

Music Genre Classification

Speech Recognition

Kelompok: 11

- Jovan Amarta Liem - 2602058932
- Frederik Hartono - 2602064481
- Christopher Alexander Chandra - 2602081462

I. Latar Belakang

Di era modern ini, musik telah diproduksi secara pesat. Jumlah musik yang tersedia sangat banyak, dan musik - musik tersebut juga memiliki genre yang sangat beragam. Sebagai contoh, rock, jazz, classic, pop dan lain sebagainya. Musik dapat diorganisir dan dikelompokkan berdasarkan genrenya, dan manfaat pengelompokkan musik tersebut antara lain:

1. Meningkatkan kenyamanan saat mendengarkan sekelompok musik
2. Mempermudah artis untuk menemukan musik baru yang sesuai dengan genre
3. Penempatan rekomendasi yang lebih relevan dan terpersonalisasi
4. Analisis tren musik menjadi lebih mudah
5. Kemudahan dalam pengembangan teknologi baru pada bidang musik

Pengelompokkan musik berdasarkan genre secara manual akan sangat memakan waktu. Maka dari itu, pengelompokkan musik menggunakan model machine learning dan deep learning akan memudahkan proses pengelompokkan musik tersebut. Project ini dilakukan dengan tujuan untuk mengelompokkan atau mengklasifikasikan musik ke dalam genre menggunakan model machine learning dan deep learning.

Model machine learning adalah sebuah program yang dapat menemukan sebuah pola dan membuat output berdasarkan kumpulan data atau dataset. Model machine learning dapat digunakan untuk 2 task yaitu klasifikasi dan regresi. Model machine learning juga dibagi ke dalam 3 kategori yaitu supervised, unsupervised, dan reinforcement learning.

Pada projek kali ini model machine learning yang digunakan adalah klasifikasi, karena model machine learning tersebut akan mengeluarkan output berupa genre dari musik. Kemudian kategori machine learning yang digunakan adalah supervised learning. Contoh dari model yang dapat digunakan untuk klasifikasi supervised learning adalah XGBoost, Random Forest, Naive Bayes, dan lain sebagainya.

Sedangkan, model deep learning mirip dengan model machine learning. Namun, model deep learning memiliki arsitektur neural network yang memiliki lapisan tersembunyi atau hidden layer dan lapisan output atau output layer. Model deep learning memungkinkan untuk mempelajari representasi data melalui layer atau lapisan-lapisan tersebut. Model

machine learning yang populer digunakan adalah CNN (Convolutional Neural Network), RNN (Reccurent Neural Network), Transformer, LSTM (Long Short Term Memory) dan lain sebagainya.

II. Rumusan Masalah

Mengembangkan suatu model untuk tugas klasifikasi genre musik menghadirkan beberapa pertanyaan. Proyek ini akan mendalami dan menjawab beberapa pertanyaan dan masalah, antara lain:

1. Bagaimana cara mengembangkan model machine learning yang mampu mendeteksi genre musik dari file audio?
2. Fitur audio apa saja yang paling efektif untuk digunakan dalam klasifikasi genre musik?
3. Seberapa akurat model tersebut dalam mengklasifikasikan genre musik pada dataset yang berbeda?
4. Apa saja tantangan yang dihadapi dalam pengembangan dan evaluasi model tersebut?

III. Tujuan Project

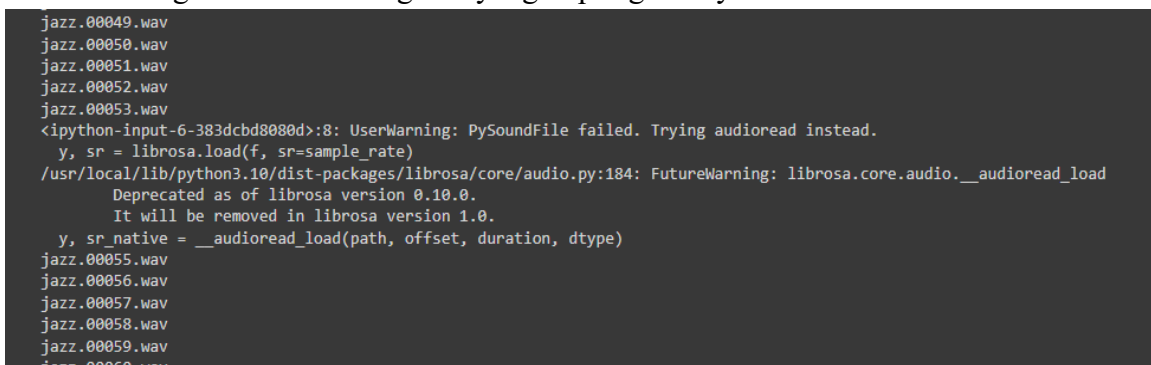
Proyek ini bertujuan untuk mengembangkan sebuah model machine learning dan deep learning berbasis python yang mampu mendeteksi genre musik dengan tingkat akurasi yang tinggi dari suatu lagu berdasarkan fitur-fitur audio lagu tersebut. Proyek ini diharapkan dapat mempermudah pengguna untuk mengelompokkan dan menemukan genre musik yang mereka sukai sehingga meningkatkan kenyamanan saat mendengarkan musik, proyek ini juga bertujuan untuk mengidentifikasi dan mengevaluasi fitur audio apa saja yang efektif untuk melakukan pengelompokkan musik, tingkat keakuratan model yang digunakan, dan tantangan dalam pengembangan klasifikasi genre musik.

IV. Ruang Lingkup Project

1. Dataset yang digunakan merupakan dataset dari kaggle. Dataset yang digunakan memiliki 1 file corrupt pada file jazz, sehingga ketika dilakukan feature extraction audio jazz nomor 54 tidak dapat diekstraksi. Namun, karena pemilik dari dataset tersebut sudah menyediakan file csv yang berisi fitur-fitur yang sudah diekstraksi dari audionya, hal ini tidak menjadi masalah yang terlalu besar.
2. Model yang digunakan adalah MLP (Multi Layer Perceptron), XGBoost, Random Forest, LSTM (Long Short Term Memory) dengan arsitektur Convolutional. Model-model ini nantinya akan digunakan untuk tugas klasifikasi. Untuk model XGBoost dan Random Forest, model XGBoost hanya menggunakan parameter `n_estimator` 1000, dan untuk model Random Forest tidak menggunakan parameter apapun. Untuk model LSTM, MLP diperlukan untuk membuat arsitekturnya dan juga mencoba berbagai macam parameter seperti filter dan kernel.
3. Proyek ini hanya mencakup pengembangan codingan backend tanpa mencakup pengembangan User Interface dan integrasi ke dalam aplikasi / platform musik.

V. Deskripsi Dataset

Dataset yang digunakan berasal dari kaggle. Jumlah Genre yang terdapat pada dataset tersebut berjumlah 10 genre dan setiap genrenya berisi 100 lagu. Namun seperti yang disampaikan pada bab 4, audio pada genre jazz nomor 54 corrupt sehingga tidak dapat digunakan. Hal tersebut tidak menjadi masalah karena pemilik dataset sudah menyediakan fitur yang sudah diekstraksi sebelumnya. Setiap lagu memiliki durasi 30 detik, dan dari audio berdurasi 30 detik tersebut dilakukan feature extraction. Fitur-fitur yang diekstrak antara lain: chroma, root mean square energy, spectral centroid, spectral bandwidth, roll off, zero crossing rate, harmony, perceptual, dan tempo. Fitur-fitur yang diekstrak tersebut akan dimasukkan ke dalam file csv yang nantinya, file csv tersebut akan digunakan untuk training dan testing model. Hal ini juga berlaku sama untuk dataset kedua yang merupakan dataset dari lagu yang berisi 3 detik, asal lagu 3 detik tersebut adalah lagu yang berdurasi 30 detik dibagi ke dalam 10 segmen yang tiap segmennya berdurasi 3 detik.



```
jazz.00049.wav
jazz.00050.wav
jazz.00051.wav
jazz.00052.wav
jazz.00053.wav
<ipython-input-6-383dcdbd8080d>:8: UserWarning: PySoundFile failed. Trying audioread instead.
  y, sr = librosa.load(f, sr=sample_rate)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:184: FutureWarning: librosa.core.audio.__audioread_load
  Deprecated as of librosa version 0.10.0.
  It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
jazz.00055.wav
jazz.00056.wav
jazz.00057.wav
jazz.00058.wav
jazz.00059.wav
jazz.00060.wav
```

Gambar ketika proses ekstraksi fitur dilakukan, jazz nomor 54 mengalami error.

```
1 data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9990 entries, 0 to 9989
Data columns (total 60 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   filename                             9990 non-null   object
1   length                               9990 non-null   int64
2   chroma_stft_mean                     9990 non-null   float64
3   chroma_stft_var                      9990 non-null   float64
4   rms_mean                             9990 non-null   float64
5   rms_var                              9990 non-null   float64
6   spectral_centroid_mean               9990 non-null   float64
7   spectral_centroid_var                9990 non-null   float64
8   spectral_bandwidth_mean              9990 non-null   float64
9   spectral_bandwidth_var               9990 non-null   float64
10  rolloff_mean                         9990 non-null   float64
11  rolloff_var                          9990 non-null   float64
12  zero_crossing_rate_mean               9990 non-null   float64
13  zero_crossing_rate_var                9990 non-null   float64
14  harmony_mean                         9990 non-null   float64
15  harmony_var                          9990 non-null   float64
16  perceptr_mean                        9990 non-null   float64
17  perceptr_var                         9990 non-null   float64
18  tempo                               9990 non-null   float64
19  mfcc1_mean                          9990 non-null   float64
20  mfcc1_var                           9990 non-null   float64
21  mfcc2_mean                          9990 non-null   float64
22  mfcc2_var                           9990 non-null   float64
23  mfcc3_mean                          9990 non-null   float64
24  mfcc3_var                           9990 non-null   float64
25  mfcc4_mean                          9990 non-null   float64
26  mfcc4_var                           9990 non-null   float64
27  mfcc5_mean                          9990 non-null   float64
28  mfcc5_var                           9990 non-null   float64
29  mfcc6_mean                          9990 non-null   float64
30  mfcc6_var                           9990 non-null   float64
31  mfcc7_mean                          9990 non-null   float64
32  mfcc7_var                           9990 non-null   float64
33  mfcc8_mean                          9990 non-null   float64
34  mfcc8_var                           9990 non-null   float64
35  mfcc9_mean                          9990 non-null   float64
36  mfcc9_var                           9990 non-null   float64
37  mfcc10_mean                         9990 non-null   float64
38  mfcc10_var                          9990 non-null   float64
```

Gambar dataset info

```

39 mfcc11_mean          9990 non-null float64
40 mfcc11_var          9990 non-null float64
41 mfcc12_mean          9990 non-null float64
42 mfcc12_var          9990 non-null float64
43 mfcc13_mean          9990 non-null float64
44 mfcc13_var          9990 non-null float64
45 mfcc14_mean          9990 non-null float64
46 mfcc14_var          9990 non-null float64
47 mfcc15_mean          9990 non-null float64
48 mfcc15_var          9990 non-null float64
49 mfcc16_mean          9990 non-null float64
50 mfcc16_var          9990 non-null float64
51 mfcc17_mean          9990 non-null float64
52 mfcc17_var          9990 non-null float64
53 mfcc18_mean          9990 non-null float64
54 mfcc18_var          9990 non-null float64
55 mfcc19_mean          9990 non-null float64
56 mfcc19_var          9990 non-null float64
57 mfcc20_mean          9990 non-null float64
58 mfcc20_var          9990 non-null float64
59 label                9990 non-null object
dtypes: float64(57), int64(1), object(2)
memory usage: 4.6+ MB

```

Gambar dataset info 2

```

1 data.shape
(9990, 60)

```

Gambar dataset shape untuk dataset 3 detik

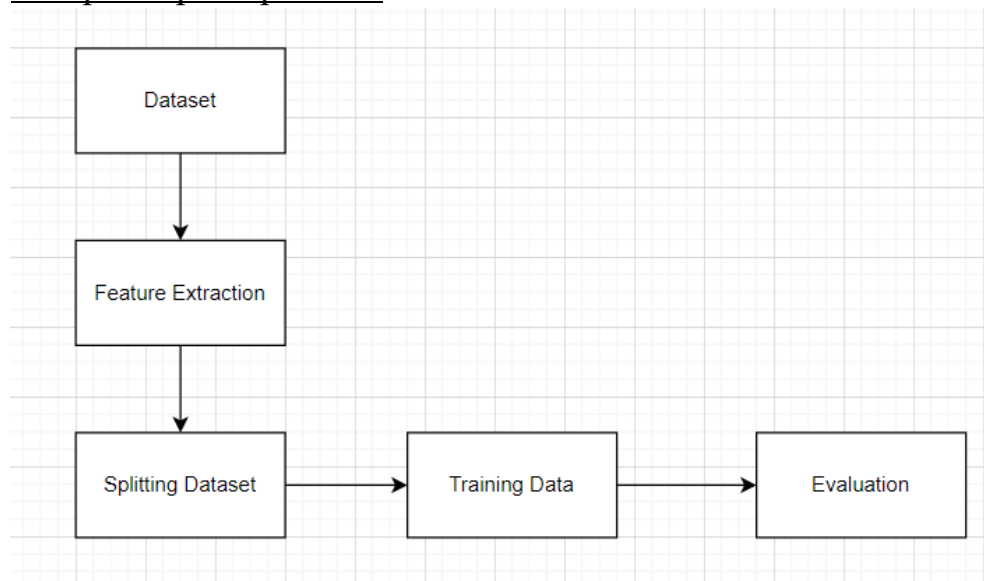
```

1 data.shape
(1000, 60)

```

Gambar dataset shape untuk dataset 30 detik

VI. Tahap-tahap Eksperimen



Berikut ini tahap-tahap eksperimen yang kami lakukan:

1. Dataset
Mencari topik permasalahan berupa klasifikasi genre musik dan mengumpulkan dataset dari topik terkait.
2. Feature Extraction
Melakukan preprocessing data dan mengekstraksi fitur-fitur dari dataset audio.
3. Splitting dataset
Melakukan splitting pada dataset menjadi training data, validation data, dan testing data.
4. Training data
Merancang model AI berupa XGBoost, Random Forest Classifier, MLP(Multi Layer Perceptron), dan LSTM (Long Short Term Memory) dan melakukan training model dari dataset yang telah diolah.
5. Evaluation
Menganalisis hasil dari setiap hasil training model dan mengevaluasi metrik akurasi dari setiap model.

VII. Preprocessing

1. Read dataset
Dataset dibaca dari file CSV menggunakan 'pd.read_csv', Dataset berisi fitur-fitur yang telah diekstraksi dari file audio.

```
#Reading the csv file
data = pd.read_csv("/content/features_3_sec.csv")
```

2. Hapus kolom yang tidak diperlukan

```
] datas = data.drop(labels='filename',axis=1)
```

3. Normalization

Fitur-fitur dinormalisasi menggunakan StandardScaler untuk memastikan setiap fitur memiliki skala yang sama, hal ini penting untuk algoritma machine learning.

```
#Standard scaler is used to standardize features & look like standard
fit = StandardScaler()
X = fit.fit_transform(np.array(datas.iloc[:, :-1], dtype = float))
```

4. Split dataset ke dalam training dan testing

Dataset dibagi menjadi set pelatihan dan set pengujian menggunakan train_test_split untuk mengevaluasi performa model pada data yang belum pernah dilihat oleh model.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

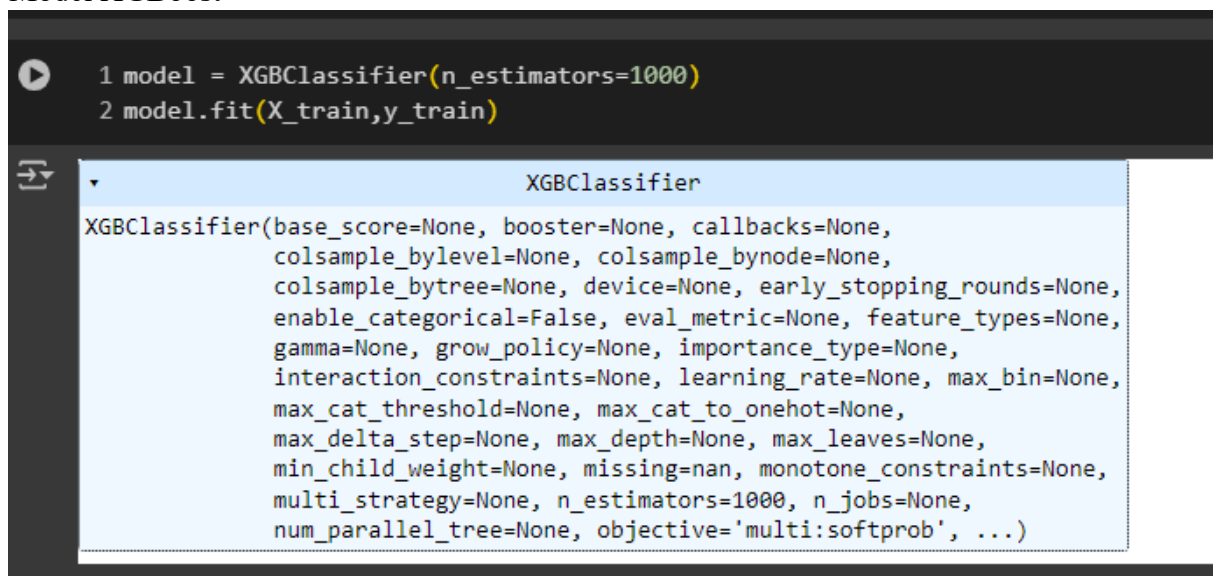
Dataset tersebut dibagi menjadi 80% untuk training dan 20% untuk testing.

Namun, hal ini hanya digunakan untuk training model machine learning, untuk model deep learning dataset dibagi menjadi 80:10:10

```
[24] 1 # Splitting dataset to 80% training and 20% testing
      2 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
      3 X_test_1, X_val, y_test_1, y_val = train_test_split(X_test, y_test, test_size=0.5, random_state=42)
```

VIII. Rancangan Model

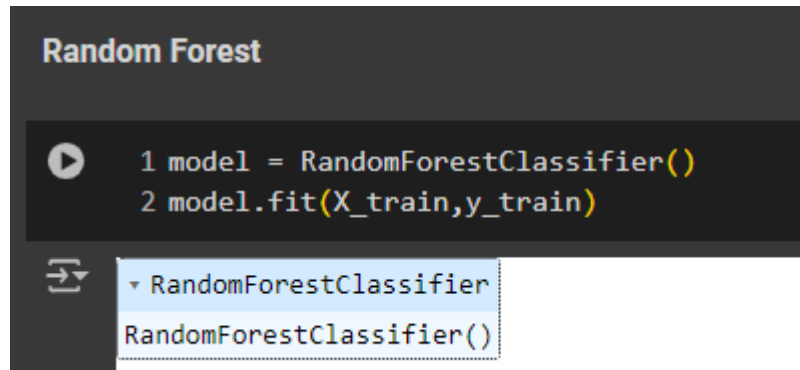
Model XGBoost



Gambar model XGBoost

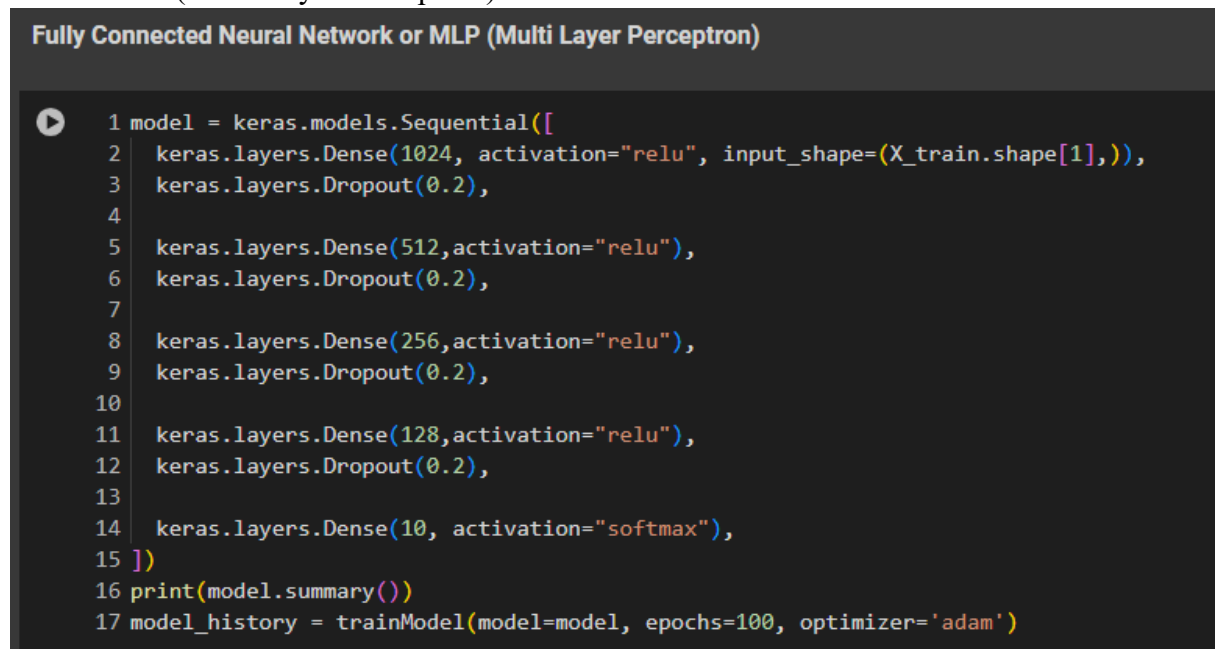
Untuk model XGBoost yang digunakan hanya menggunakan parameter `n_estimators` sebesar 1000

Model Random Forest Classifier



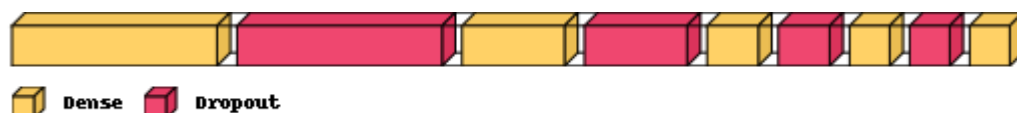
Gambar model Random Forest Classifier

Model MLP (Multi Layer Perceptron)



Gambar model Multi Layer Perceptron

Pada model MLP, kami menggunakan activation ReLu untuk input sampai hidden layer, sedangkan untuk output layer menggunakan activation softmax, karena outputnya adalah genre, yang mana pada dataset terdapat 10 genre. Bagian Dropout pada model digunakan untuk mencegah terjadinya overfitting.



Gambar arsitektur model MLP

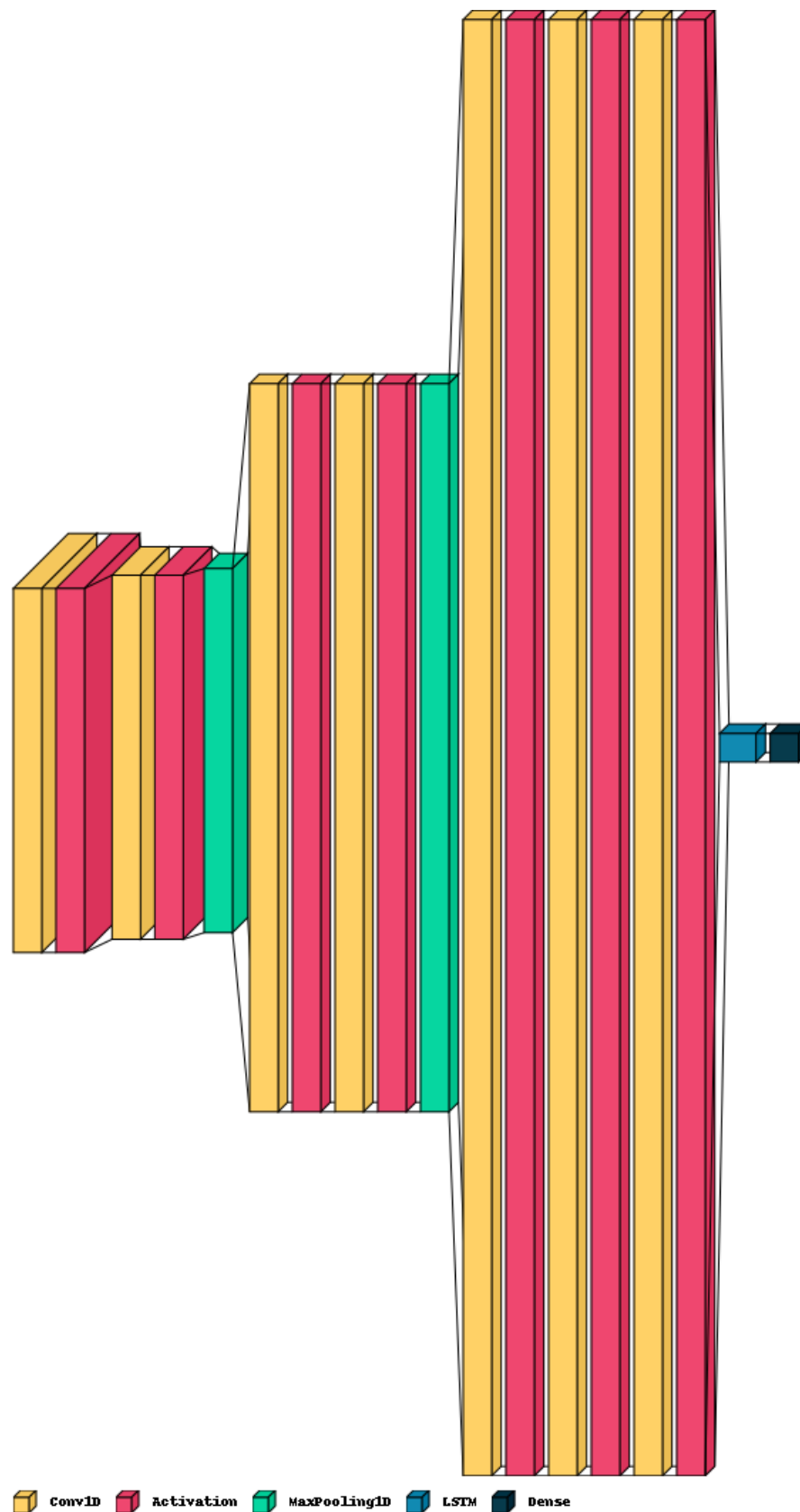
Model CNN-LSTM

```
CNN-LSTM Architecture

1 lstm_model = keras.models.Sequential([
2     keras.layers.Input(shape=(58, 1,)),
3
4     keras.layers.Conv1D(filters=64, kernel_size=3, strides=1, padding='same'),
5     keras.layers.Activation('relu'),
6     keras.layers.Conv1D(filters=64, kernel_size=3, strides=1, padding='same'),
7     keras.layers.Activation('relu'),
8     keras.layers.MaxPooling1D(pool_size=2, strides=2, padding='same'),
9
10    keras.layers.Conv1D(filters=128, kernel_size=3, strides=1, padding='same'),
11    keras.layers.Activation('relu'),
12    keras.layers.Conv1D(filters=128, kernel_size=3, strides=1, padding='same'),
13    keras.layers.Activation('relu'),
14    keras.layers.MaxPooling1D(pool_size=2, strides=2, padding='same'),
15
16    keras.layers.Conv1D(filters=256, kernel_size=3, strides=1, padding='same'),
17    keras.layers.Activation('relu'),
18    keras.layers.Conv1D(filters=256, kernel_size=3, strides=1, padding='same'),
19    keras.layers.Activation('relu'),
20    keras.layers.Conv1D(filters=256, kernel_size=3, strides=1, padding='same'),
21    keras.layers.Activation('relu'),
22
23    keras.layers.LSTM(units=256),
24
25    keras.layers.Dense(units=10, activation='softmax'),
26 ])
27 print(lstm_model.summary())
28 model_history = trainModel(model=lstm_model, epochs=100, optimizer='adam')
```

Gambar model CNN-LSTM

Model ini memiliki lapisan convolution yaitu pada Conv1D dengan filter sebanyak 64 dan kernel sebesar 3. Activation yang digunakan juga sama seperti sebelumnya yaitu menggunakan ReLu sedangkan untuk outputnya menggunakan activation softmax. Convolutional layer yang terdapat pada model berfungsi untuk mengekstraksi fitur dari data spasial. Sedangkan untuk maxpooling yang terdapat pada model berfungsi untuk mereduksi dimensi spasial dari fitur yang telah diekstraksi dari lapisan convolution. Pada model ini juga menggunakan LSTM layer dengan 256 unit. Fungsi dari LSTM layer ini adalah untuk menangkap atau mengenali pola antar data.



Berikut adalah gambar arsitektur model CNN-LSTM

IX. Proses Pelatihan Model

Parameter model deep learning:

Hyperparameter	Settings
Batch size	64
Optimizer	Adam
Metrics	Accuracy
Epochs	100
Loss	Sparse Categorical Cross Entropy

Parameter tersebut adalah parameter yang digunakan untuk model MLP dan LSTM-CNN

```

1 # The loss is calculated using sparse_categorical_crossentropy function
2 def trainModel(model, epochs, optimizer):
3     batch_size = 64
4     model.compile(optimizer=optimizer,
5                   loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
6                   metrics='accuracy'
7     )
8     return model.fit(X_train, y_train, validation_data=(X_val, y_val), epochs=epochs,
9                     batch_size=batch_size)

```

Gambar setting hyperparameter deep learning model.

Parameter tersebut digunakan untuk kedua dataset baik dataset 30 detik dan 3 detik.

X. Hasil Evaluasi dan Analisis

Dataset 3 detik

XGBoost

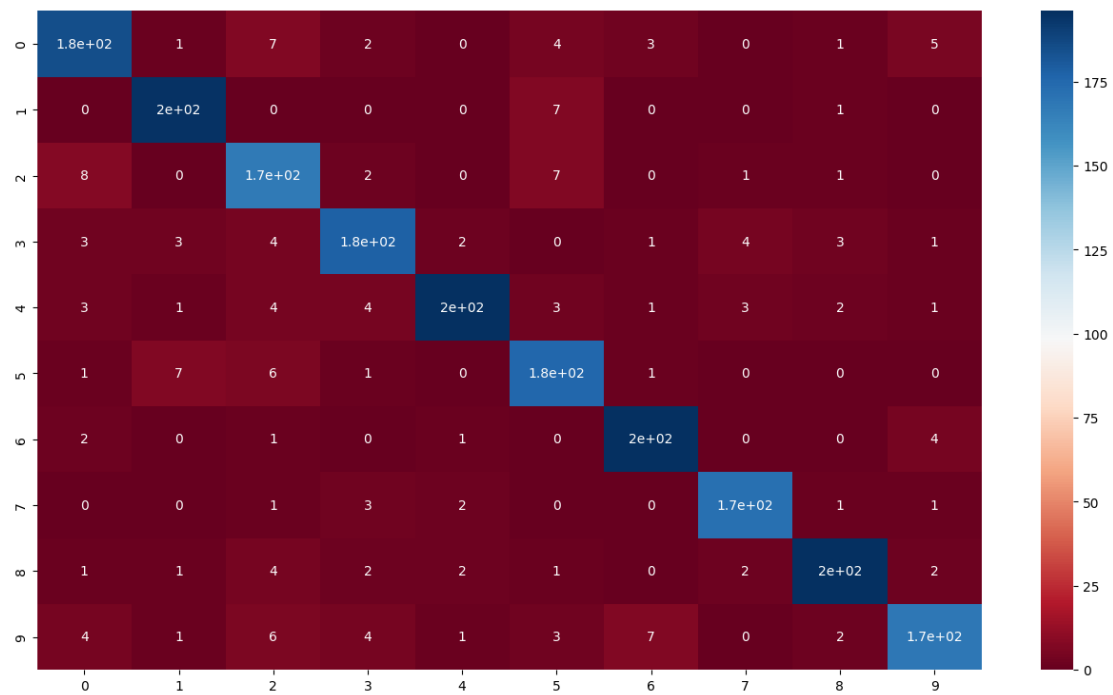
```

Testing accuracy: 0.9159159159159159
Testing:

```

	precision	recall	f1-score	support
0	0.89	0.89	0.89	208
1	0.93	0.96	0.95	203
2	0.83	0.90	0.87	186
3	0.91	0.89	0.90	199
4	0.96	0.90	0.93	218
5	0.88	0.92	0.90	192
6	0.94	0.96	0.95	204
7	0.95	0.96	0.95	180
8	0.95	0.93	0.94	211
9	0.92	0.86	0.89	197
accuracy			0.92	1998
macro avg	0.92	0.92	0.92	1998
weighted avg	0.92	0.92	0.92	1998

Gambar hasil classification report model XGBoost

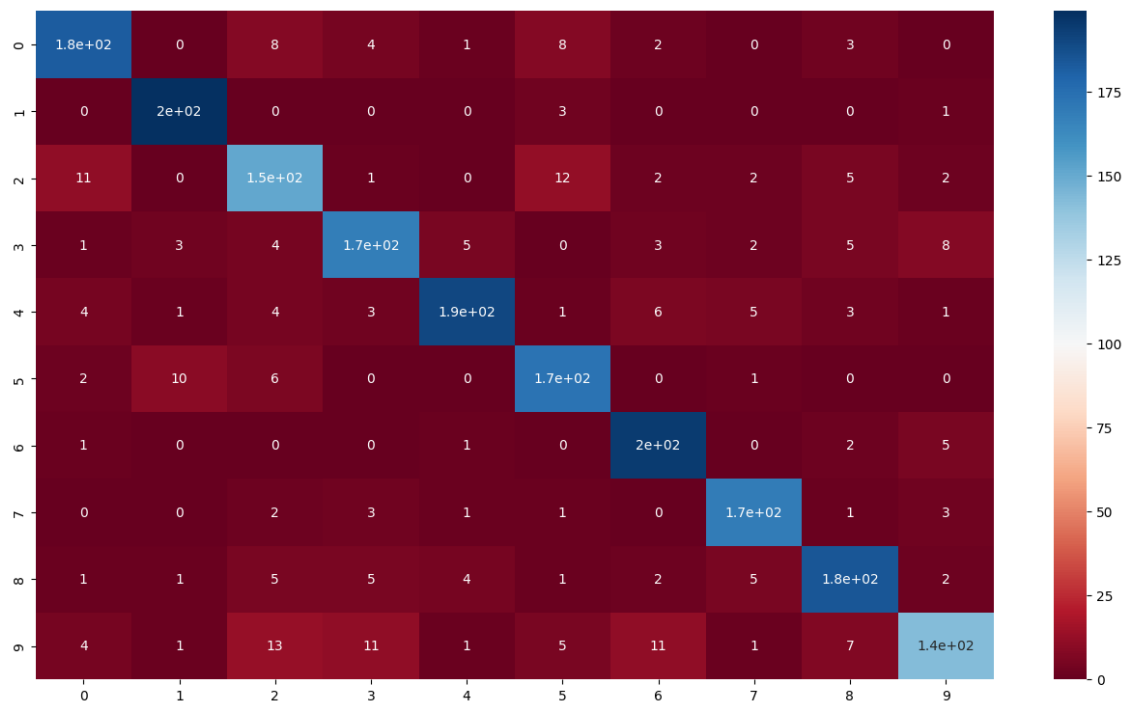


Gambar confusion matrix untuk model XGBoost

Random Forest Classifier

Testing accuracy: 0.8783783783783784					
Testing:					
	precision	recall	f1-score	support	
0	0.88	0.88	0.88	208	
1	0.93	0.98	0.95	203	
2	0.78	0.81	0.80	186	
3	0.86	0.84	0.85	199	
4	0.94	0.87	0.90	218	
5	0.85	0.90	0.87	192	
6	0.88	0.96	0.92	204	
7	0.91	0.94	0.93	180	
8	0.88	0.88	0.88	211	
9	0.87	0.73	0.79	197	
accuracy			0.88	1998	
macro avg	0.88	0.88	0.88	1998	
weighted avg	0.88	0.88	0.88	1998	

Gambar classification report untuk model Random Forest Classifier

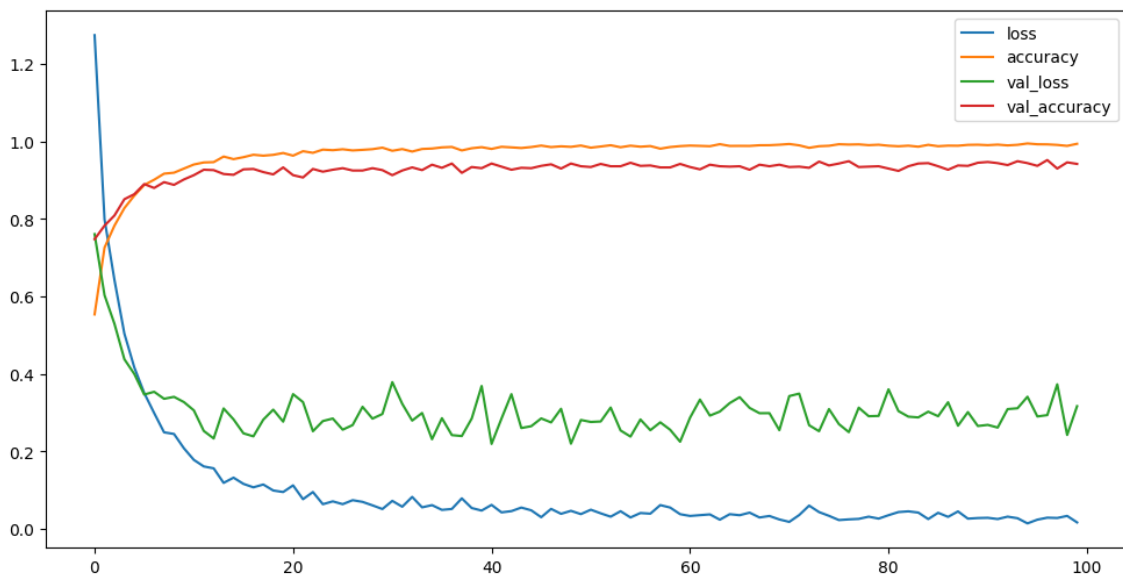


Gambar classification report untuk model Random Forest Classifier

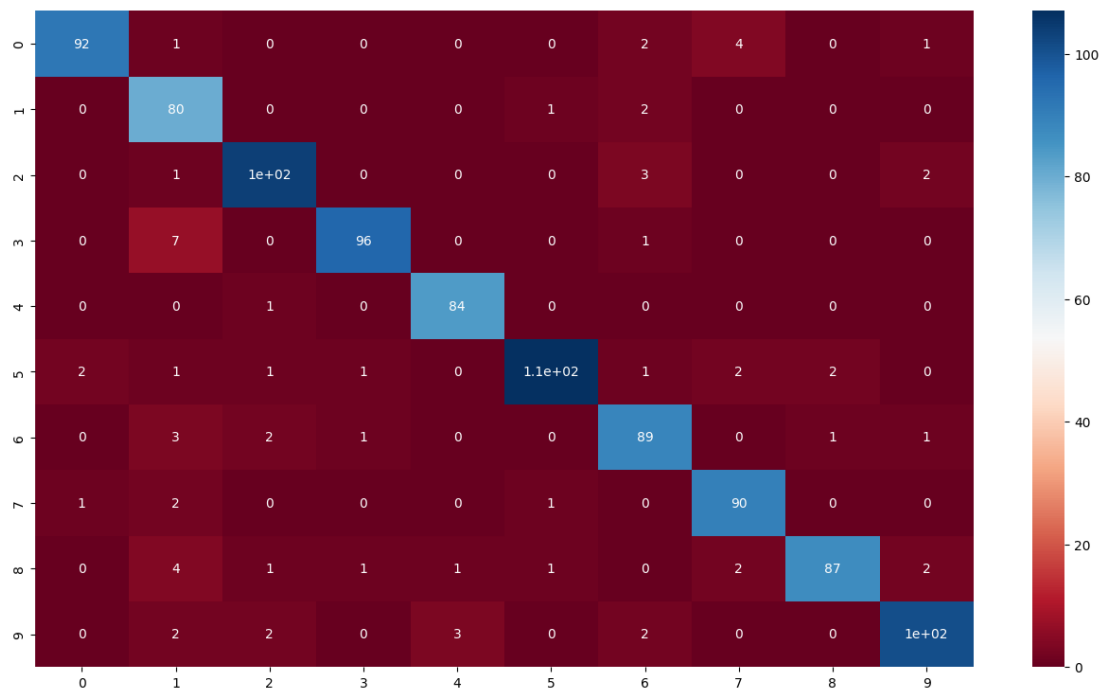
Multi Layer Perceptron (MLP)

```
16/16 [=====] - 0s 2ms/step - loss: 0.4081 - accuracy: 0.9309
The test loss is : 0.4081197679042816
The test Accuracy is : 93.09309124946594
```

Gambar accuracy model MLP



Gambar loss & accuracy curves untuk training dan validation

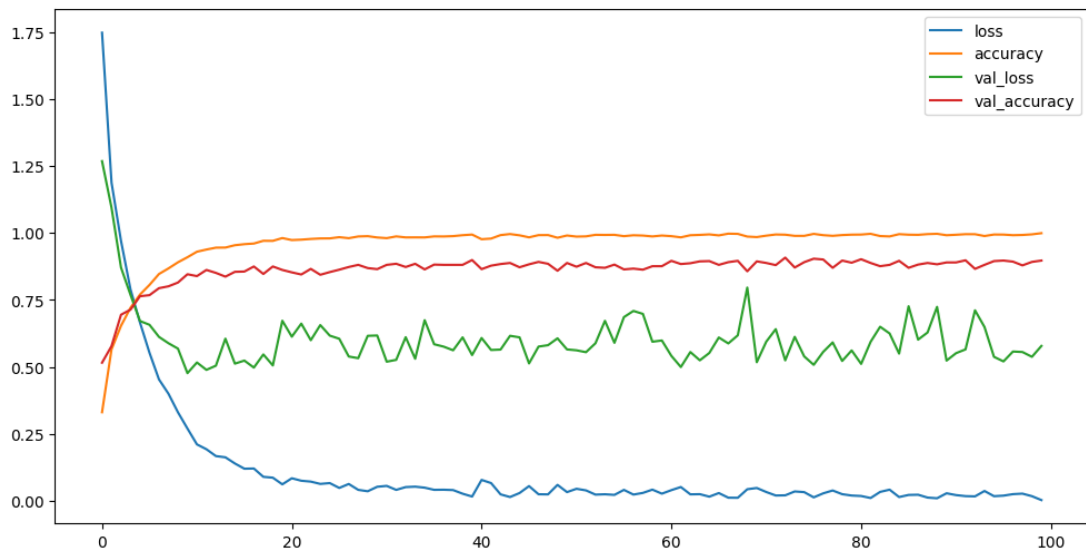


Confusion Matrix untuk model MLP

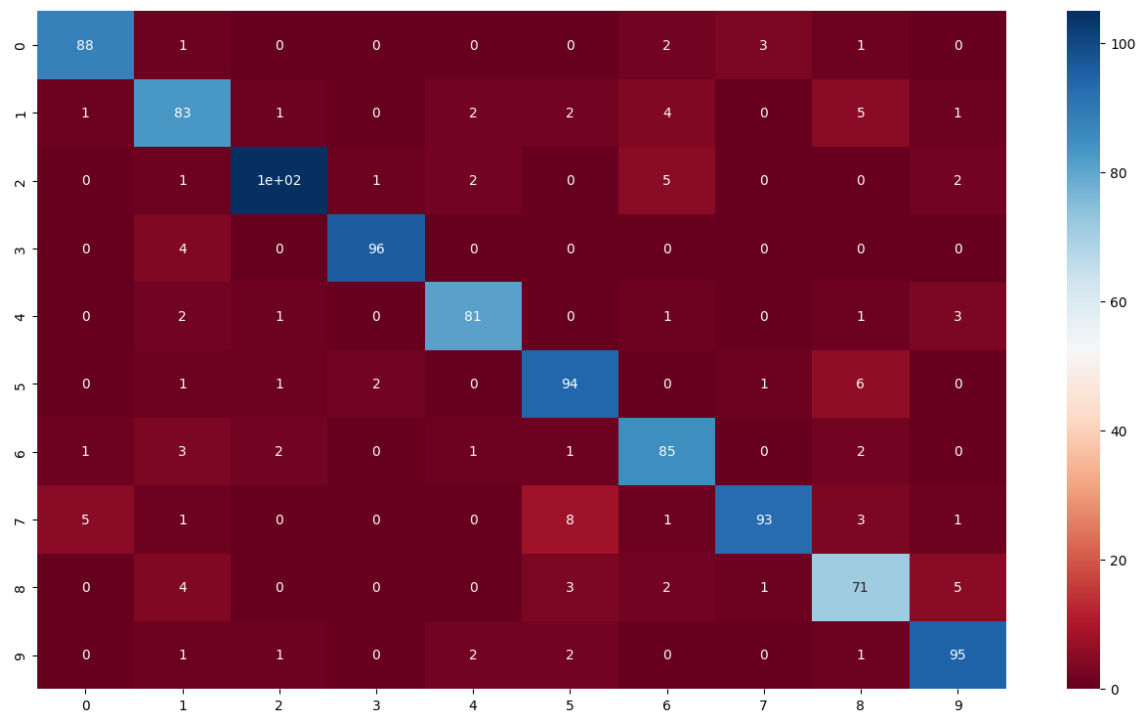
LSTM – CNN

```
16/16 [=====] - 0s 4ms/step - loss: 0.6818 - accuracy: 0.8919
The test loss is : 0.6818191409111023
The test Accuracy is : 89.18918967247009
```

Gambar accuracy model LSTM-CNN



Gambar loss & accuracy curves untuk training dan validation



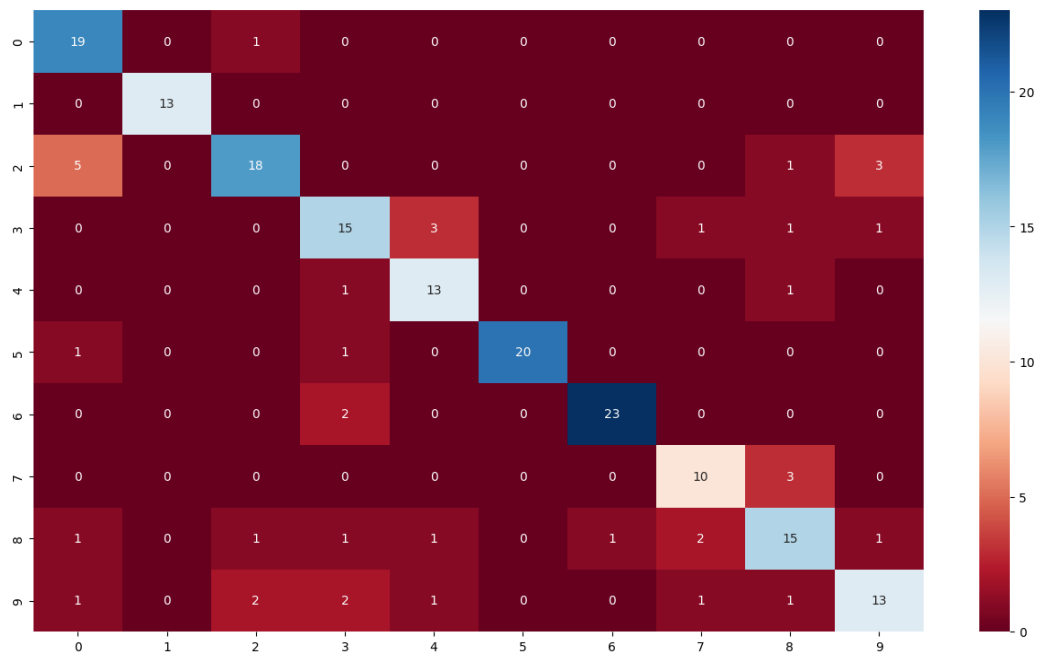
Confusion Matrix untuk model LSTM-CNN

Dataset 30 detik

XGBoost

Testing accuracy: 0.795				
Testing:				
	precision	recall	f1-score	support
0	0.70	0.95	0.81	20
1	1.00	1.00	1.00	13
2	0.82	0.67	0.73	27
3	0.68	0.71	0.70	21
4	0.72	0.87	0.79	15
5	1.00	0.91	0.95	22
6	0.96	0.92	0.94	25
7	0.71	0.77	0.74	13
8	0.68	0.65	0.67	23
9	0.72	0.62	0.67	21
accuracy			0.80	200
macro avg	0.80	0.81	0.80	200
weighted avg	0.80	0.80	0.79	200

Gambar classification report XGBoost



Gambar confusion matrix XGBoost

Random Forest Classifier

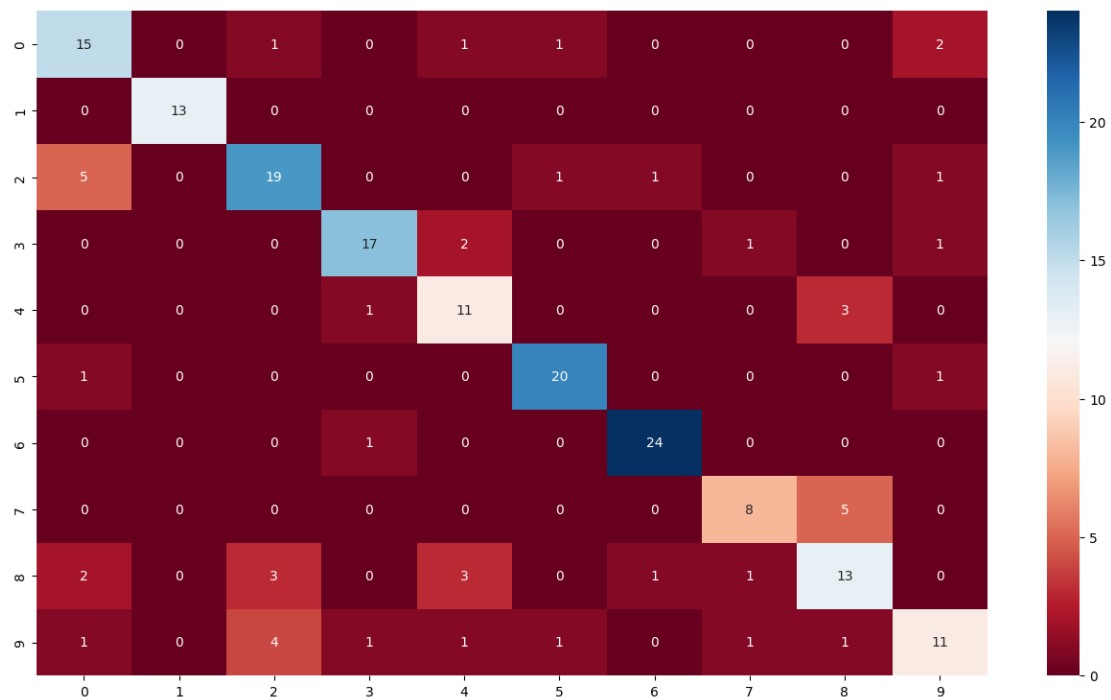
```

Testing accuracy: 0.755
Testing:

```

	precision	recall	f1-score	support
0	0.62	0.75	0.68	20
1	1.00	1.00	1.00	13
2	0.70	0.70	0.70	27
3	0.85	0.81	0.83	21
4	0.61	0.73	0.67	15
5	0.87	0.91	0.89	22
6	0.92	0.96	0.94	25
7	0.73	0.62	0.67	13
8	0.59	0.57	0.58	23
9	0.69	0.52	0.59	21
accuracy			0.76	200
macro avg	0.76	0.76	0.76	200
weighted avg	0.76	0.76	0.75	200

Gambar classification report Random Forest Classifier

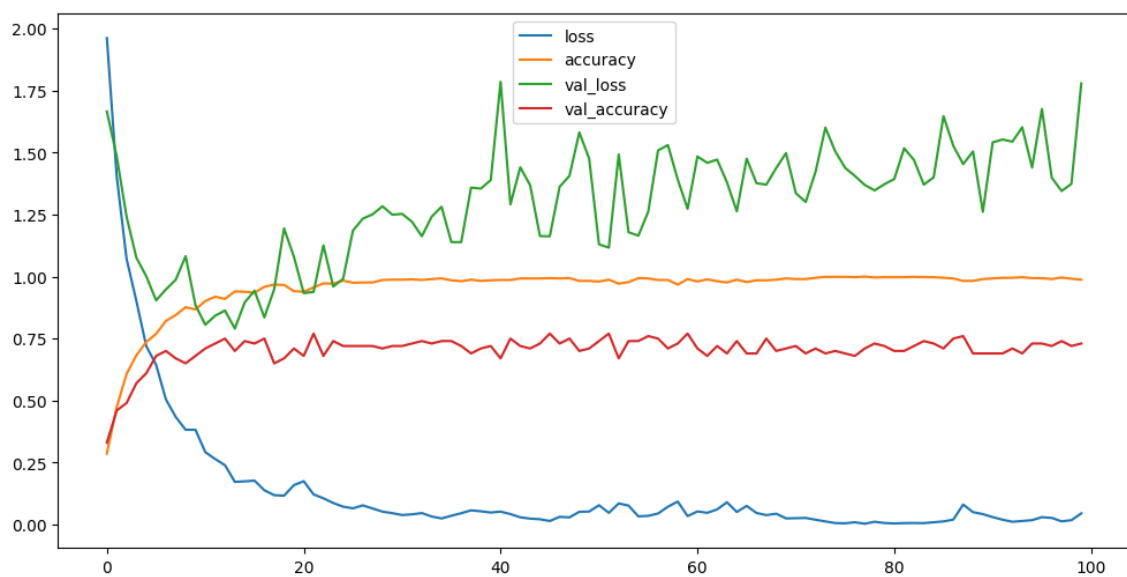


Gambar confusion matrix Random Forest Classifier

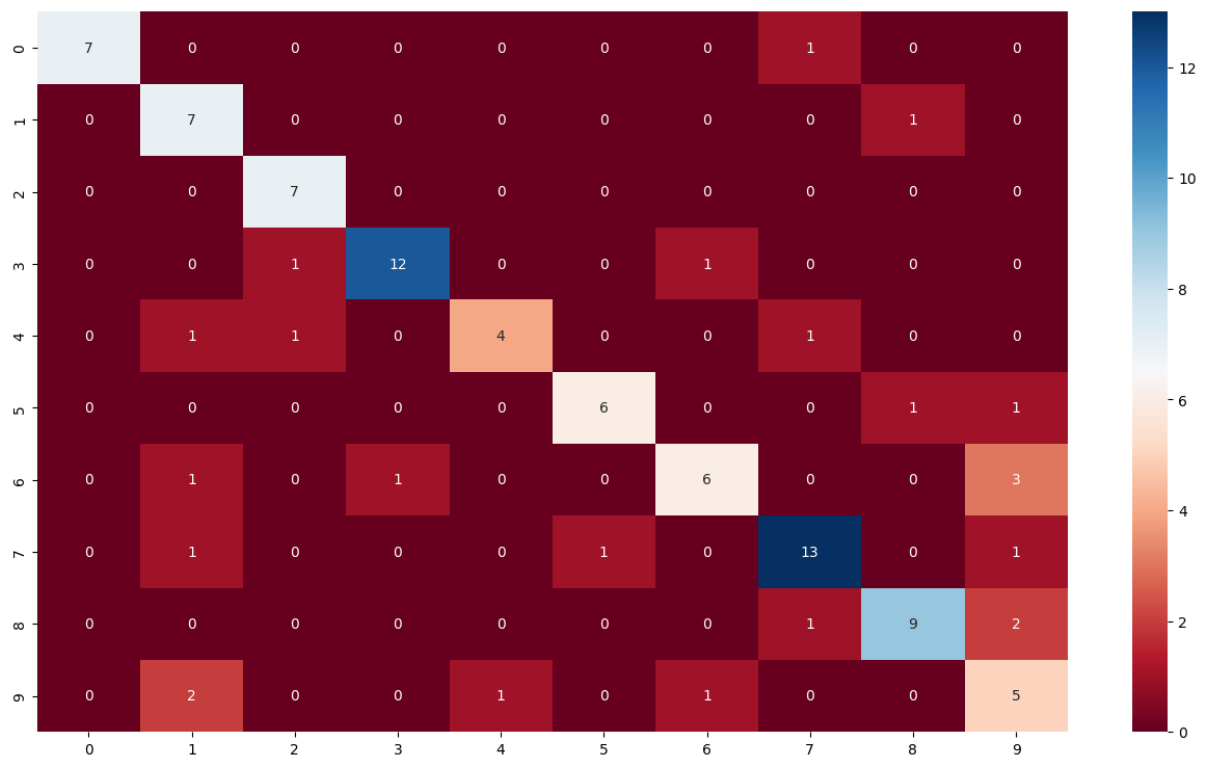
Multi Layer Perceptron (MLP)

```
2/2 [=====] - 0s 6ms/step - loss: 1.9290 - accuracy: 0.7600
The test loss is : 1.9290298223495483
The test Accuracy is : 75.99999904632568
```

Gambar accuracy model MLP



Gambar loss and accuracy untuk training and validation MLP



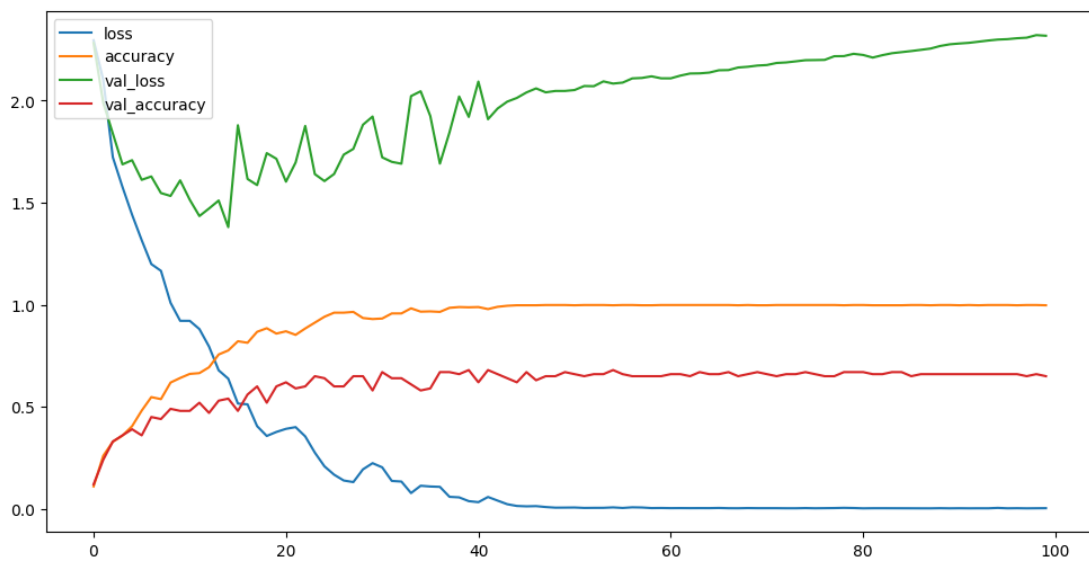
Gambar confusion matrix model MLP

LSTM - CNN

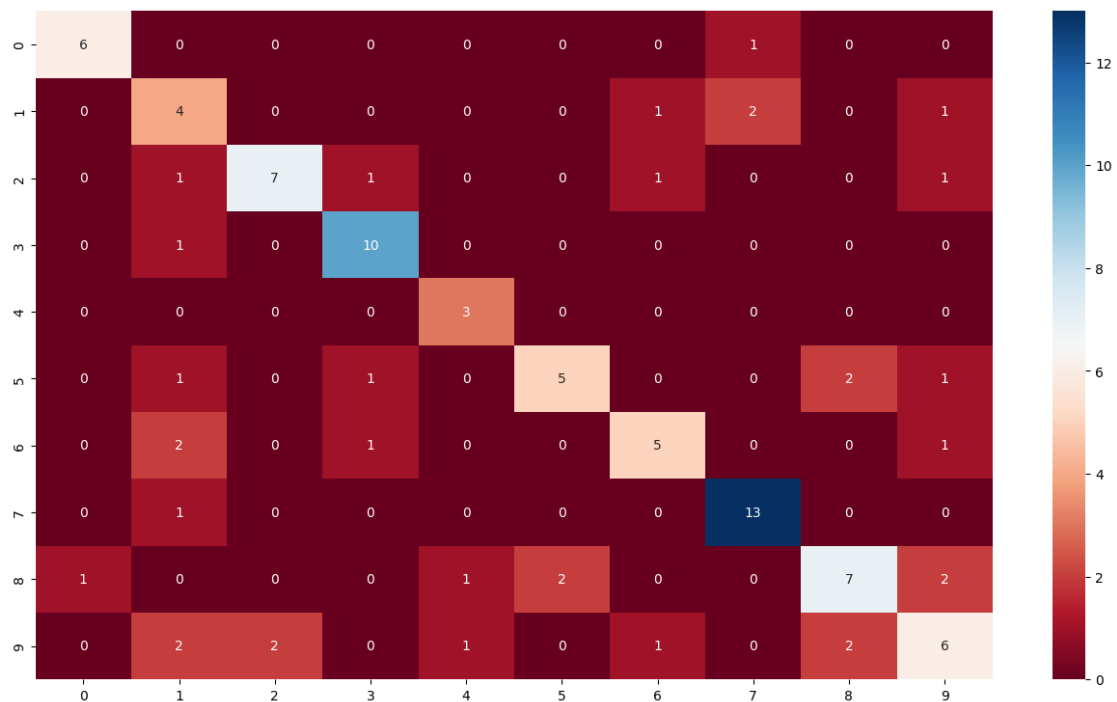
```
2/2 [=====] - 0s 8ms/step - loss: 2.4314 - accuracy: 0.6600
The test loss is : 2.431408882141133

The test Accuracy is : 66.00000262260437
```

Gambar accuracy LSTM-CNN



Gambar loss and accuracy untuk training and validation LSTM-CNN



Gambar confusion matrix model LSTM-CNN

Dataset 3 detik

Model Name	Accuracy
XGBoost	91.5%
Random Forest Classifier	87.8%
Multi Layer Perceptron (MLP)	93.1%
LSTM - CNN	89.2%

Dataset 30 detik

Model Name	Accuracy
XGBoost	79.5%
Random Forest Classifier	75.5%
Multi Layer Perceptron (MLP)	75.9%
LSTM - CNN	66%

Dari hasil akurasi yang didapatkan, model MLP memiliki kinerja yang paling bagus diantara semua model yang telah dibuat, akurasi 93.1% untuk model MLP didapatkan dengan dataset 3 detik. Namun, jika kita menggunakan dataset 30 detik dapat dilihat bahwa akurasi yang didapatkan setiap model tidak maksimal. Bahkan model LSTM-CNN yang sebelumnya mendapatkan akurasi yang cukup baik yaitu 89.2% pada dataset 30 detik turun menjadi 66%. Model LSTM-CNN dengan dataset 30 detik menjadi model dengan akurasi terburuk.

Untuk dataset 3 detik model yang mendapatkan hasil terburuk adalah model Random Forest Classifier dengan akurasi 87.8% sedangkan yang terbaik adalah MLP dengan akurasi 93.1%. Sedangkan pada dataset 30 detik hasil terbaik didapatkan model XGBoost

dengan akurasi 79.5% dan hasil terburuk didapatkan oleh model LSTM-CNN dengan akurasi 66%. Namun, model machine learning dan deep learning memiliki perbedaan pada splitting datasetnya. Untuk model machine learning XGBoost dan Random Forest Classifier dataset dibagi menjadi 80:20, sedangkan untuk model deep learning MLP dan LSTM-CNN dataset dibagi menjadi 80:10:10.

Membagi dataset 30 detik ke dalam 10 segmen memberikan hasil yang signifikan pada model yang dibuat, hal ini disebabkan karena fitur yang diambil bisa menjadi lebih konsisten. Satu lagu dapat memiliki dinamika dan tempo yang berbeda-beda, sehingga hal ini dapat membuat model machine learning atau deep learning menjadi lebih sulit mengenali musik atau lagu. Beda halnya dengan membagi lagu tersebut ke dalam beberapa segmen, lagu yang telah dibagi ke dalam beberapa segmen dapat dengan lebih mudah dikenali dan dipahami oleh model machine learning dan deep learning, karena perubahan pada dinamika dan tempo yang terjadi akan lebih sedikit.

XI. Kesimpulan dan Saran

Dapat disimpulkan bahwa, membagi dataset ke dalam beberapa segmen dapat meningkatkan akurasi semua model secara signifikan. Hasil akurasi dan performa terbaik untuk tugas klasifikasi genre musik adalah model Multi Layer Perceptron (MLP). Kedepannya, dapat dicoba model-model lainnya selain model yang sudah digunakan saat ini. Kemudian, parameter pada model machine learning yang digunakan baik XGBoost dan Random Forest Classifier juga bisa diubah agar mendapatkan akurasi yang lebih maksimal lagi.

XII. Lampiran

Link feature extraction 30 detik:

<https://colab.research.google.com/drive/1fbGsg6n3NT4g9NEyi36527pRi2V9kKkq?usp=sharing>

Link feature extraction 3 detik:

<https://colab.research.google.com/drive/1qTNfMY81AD1OvDoFnS0hNywYPGMvk8RR?usp=sharing>

Link model:

https://colab.research.google.com/drive/1HU8jOTpDBh_8WTzgxsCKT0qqC53Y-D9Q?usp=sharing

Link dataset:

<https://www.kaggle.com/datasets/andradaolteanu/gtzan-dataset-music-genre-classification>

** Berikut adalah link dataset yang kami gunakan pada google colab kami, sumbernya dari link diatas.

<https://drive.google.com/file/d/13wwFF1alQeoGvv7MFyDMRlDbTedgJUin/view?usp=sharing>

Link dataset yang sudah diekstraksi (30 detik):

https://drive.google.com/file/d/12zp_vICcuIN4N8wthaeW5ijoKFQVslc3/view?usp=sharing

Link dataset yang sudah diekstraksi (3 detik):

<https://drive.google.com/file/d/17jeQXivuloPTOS2bG6c64Dmj9ahNhb9i/view?usp=sharing>