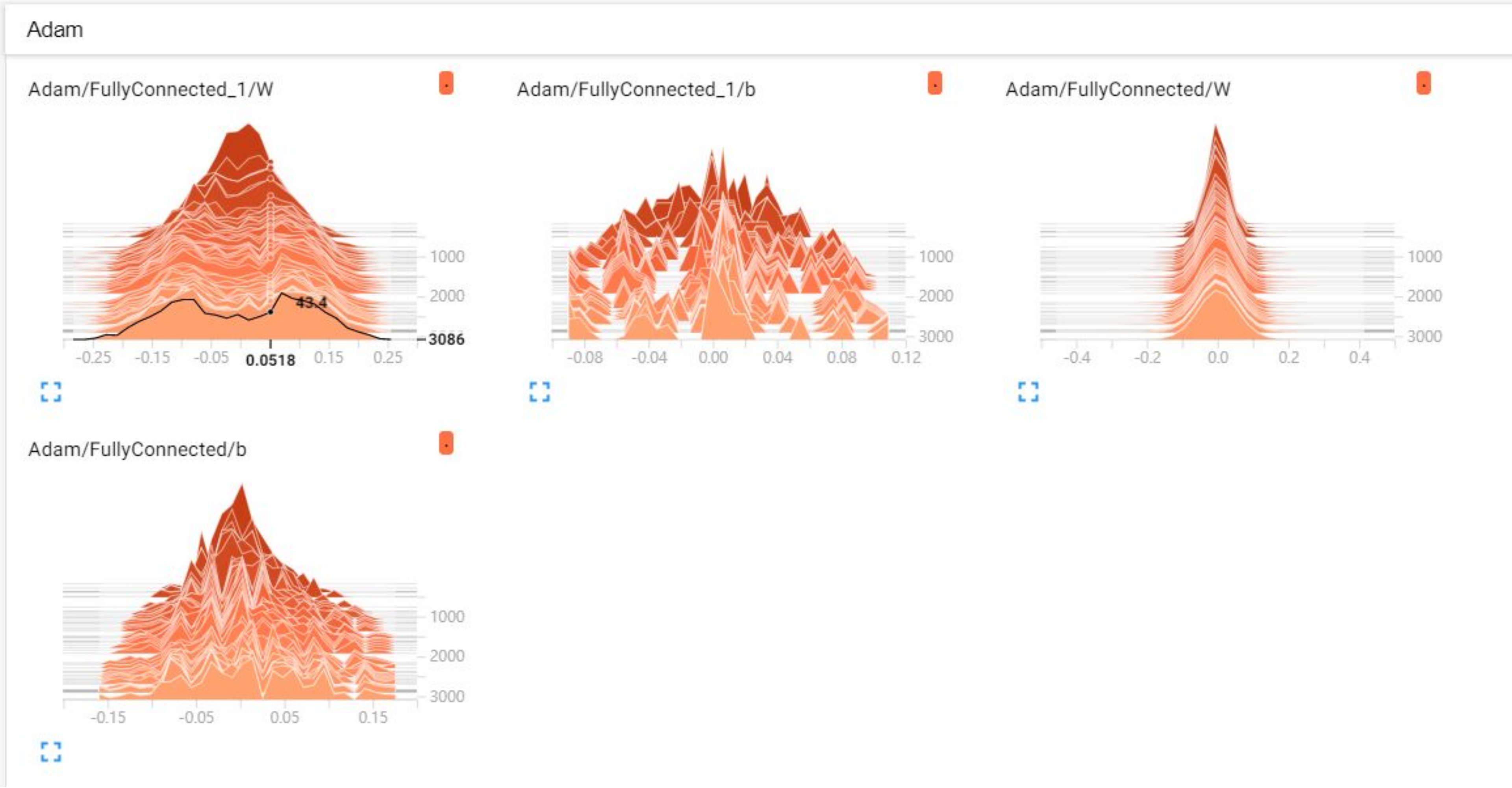


FullyConnected/W/Gradients



FullyConnected/b/Gradients




Adam
Adam/FullyConnected_1/W
1000
2000
3086
43.4
-0.25
-0.15
-0.05
0.0518
0.15
0.25
Adam/FullyConnected_1/b
1000
2000
3000
-0.08
-0.04
0.00
0.04
0.08
0.12
Adam/FullyConnected/W
1000
2000
3000
-0.4
-0.2
0.0
0.2
0.4
Adam/FullyConnected/b
1000
2000
3000
-0.15
-0.05
0.05
0.15


FullyConnected_1
FullyConnected_1/Softmax/Activations
4.00
182
0.353
0.1 0.5 0.7 0.9 1.1
1000 2000 3000
FullyConnected_1/W/Gradients
-1.2 -0.8 -0.4 0.0 0.4 0.8 1.2
1000 2000 3000
FullyConnected_1/b/Gradients
-0.10 -0.06 -0.02 0.02 0.06 0.10
1000 2000 3000
FullyConnected
FullyConnected/BiasAdd/Activations
-30 -10 10 30 50
1000 2000 3000
FullyConnected/W/Gradients
-0.015 -0.005 0.005 0.015
1000 2000 3000
FullyConnected/b/Gradients
-0.015 -0.005 0.005 0.015
1000 2000 3000