# PENGECAMAN MUZIK MENGGUNAKAN RANGKAIAN NEURAL LSTM DAN CAP JARI SIFT

Ooi Heap Sheng
Ts. Dr. Afzan Adam

*Fakulti Teknologi dan Sains Maklumat, Universiti Kebangsaan Malaysia*

**ABSTRAK**

Setakat ini, algoritma yang wujud untuk pengecaman karya muzik digital masih mengalami masalah pengecaman yang tidak tepat apabila bunyi lain ditambah ke muzik pertanyaan. Kajian-kajian lepas menggunakan kaedah mengira purata kadar pintasan sifar *average zero crossing rate* yang menganalisis gelombang asas muzik sahaja dan memperoleh keputusan yang tidak memuaskan. Kaedah pengecaman muzik yang berasaskan persepsi pendengaran manusia, dengan mengira komposisi frekuensi, dapat memberi keputusan yang lebih baik apabila bunyi ditambah ke muzik pertanyaan. Oleh itu, kajian ini akan mengaplikasikan kaedah algoritma kecerdasan buatan untuk mengecam karya-karya muzik yang telah diubahsuai untuk membantu pengguna mencari karya muzik yang hampir sama atau memelihara hak cipta karya muzik artis asal dari ditiru. Kajian ini akan menggunakan *Short Time Fourier Transform* untuk mendapat komposisi frekuensi mengikut masa (*Spectrogram)*, Rangkaian Neural *LSTM (Long Short Term Memory)* untuk mendapatkan melodi muzik, menggunakan *Scale Invarient Feature Transform* untuk mendapatkan Cap Jari muzik, dan menggunakan *Locality Sensitive Hashing* untuk kegunaan pengecaman muzik dan penyimpanan Cap Jari muzik dalam pangkalan data muzik. Setakat ini, algoritma *Short Time Fourier Transform (STFT)* berjaya menghasilkan *Spectrogram* yang jelas, dan memenuhi Objektif 1. Rangkaian Neural *LSTM* berjaya menghasilkan melodi muzik untuk muzik yang dimain dengan satu suara, dan memenuhi Objektif 2. Algoritma *SIFT* pula tidak dapat menyari Cap Jari Melodi Muzik yang berperwakilan dengan konsisten, dan tidak memenuhi Objektif 3. Oleh sebab masa kajian yang terhad, kajian ini terpaksa dihabiskan setakat ini sahaja. Kajian ini akan menjelaskan dengan lebih teliti punca kegagalan algoritma *SIFT.*



## 1 PENGENALAN

Para artis pada era digital menghasilkan karya-karya muzik mereka dan mengedarkannya menggunakan platform seperti *YouTube, SoundCloud* dan sebagainya. Artis mendapat keuntungan daripada platform tersebut daripada jumlah masa yang ditonton dan bilangan penonton. Walau bagaimanapun, terdapat juga pengguna yang meniru karya muzik artis lain dan memuatnaik ke platform yang sama untuk mencuri keuntungan. Muzik dalam bentuk

elektronik senang untuk ditiru dan dihasilkan salinan baru untuk diedarkan dalam platform yang ada. Oleh sebab itu, banyak platform muzik yang membenarkan para artis untuk mengedarkan karya muzik mereka telahpun menggunakan algoritma untuk mengecam karya muzik yang sama.

Terdapat beberapa jenis Hak Cipta Terpelihara untuk karya muzik. Antara 2 jenis yang paling penting ialah Hak Cipta komposisi/melodi muzik *(Performing Arts)*, dan Hak Cipta rakaman muzik *(Sound Recordings)* seperti yang dijelaskan oleh (Majewski 2019).

Platform seperti *YouTube* (2019) mempunyai algoritma untuk mengecam karya-karya muzik yang sama dan algoritma ini telah pun banyak digunakan untuk mengecam muzik yang digunakan atau ditiru oleh artis lain tanpa kebenaran. Walau bagaimanapun, algoritma *YouTube* hanya dapat mengecam muzik yang dimain dengan alat muzik yang sama, dan sebab itu hanya dapat memelihara Hak Cipta rakaman muzik. Hak Cipta komposisi/melodi muzik tidak dapat diperlihara secara automatik dan perlu dipantau secara manual. Hal ini telah membenarkan karya muzik sesetengah artis ditolak atas tuntutan palsu walaupun mereka tidak meniru atau menggunakan karya muzik daripada artis lain tanpa kebenaran (Jackie 2018).

Kajian ini akan mendalami kaedah-kaedah algoritma yang boleh digunakan untuk mengecam komposisi/melodi muzik yang telah ditiru dan berpotensi sebagai algoritma untuk memelihara hak cipta karya muzik artis sebenar.

## 2 PENYATAAN MASALAH

Masalah peniruan karya muzik dan penggunaan karya muzik tanpa kebenaran artis sebenar telah semakin meluas di dunia atas talian ini. Kaedah-kaedah yang wujud setakat ini untuk mengecam komposisi/melodi muzik yang sama masih mempunyai kelemahan pengecaman untuk muzik yang dimain dengan alat muzik yang berbeza. Kaedah pengecaman yang wujud dan boleh dikaji secara terbuka buat setakat ini, seperti Shazam (Wang 2003), menggunakan kaedah *Spectrogram* dan cap jari *Spectrogram* dengan titik nilai tertinggi atau menggunakan *Scale Invarient Feature Transform* (X. Zhang et al. 2015). Kaedah ini tidak dapat mengecam komposisi/melodi muzik yang sama.

## 3 OBJEKTIF KAJIAN

1. Menggunakan algoritma yang mampu menghasilkan *Spectrogram* muzik yang jelas.
2. Menggunakan algoritma yang mampu menyari melodi muzik yang berperwakilan daripada *Spectrogram*.
3. Menguji algoritma *SIFT* untuk menyari cap jari melodi muzik.

# 4 METOD KAJIAN

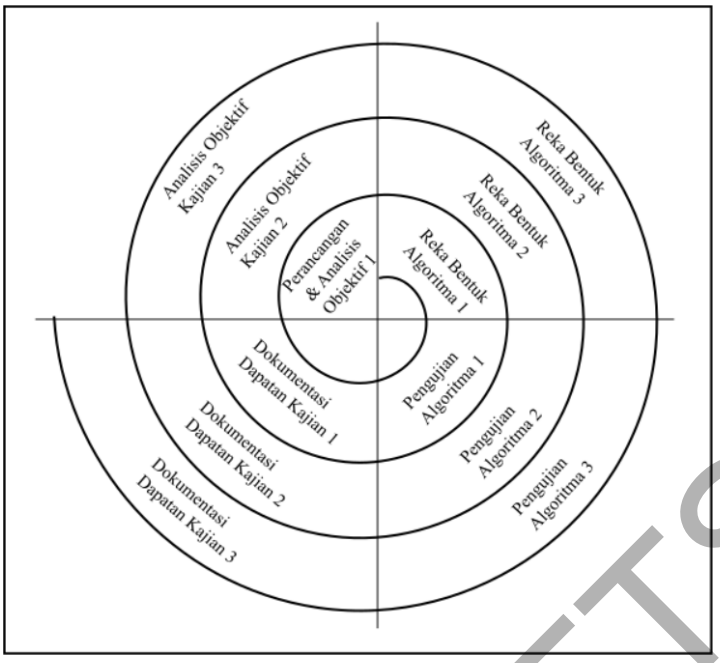
## 4.1 Fasa Perancangan

Kajian ini hanya merangkumi algoritma yang mampu menganalisis karya muzik, dan mengecam karya muzik yang sama. Hasil kajian ini adalah algoritma mentah yang mampu menjalankan objektif tersebut sahaja.

Kajian ini akan menggunakan 2000 karya muzik yang boleh didapati dari platform *YouTube*, dan dataset muzik *NSynth* (Jesse Engel, Cinjon Resnick, Adam Roberts, Sander Dieleman, Douglas Eck Karen Simonyan 2017) untuk melatih algoritma kecerdasan buatan.

Algoritma kecerdasan buatan ini akan dihasilkan menggunakan proses model *Spiral*. Hal ini demikian kerana algoritma kecerdasan buatan ini memerlukan pengubahsuaian yang banyak dalam proses pembangunannya. Oleh itu, pembangunan algoritma ini memerlukan proses model yang lebih senang untuk diubahsuai.

Secara kasarnya, proses model kajian ini akan merangkumi tugasan tersebut dalam bentuk *Spiral*, tugasan tersebut akan diulangi dengan cepat sehingga objektif dicapai.




Rajah 1.1: Proses Model *Spiral*

## 4.2 Fasa Analisis

Kajian-kajian lepas menggunakan kaedah mengira purata kadar pintasan sifar *average zero crossing rate* yang menganalisis statistik gelombang asas muzik sahaja dan memperoleh keputusan yang tidak memuaskan apabila bunyi ditambah ke muzik pertanyaan. Kaedah pengecaman muzik yang berasaskan persepsi pendengaran manusia, dengan mengira komposisi frekuensi, dapat memberi keputusan yang lebih baik apabila bunyi ditambah ke muzik pertanyaan. Keberkesanan teknik-teknik pengecaman muzik tersebut boleh dirujuk dalam Jadual 4.2.1.

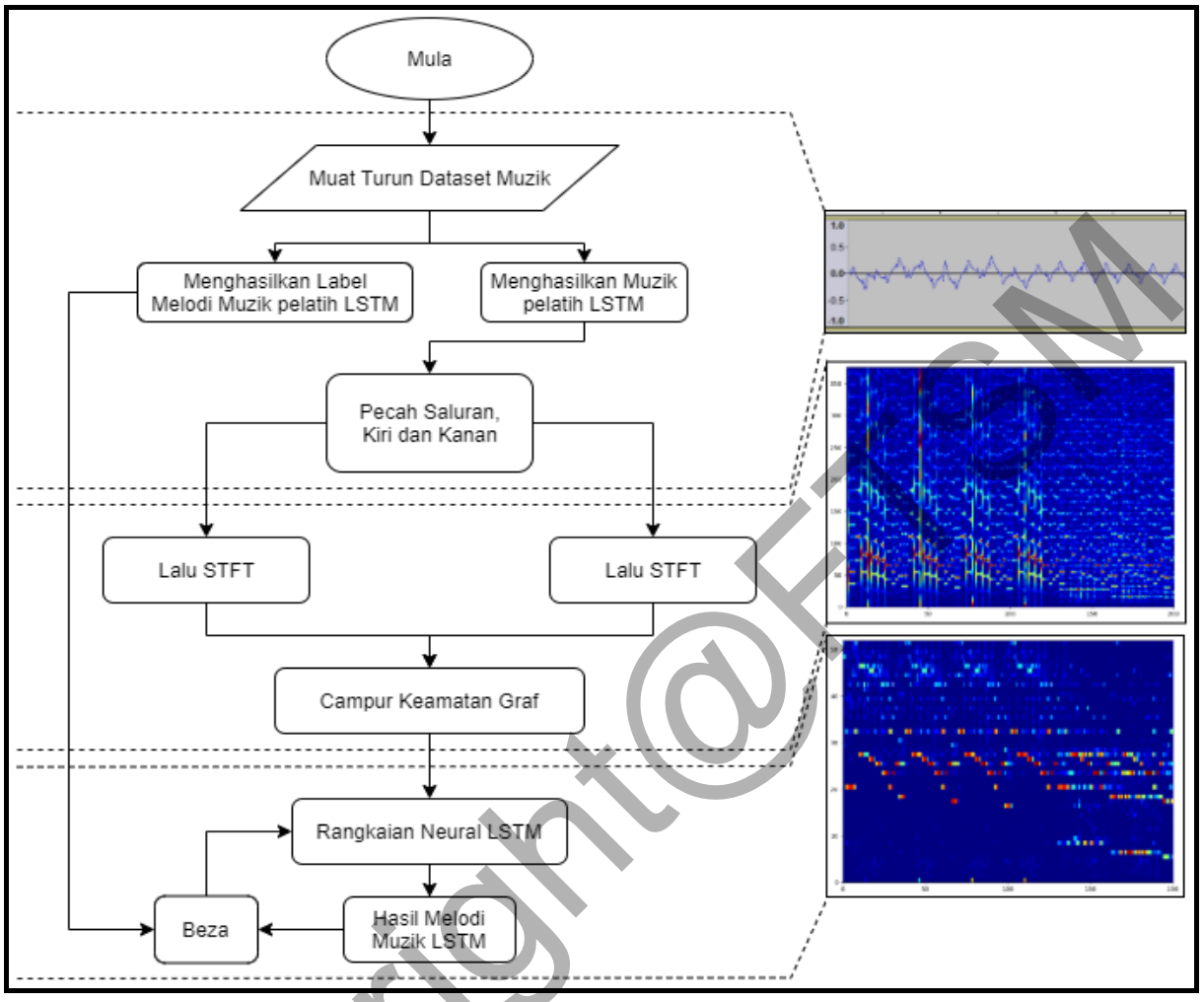
Jadual 4.2.1 Kesimpulan Keberkesanan Teknik Pengecaman

| | Perbandingan Gelombang Asas | Statistik Gelombang Asas | *Spectrogram* + Cap Jari (Shazam) |
|---|---|---|---|
| Kekuatan Muzik Beza | 0% | 80% | 99% |
| Tambah Bunyi Lain | 0% | 30% | 90% |
| Kadar Muzik Beza | 0% | 0% | 40% |
| Regangan Masa Beza | 0% | 0% | 40% |
| Kelangsingan Muzik Beza | 0% | 0% | 35% |
| Alat Muzik Beza | 0% | 0% | 0% |

Algoritma seperti *Short Time Fourier Transform (STFT)* dapat menghasilkan komposisi frekuensi mengikut masa (*Spectrogram)* yang agak memuaskan. Cap Jari muzik daripada

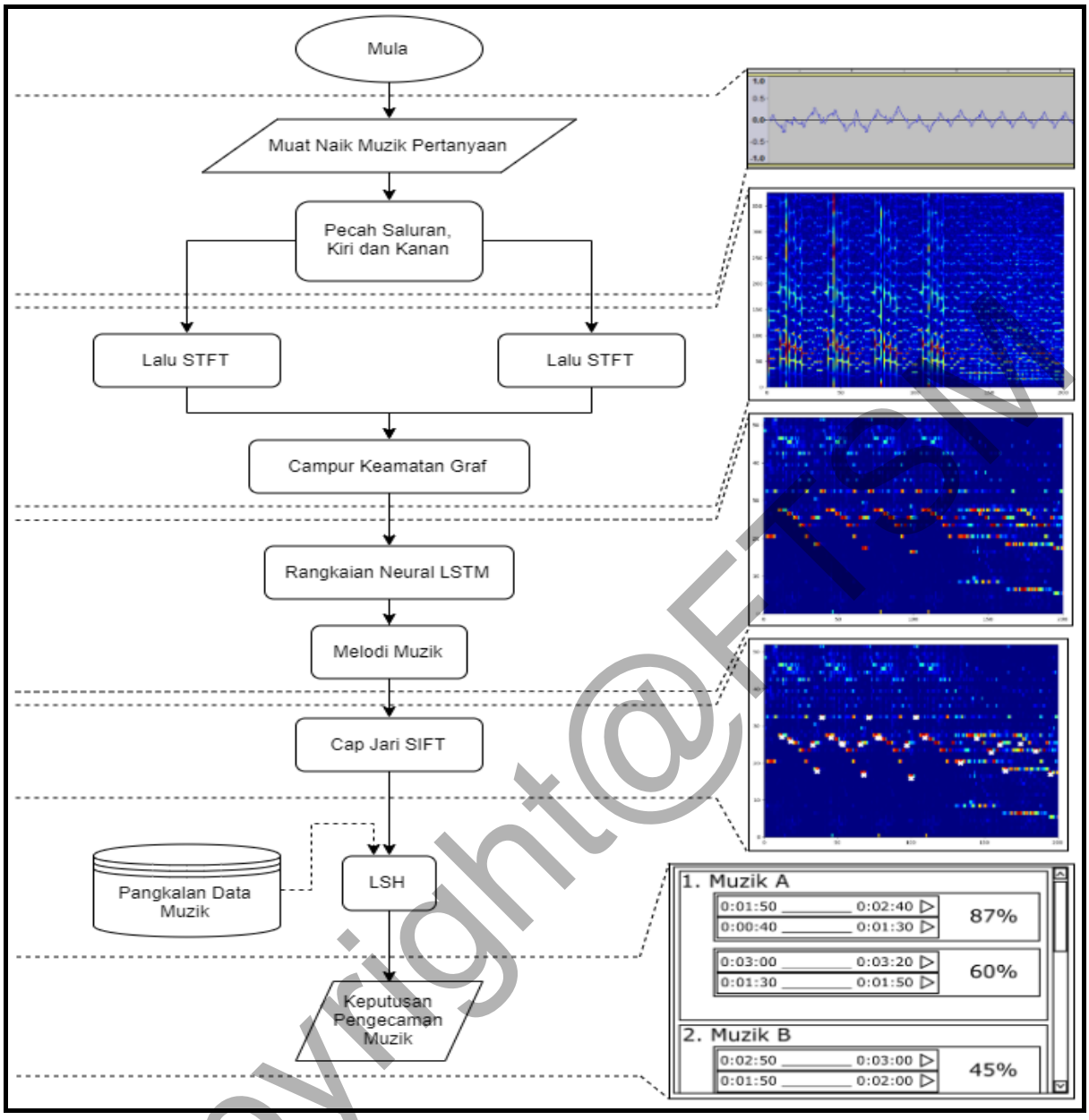
*Spectrogram* boleh didapati dengan kaedah *Scale Invarient Feature Transform (SIFT)*, dan *Locality Sensitive Hashing (LSH)* boleh digunakan untuk pengecaman muzik dan penyimpanan cap jari muzik dalam pangkalan data muzik. Aplikasi sedia ada seperti Shazam boleh digunakan sebagai perbandingan ketepatan pengecaman muzik untuk algoritma yang akan dibangunkan dalam kajian ini.

## 4.3 Fasa Reka Bentuk




Rajah 4.3.1: Reka bentuk algoritma melatih sistem pengecaman muzik (pemaju sistem)

Rajah 4.3.1 menunjukkan reka bentuk algoritma bagi pemaju sistem melatih rangkaian neural *LSTM* sistem pengecaman muzik. Rangkaian neural *LSTM* akan dilatih menggunakan kaedah *Supervised Learning*. Pemaju sistem perlu memuat turun dan dapatkan Dataset Muzik, Dataset Muzik tersebut akan dicampur dengan Dataset Muzik yang lain untuk menghasilkan muzik pelatih *LSTM*, muzik tersebut akan melalui *STFT* untuk dapatkan *Spectrogram*, *Spectrogram* akan diberi kepada rangkaian neural *LSTM* untuk mendapatkan melodi muzik, dan perbezaan antara Hasil Melodi Muzik *LSTM* dengan Label Melodi Muzik Pelatih *LSTM* akan digunakan untuk melatih rangkaian neural *LSTM*.

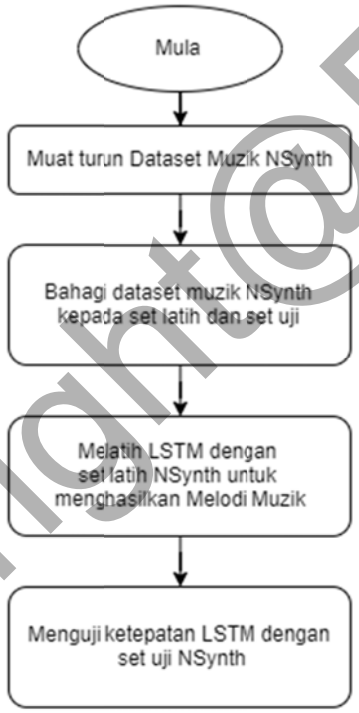



Rajah 4.3.2: Reka bentuk algoritma mengecam muzik (pengguna)

Rajah 4.3.2 menunjukkan reka bentuk algoritma bagi pengguna mengecam muzik menggunakan sistem pengecaman muzik. Pengguna perlu memuat naik muzik pertanyaan ke dalam sistem pengecaman muzik. Muzik pertanyaan tersebut akan melalui *STFT*, rangkaian neural *LSTM* untuk mendapatkan melodi muzik, melodi muzik akan melalui *SIFT* untuk dapatkan cap jari muzik, cap jari tersebut akan dibandingkan dengan Pangkalan Data Muzik menggunakan *LSH* untuk mendapatkan Keputusan Pengecaman Muzik, dan seterusnya dipaparkan kepada pengguna.

## 4.4 Fasa Pengujian

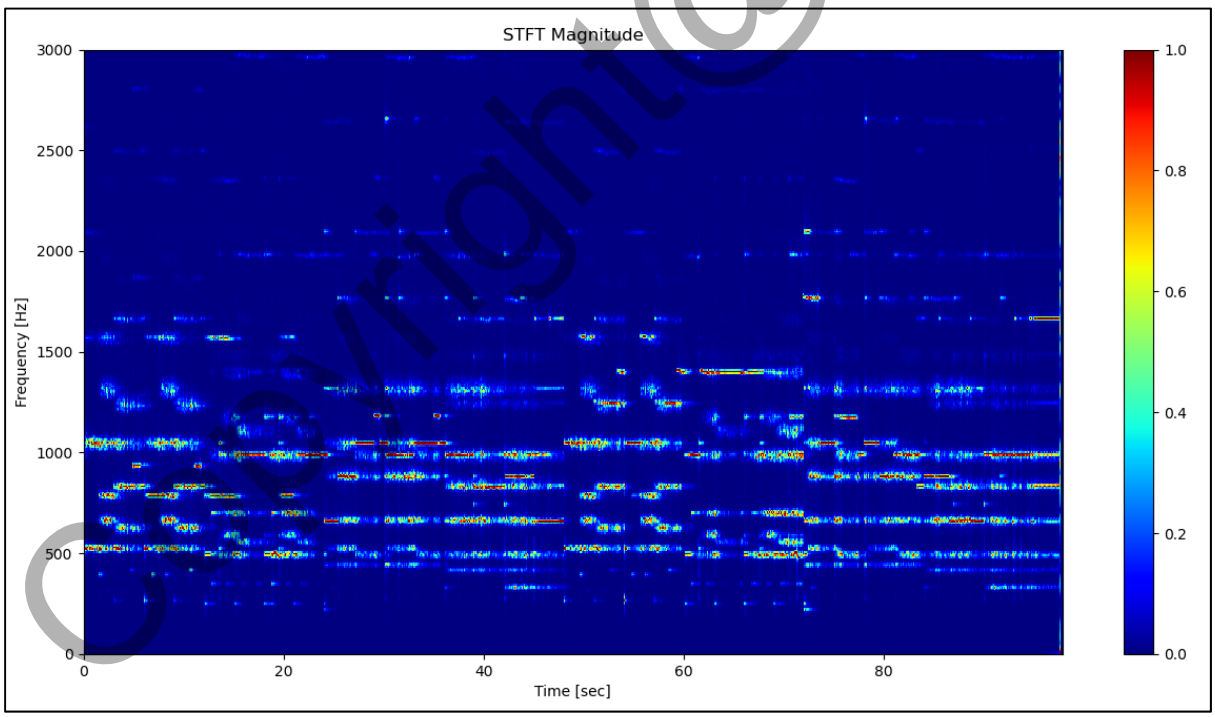
Kaedah pengujian yang akan dijalankan untuk mengenalpasti keberkesanan algoritma sistem ini dalam kegiatan pengecaman muzik ialah pengujian *LSTM* menggunakan set uji *NSynth*. Pengujian ini boleh dilakukan dengan menggunakan set uji dataset *NSynth.* Dataset muzik *NSynth* telahpun mengandungi dua bahagian iaitu bahagian untuk melatih dan bahagian untuk menguji. Muzik yang wujud di dalam bahagian melatih tidak wujud di bahagian menguji. Oleh itu, dataset muzik *NSynth* boleh digunakan untuk menguji ketepatan pengecaman melodi *LSTM* setelah habis dilatih menggunakan set latih. Hasil pengujian pertama ini akan dibincangkan di Bab 5 nanti. Rajah 4.4.1 merupakan Rajah aliran untuk proses pengujian ini.




Rajah 4.4.1: Rajah aliran proses pengujian menggunakan set uji *NSynth*

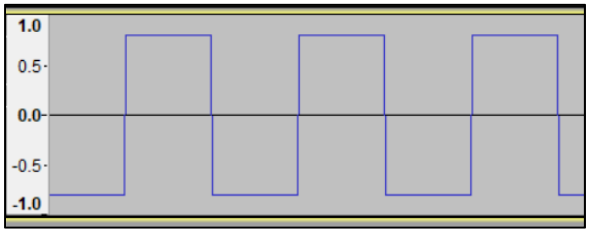
# 5 HASIL KAJIAN

Objektif 1 kajian ini adalah menggunakan algoritma yang mampu menghasilkan *Spectrogram* muzik yang jelas daripada audio muzik mentah. *STFT* merupakan salah satu algoritma yang boleh digunakan untuk menhasilkan *Spectrogram* muzik. Untuk menghasilkan *Spectrogram,* bahasa pengaturcaraan *Python* telah digunakan untuk tujuan ini. Kod *Python* tersebut menghasilkan *Spectrogram* hanya dari frekuensi 1Hz sampai 3000Hz untuk menggurangkan saiz *Spectrogram STFT* tersebut. Frekuensi dari 1Hz sampai 3000Hz telah merangkumi kebanyakan maklumat daripada muzik tersebut dan kebanyakan muzik tidak menggunakan frekuensi yang terlalu tinggi. Oleh itu, frekuensi yang lebih tinggi daripada 3000Hz boleh diabaikan. Selain itu, nilai *STFT* ini telah dihadkan di dalam julat 0 sehingga 1. Hal ini demikian untuk mempermudahkan Rangkaian Neural *LSTM* untuk memproses data tersebut dalam langkah yang seterusnya. Kod *Python* tersebut mampu menghasilkan *Spectrogram* yang agak jelas seperti yang ditunjukkan dalam Rajah 5.1.

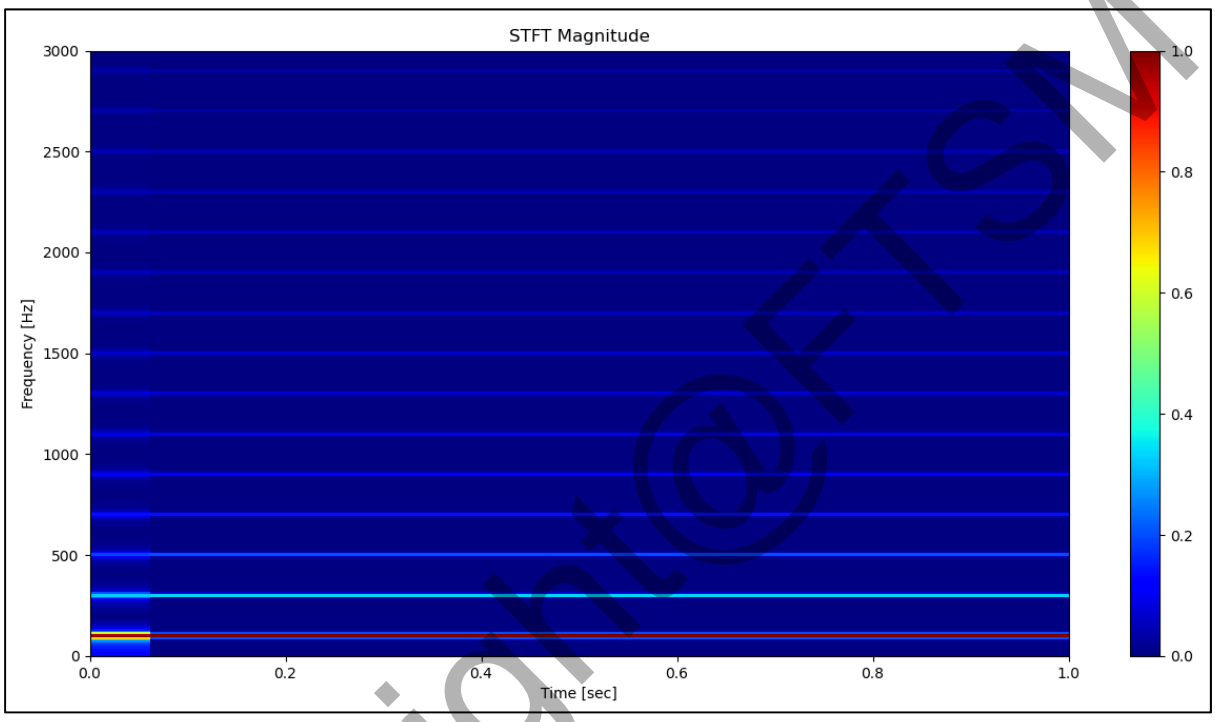



Rajah 5.1: *Spectrogram STFT* yang dihasilkan menggunakan Kod *Python*

*Spectrogram* yang dihasilkan menggunakan algoritma *STFT* adalah agak jelas dalam kebanyakan situasi. Algoritma *STFT* juga mampu untuk menganalisis gelombang buatan seperti Gelombang Persegi. Rajah 5.2 menunjukkan Gelombang Persegi audio mentah.



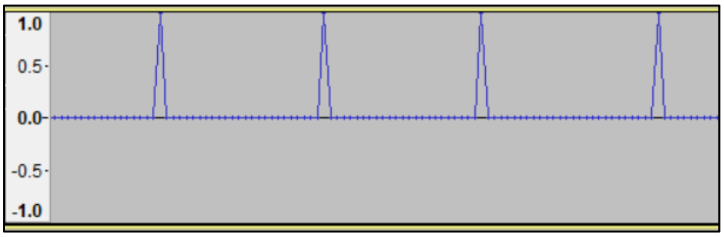

Rajah 5.2: Gelombang Persegi



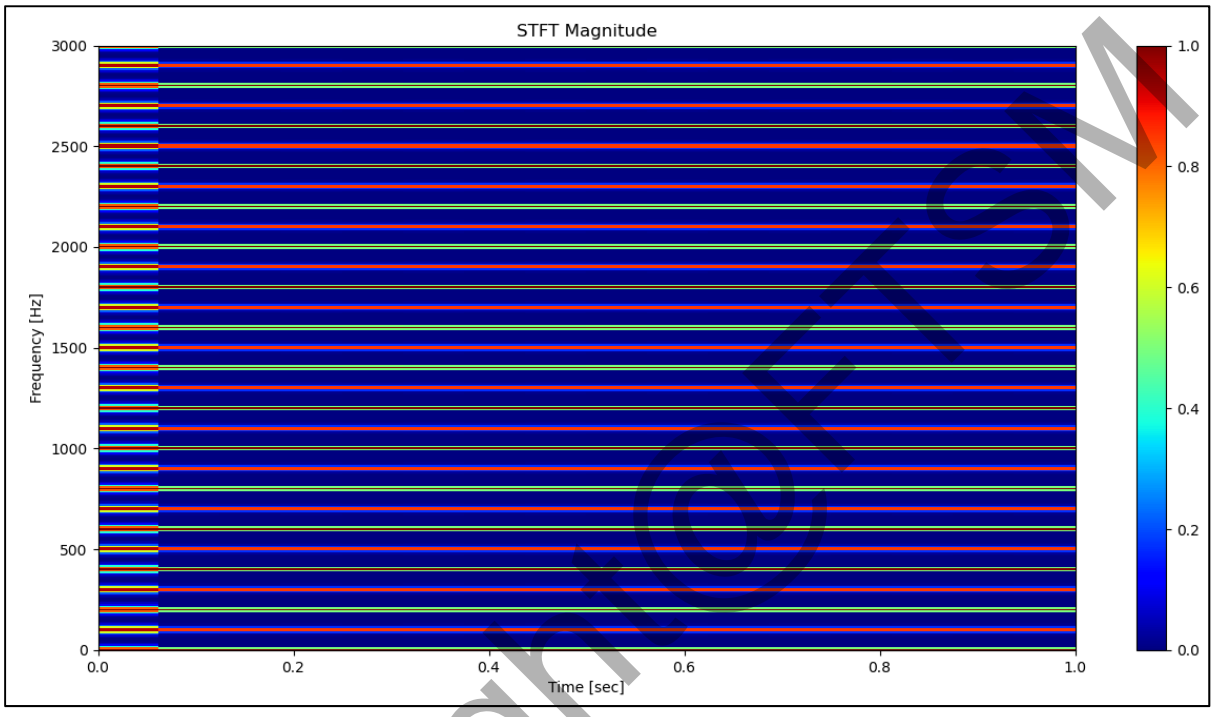

Rajah 5.3: *Spectrogram STFT* untuk Gelombang Persegi 100Hz

Seperti yang ditunjukkan di dalam Rajah 5.3, *STFT* mampu menghasilkan *Spectrogram* yang betul untuk gelombang persegi. Gelombang persegi yang telah digunakan merupakan gelombang 100Hz, dan *Spectrogram* tersebut telah berjaya menunjukkan satu garisan merah yang paling kuat nilainya di paksi frekuensi posisi 100Hz, dan menghasilkan garisan yang lebih lemah di setiap posisi yang boleh dibahagi oleh 100, iaitu di posisi 100Hz, 200Hz, 300Hz, 400Hz, 500Hz dan sebagainya. Garisan yang lebih lemah ini juga digelar sebagai timbre atau warna suara.

Walau bagaimanapun, terdapat satu situasi yang mampu menggalahkan algoritma ini. Sebagai contohnya, algoritma *STFT* tidak dapat menghasilkan *Spectrogram* yang betul apabila muzik yang diberi menggunakan gelombang nadi, seperti yang ditunjukkan dalam Rajah 5.4.




Rajah 5.4: Gelombang Nadi




Rajah 5.5: *Spectrogram STFT* untuk Gelombang Nadi 100Hz

Seperti yang ditunjukkan di dalam Rajah 5.5, *STFT* tidak mampu menghasilkan *Spectrogram* yang betul untuk gelombang nadi. Gelombang Nadi yang telah digunakan merupakan gelombang 100Hz, dan *Spectrogram* tersebut sepatutnya menunjukkan satu garisan merah yang paling kuat nilainya di paksi frekuensi posisi 100Hz. Tetapi *STFT* telah menghasilkan *Spectrogram* yang mempunyai banyak garisan merah yang sama kekuatannya di setiap posisi yang boleh dibahagi oleh 100, iaitu di posisi 100Hz, 200Hz, 300Hz, 400Hz, 500Hz dan sebagainya.

Hal ini boleh menjejaskan usaha pengecaman muzik pada langkah yang seterusnya kerana *Spectrogram* ini tidak tepat. Banyak muzik moden menggunakan gelombang digital buatan seperti Gelombang Nadi. Oleh sebab itu, kaedah menghasilkan *Spectrogram* menggunakan algoritma *STFT* mungkin akan gagal dalam muzik yang menggunakan gelombang digital buatan seperti Gelombang Nadi.

Walau bagaimanapun, algoritma *STFT* masih sangat berguna dalam banyak keadaan yang lain. Kebanyakan muzik tidak menggunakan Gelombang Nadi sahaja dan *STFT* masih boleh berfungsi dalam kebanyakan muzik yang lain. Oleh itu, kajian ini akan meneruskan penggunaan algoritma *STFT* untuk menghasilkan *Spectrogram.* Objektif Pertama Kajian ini telahpun selesai dicapai.

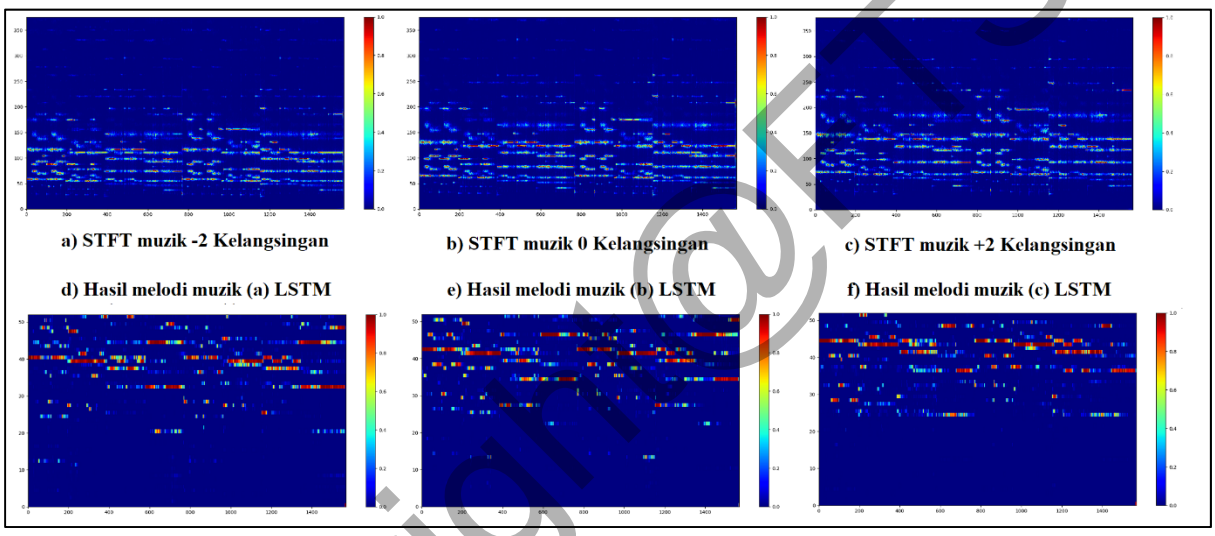
Objektif 2 kajian ini adalah menggunakan algoritma yang mampu menyari melodi muzik yang berperwakilan daripada *Spectrogram.* Selepas mendapatkan *Spectrogram* daripada langkah pertama, *Spectrogram* tersebut akan digunakan untuk menyari melodi muzik untuk kegunaan langkah yang seterusnya. Melodi muzik tersebut boleh dinyari daripada *Spectrogram* menggunakan Rangkaian Neural *LSTM* yang telah dilatih menggunakan dataset muzik *NSynth* menggunakan kaedah *Supervised Learning*.

Oleh sebab Rangkaian Neural *LSTM* perlu dilatih dengan menggunakan kaedah *Supervised Learning.* Dataset muzik *NSynth* tersebut perlu diproses untuk menghasilkan muzik yang lebih panjang. Dataset muzik *NSynth* mengandungi banyak nota muzik sepanjang 5 saat, dari 10 jenis alat muzik yang berbeza dalam jarak nada muzik MIDI 22 sehingga nada muzik MIDI 108. Maksudnya, dengan adanya dataset muzik *NSynth*, pelbagai jenis muzik dan karya muzik yang lebih panjang dan menarik boleh dihasilkan dengan menggunakan nota muzik *NSynth* tersebut, yang mengandungi 10 jenis alat muzik dalam jarak nada muzik MIDI 22 sehingga MIDI 108. Muzik yang lebih panjang ini, digelar sebagai muzik pelatih *LSTM* selepas ini, boleh digunakan untuk melatih Rangkaian Neural *LSTM.* Oleh sebab dataset *NSynth* mempunyai semua label nada muzik MIDI, *Supervised Learning* boleh dilakukan dengan menggunakan label tersebut apabila sedang menghasilkan muzik pelatih *LSTM*.

Kaedah yang digunakan untuk menghasilkan Muzik Pelatih Rangkaian Neural *LSTM* adalah dengan langkah yang berikut. Pertama, mengambil nada muzik *NSynth* tersebut secara rawak tetapi memastikan pergerakkan nada muzik untuk nota muzik yang seterusnya tidak berlaku secara mendadak dan berada di dalam jarak 6 sehingga 12 nada muzik MIDI. Kedua, memastikan semua nada muzik adalah sama rata dari segi jumlahnya supaya muzik pelatih *LSTM* ini tidak memberat sebelah sesuatu jenis nada muzik. Ketiga, menggunakan nota muzik yang berlainan panjangnya seperti 3 saat, atau 5 saat, atau 1 saat untuk mempelbagaikan variasi muzik supaya *LSTM* tidak menganggap semua nota muzik adalah 5 saat panjangnya. Keempat, dua nota muzik boleh dimain pada masa yang sama supaya *LSTM* belajar untuk membezakan dua nota muzik yang dimain pada masa yang sama. Kelima, muzik pelatih ini perlu mempunyai kord muzik supaya *LSTM* dapat memahami nada kord

yang dimain menggunakan banyak nota muzik yang berlainan nada pada masa yang sama. Keenam, nada muzik MIDI perlu dihadkan diantara jarak nada muzik MIDI 31 dan nada muzik MIDI 81 supaya dapat dipaparkan di dalam *Spectrogram* yang mempunyai had 3000Hz.

Kaedah ini merupakan kaedah yang berkesan. Rajah 5.6 menunjukkan hasil melodi muzik *LSTM* dengan menggunakan muzik pelatih kaedah tersebut. Melodi muzik yang mampu dihasilkan oleh *LSTM* dari *Spectrogram STFT* adalah agak jelas dan tepat. Ketiga-tiga melodi muzik yang telah dihasilkan tersebut adalah agak konsisten dengan peratusan persamaan pada 71%.



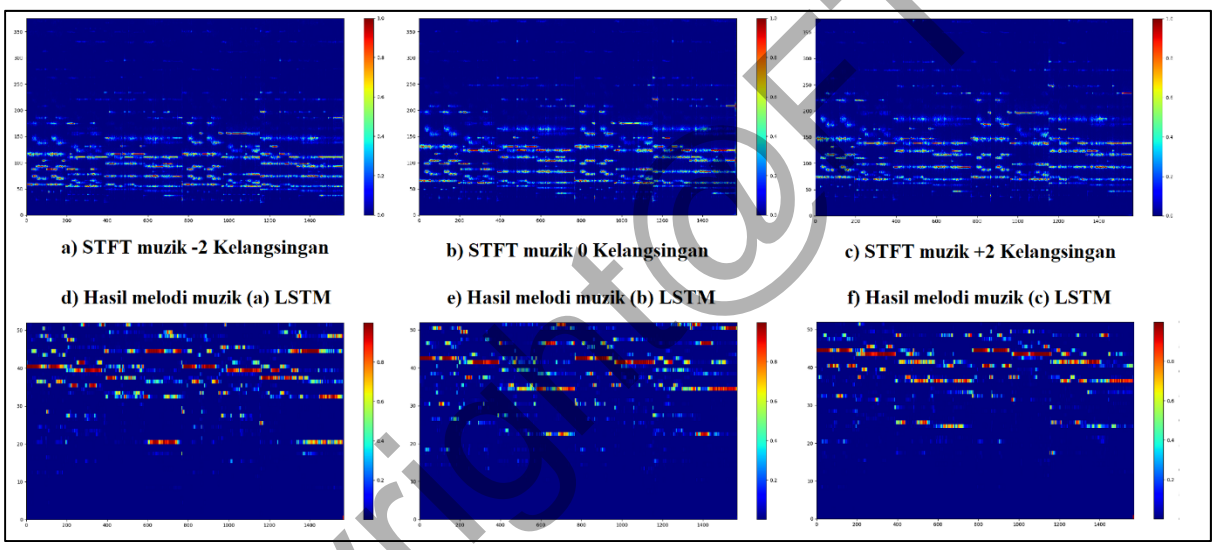

Rajah 5.6: Hasil melodi muzik *LSTM* menggunakan muzik pelatih

Rangkaian Neural *LSTM* boleh dibangunkan dengan menggunakan rangkaian neural yang berlainan saiz dari segi keluasan dan kedalamannya. Oleh sebab melatih rangkaian neural *LSTM* yang besar memerlukan unit pemprosesan grafik komputer yang mempunyai memori yang besar, usaha untuk melatih rangkaian neural *LSTM* yang besar adalah mustahil tanpa sumber pemprosesan grafik komputer yang boleh menampungnya.

Dengan adanya unit pemprosesan grafik yang lebih besar memorinya, iaitu Nvidia GTX 1080 ti yang mempunyai jumlah memori grafik sebanyak 11GB, kajian ini boleh berlangsung dengan rangkaian neural *LSTM* yang bersaiz besar. *LSTM* ini menggunakan lapisan neural *LSTM* pertama yang bersaiz 1128, lapisan neural *LSTM* kedua yang bersaiz

1128, lapisan neural *LSTM* ketiga yang bersaiz 1128, dan lapisan neural *LSTM* terakhir yang bersaiz 52.

Rangkaian neural *LSTM* yang bersaiz besar ini dapat belajar corak muzik dan menghasilkan melodi muzik daripada *Spectrogram* yang diberi. Melodi muzik yang dapat dihasilkan agak konsisten dengan muzik yang sama yang telah dinaik dan diturunkan kelangsingan. Nilai keluaran untuk melodi muzik kebanyakannya mempunyai nilai lebih tinggi daripada 0.6, dan ini menunjukkan bahawa *LSTM* yakin dengan hasil melodi muzik yang dikeluarkan. Nilai keluaran melodi muzik juga lebih jelas dan menunjukkan melodi yang terdapat di dalam muzik tersebut. Rajah 5.7 menunjukkan hasil melodi muzik *LSTM* bersaiz besar tersebut.




Rajah 5.7: Hasil melodi muzik *LSTM* bersaiz besar

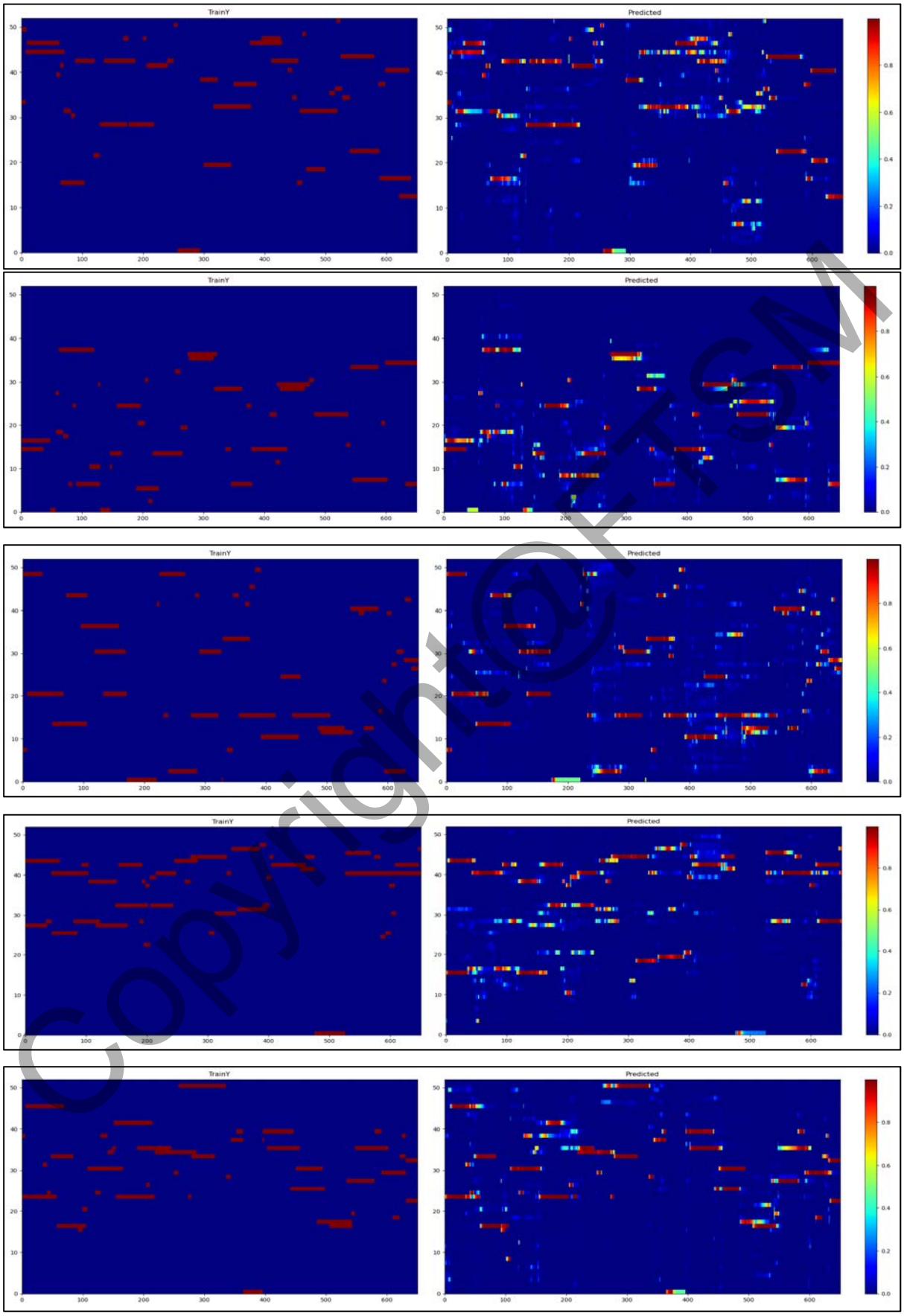
Rangkaian neural *LSTM* ini telah dilatih dengan set latih *NSynth* dengan masa yang panjang. Kajian ini mendapati bahawa kegiatan latihan ini mampu mengeluarkan hasil yang bagus dengan masa latihan selama 6 jam, dan memberi keputusan *Categorical Crossentropy* pada nilai 3.7012, dan keputusan *Accuracy* pada nilai 49.79%. Masa latihan selama 13 jam pula memberi keputusan *Categorical Crossentropy* pada nilai 3.4709, dan keputusan *Accuracy* pada nilai 56.18%. Masa latihan selama 19 jam pula member keputusan *Categorical Crossentropy* pada nilai 3.3108, dan keputusan *Accuracy* pada nilai 56.54%. Masa latihan selama 36 jam pula member keputusan *Categorical Crossentropy* pada nilai 3.0580, dan keputusan *Accuracy* pada nilai 57.54%. Masa latihan selama 60 jam pula

member keputusan *Categorical Crossentropy* pada nilai 2.7341, dan keputusan *Accuracy* pada nilai 56.58%. Jadual 5.1 menunjukkan kesimpulan keputusan rangkaian neural *LSTM* ketika tengah melatih menggunakan set latih *NSynth*.

Jadual 5.1: Kesimpulan Keputusan rangkaian neural *LSTM* ketika tengah melatih

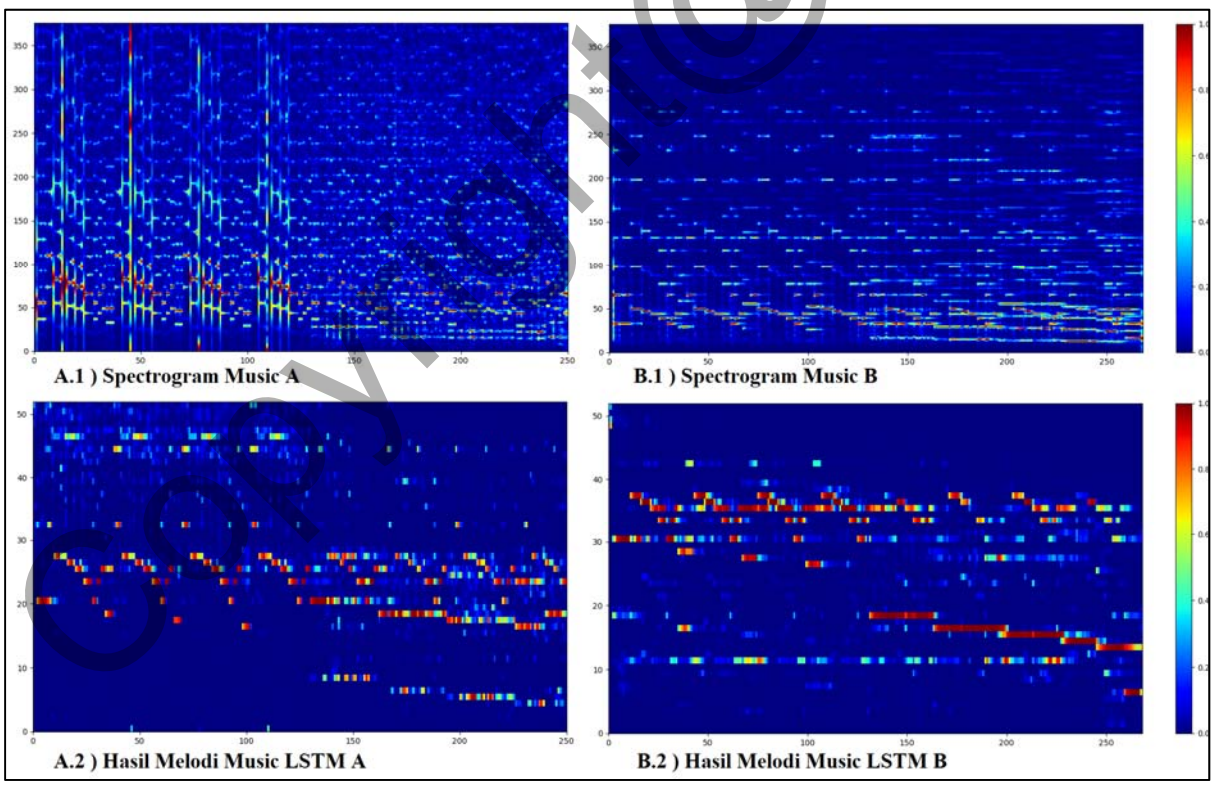
| Masa Melatih *LSTM* | *Categorical Crossentropy* | *Accuracy* |
|---|---|---|
| 6 jam | 3.7012 | 49.79% |
| 13 Jam | 3.4709 | 56.18% |
| 19 Jam | 3.3108 | 56.54% |
| 36 Jam | 3.0580 | 57.54% |
| 60 Jam | 2.7341 | 56.58% |

Selain itu, pengujian rangkaian neural *LSTM* ini boleh dilakukan dengan menggunakan set uji *NSynth*. Seperti yang dibincangkan di bab 4.3 Rangkaian neural *LSTM* akan mengambil *Spectrogram* muzik untuk diproses dan mengeluarkan melodi muzik yang terkandung di dalam *Spectrogram* muzik tersebut. Melodi muzik boleh dilukis di atas graf dengan menggunakan nada muzik MIDI sebagai paksi y dan menggunakan masa sebagai paksi x. Nada muzik MIDI mula dengan nada muzik MIDI 31 sehingga nada muzik MIDI 81, tatapi dilukis di dalam graf mula dengan paksi nilai 1 sehingga 51. Rajah 5.8 menunjukkan visualisasi label melodi muzik pelatih *LSTM* di sebelah kiri dan hasil melodi muzik *LSTM* di sebelah kanan. Secara kesuluruhannya, pengujian *LSTM* menggunakan set uji *NSynth* telah memberi keputusan yang berikut. Keputusan *Categorical Crossentropy* pada nilai 2.2386, dan keputusan *Accuracy* pada nilai 59.54%. Hal ini menunjukkan bahawa rangkaian neural *LSTM* ini tidak begitu tepat dengan penghasilan melodi muzik daripada *Spectrogram* muzik, tetapi masih dapat menghasilkan melodi muzik yang boleh digunakan.



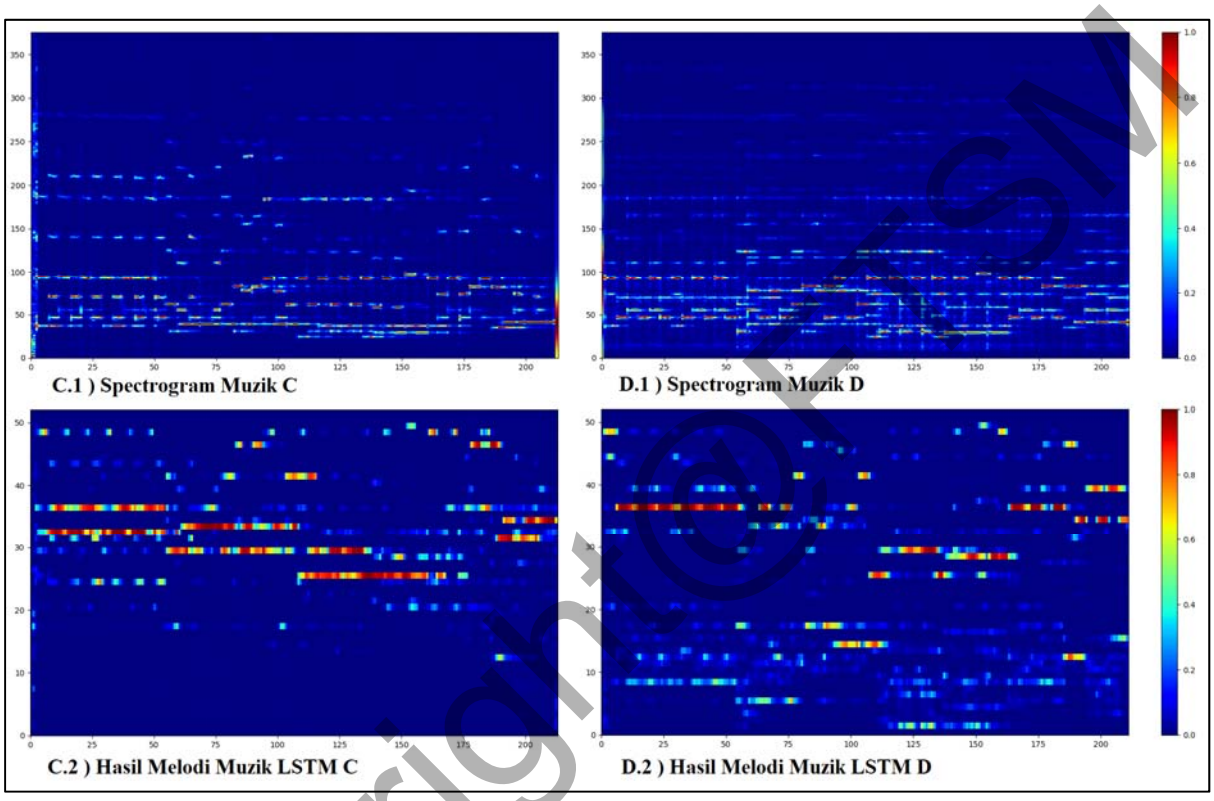

Rajah 5.8: label melodi muzik pelatih *LSTM* (kiri), hasil melodi muzik *LSTM* (kanan)

Pengujian yang seterusnya boleh dilakukan dengan menggunakan muzik yang dimain dengan menggunakan alat muzik yang berbeza. Sebagai contohnya, muzik yang bernama "megalovania" boleh didapati dalam platform *YouTube* dalam pelbagai jenis perisa, yang dimain dengan menggunakan alat muzik dan campuran yang berbeza, juga digelar sebagai *remix*. Seperti yang ditunjuk di Rajah 5.9, Muzik A dan Muzik B merupakan muzik yang sama tetapi dimain dengan alat muzik yang berbeza dan campuran yang berbeza. Hal ini sangat jelas dengan meneliti *Spectrogram* Muzik A dan Muzik B, *Spectrogram* kedua-dua muzik tersebut adalah tidak sama kerana alat muziknya berbeza. Selepas *Spectrogram* tersebut diproses oleh rangkaian neural *LSTM*, hasil melodi muzik *LSTM* untuk kedua-dua muzik tersebut adalah hampir sama dari segi bentuknya. Rajah tersebut juga menunjukkan bahawa Muzik B dimain dengan menggunakan nada (kelangsingan) yang lebih tinggi berbanding dengan Muzik A. Muzik A dan Muzik B merupakan muzik yang dimain dengan satu suara sahaja, bermaksud hanya satu alat muzik yang bersuara pada satu masa. Pengujian yang seterusnya akan menggunakan muzik yang dimain dengan dua suara.




Rajah 5.9: Penghasilan Melodi Muzik *LSTM* menggunakan *Spectrogram* muzik yang dimain dengan alat muzik dan nada muzik yang berbeza dengan satu suara.

Pengujian yang seterus ini akan menggunakan muzik yang dimain dengan dua suara, bermaksud terdapat dua alat muzik yang bersuara pada satu masa. Seperti yang ditunjukkan di Rajah 5.10. Hasil melodi muzik *LSTM* tidak konsisten antara Muzik C dan Muzik D. Walau bagaimanapun, masih boleh kelihatan beberapa bahagian yang konsisten, seperti bahagian atas paksi y dari nilai 40 hingga paksi y nilai 51.




Rajah 5.10: Penghasilan Melodi Muzik *LSTM* menggunakan *Spectrogram* muzik yang dimain dengan alat muzik dan nada muzik yang berbeza dengan dua suara.

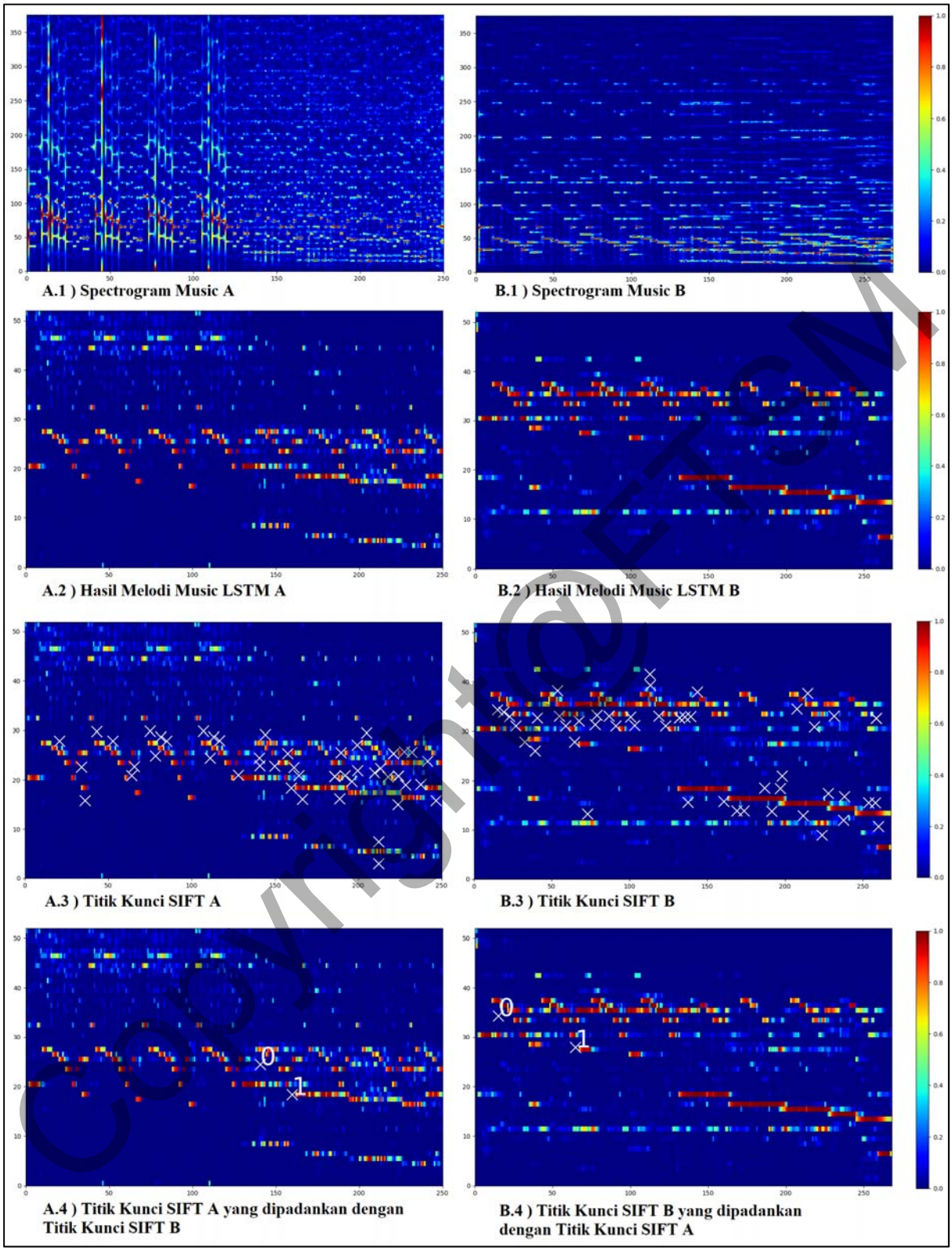
Secara kesimpulannya, pengujian Objektif 2, iaitu untuk menggunakan algoritma yang mampu menyari melodi muzik yang berperwakilan daripada *Spectrogram*, dapat dicapai hanya untuk muzik yang dimain dengan menggunakan satu suara pada satu masa sahaja. Jika muzik tersebut dimain dengan dua suara, bahagian yang mengandungi dua suara itu akan mendapat hasil melodi muzik *LSTM* yang tidak konsisten. Objektif 2 kajian ini boleh dicapai pada kajian masa depan dengan mengasingkan bunyi alat-alat muzik dahulu sebelum mencuba untuk menghasilkan melodi muzik tersebut.

Objektif 3 kajian ini adalah menguji algoritma *SIFT* untuk menyari cap jari melodi muzik. Dengan kata yang lain, hasil melodi muzik *LSTM* perlu digunakan untuk kegiatan pengecaman dengan menyimpannya dalam pangkalan data dengan kaedah yang sesuai, dan mencapainya dari pangkalan data dengan kaedah yang sesuai. Hasil melodi muzik *LSTM* tersebut merupakan rajah 2D dan perlu diproses untuk menjadi format nombor yang boleh disimpan di dalam pangkalan data. Kaedah yang dicadangkan adalah menggunakan *SIFT* untuk menyari cap jari melodi muzik kepada format nombor, yang kemudian disimpan di dalam pangkalan data menggunakan kaedah *LSH.*

Pengujian yang pertama adalah untuk mengaji ketepatan algoritma *SIFT* untuk menyari cap jari melodi muzik. Algoritma *SIFT* akan memilih Titik Kunci yang berperwakilan dalam rajah 2D hasil melodi muzik *LSTM* tersebut. Selepas siap memilih Titik Kunci tersebut, *SIFT* akan mengira nilai yang berperwakilan di sekitar Titik Kunci tersebut. Nilai Titik Kunci tersebut boleh digunakan untuk dibandingkan dengan Nilai Titik Kunci yang lain. Kalau dua Titik Kunci mempunyai nilai yang hampir sama, ia bermaksud melodi di sekitar Titik Kunci tersebut adalah hampir sama. Oleh sebab itu, Titik Kunci *SIFT* juga digelar sebagai cap jari melodi muzik dalam kajian ini. Rajah 5.11 menunjukkan proses penjanaan cap jari melodi muzik menggunakan *SIFT.*

Seperti yang ditunjukkan di Rajah 5.11, Titik Kunci *SIFT* yang dipilih adalah tidak konsisten antara dua hasil melodi muzik *LSTM* tersebut. Oleh sebab itu, nilai Titik Kunci yang dikira tidak dapat digunakan antara dua muzik tersebut. Hanya dua Titik Kunci (cap jari) yang mampu dibandingkan dan dipadankan antara dua muzik tersebut. Dari dua Titik Kunci yang dipadankan, Titik Kunci yang dilebel ‘0’ tersebut berjaya mengecam melodi sekitar dengan betul, Titik Kunci yang dilebel ‘1’ tersebut pula megecam melodi sekitar di lokasi yang salah.

Oleh sebab lokasi Titik Kunci yang dipilih sangat tidak konsisten, nilai Titik Kunci juga akan tidak sama. Kajian ini tidak dapat diteruskan kerana *SIFT* tidak dapat menghasilkan Titik Kunci (cap jari) yang berperwakilan daripada rajah 2D melodi muzik *LSTM* tersebut. Keupayaan dan kelemahan algoritma *SIFT* akan dibincang dengan lebih teliti seterusnya.

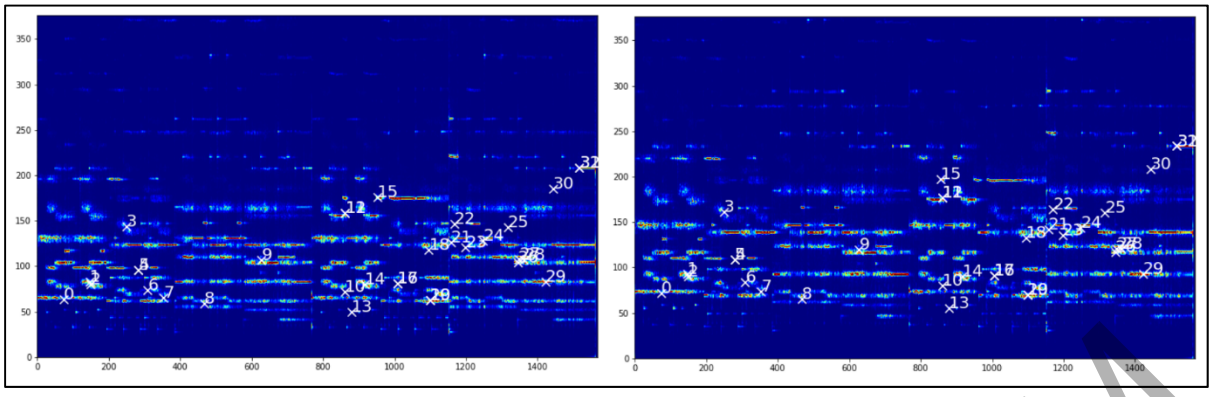



Rajah 5.11: Penyarian cap jari (Titik Kunci) melodi muzik menggunakan *SIFT*

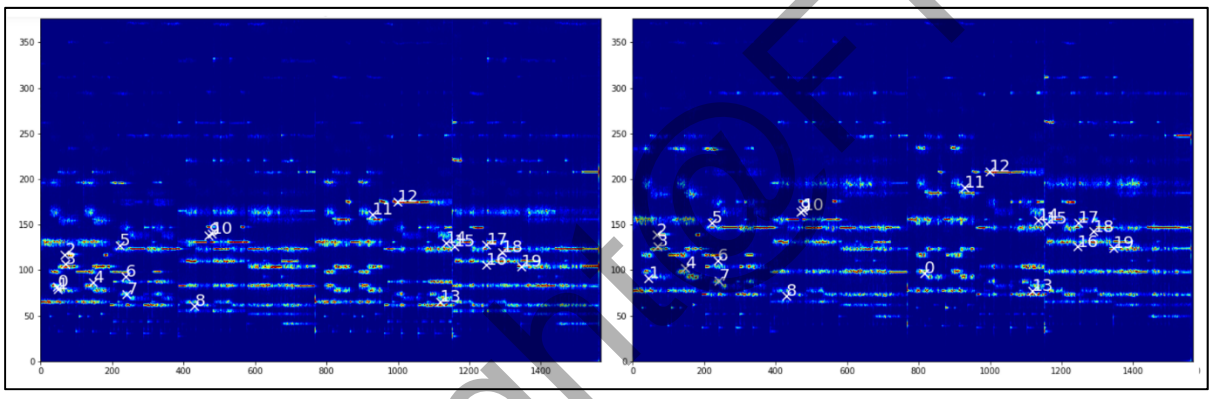
Kajian ini telah mencadangkan kaedah *SIFT* digunakan untuk menyari cap jari melodi muzik, kerana kajian lepas dari (X. Zhang et al. 2015), telah menunjukkan keberkesanan algoritma *SIFT* dalam proses penyarian cap jari *Spectrogram* muzik. Kajian ini pula menggunakan algoritma *SIFT* dalam proses penyarian cap jari melodi muzik daripada graf Melodi. Algoritma *SIFT* dapat menyari cap jari daripada *Spectrogram* tetapi tidak dapat menyari cap jari secara berkesan daripada graf Melodi. Bab ini akan menguji algoritma *SIFT* secara lanjut untuk mengenal pasti keupayaan algoritma *SIFT* dengan lebih teliti dalam proses penyarian cap jari muzik.

Pengujian yang pertama adalah menggunakan algoritma *SIFT* untuk menyari cap jari *Spectrogram* muzik seperti yang dicadangkan oleh kajian lepas dari (X. Zhang et al. 2015). Rajah 5.12 dan Rajah 5.13 menunjukkan algoritma *SIFT* berjaya menyari cap jari *Spectrogram* muzik dengan konsisten untuk muzik yang telah dinaik kelangsingan sebanyak +2 dan +3. Rajah 5.14 pula menunjukkan algoritma *SIFT* tidak mampu menyari cap jari *Spectrogram* muzik dengan mencukupi apabila muzik dinaik kelangsingan sebanyak +4. Rajah 5.15 dan Rajah 5.16 pula menunjukkan algoritma *SIFT* berjaya menyari cap jari *Spectrogram* muzik dengan konsisten untuk muzik yang telah melalui regangan masa sebanyak 1.2 dan 1.3. Rajah 5.17 pula menunjukkan algoritma *SIFT* tidak mampu menyari cap jari *Spectrogram* muzik dengan mencukupi apabila muzik melalui regangan masa sebanyak 1.4.

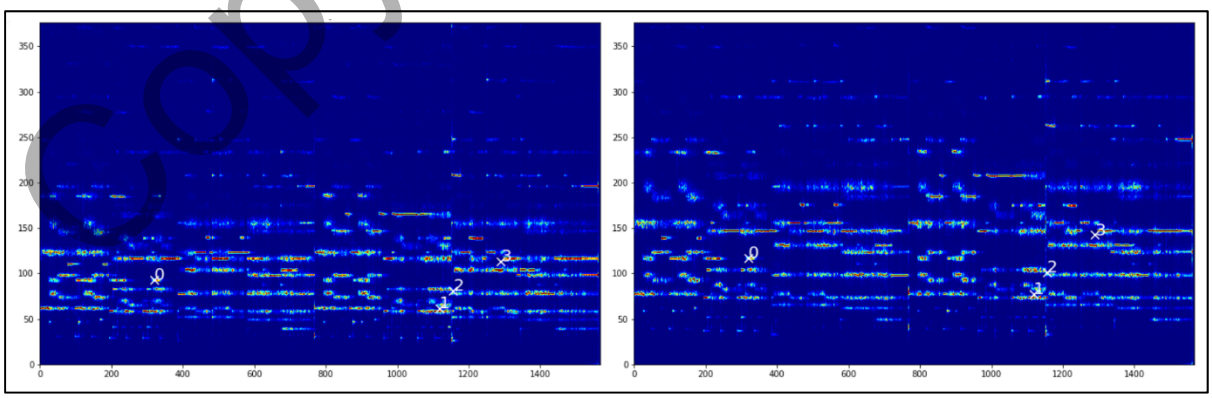
Pengujian pertama ini telah menunjukkan kelemahan algoritma *SIFT* dalam usaha menyari cap jari *Spectrogram* muzik. Algoritma *SIFT* hanya mampu menyari cap jari *Spectrogram* apabila perubahan bentuk rajah tidak berubah secara banyak. Regangan bentuk rajah tersebut secara sedikit, dengan nilai 1.4, akan memberi kesan negatif yang tinggi kepada ketepatan penyarian cap jari. Hal ini telah menunjukkan bahawa keupayaan algoritma *SIFT* dalam menyari cap jari muzik rajah yang telah melalui regangan adalah sangat terhad. Penemuan ini adalah konsisten dengan kajian lepas dari (X. Zhang et al. 2015).



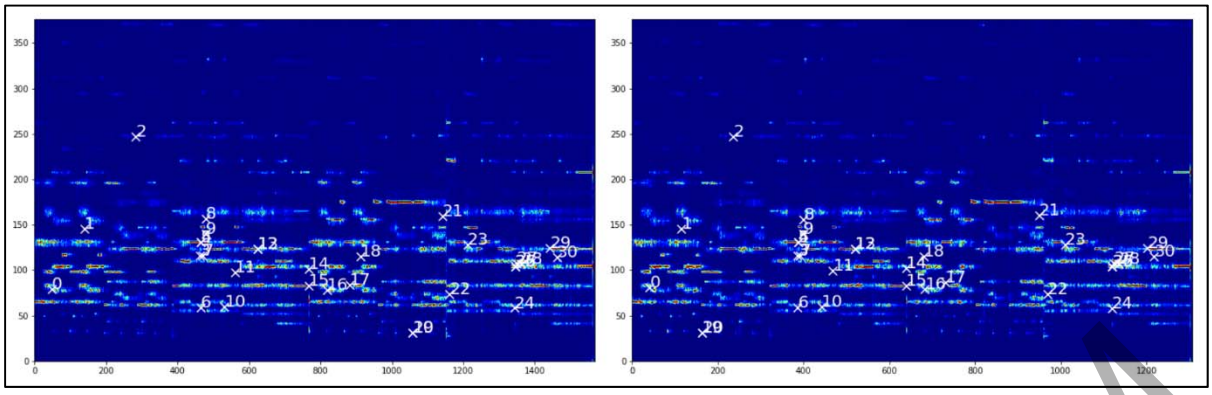
Rajah 5.12: *SIFT* berjaya menyari cap jari *Spectrogram* muzik dengan konsisten untuk muzik yang telah dinaik kelangsingan +2 di kanan, muzik asal di kiri.



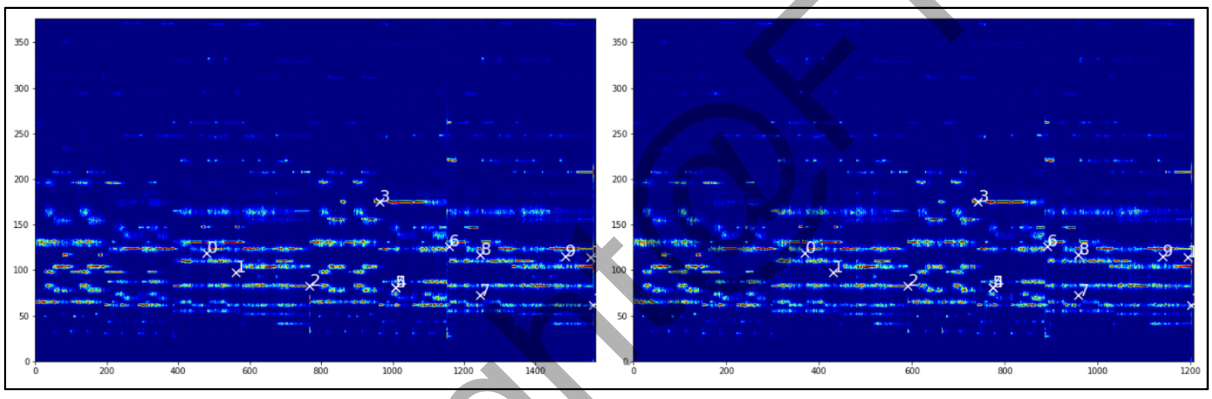
Rajah 5.13: *SIFT* berjaya menyari cap jari *Spectrogram* muzik dengan konsisten untuk muzik yang telah dinaik kelangsingan +3 di kanan, muzik asal di kiri.



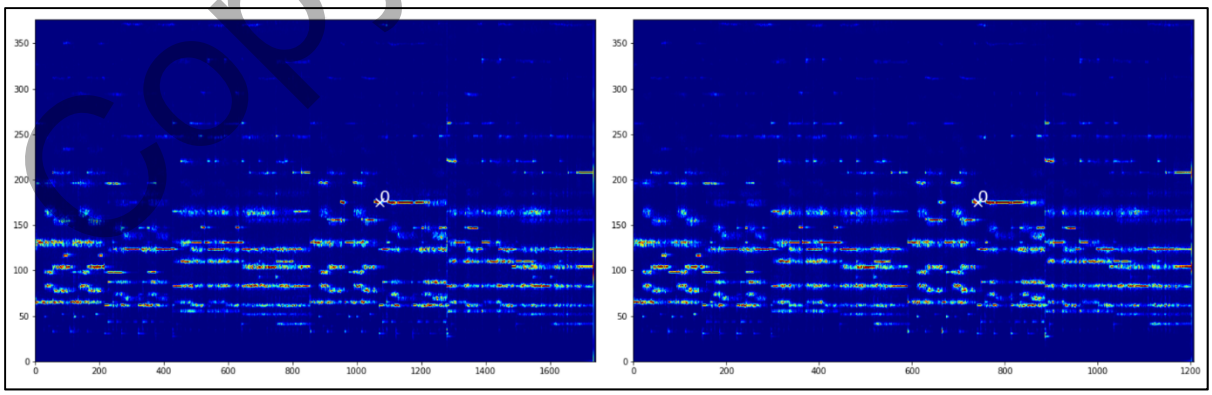
Rajah 5.14: *SIFT* tidak mampu menyari cap jari *Spectrogram* muzik dengan mencukupi apabila muzik dinaik kelangsingan +4 di kanan, muzik asal di kiri.



Rajah 5.15: *SIFT* berjaya menyari cap jari *Spectrogram* muzik dengan konsisten untuk muzik yang telah melalui regangan masa 1.2 di kanan, muzik asal di kiri.



Rajah 5.16: *SIFT* berjaya menyari cap jari *Spectrogram* muzik dengan konsisten untuk muzik yang telah melalui regangan masa 1.3 di kanan, muzik asal di kiri.

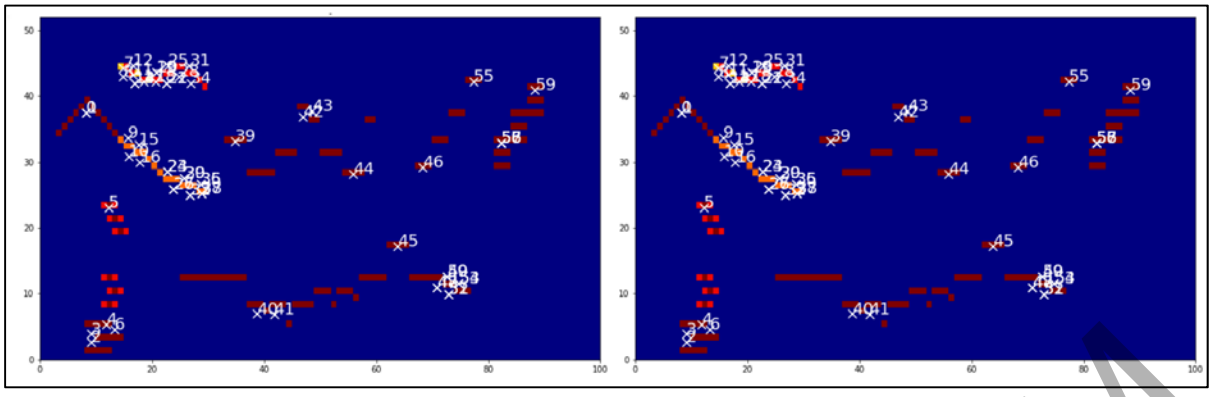


Rajah 5.17: *SIFT* tidak mampu menyari cap jari *Spectrogram* muzik dengan mencukupi apabila muzik melalui regangan masa 1.4 di kanan, muzik asal di kiri.

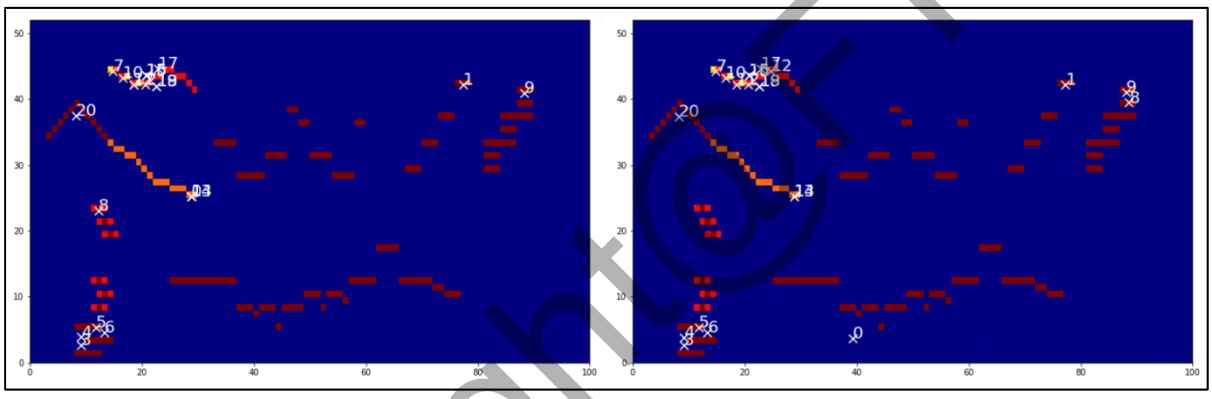
Pengujian yang kedua adalah menggunakan algoritma *SIFT* untuk menyari cap jari melodi muzik, iaitu kaedah penyarian cap jari muzik objektif 3 kajian ini. Rajah 5.18 menunjukkan *SIFT* mampu menyari cap jari melodi muzik dengan konsisten untuk graf melodi muzik yang 100% sama. Rajah 5.19 pula menunjukkan *SIFT* semakin susah menyari cap jari melodi muzik dengan konsisten untuk graf melodi yang ditambah bunyi bising sebanyak 0.001. Rajah 5.20 pula menunjukkan *SIFT* tidak mampu menyari cap jari melodi muzik untuk graf melodi yang ditambah bunyi bising sebanyak 0.005. Pengujian ini menunjukkan bahawa algoritma *SIFT* sangat sensitif dengan bunyi bising apabila menyari cap jari melodi muzik. Penemuan ini adalah terbalik dengan kajian lepas dari (X. Zhang et al. 2015), apabila *SIFT* digunakan untuk *Spectrogram*, bukan graf Melodi.

Rajah 5.21 dan Rajah 5.22 menunjukkan *SIFT* mampu menyari cap jari melodi muzik dengan konsisten untuk graf melodi muzik yang diubah amplitud nilainya secara sedikit dan sederhana. Rajah 5.23 pula menunjukkan *SIFT* semakin susah menyari cap jari melodi muzik dengan konsisten untuk graf melodi muzik yang diubah amplitud nilainya secara banyak. Pengujian ini menunjukkan bahawa algoritma *SIFT* mampu menyari cap jari melodi muzik dengan konsisten walaupun nilai melodi telah diturunkan, sekiranya bentuk rajah tersebut masih sama, dan tanpa bunyi bising di ruang sekitar nilai melodi.

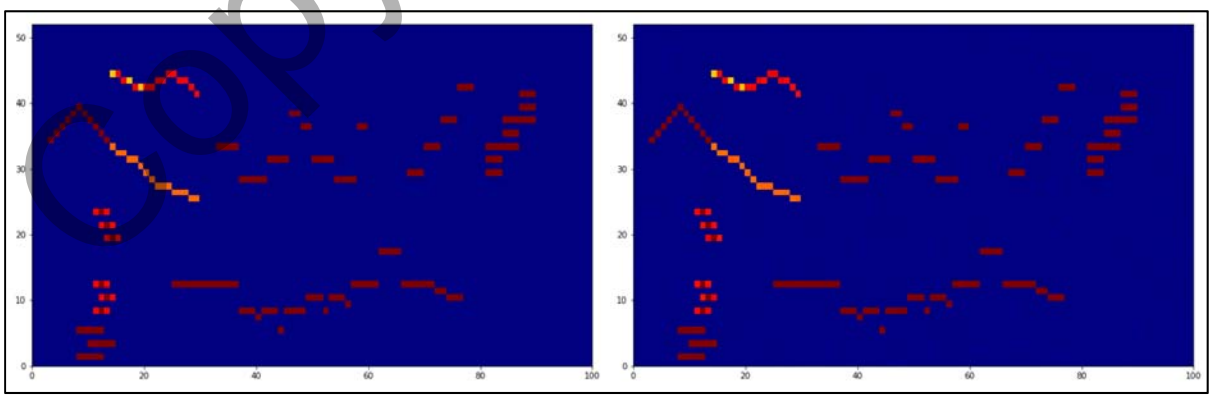
Pengujian kedua ini telah menunjukkan kelemahan algoritma *SIFT* dalam usaha menyari cap jari melodi muzik. Algoritma *SIFT* hanya mampu menyari cap jari melodi muzik apabila bentuk rajah tidak berubah secara banyak, dan tiada bunyi bising di ruang sekitar nilai melodi. Algoritma *SIFT* tidak mampu menyari cap jari melodi muzik dengan konsisten apabila bunyi bising ditambah di ruang sekitar nilai melodi. Hal ini telah menunjukkan bahawa keupayaan algoritma *SIFT* dalam menyari cap jari melodi muzik adalah sangat terhad.



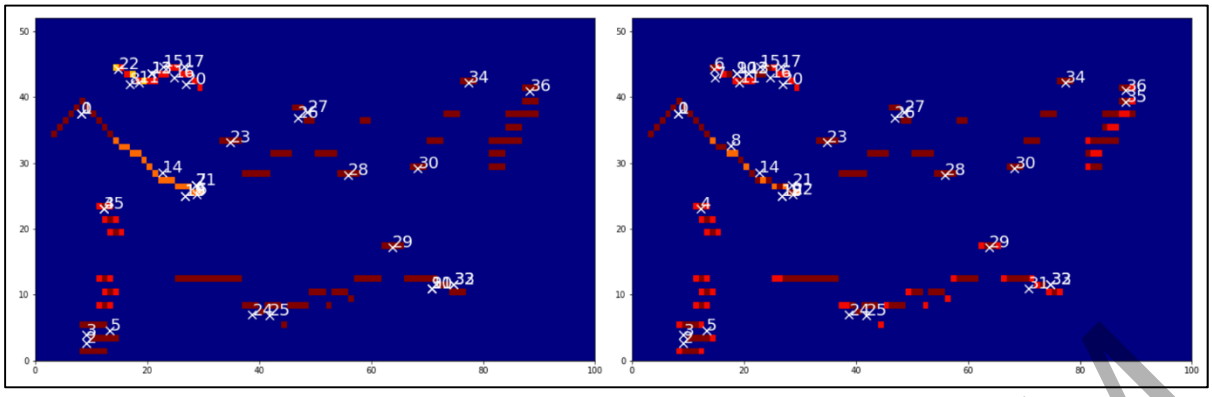
Rajah 5.18: *SIFT* mampu menyari cap jari melodi muzik dengan konsisten untuk graf melodi muzik yang 100% sama.



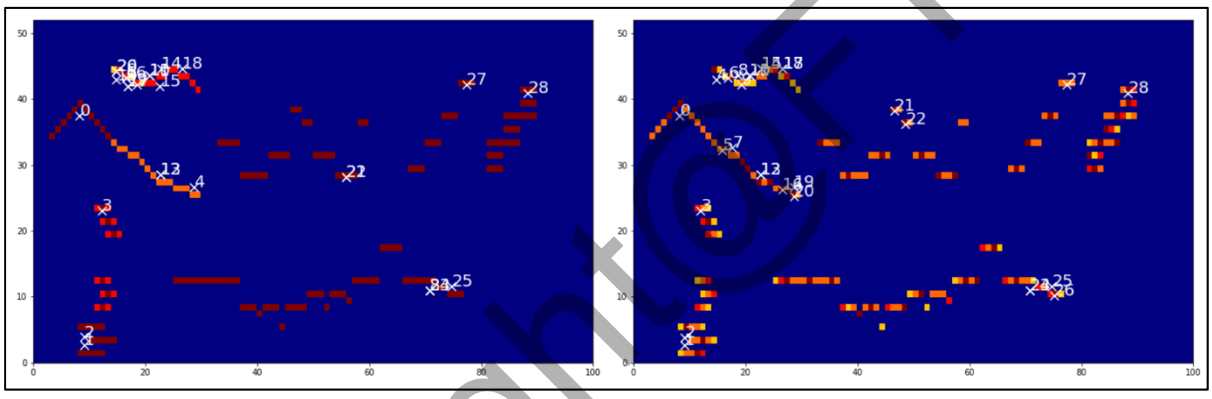
Rajah 5.19: *SIFT* semakin susah menyari cap jari melodi muzik dengan konsisten untuk graf melodi yang ditambah bunyi bising sebanyak 0.001 di kanan, muzik asal di kiri.



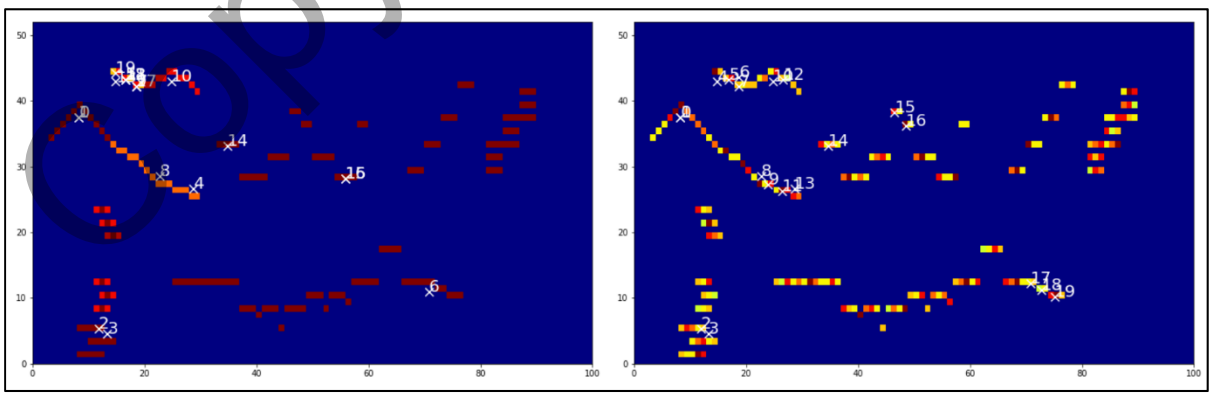
Rajah 5.20: *SIFT* tidak mampu menyari cap jari melodi muzik untuk graf melodi yang ditambah bunyi bising sebanyak 0.005 di kanan, muzik asal di kiri.



Rajah 5.21: *SIFT* mampu menyari cap jari melodi muzik dengan konsisten untuk graf melodi muzik yang diubah amplitud nilainya secara sedikit di kanan, muzik asal di kiri.



Rajah 5.22: *SIFT* mampu menyari cap jari melodi muzik dengan konsisten untuk graf melodi muzik yang diubah amplitud nilainya secara sederhana di kanan, muzik asal di kiri.



Rajah 5.23: *SIFT* semakin susah menyari cap jari melodi muzik dengan konsisten untuk graf melodi muzik yang diubah amplitud nilainya secara banyak di kanan, muzik asal di kiri.

Secara ringkasnya, pengujian lanjutan *SIFT* telah menunjukkan bahawa algoritma *SIFT* hanya berkesan untuk menyari cap jari *Spectrogram* muzik apabila muzik tersebut tidak melalui regangan yang banyak. Algoritma *SIFT* tidak berkesan untuk menyari cap jari Melodi muzik kerana perubahan nilai graf Melodi adalah terlalu tinggi untuk diproses dengan kaedah *SIFT*. Oleh sebab itu, objektif 3 kajian ini perlu dicapai dengan menggunakan algoritma yang lain pada kajian masa akan datang.

Kesimpulannya, pengujian yang telah dijalankan mendapati bahawa Objektif 1 dan Objektif 2 dapat dipenuhi oleh kajian ini, tetapi Objektif 3 masih memerlukan lebih banyak penyelidikan untuk dihabiskan pada masa akan datang. Objektif 1 dapat dipenuhi dengan menggunakan *STFT*. Objektif 2 dapat dipenuhi dengan menggunakan rangkaian neural *LSTM* yang bersaiz besar, dengan lapisan 1128 – 1128 – 1128 – 52. Rangkaian neural *LSTM* juga perlu dilatih dengan muzik pelatih yang menggunakan kaedah berikut. Pertama, mengambil nada muzik *NSynth* tersebut secara rawak tetapi memastikan pergerakkan nada muzik untuk nota muzik yang seterusnya tidak berlaku secara mendadak dan berada di dalam jarak 6 sehingga 12 nada muzik MIDI. Kedua, memastikan semua nada muzik adalah sama rata dari segi jumlahnya supaya muzik pelatih *LSTM* ini tidak memberat sebelah sesuatu jenis nada muzik. Ketiga, menggunakan nota muzik yang berlainan panjangnya seperti 3 saat, atau 5 saat, atau 1 saat untuk mempelbagaikan variasi muzik supaya *LSTM* tidak menganggap semua nota muzik adalah 5 saat panjangnya. Keempat, dua nota muzik boleh dimain pada masa yang sama supaya *LSTM* belajar untuk membezakan dua nota muzik yang dimain pada masa yang sama. Kelima, muzik pelatih ini perlu mempunyai kord muzik supaya *LSTM* dapat memahami nada kord yang dimain menggunakan banyak nota muzik yang berlainan nada pada masa yang sama. Keenam, nada muzik MIDI perlu dihadkan diantara jarak nada muzik MIDI 31 dan nada muzik MIDI 81 supaya dapat dipaparkan di dalam *Spectrogram* yang mempunyai had 3000Hz.

# 6 KESIMPULAN

Secara Kesimpulannya, kajian ini mendalami kaedah algoritma kecerdasan buatan untuk mengecam karya-karya muzik yang sama dan berpotensi membantu pengguna mencari karya muzik yang hampir sama atau memelihara hak cipta karya muzik artis asal. Algoritma yang dicadangkan adalah seperti berikut, menggunakan *Short Time Fourier Transform (STFT)* untuk mendapat komposisi frekuensi mengikut masa (*Spectrogram)*, menggunakan Rangkaian Neural *LSTM* untuk mendapatkan Melodi Muzik, menggunakan *Scale Invarient Feature Transform (SIFT)* untuk menyari Cap Jari Melodi Muzik, dan menggunakan *Locality Sensitive Hashing (LSH)* untuk kegunaan pengecaman muzik dan penyimpanan Cap Jari Melodi Muzik dalam pangkalan data muzik.

Setakat ini, algoritma *Short Time Fourier Transform (STFT)* berjaya menghasilkan *Spectrogram* yang jelas, dan memenuhi Objektif 1. Rangkaian Neural *LSTM* berjaya menghasilkan melodi muzik untuk muzik yang dimain dengan satu suara, dan memenuhi Objektif 2. Algoritma *SIFT* pula tidak dapat menyari Cap Jari Melodi Muzik yang berperwakilan dengan konsisten, dan tidak memenuhi Objektif 3. Oleh sebab masa kajian yang terhad, kajian ini terpaksa dihabiskan setakat ini sahaja.

# 7 RUJUKAN

Balan, R., Casazza, P. & Edidin, D. 2006. On signal reconstruction without phase 20: 345–356. doi:10.1016/j.acha.2005.07.001

Ehret, G. 1978. Stiffness gradient along the basilar membrane as a way for spatial frequency analysis within the cochlea. *Fachbereich Biologie, Universitat Konstanz, D-7750 Konstanz, Federal Republic of Germany*.

Jackie. 2018. How I Deal With Fake YouTube Copyright Claims by Believe Music. https://medium.com/@JackieM/how-i-deal-with-fake-youtube-copyright-claims-by-believe-music-b9cb9067fff9 [8 December 2019].

Jesse Engel, Cinjon Resnick, Adam Roberts, Sander Dieleman, Douglas Eck Karen Simonyan, and M. N. 2017. Neural Audio Synthesis of Musical Notes with WaveNet Autoencoders.

Majewski, G. 2019. Music Copyright 101: How to copyright your music as an artist, songwriter, or producer. https://diymusician.cdbaby.com/music-rights/copyright-for-musicians/ [14 May 2020].

Martin F. McKinney, J. B. 1973. Features for Audio and Music Classification 4.

Wang, A. L. 2003. An Industrial-Strength Audio Search Algorithm.

You, S. D., Chen, W. & Chen, W. 2013. Music Identification System Using MPEG-7 Audio Signature Descriptors 2013.

Zhang, T. & Kuo, C. J. 1998. Content-Based Classification and Retrieval of Audio. *Integrated Media Systems Center and Department of Electrical Engineering-Systems, University of Southern California, Los Angeles* (1).

Zhang, X., Zhu, B., Li, L., Li, W., Li, X., Wang, W., Lu, P., et al. 2015. SIFT-based local spectrogram image descriptor : a novel feature for robust music identification. doi:10.1186/s13636-015-0050-0

Andrej Karpathy, 2015. The Unreasonable Effectiveness of Recurrent Neural Networks. http://karpathy.github.io/2015/05/21/rnn-effectiveness/ [21 Sept 2019]

Christopher Olah, 2015. Understanding LSTM Networks. https://colah.github.io/posts/2015-08-Understanding-LSTMs/ [22 Sept 2019]