Show code

```
Start coding or generate with AI.
```

Generate music with an RNN

Double-click (or enter) to edit

Double-click (or enter) to edit

```
Start coding or generate with AI.
```

```
Start coding or generate with AI.
```

 View on TensorFlow.org  Run in Google Colab  View source on GitHub  Download notebook

This tutorial shows you how to generate musical notes using a simple recurrent neural network (RNN). You will train a model using a collection of piano MIDI files from the MAESTRO dataset. Given a sequence of notes, your model will learn to predict the next note in the sequence. You can generate longer sequences of notes by calling the model repeatedly.

This tutorial contains complete code to parse and create MIDI files. You can learn more about how RNNs work by visiting the Text generation with an RNN tutorial.

Setup

This tutorial uses the `pretty_midi` library to create and parse MIDI files, and `pyfluidsynth` for generating audio playback in Colab.

```
!sudo apt install -y fluidsynth
```

```
[progress bars] 8update-alternatives: using /usr/share/sou
update-alternatives: using /usr/share/sounds/sf2/TimGM6mb.sf2 to provide /usr/share/sounds/sf3/default-GM.sf3 (default-GM.sf3) i
[progress bars] 8Setting up libinstpatch-1.0-2:amd64 (1.1.
[progress bars] 8Created symlink /etc/systemd/user/multi-u
[progress bars] 8Setting up libqt5svg5:amd64 (5.12.8-0ubun
Processing triggers for mime-support (3.64ubuntu1) ...
Processing triggers for hicolor-icon-theme (0.17-2) ...
Processing triggers for gnome-menus (3.36.0-1ubuntu1) ...
Processing triggers for libc-bin (2.31-0ubuntu9.9) ...
Processing triggers for man-db (2.9.1-1) ...

[escape sequences]
```

```
!pip install --upgrade pyfluidsynth
```

```
Collecting pyfluidsynth
  Downloading pyFluidSynth-1.3.2-py3-none-any.whl (19 kB)
Requirement already satisfied: numpy in /tmpfs/src/tf_docs_env/lib/python3.9/site-packages (from pyfluidsynth) (1.26.1)
Installing collected packages: pyfluidsynth
Successfully installed pyfluidsynth-1.3.2
```

```
!pip install pretty_midi
```

```
Collecting pretty_midi
  Downloading pretty_midi-0.2.10.tar.gz (5.6 MB)
  Preparing metadata (setup.py) ... -done
Requirement already satisfied: numpy>=1.7.0 in /tmpfs/src/tf_docs_env/lib/python3.9/site-packages (from pretty_midi) (1.26.1)
Collecting mido>=1.1.16 (from pretty_midi)
  Downloading mido-1.3.0-py3-none-any.whl.metadata (5.1 kB)
Requirement already satisfied: six in /tmpfs/src/tf_docs_env/lib/python3.9/site-packages (from pretty_midi) (1.16.0)
Requirement already satisfied: packaging~=23.1 in /tmpfs/src/tf_docs_env/lib/python3.9/site-packages (from mido>=1.1.16->pretty_mid
Downloading mido-1.3.0-py3-none-any.whl (50 kB)
Building wheels for collected packages: pretty_midi
  Building wheel for pretty_midi (setup.py) ... -\|done
  Created wheel for pretty_midi: filename=pretty_midi-0.2.10-py3-none-any.whl size=5592287 sha256=f7d88e5b16376925e8b98b6963d75a807
  Stored in directory: /home/kbuilder/.cache/pip/wheels/75/ec/20/b8e937a5bcf1de547ea5ce465db7de7f6761e15e6f0a01e25f
Successfully built pretty_midi
Installing collected packages: mido, pretty_midi
Successfully installed mido-1.3.0 pretty_midi-0.2.10
```

```
import collections
import datetime
import fluidsynth
import glob
import numpy as np
import pathlib
import pandas as pd
import pretty_midi
import seaborn as sns
import tensorflow as tf

from IPython import display
from matplotlib import pyplot as plt
from typing import Optional
```

```
2023-10-27 05:49:15.925119: E external/local_xla/xla/stream_executor/cuda/cuda_dnn.cc:9261] Unable to register cuDNN factory: Attem
2023-10-27 05:49:15.925168: E external/local_xla/xla/stream_executor/cuda/cuda_fft.cc:607] Unable to register cuFFT factory: Attemp
2023-10-27 05:49:15.926725: E external/local_xla/xla/stream_executor/cuda/cuda_blas.cc:1515] Unable to register cuBLAS factory: Att
```

```
seed = 42
tf.random.set_seed(seed)
np.random.seed(seed)

# Sampling rate for audio playback
_SAMPLING_RATE = 16000
```

```
data_dir = pathlib.Path('data/maestro-v2.0.0')
if not data_dir.exists():
  tf.keras.utils.get_file(
      'maestro-v2.0.0-midi.zip',
      origin='https://storage.googleapis.com/magentadata/datasets/maestro/v2.0.0/maestro-v2.0.0-midi.zip',
      extract=True,
      cache_dir='.', cache_subdir='data',
  )
```

```
Downloading data from https://storage.googleapis.com/magentadata/datasets/maestro/v2.0.0/maestro-v2.0.0-midi.zip

    8192/59243107 [..............................] - ETA: 0s
 4202496/59243107 [=>............................] - ETA: 1s
17809408/59243107 [========>.....................] - ETA: 0s
32882688/59243107 [===============>..............] - ETA: 0s
48955392/59243107 [=======================>......] - ETA: 0s
50339840/59243107 [========================>.....] - ETA: 0s
59243107/59243107 [==============================] - 0s 0us/step
```

```
filenames = glob.glob(str(data_dir/'**/*.mid*'))
print('Number of files:', len(filenames))
```

```
Number of files: 1282
```

```
sample_file = filenames[1]
print(sample_file)
```

```
data/maestro-v2.0.0/2008/MIDI-Unprocessed_05_R1_2008_01-04_ORIG_MID--AUDIO_05_R1_2008_wav--4.midi
```

```
pm = pretty_midi.PrettyMIDI(sample_file)
```

```
def display_audio(pm: pretty_midi.PrettyMIDI, seconds=30):
  waveform = pm.fluidsynth(fs=_SAMPLING_RATE)
  # Take a sample of the generated waveform to mitigate kernel resets
  waveform_short = waveform[:seconds*_SAMPLING_RATE]
  return display.Audio(waveform_short, rate=_SAMPLING_RATE)
```

```
display_audio(pm)
```

```
fluidsynth: warning: SDL2 not initialized, SDL2 audio driver won't be usable
fluidsynth: error: Unknown integer parameter 'synth.sample-rate'
```

0:00 / 0:30

```
print('Number of instruments:', len(pm.instruments))
instrument = pm.instruments[0]
instrument_name = pretty_midi.program_to_instrument_name(instrument.program)
print('Instrument name:', instrument_name)
```

```
Number of instruments: 1
Instrument name: Acoustic Grand Piano
```

```
for i, note in enumerate(instrument.notes[:10]):
  note_name = pretty_midi.note_number_to_name(note.pitch)
  duration = note.end - note.start
  print(f'{i}: pitch={note.pitch}, note_name={note_name},'
        f' duration={duration:.4f}')
```

```
0: pitch=54, note_name=F#3, duration=0.0612
1: pitch=51, note_name=D#3, duration=0.0781
2: pitch=58, note_name=A#3, duration=0.0898
3: pitch=39, note_name=D#2, duration=0.0703
4: pitch=46, note_name=A#2, duration=0.1029
5: pitch=39, note_name=D#2, duration=0.0495
6: pitch=51, note_name=D#3, duration=0.0599
7: pitch=46, note_name=A#2, duration=0.0443
8: pitch=54, note_name=F#3, duration=0.0651
9: pitch=63, note_name=D#4, duration=0.9219
```

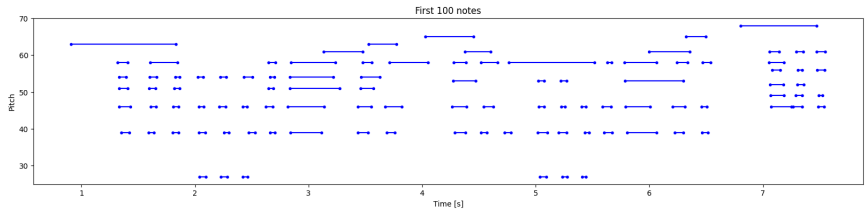
```
def midi_to_notes(midi_file: str) -> pd.DataFrame:
  pm = pretty_midi.PrettyMIDI(midi_file)
  instrument = pm.instruments[0]
  notes = collections.defaultdict(list)

  # Sort the notes by start time
  sorted_notes = sorted(instrument.notes, key=lambda note: note.start)
  prev_start = sorted_notes[0].start

  for note in sorted_notes:
    start = note.start
    end = note.end
    notes['pitch'].append(note.pitch)
    notes['start'].append(start)
    notes['end'].append(end)
    notes['step'].append(start - prev_start)
    notes['duration'].append(end - start)
    prev_start = start

  return pd.DataFrame({name: np.array(value) for name, value in notes.items()})


raw_notes = midi_to_notes(sample_file)
raw_notes.head()
```

	pitch	start	end	step	duration
0	63	0.910156	1.832031	0.000000	0.921875
1	58	1.320312	1.410156	0.410156	0.089844
2	51	1.330729	1.408854	0.010417	0.078125
3	46	1.330729	1.433594	0.000000	0.102865
4	54	1.334635	1.395833	0.003906	0.061198

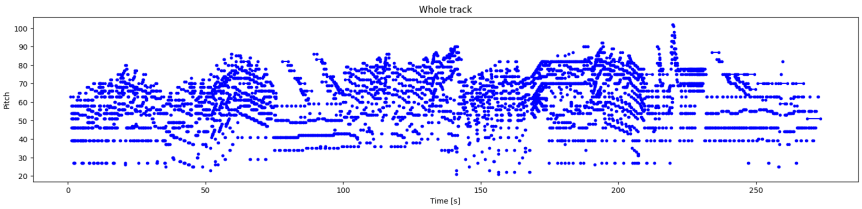
```
get_note_names = np.vectorize(pretty_midi.note_number_to_name)
sample_note_names = get_note_names(raw_notes['pitch'])
sample_note_names[:10]
```

```
array(['D#4', 'A#3', 'D#3', 'A#2', 'F#3', 'D#2', 'D#2', 'D#3', 'F#3',
       'A#3'], dtype='<U3')
```

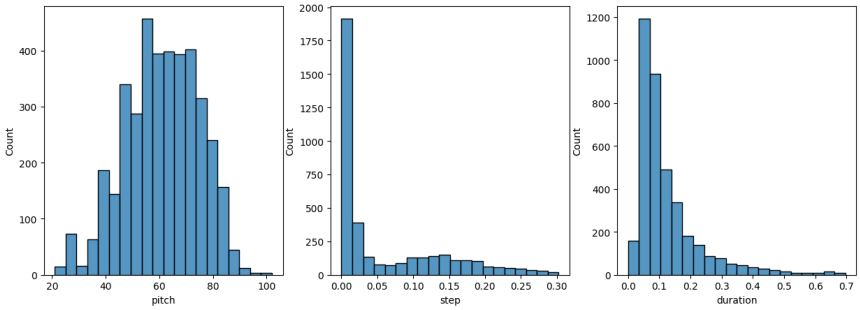
```
def plot_piano_roll(notes: pd.DataFrame, count: Optional[int] = None):
  if count:
    title = f'First {count} notes'
  else:
    title = f'Whole track'
    count = len(notes['pitch'])
  plt.figure(figsize=(20, 4))
  plot_pitch = np.stack([notes['pitch'], notes['pitch']], axis=0)
  plot_start_stop = np.stack([notes['start'], notes['end']], axis=0)
  plt.plot(
      plot_start_stop[:, :count], plot_pitch[:, :count], color="b", marker=".")
  plt.xlabel('Time [s]')
  plt.ylabel('Pitch')
  _ = plt.title(title)


plot_piano_roll(raw_notes, count=100)
```




```
plot_piano_roll(raw_notes)
```




```
def plot_distributions(notes: pd.DataFrame, drop_percentile=2.5):
  plt.figure(figsize=[15, 5])
  plt.subplot(1, 3, 1)
  sns.histplot(notes, x="pitch", bins=20)

  plt.subplot(1, 3, 2)
  max_step = np.percentile(notes['step'], 100 - drop_percentile)
  sns.histplot(notes, x="step", bins=np.linspace(0, max_step, 21))

  plt.subplot(1, 3, 3)
  max_duration = np.percentile(notes['duration'], 100 - drop_percentile)
  sns.histplot(notes, x="duration", bins=np.linspace(0, max_duration, 21))


plot_distributions(raw_notes)
```



```
def notes_to_midi(
  notes: pd.DataFrame,
  out_file: str,
  instrument_name: str,
  velocity: int = 100,  # note loudness
) -> pretty_midi.PrettyMIDI:

  pm = pretty_midi.PrettyMIDI()
  instrument = pretty_midi.Instrument(
      program=pretty_midi.instrument_name_to_program(
          instrument_name))

  prev_start = 0
  for i, note in notes.iterrows():
    start = float(prev_start + note['step'])
    end = float(start + note['duration'])
    note = pretty_midi.Note(
        velocity=velocity,
        pitch=int(note['pitch']),
        start=start,
        end=end,
    )
    instrument.notes.append(note)
    prev_start = start

  pm.instruments.append(instrument)
  pm.write(out_file)
  return pm


example_file = 'example.midi'
example_pm = notes_to_midi(
    raw_notes, out_file=example_file, instrument_name=instrument_name)


display_audio(example_pm)
```

```
fluidsynth: warning: SDL2 not initialized, SDL2 audio driver won't be usable
fluidsynth: error: Unknown integer parameter 'synth.sample-rate'
```

0:00 / 0:30

```
num_files = 5
all_notes = []
for f in filenames[:num_files]:
  notes = midi_to_notes(f)
  all_notes.append(notes)

all_notes = pd.concat(all_notes)


n_notes = len(all_notes)
print('Number of notes parsed:', n_notes)
```

```
Number of notes parsed: 15315
```

```
key_order = ['pitch', 'step', 'duration']
train_notes = np.stack([all_notes[key] for key in key_order], axis=1)


notes_ds = tf.data.Dataset.from_tensor_slices(train_notes)
notes_ds.element_spec
```

```
TensorSpec(shape=(3,), dtype=tf.float64, name=None)
```

```
def create_sequences(
    dataset: tf.data.Dataset,
    seq_length: int,
    vocab_size = 128,
) -> tf.data.Dataset:
  """Returns TF Dataset of sequence and label examples."""
  seq_length = seq_length+1

  # Take 1 extra for the labels
  windows = dataset.window(seq_length, shift=1, stride=1,
                           drop_remainder=True)

  # `flat_map` flattens the" dataset of datasets" into a dataset of tensors
  flatten = lambda x: x.batch(seq_length, drop_remainder=True)
  sequences = windows.flat_map(flatten)
```

```
    # Normalize note pitch
    def scale_pitch(x):
      x = x/[vocab_size,1.0,1.0]
      return x

    # Split the labels
    def split_labels(sequences):
      inputs = sequences[:-1]
      labels_dense = sequences[-1]
      labels = {key:labels_dense[i] for i,key in enumerate(key_order)}

      return scale_pitch(inputs), labels

    return sequences.map(split_labels, num_parallel_calls=tf.data.AUTOTUNE)


seq_length = 25
vocab_size = 128
seq_ds = create_sequences(notes_ds, seq_length, vocab_size)
seq_ds.element_spec
```

```
(TensorSpec(shape=(25, 3), dtype=tf.float64, name=None),
 {'pitch': TensorSpec(shape=(), dtype=tf.float64, name=None),
  'step': TensorSpec(shape=(), dtype=tf.float64, name=None),
  'duration': TensorSpec(shape=(), dtype=tf.float64, name=None)})
```

```
for seq, target in seq_ds.take(1):
  print('sequence shape:', seq.shape)
  print('sequence elements (first 10):', seq[0: 10])
  print()
  print('target:', target)
```

```
sequence shape: (25, 3)
sequence elements (first 10): tf.Tensor(
[[0.625      0.         0.23828125]
 [0.6015625  0.04036458 0.2421875 ]
 [0.5859375  0.22395833 0.06510417]
 [0.5625     0.09505208 0.0703125 ]
 [0.53125    0.11067708 0.1640625 ]
 [0.5859375  0.05598958 0.12760417]
 [0.5625     0.09244792 0.08333333]
 [0.625      0.08333333 0.17317708]
 [0.6015625  0.01822917 0.15104167]
 [0.5859375  0.109375   0.04036458]], shape=(10, 3), dtype=float64)

target: {'pitch': <tf.Tensor: shape=(), dtype=float64, numpy=68.0>, 'step': <tf.Tensor: shape=(), dtype=float64, numpy=0.1158854166
```

```
batch_size = 64
buffer_size = n_notes - seq_length  # the number of items in the dataset
train_ds = (seq_ds
            .shuffle(buffer_size)
            .batch(batch_size, drop_remainder=True)
            .cache()
            .prefetch(tf.data.experimental.AUTOTUNE))


train_ds.element_spec
```

```
(TensorSpec(shape=(64, 25, 3), dtype=tf.float64, name=None),
 {'pitch': TensorSpec(shape=(64,), dtype=tf.float64, name=None),
  'step': TensorSpec(shape=(64,), dtype=tf.float64, name=None),
  'duration': TensorSpec(shape=(64,), dtype=tf.float64, name=None)})
```

```
def mse_with_positive_pressure(y_true: tf.Tensor, y_pred: tf.Tensor):
  mse = (y_true - y_pred) ** 2
  positive_pressure = 10 * tf.maximum(-y_pred, 0.0)
  return tf.reduce_mean(mse + positive_pressure)
```

```
input_shape = (seq_length, 3)
learning_rate = 0.005

inputs = tf.keras.Input(input_shape)
x = tf.keras.layers.LSTM(128)(inputs)

outputs = {
  'pitch': tf.keras.layers.Dense(128, name='pitch')(x),
  'step': tf.keras.layers.Dense(1, name='step')(x),
  'duration': tf.keras.layers.Dense(1, name='duration')(x),
}

model = tf.keras.Model(inputs, outputs)

loss = {
      'pitch': tf.keras.losses.SparseCategoricalCrossentropy(
          from_logits=True),
      'step': mse_with_positive_pressure,
      'duration': mse_with_positive_pressure,
}

optimizer = tf.keras.optimizers.Adam(learning_rate=learning_rate)

model.compile(loss=loss, optimizer=optimizer)

model.summary()
```

```
Model: "model"
__________________________________________________________________________________________________
 Layer (type)                Output Shape                 Param #   Connected to
==================================================================================================
 input_1 (InputLayer)        [(None, 25, 3)]              0         []

 lstm (LSTM)                 (None, 128)                  67584     ['input_1[0][0]']

 duration (Dense)            (None, 1)                    129       ['lstm[0][0]']

 pitch (Dense)               (None, 128)                  16512     ['lstm[0][0]']

 step (Dense)                (None, 1)                    129       ['lstm[0][0]']

==================================================================================================
Total params: 84354 (329.51 KB)
Trainable params: 84354 (329.51 KB)
Non-trainable params: 0 (0.00 Byte)
__________________________________________________________________________________________________
```

```
losses = model.evaluate(train_ds, return_dict=True)
losses
```

```
  1/Unknown - 3s 3s/step - loss: 6.2279 - duration_loss: 0.7264 - pitch_loss: 4.8595 - step_loss: 0.6420
 19/Unknown - 3s 3ms/step - loss: 6.3166 - duration_loss: 0.7402 - pitch_loss: 4.8553 - step_loss: 0.7211
 38/Unknown - 3s 3ms/step - loss: 6.2207 - duration_loss: 0.6496 - pitch_loss: 4.8551 - step_loss: 0.7160
 57/Unknown - 3s 3ms/step - loss: 6.1262 - duration_loss: 0.5841 - pitch_loss: 4.8542 - step_loss: 0.6879
 76/Unknown - 3s 3ms/step - loss: 6.2462 - duration_loss: 0.7150 - pitch_loss: 4.8544 - step_loss: 0.6768
 95/Unknown - 3s 3ms/step - loss: 6.2315 - duration_loss: 0.6930 - pitch_loss: 4.8542 - step_loss: 0.6844
115/Unknown - 3s 3ms/step - loss: 6.2007 - duration_loss: 0.6628 - pitch_loss: 4.8540 - step_loss: 0.6839
134/Unknown - 3s 3ms/step - loss: 6.1725 - duration_loss: 0.6322 - pitch_loss: 4.8541 - step_loss: 0.6862
154/Unknown - 3s 3ms/step - loss: 6.1698 - duration_loss: 0.6366 - pitch_loss: 4.8542 - step_loss: 0.6790
173/Unknown - 3s 3ms/step - loss: 6.1571 - duration_loss: 0.6253 - pitch_loss: 4.8544 - step_loss: 0.6774
193/Unknown - 3s 3ms/step - loss: 6.1567 - duration_loss: 0.6281 - pitch_loss: 4.8543 - step_loss: 0.6742
213/Unknown - 4s 3ms/step - loss: 6.1349 - duration_loss: 0.6098 - pitch_loss: 4.8543 - step_loss: 0.6708
232/Unknown - 4s 3ms/step - loss: 6.1322 - duration_loss: 0.6080 - pitch_loss: 4.8544 - step_loss: 0.6698
238/238 [==============================] - 4s 3ms/step - loss: 6.1272 - duration_loss: 0.6039 - pitch_loss: 4.8544 - step_loss: 0.6
{'loss': 6.127169609069824,
 'duration_loss': 0.603919267654419,
 'pitch_loss': 4.854353427886963,
 'step_loss': 0.6688962578773499}
```

```
model.compile(
    loss=loss,
    loss_weights={
        'pitch': 0.05,
        'step': 1.0,
        'duration':1.0,
    },
    optimizer=optimizer,
)
```

```
Start coding or generate with AI.
```

```
model.evaluate(train_ds, return_dict=True)
```

```
 1/Unknown - 1s 852ms/step - loss: 1.6113 - duration_loss: 0.7264 - pitch_loss: 4.8595 - step_loss: 0.6420
19/Unknown - 1s 3ms/step - loss: 1.7041 - duration_loss: 0.7402 - pitch_loss: 4.8553 - step_loss: 0.7211
38/Unknown - 1s 3ms/step - loss: 1.6083 - duration_loss: 0.6496 - pitch_loss: 4.8551 - step_loss: 0.7160
56/Unknown - 1s 3ms/step - loss: 1.5202 - duration_loss: 0.5883 - pitch_loss: 4.8543 - step_loss: 0.6892
74/Unknown - 1s 3ms/step - loss: 1.5717 - duration_loss: 0.6508 - pitch_loss: 4.8543 - step_loss: 0.6782
92/Unknown - 1s 3ms/step - loss: 1.6320 - duration_loss: 0.7031 - pitch_loss: 4.8541 - step_loss: 0.6862
110/Unknown - 1s 3ms/step - loss: 1.5971 - duration_loss: 0.6757 - pitch_loss: 4.8542 - step_loss: 0.6787
129/Unknown - 1s 3ms/step - loss: 1.5699 - duration_loss: 0.6385 - pitch_loss: 4.8541 - step_loss: 0.6886
147/Unknown - 1s 3ms/step - loss: 1.5422 - duration_loss: 0.6181 - pitch_loss: 4.8541 - step_loss: 0.6814
166/Unknown - 1s 3ms/step - loss: 1.5491 - duration_loss: 0.6275 - pitch_loss: 4.8543 - step_loss: 0.6788
184/Unknown - 1s 3ms/step - loss: 1.5347 - duration_loss: 0.6170 - pitch_loss: 4.8543 - step_loss: 0.6750
202/Unknown - 1s 3ms/step - loss: 1.5347 - duration_loss: 0.6193 - pitch_loss: 4.8543 - step_loss: 0.6726
220/Unknown - 1s 3ms/step - loss: 1.5230 - duration_loss: 0.6099 - pitch_loss: 4.8543 - step_loss: 0.6704
238/238 [==============================] - 2s 3ms/step - loss: 1.5155 - duration_loss: 0.6039 - pitch_loss: 4.8544 - step_loss: 0.6
{'loss': 1.515533447265625,
 'duration_loss': 0.603919267654419,
 'pitch_loss': 4.854353427886963,
 'step_loss': 0.6688962578773499}
```

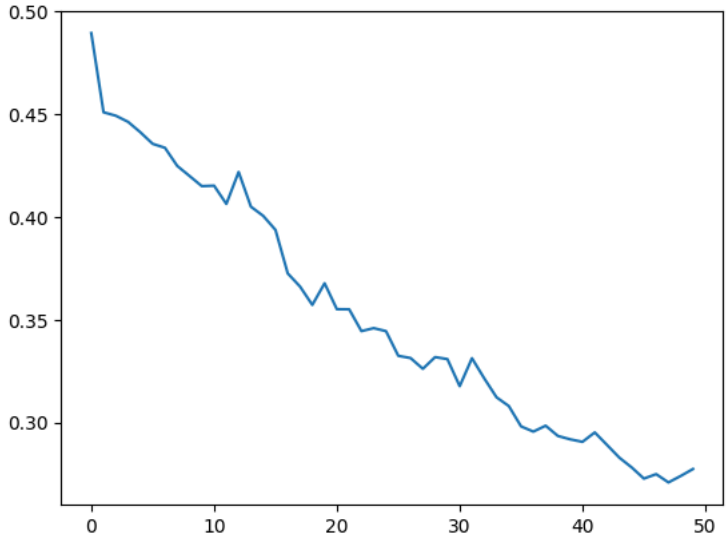
```
callbacks = [
    tf.keras.callbacks.ModelCheckpoint(
        filepath='./training_checkpoints/ckpt_{epoch}',
        save_weights_only=True),
    tf.keras.callbacks.EarlyStopping(
        monitor='loss',
        patience=5,
        verbose=1,
        restore_best_weights=True),
]
```

```
%%time
epochs = 50

history = model.fit(
    train_ds,
    epochs=epochs,
    callbacks=callbacks,
)
```

```
Epoch 1/50
WARNING: All log messages before absl::InitializeLog() is called are written to STDERR
I0000 00:00:1698385783.436050  470386 device_compiler.h:186] Compiled cluster using XLA!  This line is logged at most once for t

 1/Unknown - 3s 3s/step - loss: 1.6113 - duration_loss: 0.7264 - pitch_loss: 4.8595 - step_loss: 0.6420
12/Unknown - 3s 5ms/step - loss: 0.9619 - duration_loss: 0.4725 - pitch_loss: 4.7879 - step_loss: 0.2499
24/Unknown - 3s 5ms/step - loss: 0.7161 - duration_loss: 0.3150 - pitch_loss: 4.7369 - step_loss: 0.1643
36/Unknown - 3s 5ms/step - loss: 0.6782 - duration_loss: 0.3090 - pitch_loss: 4.6849 - step_loss: 0.1350
48/Unknown - 3s 5ms/step - loss: 0.6141 - duration_loss: 0.2756 - pitch_loss: 4.6355 - step_loss: 0.1068
60/Unknown - 3s 4ms/step - loss: 0.5554 - duration_loss: 0.2374 - pitch_loss: 4.5867 - step_loss: 0.0887
72/Unknown - 3s 4ms/step - loss: 0.5866 - duration_loss: 0.2822 - pitch_loss: 4.5376 - step_loss: 0.0776
84/Unknown - 3s 4ms/step - loss: 0.5916 - duration_loss: 0.2957 - pitch_loss: 4.4954 - step_loss: 0.0711
96/Unknown - 3s 4ms/step - loss: 0.5848 - duration_loss: 0.2857 - pitch_loss: 4.4632 - step_loss: 0.0759
108/Unknown - 3s 4ms/step - loss: 0.5727 - duration_loss: 0.2808 - pitch_loss: 4.4347 - step_loss: 0.0701
120/Unknown - 3s 4ms/step - loss: 0.5644 - duration_loss: 0.2634 - pitch_loss: 4.4027 - step_loss: 0.0809
133/Unknown - 3s 4ms/step - loss: 0.5398 - duration_loss: 0.2462 - pitch_loss: 4.3736 - step_loss: 0.0749
145/Unknown - 3s 4ms/step - loss: 0.5226 - duration_loss: 0.2343 - pitch_loss: 4.3511 - step_loss: 0.0707
157/Unknown - 4s 4ms/step - loss: 0.5274 - duration_loss: 0.2443 - pitch_loss: 4.3326 - step_loss: 0.0665
170/Unknown - 4s 4ms/step - loss: 0.5240 - duration_loss: 0.2440 - pitch_loss: 4.3190 - step_loss: 0.0640
183/Unknown - 4s 4ms/step - loss: 0.5090 - duration_loss: 0.2326 - pitch_loss: 4.3080 - step_loss: 0.0609
196/Unknown - 4s 4ms/step - loss: 0.5153 - duration_loss: 0.2419 - pitch_loss: 4.2967 - step_loss: 0.0586
208/Unknown - 4s 4ms/step - loss: 0.5059 - duration_loss: 0.2355 - pitch_loss: 4.2841 - step_loss: 0.0563
221/Unknown - 4s 4ms/step - loss: 0.4985 - duration_loss: 0.2304 - pitch_loss: 4.2721 - step_loss: 0.0544
234/Unknown - 4s 4ms/step - loss: 0.4922 - duration_loss: 0.2256 - pitch_loss: 4.2603 - step_loss: 0.0537
238/238 [==============================] - 4s 5ms/step - loss: 0.4896 - duration_loss: 0.2237 - pitch_loss: 4.2569 - step_loss:
Epoch 2/50

  1/238 [..............................] - ETA: 1s - loss: 0.5822 - duration_loss: 0.3656 - pitch_loss: 4.0392 - step_loss: 0.01
 13/238 [>.............................] - ETA: 0s - loss: 0.6068 - duration_loss: 0.2678 - pitch_loss: 4.0370 - step_loss: 0.13
 26/238 [==>...........................] - ETA: 0s - loss: 0.4860 - duration_loss: 0.1841 - pitch_loss: 4.0783 - step_loss: 0.09
 38/238 [===>..........................] - ETA: 0s - loss: 0.5261 - duration_loss: 0.2306 - pitch_loss: 4.0937 - step_loss: 0.09
 50/238 [=====>........................] - ETA: 0s - loss: 0.4850 - duration_loss: 0.2077 - pitch_loss: 4.0992 - step_loss: 0.07
 62/238 [======>.......................] - ETA: 0s - loss: 0.4475 - duration_loss: 0.1814 - pitch_loss: 4.0968 - step_loss: 0.06
 75/238 [========>.....................] - ETA: 0s - loss: 0.4869 - duration_loss: 0.2273 - pitch_loss: 4.0966 - step_loss: 0.05
 87/238 [=========>....................] - ETA: 0s - loss: 0.5005 - duration_loss: 0.2329 - pitch_loss: 4.0949 - step_loss: 0.06
 99/238 [===========>..................] - ETA: 0s - loss: 0.4925 - duration_loss: 0.2299 - pitch_loss: 4.0931 - step_loss: 0.05
111/238 [============>.................] - ETA: 0s - loss: 0.4889 - duration_loss: 0.2296 - pitch_loss: 4.0913 - step_loss: 0.05
123/238 [==============>...............] - ETA: 0s - loss: 0.4886 - duration_loss: 0.2170 - pitch_loss: 4.0838 - step_loss: 0.06
136/238 [================>.............] - ETA: 0s - loss: 0.4730 - duration_loss: 0.2062 - pitch_loss: 4.0814 - step_loss: 0.06
148/238 [=================>............] - ETA: 0s - loss: 0.4606 - duration_loss: 0.1973 - pitch_loss: 4.0805 - step_loss: 0.05
161/238 [===================>..........] - ETA: 0s - loss: 0.4726 - duration_loss: 0.2128 - pitch_loss: 4.0799 - step_loss: 0.05
174/238 [====================>.........] - ETA: 0s - loss: 0.4679 - duration_loss: 0.2097 - pitch_loss: 4.0810 - step_loss: 0.05
187/238 [======================>.......] - ETA: 0s - loss: 0.4570 - duration_loss: 0.2011 - pitch_loss: 4.0814 - step_loss: 0.05
```

```
200/238 [========================>.....] - ETA: 0s - loss: 0.4674 - duration_loss: 0.2132 - pitch_loss: 4.0811 - step_loss: 0.05
213/238 [=========================>....] - ETA: 0s - loss: 0.4587 - duration_loss: 0.2066 - pitch_loss: 4.0779 - step_loss: 0.04
225/238 [==========================>..] - ETA: 0s - loss: 0.4562 - duration_loss: 0.2056 - pitch_loss: 4.0773 - step_loss: 0.04
238/238 [==============================] - ETA: 0s - loss: 0.4510 - duration_loss: 0.2010 - pitch_loss: 4.0747 - step_loss: 0.04
238/238 [==============================] - 1s 4ms/step - loss: 0.4510 - duration_loss: 0.2010 - pitch_loss: 4.0747 - step_loss:
Epoch 3/50

 1/238 [..............................] - ETA: 1s - loss: 0.5760 - duration_loss: 0.3605 - pitch_loss: 4.0178 - step_loss: 0.01
13/238 [>.............................] - ETA: 0s - loss: 0.6024 - duration_loss: 0.2696 - pitch_loss: 4.0182 - step_loss: 0.13
26/238 [==>...........................] - ETA: 0s - loss: 0.4812 - duration_loss: 0.1840 - pitch_loss: 4.0576 - step_loss: 0.09
39/238 [===>..........................] - ETA: 0s - loss: 0.5153 - duration_loss: 0.2259 - pitch_loss: 4.0697 - step_loss: 0.08
51/238 [=====>........................] - ETA: 0s - loss: 0.4762 - duration_loss: 0.2038 - pitch_loss: 4.0695 - step_loss: 0.06
63/238 [======>.......................] - ETA: 0s - loss: 0.4428 - duration_loss: 0.1807 - pitch_loss: 4.0675 - step_loss: 0.05
```

```
plt.plot(history.epoch, history.history['loss'], label='total loss')
plt.show()
```




```
def predict_next_note(
    notes: np.ndarray,
    model: tf.keras.Model,
    temperature: float = 1.0) -> tuple[int, float, float]:
  """Generates a note as a tuple of (pitch, step, duration), using a trained sequence model."""

  assert temperature > 0

  # Add batch dimension
  inputs = tf.expand_dims(notes, 0)

  predictions = model.predict(inputs)
  pitch_logits = predictions['pitch']
  step = predictions['step']
  duration = predictions['duration']

  pitch_logits /= temperature
  pitch = tf.random.categorical(pitch_logits, num_samples=1)
  pitch = tf.squeeze(pitch, axis=-1)
  duration = tf.squeeze(duration, axis=-1)
  step = tf.squeeze(step, axis=-1)

  # `step` and `duration` values should be non-negative
  step = tf.maximum(0, step)
  duration = tf.maximum(0, duration)

  return int(pitch), float(step), float(duration)
```

```
temperature = 2.0
num_predictions = 120

sample_notes = np.stack([raw_notes[key] for key in key_order], axis=1)

# The initial sequence of notes; pitch is normalized similar to training
# sequences
input_notes = (
    sample_notes[:seq_length] / np.array([vocab_size, 1, 1]))

generated_notes = []
prev_start = 0
for _ in range(num_predictions):
  pitch, step, duration = predict_next_note(input_notes, model, temperature)
  start = prev_start + step
  end = start + duration
  input_note = (pitch, step, duration)
  generated_notes.append((*input_note, start, end))
  input_notes = np.delete(input_notes, 0, axis=0)
  input_notes = np.append(input_notes, np.expand_dims(input_note, 0), axis=0)
  prev_start = start

generated_notes = pd.DataFrame(
    generated_notes, columns=(*key_order, 'start', 'end'))
```

```
 1/1 [==============================] - ETA: 0s
 1/1 [==============================] - 0s 386ms/step

 1/1 [==============================] - ETA: 0s
 1/1 [==============================] - 0s 42ms/step

 1/1 [==============================] - ETA: 0s
 1/1 [==============================] - 0s 41ms/step

 1/1 [==============================] - ETA: 0s
 1/1 [==============================] - 0s 41ms/step

 1/1 [==============================] - ETA: 0s
 1/1 [==============================] - 0s 41ms/step

 1/1 [==============================] - ETA: 0s
 1/1 [==============================] - 0s 42ms/step

 1/1 [==============================] - ETA: 0s
 1/1 [==============================] - 0s 41ms/step

 1/1 [==============================] - ETA: 0s
 1/1 [==============================] - 0s 41ms/step

 1/1 [==============================] - ETA: 0s
 1/1 [==============================] - 0s 41ms/step

 1/1 [==============================] - ETA: 0s
 1/1 [==============================] - 0s 42ms/step

 1/1 [==============================] - ETA: 0s
 1/1 [==============================] - 0s 41ms/step

 1/1 [==============================] - ETA: 0s
 1/1 [==============================] - 0s 41ms/step

 1/1 [==============================] - ETA: 0s
 1/1 [==============================] - 0s 42ms/step

 1/1 [==============================] - ETA: 0s
 1/1 [==============================] - 0s 42ms/step

 1/1 [==============================] - ETA: 0s
 1/1 [==============================] - 0s 41ms/step

 1/1 [==============================] - ETA: 0s
 1/1 [==============================] - 0s 42ms/step

 1/1 [==============================] - ETA: 0s
 1/1 [==============================] - 0s 41ms/step

 1/1 [==============================] - ETA: 0s
 1/1 [==============================] - 0s 42ms/step

 1/1 [==============================] - ETA: 0s
 1/1 [==============================] - 0s 41ms/step
```

```
generated_notes.head(10)
```

	pitch	step	duration	start	end
0	47	0.097688	0.179285	0.097688	0.276973
1	72	0.858761	0.163204	0.956448	1.119652
2	84	1.089159	0.000000	2.045608	2.045608
3	84	1.095052	0.000000	3.140660	3.140660
4	89	1.108157	0.000000	4.248817	4.248817
5	89	1.118506	0.000000	5.367323	5.367323
6	84	1.135083	0.000000	6.502405	6.502405
7	90	1.128108	0.000000	7.630513	7.630513
8	84	1.129703	0.000000	8.760216	8.760216
9	84	1.155163	0.019010	9.915379	9.934389

```
out_file = 'output.mid'
out_pm = notes_to_midi(
    generated_notes, out_file=out_file, instrument_name=instrument_name)
display_audio(out_pm)
```

```
fluidsynth: warning: SDL2 not initialized, SDL2 audio driver won't be usable
fluidsynth: error: Unknown integer parameter 'synth.sample-rate'
```

0:00 / 0:30