01

Audio Generation using Neutral Network

Intelligence Artificielle, M2 Informatique

Objectif

A l'aide d'algorithmes d'apprentissage, réussir à générer de la musique en utilisant le language Python

Template Kaggle

Audio Generation Using Neural Net

Dataset

Multiples compositions de Beethoven

Architectures utilisées

LSTM : Obtention des premiers résultats assez rapidement, mais il s'agit d'un enchainement de notes qui fini par se répéter.

LSTM using embedding : On obtient de meilleurs résultats que LSTM. La musique générée est un peu plus homogène mais reste mauvaise.

GAN : En utilisant GAN, les résultats sont un peu plus satisfaisant que les précédents, mais sans être une véritable révolution.

LSTM

Code utilisé pour générer la musique via LSTM

```
model = Sequential()
model.add(LSTM(512,return_sequences=False, input_shape=(phrase_len, 3)))
model.add(Dropout(0.5))
model.add(Dense(3, activation='relu'))
model.compile(loss='mae', optimizer='adam')

model.fit(X, y, batch_size=256, epochs=70, validation_split=0.2)


def tune_generator(model, name='lstm_tune_'):
    for i in range(3):
        start = np.random.randint(0, len(X)-1)
        pattern = X[start]
        prediction_output = []

        for note_index in range(100):
            prediction_input = np.reshape(pattern, (1, len(pattern), 3))
            prediction = model.predict(prediction_input, verbose=0)
            prediction_output.append(prediction.astype(int)[0])
            pattern = np.append(pattern, prediction, axis = 0)
            pattern = pattern[1:len(pattern)]

        notes = pd.DataFrame(prediction_output, columns=['time', 'note', 'velocity'])
        notes['pause'] = 180
        notes_dict = notes.to_dict('records')
        tune_to_midi(notes_dict, midi_name=name + str(i))
```

LSTM using embedding

Code utilisé pour générer la musique via LSTM using embedding

```
n_notes = 128
embed_size = 100

notes_in = Input(shape = (phrase_len,))
durations_in = Input(shape = (phrase_len,1))

notes_embed = Embedding(n_notes, embed_size)(notes_in)

concat_model = Concatenate()([notes_embed,durations_in])
concat_model = Dropout(0.3)(concat_model)
concat_model = LSTM(512, return_sequences=False)(concat_model)

notes_out = Dense(n_notes, activation = 'softmax', name = 'note')(concat_model)
durations_out = Dense(1, activation = 'relu', name = 'duration')(concat_model)

embed_model = Model([notes_in, durations_in], [notes_out, durations_out])
embed_model.compile(loss=['sparse_categorical_crossentropy',
                    'mse'], optimizer=RMSprop(lr = 0.001))

train_chords = X[:, :, 1]
train_durations = X[:, :, 0]
target_chords = y[:, 1]
target_durations = y[:, 0]

embed_model.fit([train_chords, train_durations],
                    [target_chords, target_durations]
                    , epochs=200, batch_size=256, validation_split=0.2
                )
```

GAN

Code utilisé pour générer la musique via GAN

```
tune_len = 200
n_notes = 128

train_matrixes = []
for x in X[:1000]:
    train_matrixes.append(tune_to_matrix(x, tune_len=200))
```

C'est parti pour les écoutes !

LSTM

LSTM using Embedding

GAN

Ecoute des rendus

LSTM

LSTM using Embedding

GAN

Ecoute des rendus

LSTM

LSTM using Embedding

GAN

Ecoute des rendus

Problèmes rencontrés

1

Difficulté à trouver un **code fonctionnel et documenté** produisant des sons à peu près correct

2

Difficulté à **comprendre et manipuler** le code car certaines parties ne sont **pas documentées**

Pour conclure

- Très intéressant
- LSTM < LSTM using Embedding < GAN
- Obtention de meilleurs résultats avec plus de tests