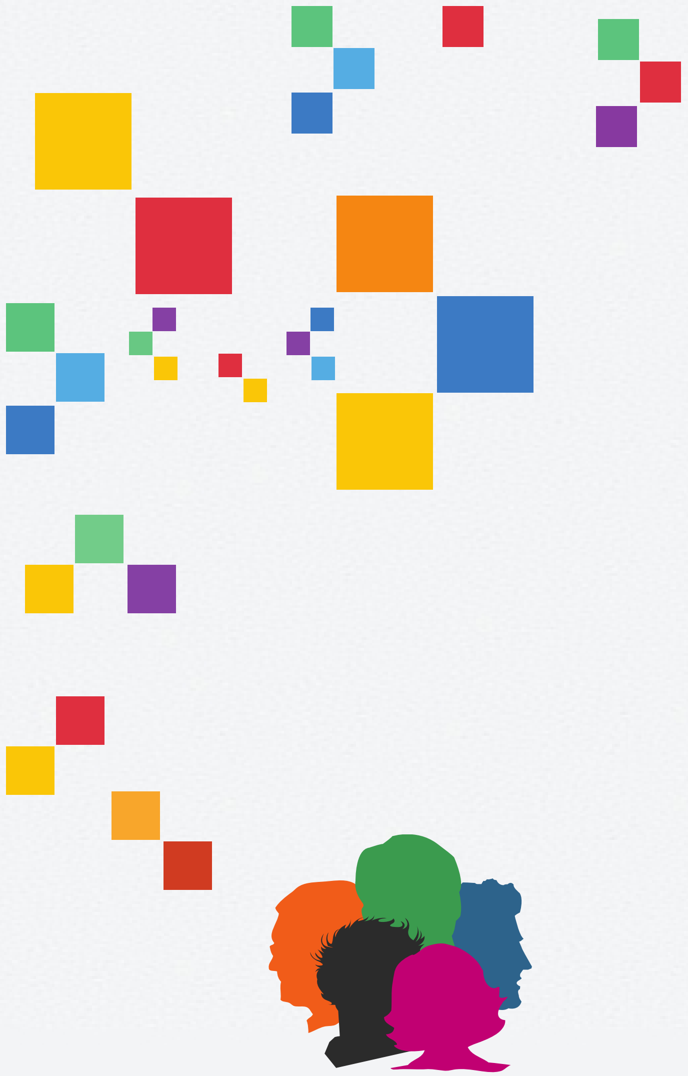
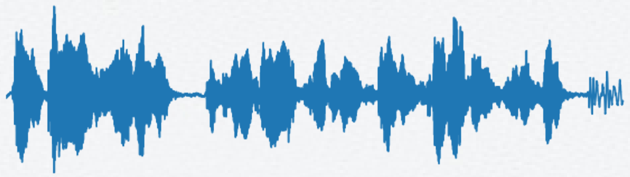
TensorFlow
在智能语音系统中的应用

李嘉璇 《TensorFlow技术解析与实战》作者

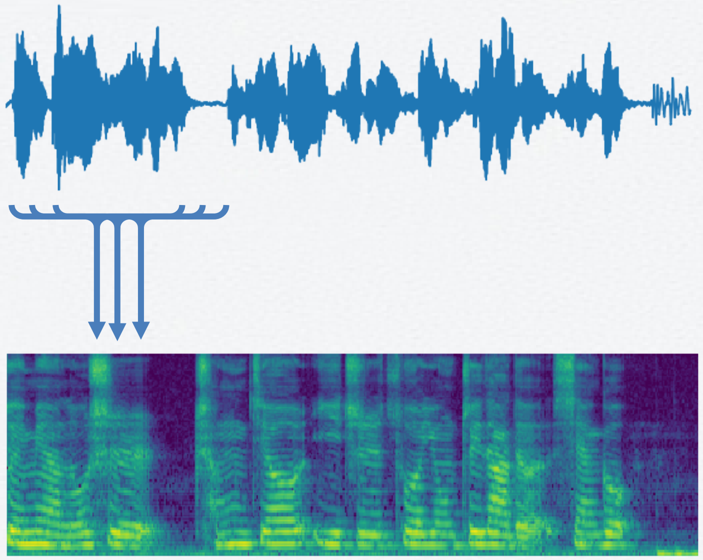


语音识别任务



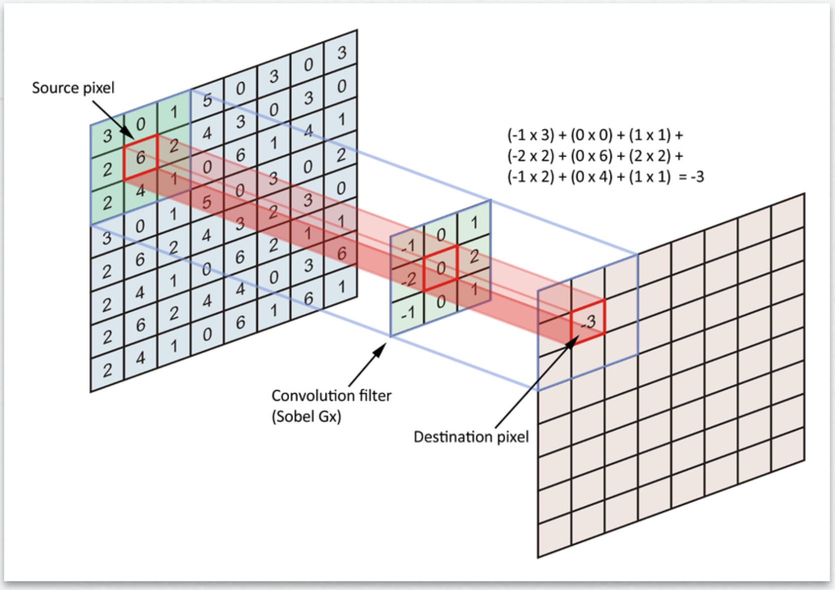
今天天气怎么样？

Mel时频谱

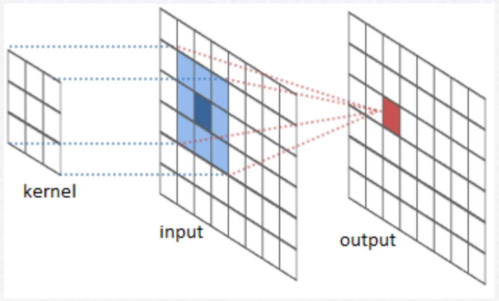


```
import librosa
...
audio, sr = librosa.load(audio_path,
sr=sample_rate)
spectrogram = librosa.feature.melspectrogram(
    audio, sr,
    n_mels=n_mels, n_fft=n_fft,
hop_length=hop_length,
)
return
np.log(np.maximum(np.transpose(spectrogram),
          INFINITESIMAL))
```

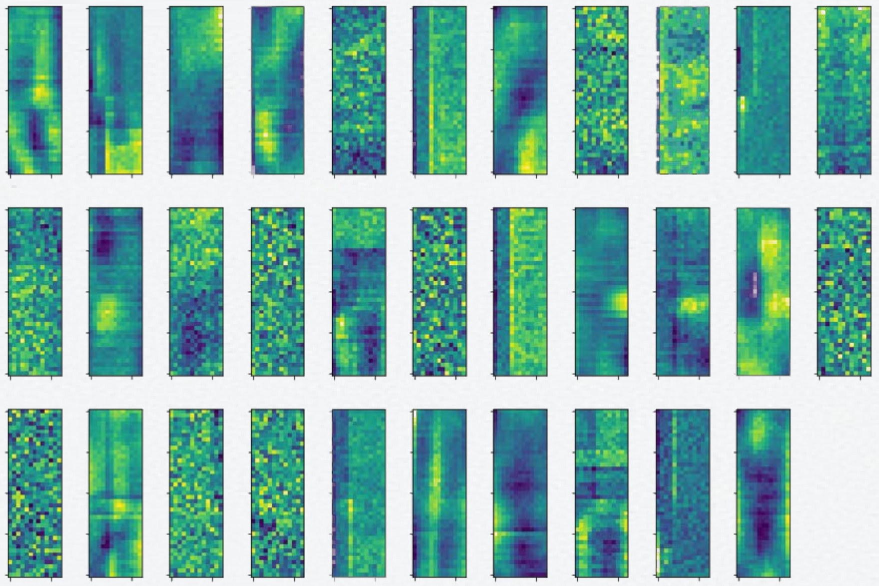
CNN卷积神经网络


Source pixel
(-1 x 3) + (0 x 0) + (1 x 1) +
(-2 x 2) + (0 x 6) + (2 x 2) +
(-1 x 2) + (0 x 4) + (1 x 1) = -3
Convolution filter
(Sobel Gx)
Destination pixel

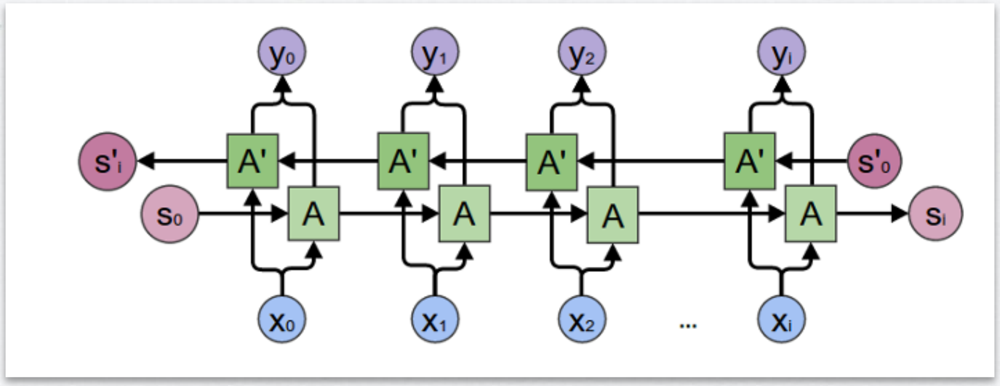
```
output = tf.nn.relu(
    tf.nn.bias_add(
        tf.nn.conv2d(
            tf.expand_dims(inputs, axis=3),
                    # [b, T, f] -> [b, T, f, 1]
            kernels,
                    # [1, w, h, K]
            [1, stride_t, stride_f, 1],
            padding=padding
        ),
        biases
    )
)
```



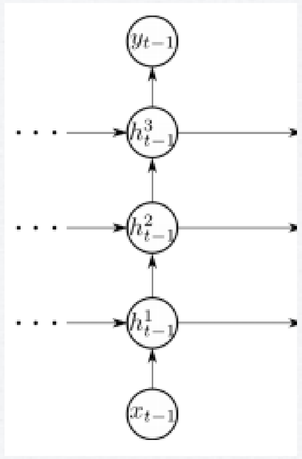
卷积核



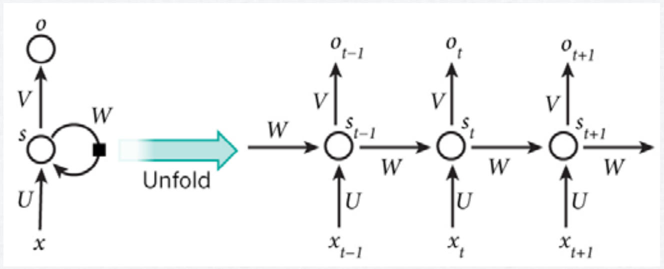
RNN 循环神经网络



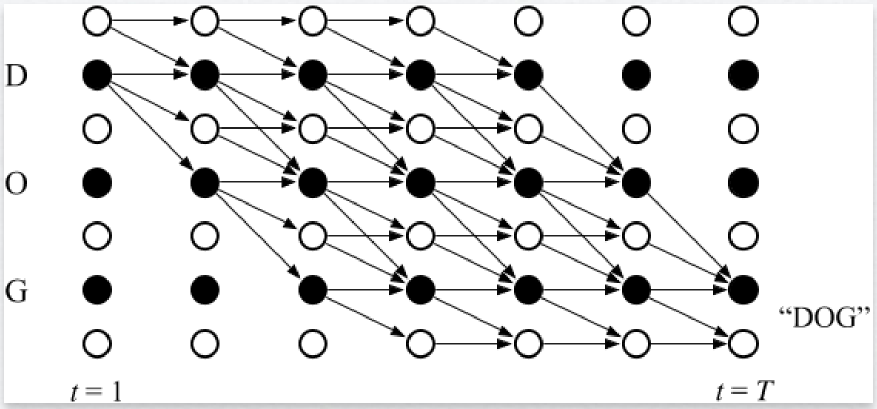
```python
def get_multi_rnn_cell():
    return tf.contrib.rnn.MultiRNNCell(
        [tf.contrib.rnn.GRUCell(num_units)
         for _ in range(num_layers)]
    )

cell_fw = get_multi_rnn_cell()
cell_bw = get_multi_rnn_cell()
```



```
state_fw = cell_fw.zero_state(batch_size,
tf.float32)
state_bw = cell_bw.zero_state(batch_size,
tf.float32)
outputs, states = tf.nn.bidirectional_dynamic_rnn(
    cell_fw, cell_bw, inputs, sequence_length,
    state_fw, state_bw
)
```

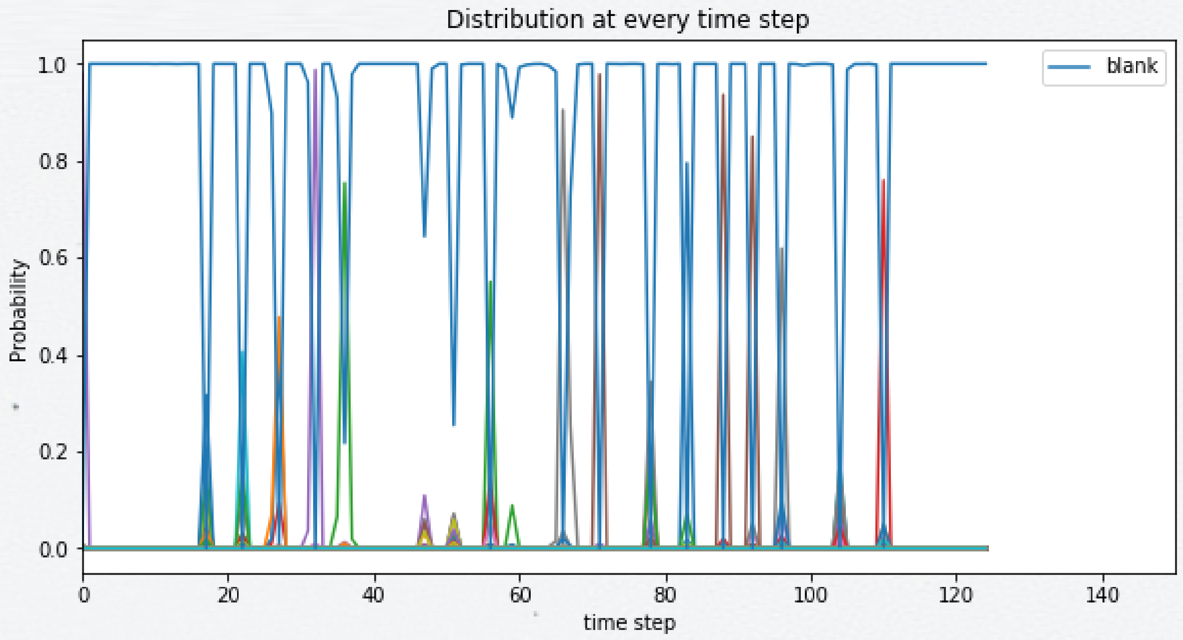


CTC: Connectionist Temporal Classification

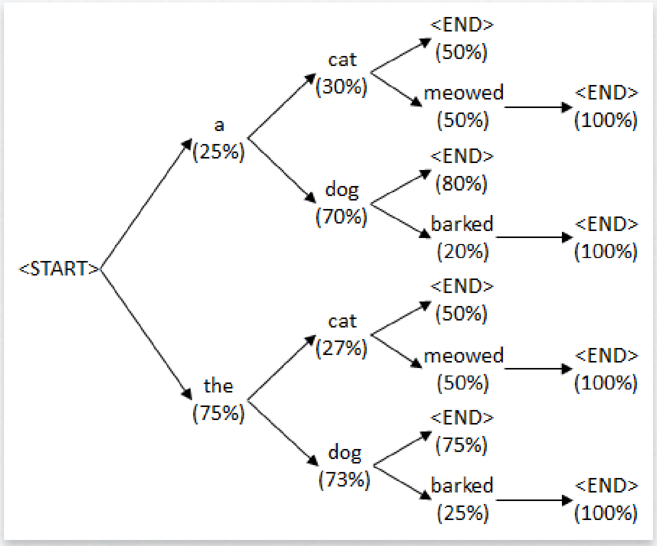


$$ln(p(\mathbf{l}|\mathbf{x})) = \sum_{t=1}^{T} ln(C_t)$$

```
loss = tf.reduce_sum(tf.nn.ctc_loss(
    inputs=inputs,  # [b, t, v+1]
    labels=labels,  # sparse [b, t]
    sequence_length=input_length,
# [b]
    ctc_merge_repeated=True,
    time_major=False
)) / batch_size
```

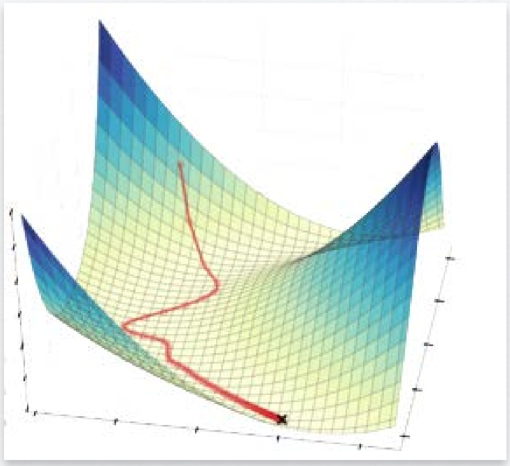

Distribution at every time step
Probability
time step
blank
1.0
0.8
0.6
0.4
0.2
0.0
0
20
40
60
80
100
120
140

CTC解码: Beam Search



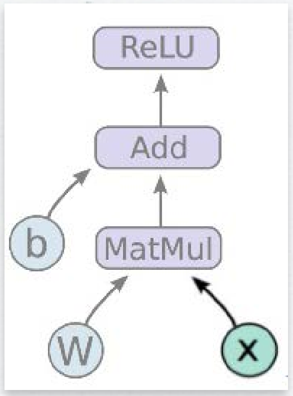
```
result, log_prob =
tf.nn.ctc_beam_search_decoder(
    inputs=inputs,
    beam_width=beam_width,
    sequence_length=sequence_length,
    top_paths=num_top_paths
)
```

使用Optimizer训练模型



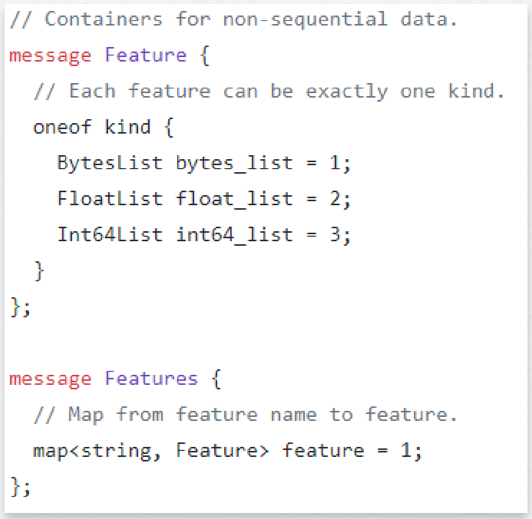
```
optimizer = tf.train.AdamOptimizer(lr)

update_ops =
tf.get_collection(tf.GraphKeys.UPDATE_OPS)
train_op =
tf.contrib.slim.learning.create_train_op(
    loss,
    optimizer,
    update_ops=update_ops,
    clip_gradient_norm=Config.max_grad_norm,
    summarize_gradients=True
)
```

数据馈入（feed）

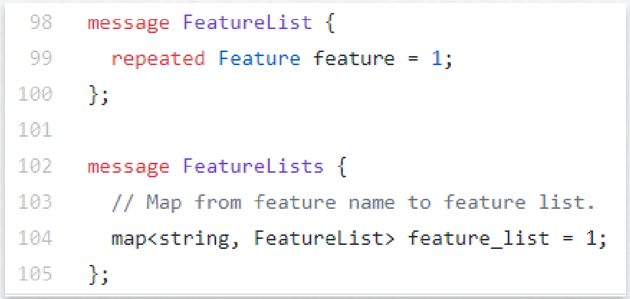


```
with tf.Graph().as_default(), tf.Session() as sess:
    spectrogram_ph = tf.Placeholder(...)
    ...
    loss_v, _ = sess.run(
        [loss, train_op],
        feed_dict={
            spectrogram_ph: spectrogram,
            ...
        }
    )
    print(loss_v)
```

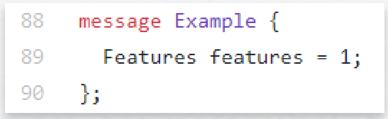
TF Record：高效率的文件读写方案



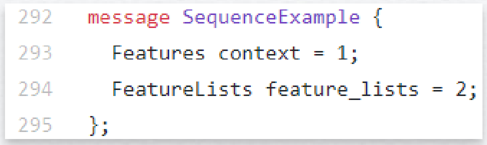
```
// Containers for non-sequential data.
message Feature {
  // Each feature can be exactly one kind.
  oneof kind {
    BytesList bytes_list = 1;
    FloatList float_list = 2;
    Int64List int64_list = 3;
  }
};

message Features {
  // Map from feature name to feature.
  map<string, Feature> feature = 1;
};
```



```
message FeatureList {
  repeated Feature feature = 1;
};

message FeatureLists {
  // Map from feature name to feature list.
  map<string, FeatureList> feature_list = 1;
};
```



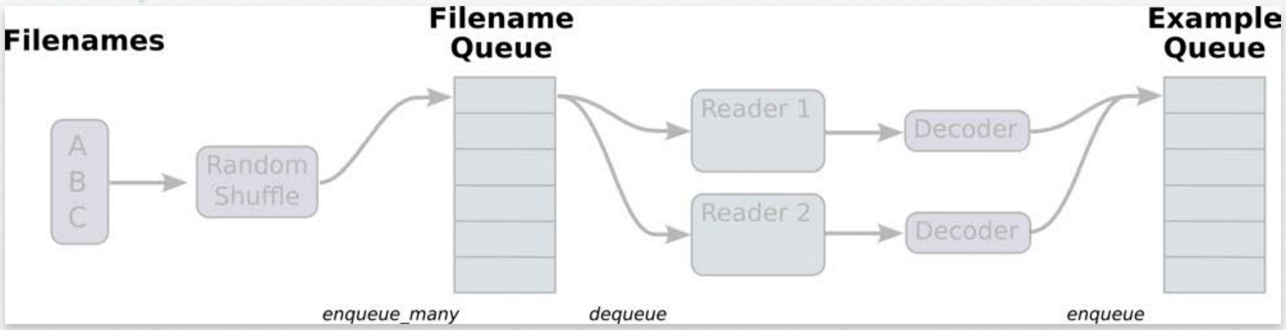
```
message Example {
  Features features = 1;
};
```



```
message SequenceExample {
  Features context = 1;
  FeatureLists feature_lists = 2;
};
```

```
with tf.python_io.TFRecordWriter(filename) as writer:
    example = tf.train.SequenceExample(
        context=tf.train.Features(feature={
            'labels': _int64_feature(labels)
        }),
        feature_lists=tf.train.FeatureLists(feature_list={
            'spectrogram': tf.train.FeatureList(feature=[
                _float_feature(frame.tolist())
                for frame in spectrogram
            ]),
        })
    )
    writer.write(example.SerializeToString())
```

基于Queue的多线程工作流



```
filename_queue = tf.train.string_input_producer(
     filenames, num_epochs=num_epochs
)
reader = tf.TFRecordReader()
_, serialized_example = reader.read(filename_queue)
```

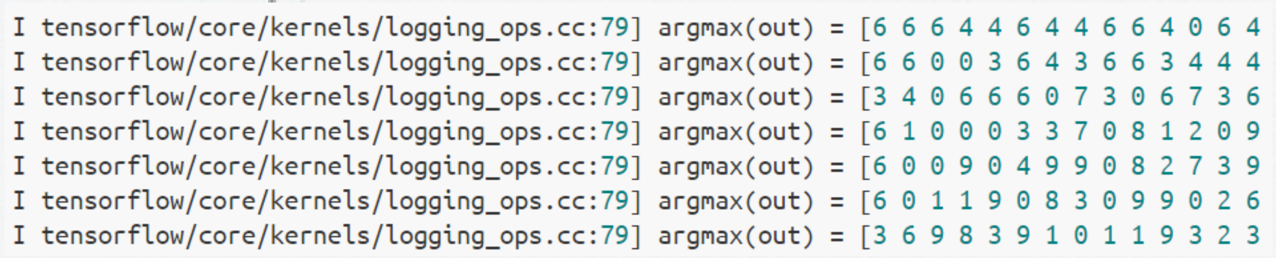
```
context, sequence = tf.parse_single_sequence_example(
    serialized_example,
    context_features={
        'labels': tf.VarLenFeature(tf.int64)
    },
    sequence_features={
        'spectrogram':
         tf.FixedLenSequenceFeature([freq_size],
                                    tf.float32),
    }
)
```

```
spectrogram, labels = tf.train.batch(
    [spectrogram, labels],
    batch_size=batch_size,
    num_threads=num_threads,
    capacity=capacity,
    dynamic_pad=True
)
```

```
coord = tf.train.Coordinator()
threads = tf.train.start_queue_runners(
    sess=sess,
    coord=coord
)

while not coord.should_stop():
    loss_v, _ = sess.run(loss, train_op)
```

使用tf.Print()显示中间结果

```
output = tf.Print(output, [output] , ‘argmax_out = ’, summary=20)
```



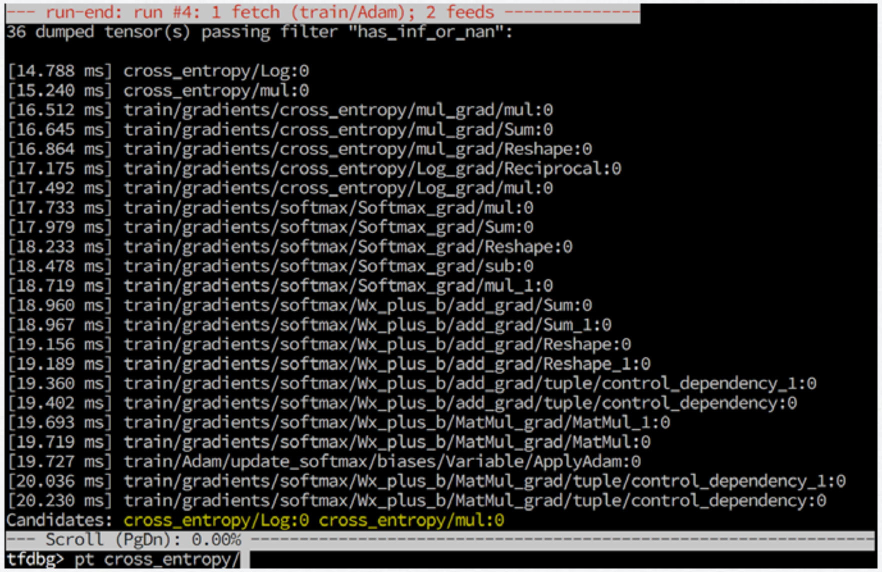
```
I tensorflow/core/kernels/logging_ops.cc:79] argmax(out) = [6 6 6 4 4 6 4 4 6 6 4 0 6 4
I tensorflow/core/kernels/logging_ops.cc:79] argmax(out) = [6 6 0 0 3 6 4 3 6 6 3 4 4 4
I tensorflow/core/kernels/logging_ops.cc:79] argmax(out) = [3 4 0 6 6 6 0 7 3 0 6 7 3 6
I tensorflow/core/kernels/logging_ops.cc:79] argmax(out) = [6 1 0 0 0 3 3 7 0 8 1 2 0 9
I tensorflow/core/kernels/logging_ops.cc:79] argmax(out) = [6 0 0 9 0 4 9 9 0 8 2 7 3 9
I tensorflow/core/kernels/logging_ops.cc:79] argmax(out) = [6 0 1 1 9 0 8 3 0 9 9 0 2 6
I tensorflow/core/kernels/logging_ops.cc:79] argmax(out) = [3 6 9 8 3 9 1 0 1 1 9 3 2 3
```

使用tf.Assert()作断言

```
assert_op = tf.Assert(tf.reduce_all(out > 0),
                      [out], name='assert_out_positive')
with tf.control_dependencies([assert_op]):
    out = tf.identity(out, name='out')
```

```
out = tf.with_dependencies([assert_op], out)
```

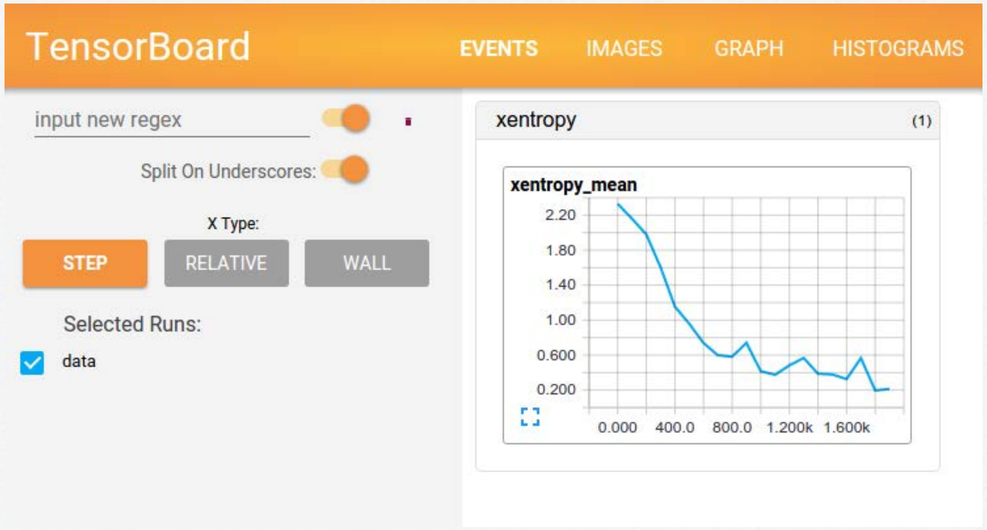
tfdbg：TensorFlow官方调试环境



```
--- run-end: run #4: 1 fetch (train/Adam); 2 feeds --------------
36 dumped tensor(s) passing filter "has_inf_or_nan":

[14.788 ms] cross_entropy/Log:0
[15.240 ms] cross_entropy/mul:0
[16.512 ms] train/gradients/cross_entropy/mul_grad/mul:0
[16.645 ms] train/gradients/cross_entropy/mul_grad/Sum:0
[16.864 ms] train/gradients/cross_entropy/mul_grad/Reshape:0
[17.175 ms] train/gradients/cross_entropy/Log_grad/Reciprocal:0
[17.492 ms] train/gradients/cross_entropy/Log_grad/mul:0
[17.733 ms] train/gradients/softmax/Softmax_grad/mul:0
[17.979 ms] train/gradients/softmax/Softmax_grad/Sum:0
[18.233 ms] train/gradients/softmax/Softmax_grad/Reshape:0
[18.478 ms] train/gradients/softmax/Softmax_grad/sub:0
[18.719 ms] train/gradients/softmax/Softmax_grad/mul_1:0
[18.960 ms] train/gradients/softmax/Wx_plus_b/add_grad/Sum:0
[18.967 ms] train/gradients/softmax/Wx_plus_b/add_grad/Sum_1:0
[19.156 ms] train/gradients/softmax/Wx_plus_b/add_grad/Reshape:0
[19.189 ms] train/gradients/softmax/Wx_plus_b/add_grad/Reshape_1:0
[19.360 ms] train/gradients/softmax/Wx_plus_b/add_grad/tuple/control_dependency_1:0
[19.402 ms] train/gradients/softmax/Wx_plus_b/add_grad/tuple/control_dependency:0
[19.693 ms] train/gradients/softmax/Wx_plus_b/MatMul_grad/MatMul_1:0
[19.719 ms] train/gradients/softmax/Wx_plus_b/MatMul_grad/MatMul:0
[19.727 ms] train/Adam/update_softmax/biases/Variable/ApplyAdam:0
[20.036 ms] train/gradients/softmax/Wx_plus_b/MatMul_grad/tuple/control_dependency_1:0
[20.230 ms] train/gradients/softmax/Wx_plus_b/MatMul_grad/tuple/control_dependency:0
Candidates: cross_entropy/Log:0 cross_entropy/mul:0
--- Scroll (PgDn): 0.00% -----------------------------------------------------------
tfdbg> pt cross_entropy/
```

```
with tf.Graph().as_default(), tf.Session() as sess:

    ...
    from tensorflow.python import debug as tf_debug
    sess = tf_debug.LocalCLIDebugWrapperSession(sess)
    sess.add_tensor_filter( “has_inf_or_nan” ,
                                  tf_debug.has_inf_or_nan)
```

使用TensorBoard可视化训练过程



```
loss = ...
tf.summary.scalar('loss', loss)

...

tvars = tf.trainable_variables()
for var in tvars:
    tf.summary.histogram(var.name, var)

...
summary = tf.summary.merge_all()
```

```
writer = tf.summary.FileWriter('summary')
writer.add_graph(sess.graph)

...
s = sess.run(summary)
writer.add_summary(s, global_step)
```

感谢您参加本届MPD！

www.mpd.org.cn

400-812-8020