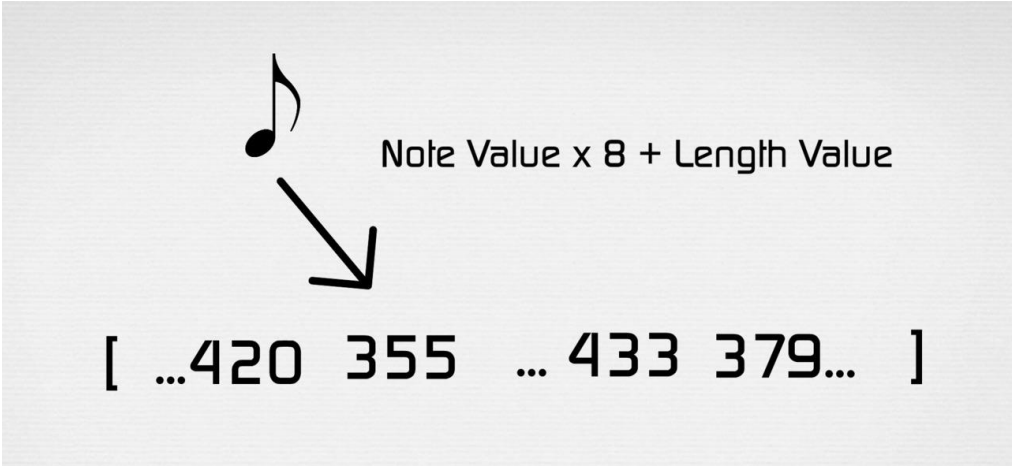
Music Generation

skid

davinnovation AT gmail.com

Music generation with RNN

http://davinnovation.github.io/midi_generate_rnn.html



inspired by aikorea.org/blog

davinnovation AT gmail.com

Music generation with RNN

http://davinnovation.github.io/midi_generate_rnn.html

$$P(w_1, \ldots, w_m) = \prod_{i=1}^{m} P(w_i \mid w_1, \ldots, w_{i-1}) \approx \prod_{i=1}^{m} P(w_i \mid w_{i-(n-1)}, \ldots, w_{i-1})$$

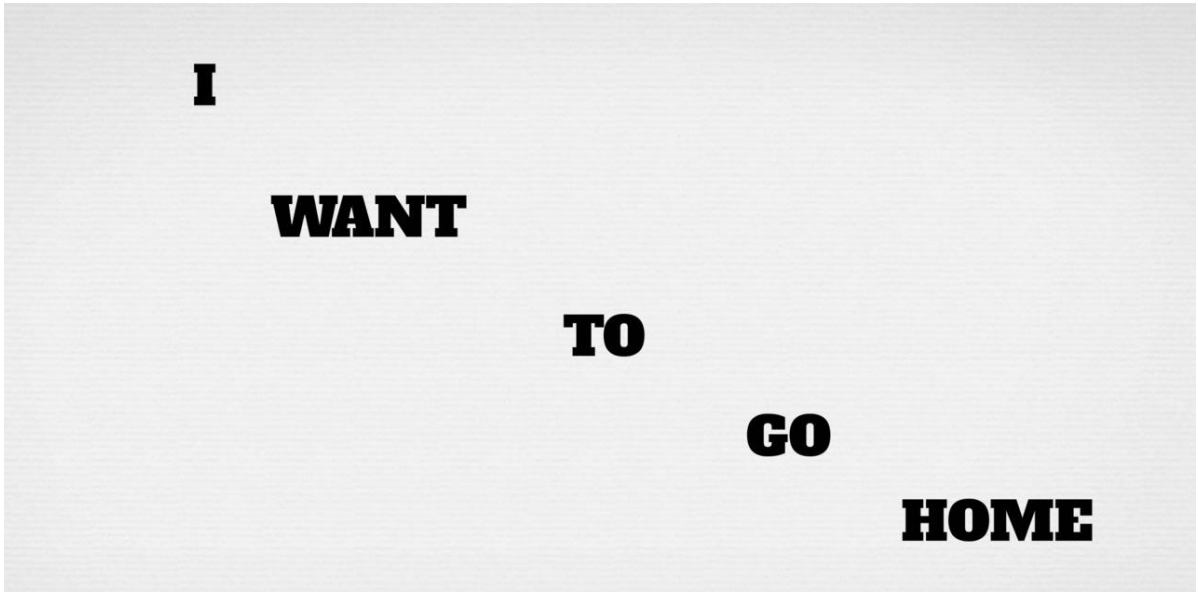
< n-gram model >

inspired by aikorea.org/blog

davinnovation AT gmail.com

Music generation with RNN

http://davinnovation.github.io/midi_generate_rnn.html

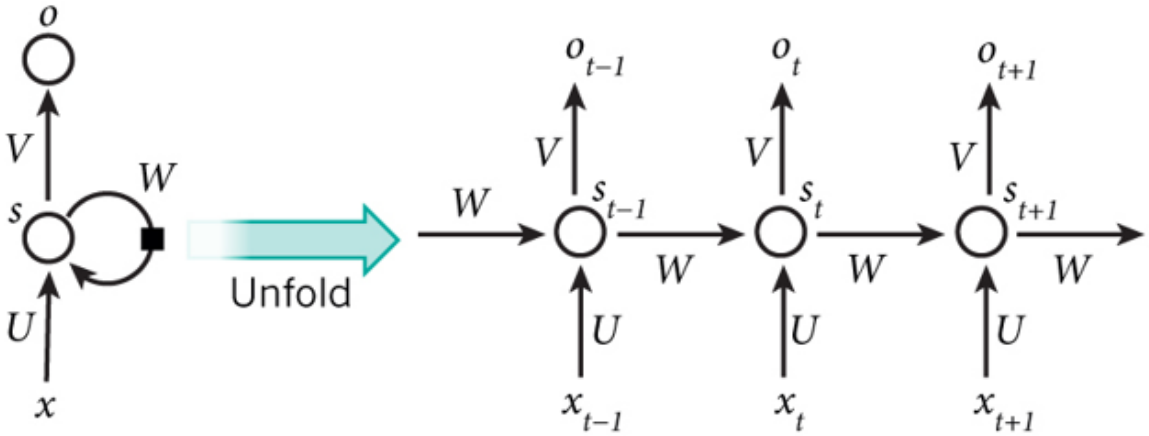


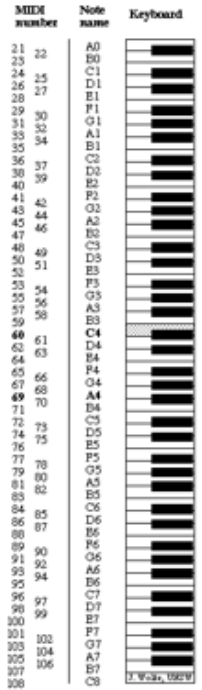
inspired by aikorea.org/blog

davinnovation AT gmail.com

Music generation with RNN

http://davinnovation.github.io/midi_generate_rnn.html





$X_t \in R^{88+2}$: Midi Pitch의 범위는 21 ~ 108 + START_TOKEN + END_TOKEN

$O_t \in R^{88+2}$: x의 output

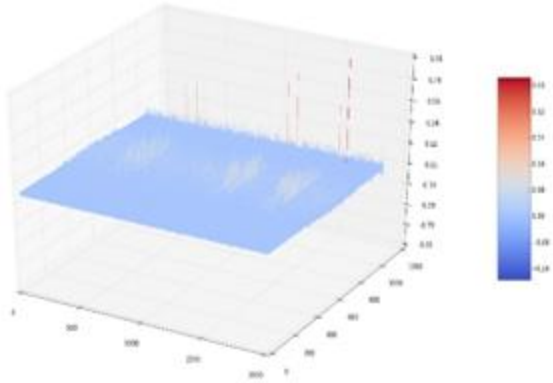
$S_t \in R^{50}$: Hidden Layer의 개수 (default 50이나 실험 결과에 따라 유동적으로 변화)

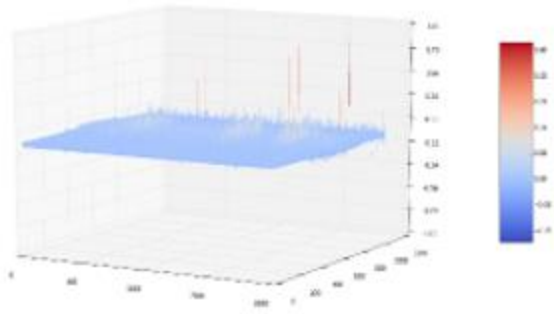
$U \in R^{50*90}$ / $V \in R^{90*50}$ / $W \in R^{50*50}$

davinnovation AT gmail.com

Music generation with RNN

http://davinnovation.github.io/midi_generate_rnn.html





Loss function $L(y, o) = -\frac{1}{N}\sum_{n \in N} y_n \log o_n$

U = U - learning rate * $\frac{\partial L}{\partial U}$

V = V - learning rate * $\frac{\partial L}{\partial V}$

W = W - learning rate * $\frac{\partial L}{\partial W}$

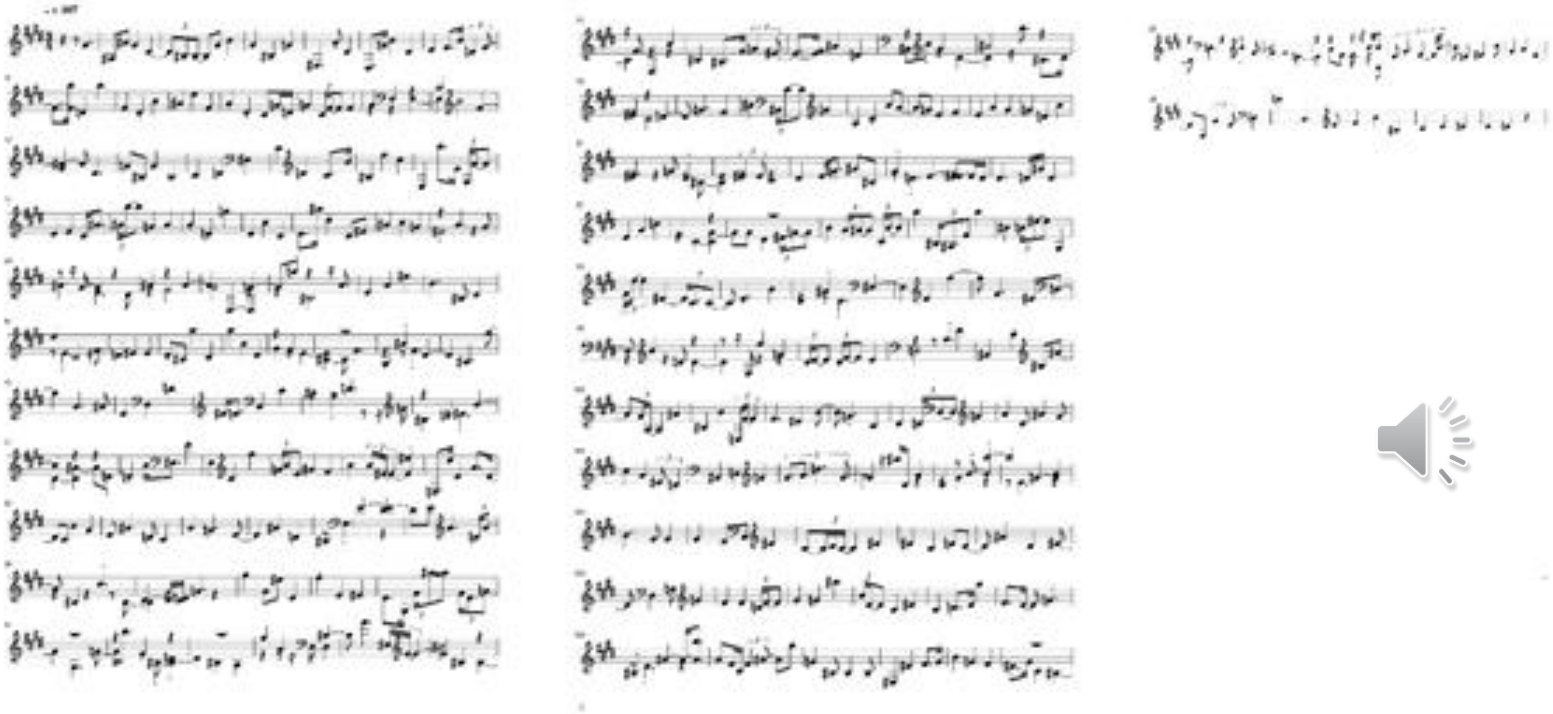
Epoch 0 loss: 6.93361 (expected $log 1026$)

Epoch 80 loss: 4.97166

davinnovation AT gmail.com

Music generation with RNN

http://davinnovation.github.io/midi_generate_rnn.html



davinnovation AT gmail.com

magenta

ML for art & music

davinnovation AT gmail.com

What is Magenta?

Use Machine learning to create art and music (June. 2016)

by Google Brain Team (Jeff Dean etc)

davinnovation AT gmail.com

Install Magenta

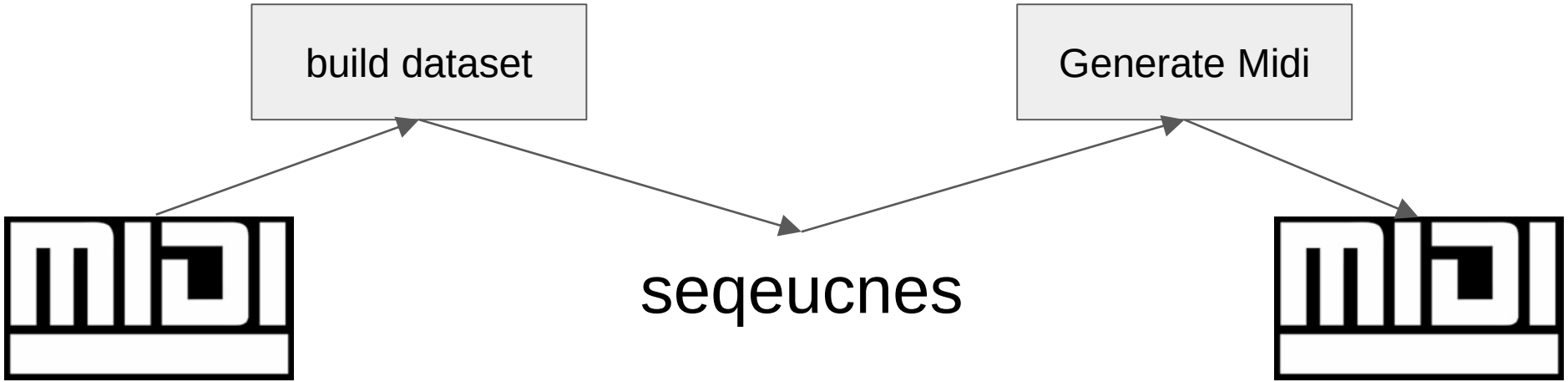
0. Only On Unbuntu 14.04 LTS

1. Bazel > 0.2.3. && Tensorflow > 0.9

2. [git clone https://github.com/tensorflow/magenta.git]

3. [bazel test //magenta/…]

Magenta Learning & Generating



Magenta - Build Dataset

conevert_midi_dir_to_note_sequences.py

> Seqeunces

Time Signatures ...about (4/4, 4/2 ~)

Key Signatures ...about key (Minor ~ Major , C0 ~ B0)

Tempo … about (note..?)

Instrument events

http://frontjang.info/entry/MIDI-%ED%8C%8C%EC%9D%BC-%EB%B6%84%EC%84%9D%ED%95%98%EA%B8%B0-2-Meta-events

http://www.deluge.co/?q=midi-tempo-bpm

Magenta - Generate Midi

Seqeunces -> Melody [note 48 to 84] //

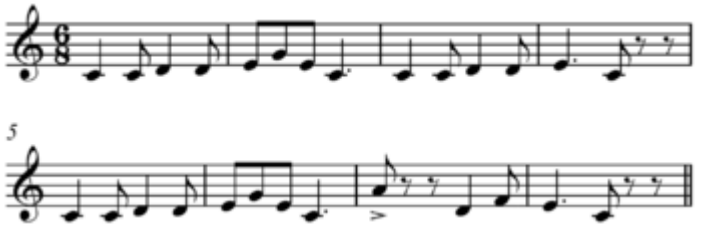
Basic RNN (shared) : LSTM cell

Lookback RNN : shared +

Attention RNN : +attention option

Basic RNN - Learn

Input : MonoPhonic Data
[note off, no event, note-on] for every pitch

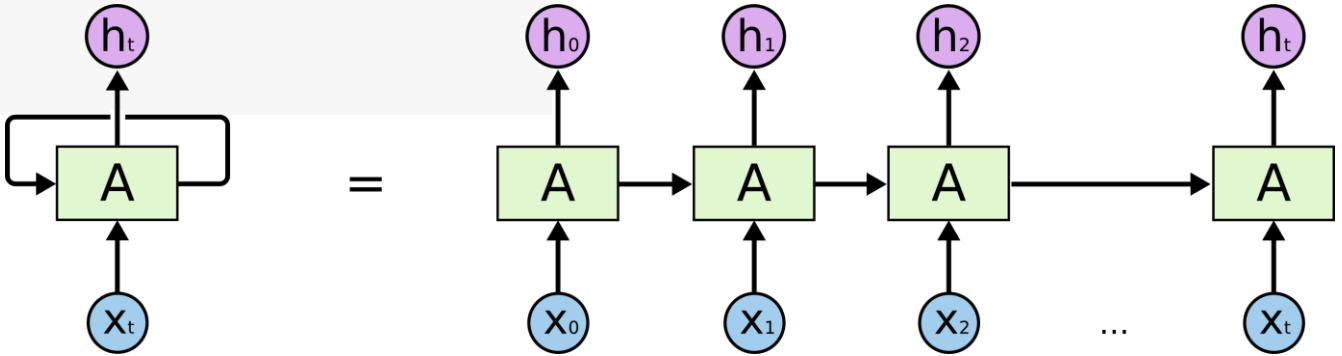


extract a melody line if it is at least **7 measures long**, and at least **5 unique pitches** (with octave equivalence). If multiple notes play at the same time, one note is kept.

Basic RNN - Generate

```
# Provide a MIDI file to use as a primer for the generation.
# The MIDI should just contain a short monophonic melody.
# primer.mid is provided as an example.
PRIMER_PATH=<absolute path of your primer MIDI file>

bazel run //magenta/models/basic_rnn:basic_rnn_generate -- \
--run_dir=/tmp/basic_rnn/logdir/run1 \
--hparams='{"rnn_layer_sizes":[50]}' \
--output_dir=/tmp/basic_rnn/generated \
--num_outputs=10 \
--num_steps=128 \
--primer_midi=$PRIMER_PATH
```



Lookback RNN - Learn

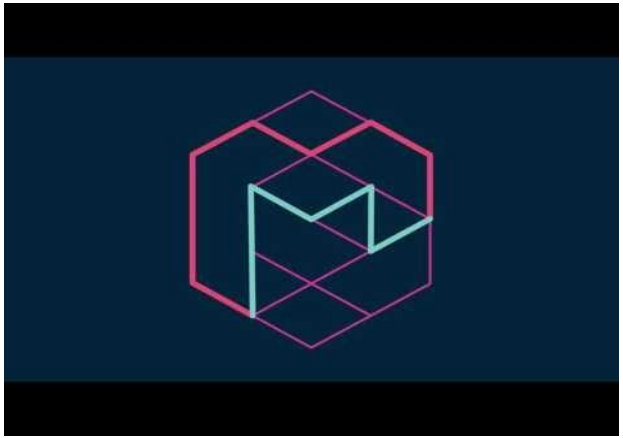
Input : MonoPhonic Data
[note off, no event, note-on] for every pitch

+

1. Input events from 1 and 2 bars ago
2. Input whether the last event was repeating the event from 1 or 2 bars before it
3. Input the current position within the measure

Lookback RNN - Generate

```
bazel run //magenta/models/lookback_rnn:lookback_rnn_generate -- \
--run_dir=/tmp/lookback_rnn/logdir/run1 \
--hparams="{'batch_size':64,'rnn_layer_sizes':[64,64]}" \
--output_dir=/tmp/lookback_rnn/generated \
--num_outputs=10 \
--num_steps=128 \
--primer_melody="[60]"
```



Attention RNN - Learn

Input : MonoPhonic Data
[note off, no event, note-on] for every pitch

\+ (from Grammer as a Foreign Language [http://arxiv.org/abs/1412.7449])

1. look at the outputs from the last n steps when generating the output for the current step

$$u_i^t = v^T \tanh(W_1' h_i + W_2' d_t)$$
$$a_i^t = \text{softmax}(u_i^t)$$
$$d_t' = \sum_{i=1}^{T_A} a_i^t h_i$$

v and matrices W_1', W_2' are learnable parameters of the model.

hi are the RNN outputs from the previous n steps

dt is the current step's RNN cell state

a mask-like vector a_i^t

Attention RNN – Learn [step by step in 1]

Gen Step 1

If we are on the 4th step

Step 1: $[1.0, 0.0, 0.0, 1.0]$
Step 2: $[0.0, 1.0, 0.0, 1.0]$
Step 3: $[0.0, 0.0, 0.5, 0.0]$

$$u_i^t = v^T \tanh(W_1' h_i + W_2' d_t)$$
$$a_i^t = \text{softmax}(u_i^t)$$
$$d_t' = \sum_{i=1}^{T_A} a_i^t h_i$$

Attention RNN – Learn [step by step in 2]

Gen Step 2

If we are on the 4th step

Attention mask

$$a_i^t = [0.7, 0.1, 0.2]$$

```
def _attention(self, state, attn_states):
  with tf.variable_scope('Attention'):
    v = tf.get_variable('V', [self._attn_vec_size])
    attn_states_flat = tf.reshape(attn_states, [-1, self._attn_size])
    attn_states_vec = tf.contrib.layers.linear(
        attn_states_flat, self._attn_vec_size, scope='AttnStatesVec')
    attn_states_vec = tf.reshape(
        attn_states_vec, [-1, self._attn_length, self._attn_vec_size])
    state_vec = tf.contrib.layers.linear(
        state, self._attn_vec_size, scope='StateVec')
    state_vec = tf.expand_dims(state_vec, 1)
    attn = tf.reduce_sum(v * tf.tanh(attn_states_vec + state_vec), 2)
    attn_mask = tf.nn.softmax(attn)
    attn_mask = tf.expand_dims(attn_mask, 2)
    return tf.reduce_sum(attn_states * attn_mask, 1)
```

Attention RNN – Learn [step by step in 3]

Gen Step 3

If we are on the 4th step

$$a_i^t = [0.7, 0.1, 0.2]$$

Step 1 (70%): $[0.7, 0.0, 0.0, 0.7]$
Step 2 (10%): $[0.0, 0.1, 0.0, 0.1]$
Step 3 (20%): $[0.0, 0.0, 0.1, 0.0]$

$$u_i^t = v^T \tanh(W_1' h_i + W_2' d_t)$$
$$a_i^t = \text{softmax}(u_i^t)$$
$$d_t' = \sum_{i=1}^{T_A} a_i^t h_i$$

Attention RNN – Learn [step by step in 4]

Gen Step 4

If we are on the 4th step

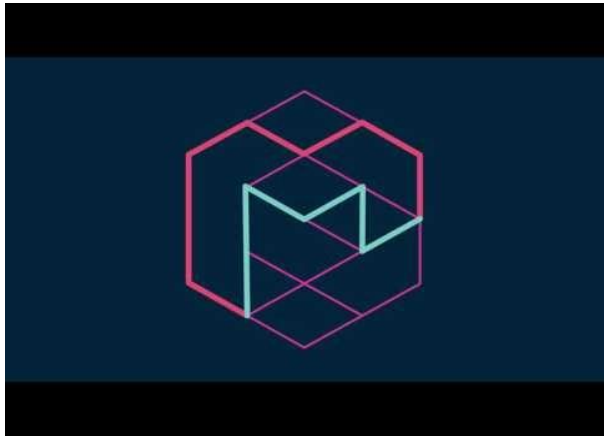
Step 1 (70%): $[0.7, 0.0, 0.0, 0.7]$
Step 2 (10%): $[0.0, 0.1, 0.0, 0.1]$
Step 3 (20%): $[0.0, 0.0, 0.1, 0.0]$

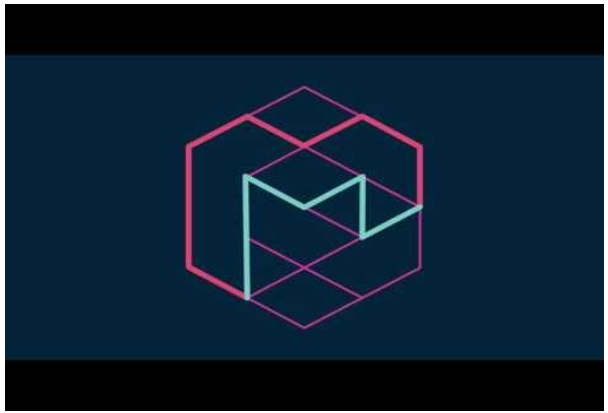
$$u_i^t = v^T \tanh(W_1' h_i + W_2' d_t)$$
$$a_i^t = \text{softmax}(u_i^t)$$
$$d_t' = \sum_{i=1}^{T_A} a_i^t h_i$$

$$d_t' = [0.7, 0.1, 0.1, 0.8]$$

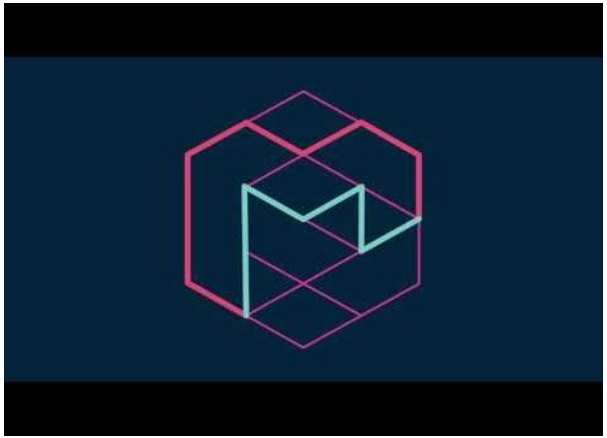
Attention RNN - Generate

```
bazel run //magenta/models/attention_rnn:attention_rnn_generate -- \
--run_dir=/tmp/attention_rnn/logdir/run1 \
--hparams="{'batch_size':64,'rnn_layer_sizes':[64,64]}" \
--output_dir=/tmp/attention_rnn/generated \
--num_outputs=10 \
--num_steps=128 \
--primer_melody="[60]"
```





Google’s First Song



Same Initial

Combine Melody 1 & 2
AABA + drums