BAB 2

Metode Penelitian

2.1. Sumber Dataset

Dalam menentuka klasifikasi genre musik tentunya dibutuhkan data musik dan genrenya sebagai dasar penentuan genre yang akan dilakukan, dataset yang digunakan adalah dataset musik yang berasal dari GTZAN dataset music genre classification yang terdapat dalam situs kaggle.com.

2.2. Peralatan Penelitian

2.2.1. Perangkat Keras

Perangkat keras yang digunakan untuk melakukan percobaan adalah sebagai berikut:

Laptop : ACER Aspire 5 A514-54
CPU : 11th Gen Intel(R) Core(TM) i5-1135G7 10nm
@ 2.40GHz 2.42GHz
Max TDP 28 Watt, L3 cache 8MB.
GPU : Intel(R) Iris(R) Xe Graphics (integrated)
RAM : DDR4 8GB SODIMM 2667 Mhz
SSD : M2 NVMe KINGSTON OM8PCP3512F-AA
512GB PCIe Gen3 Speed Read 1,9GB/s. Write 0.78GB/s

Dengan adanya informasi perangkat keras ini dapat menjadi pertimbangan mengenai performa yang akan dihasilkan, karena spesifikasi perangkat keras akan memengaruhi performa sehingga tidak dapat dibandingkan pada perangkat keras yang memiliki spesifikasi yang berbeda.

2.2.2. Perangkat Lunak

Berikut adalah perangkat lunak yang digunakan untuk melakukan percobaan:

Perangkat Lunak	Versi
Visual Studio Code	1.62.3
Python	3.9.2

Percobaan dilakukan dengan menggunakan beberapa paket (*package*) non-default dari bahasa pemrograman Python sehingga harus diinstal terlebih dahulu menggunakan pengelola package (*package manager*) python yang bernama pip, berikut adalah daftar paket non-default yang digunakan:

Paket Python	Versi
Tensorflow	2.7.0
Scikit Learn	0.24.2
Numpy	1.20.2

Matplotlib	3.4.2
Librosa	0.8.1

Berikut adalah penjelasan singkat mengenai beberapa paket yang digunakan:

a. Tensorflow

Berdasarkan halaman resminya TensorFlow adalah platform open-source untuk pembelajaran mesin (*Machine Learning*). Tensorflow memiliki ekosistem alat, pustaka, dan sumber daya komunitas yang komprehensif dan fleksibel yang memungkinkan peneliti mendorong ML mutakhir dan pengembang dengan mudah membangun dan menerapkan aplikasi yang didukung ML.

b. Scikit Learn

Berdasarkan halaman resminya Scikit learn adalah alat yang simple dan efisien untuk melakukan analisis data seperti melakukan klasifikasi, regresi, klasterisasi dan lain-lain. Scikit learn merupakan paket open-source yang dibuat dari paket python lain seperti Numpy untuk membuat array, scipy untuk perhitungan statistik, dan matplotlib untuk visualisasi data.

c. Librosa

Librosa adalah paket python untuk analisis musik dan audio. Librosa menyediakan fungsi-fungsi dan algoritma yang diperlukan untuk membuat sistem pencarian informasi musik. Librosa sangat mempermudah untuk melakukan analisis audio dengan algoritma yang lengkap untuk pemprosesan berkas audio.

2.3. Tahapan Penelitian

1. Akuisisi Data

Akuisisi Data merupakan proses awal yang dilakukan dalam penelitian. Akuisisi data dapat diartikan sebagai proses pengumpulan data. Dimana data-data tersebut nantinya akan digunakan sebagai dasar dilakukannya penelitian. Pengumpulan data dapat dilakukan dengan berbagai metode, baik melalui observasi lapangan, wawancara, penyebaran kuesioner, ataupun dari data yang sudah tersedia dari halaman resmi penyedia data.

2. Praproses Data

Data yang telah dikumpulkan sebelumnya termasuk kedalam data mentah, sebelum pembuatan model data terlebih dahulu akan ‘dibersihkan’ agar nantinya data siap dipakai untuk proses selanjutnya. Pembersihan yang dimaksud diantaranya adalah proses pemeriksaan missing value, menghapus ataupun mengganti missing value, normalisasi data, serta penyesuaian tipe data.

Data kemudian akan dibagi menjadi 3 bagian yang memiliki fungsi yang berbeda, bagian pertama adalah data tes untuk melatih model, bagian kedua adalah data evaluasi untuk mengevaluasi model yang sudah dibuat, dan yang terakhir adalah data uji untuk membandingkan data asli dengan hasil prediksi oleh model yang telah dibuat.

3. Pembuatan Model

Pada penelitian ini pembuatan model dilakukan sebanyak empat kali untuk setiap jenis algoritma yang diuji. Model dibuat dengan menggunakan fungsi-fungsi yang sudah tersedia didalam paket Keras pada Tensorflow berdasarkan data yang sudah dibagi sesuai fungsinya.

4. Perbandingan Model

   Setelah model dibuat kemudian dilakukan prediksi untuk setiap algoritma, selain itu dilakukan pengecekan performa untuk setiap algoritma untuk mengetahui algoritma dengan performa terbaik.

2.4. Landasan Teori

2.4.1. Artificial Intelligence dan Machine Learning

Kecerdasan Buatan merupakan salah satu bidang dalam ilmu komputer yang ditujukan pada pembuatan software dan hardware yang dapat berfungsi sebagai sesuatu yang dapat berpikir seperti manusia (Sunarya, Santoso, & Sentanu, 2015).

Kecerdasan buatan banyak digunakan untuk memecahkan berbagai masalah seperti bisnis (Rahardja, Roihan, & others, 2017), robotika, bahasa alami, matematika, game, persepsi, diagnosis medis, teknik, analisis keuangan, analisis sains, dan penalaran (Russell & Norvig, 2016). Machine learning dapat didefinisikan sebagai aplikasi komputer dan algoritma matematika yang diadopsi dengan cara pembelajaran yang berasal dari data dan menghasilkan prediksi di masa yang akan datang (Goldberg & Holland, 1988).

Adapun proses pembelajaran yang dimaksud adalah suatu usaha dalam memperoleh kecerdasan yang melalui dua tahap antara lain latihan (training) dan pengujian (testing) (Huang, Zhu, & Siew, 2006). Bidang machine learning berkaitan dengan pertanyaan tentang bagaimana membangun program komputer agar meningkat secara otomatis dengan berdasar dari pengalaman (Mitchell, 1997).

Penelitian terkini mengungkapkan bahwa machine learning terbagi menjadi tiga kategori: Supervised Learning, Unsupervised Learning, Reinforcement Learning (Somvanshi & Chavan, 2016).

Teknik yang digunakan oleh Supervised Learning adalah metode klasifikasi di mana kumpulan data sepenuhnya diberikan label untuk mengklasifikasikan kelas yang tidak dikenal. Sedangkan teknik Unsupervised Learning sering disebut cluster dikarenakan tidak ada kebutuhan untuk pemberian label dalam kumpulan data dan hasilnya tidak mengidentifikasi contoh di kelas yang telah ditentukan (Thupae, Isong, Gasela, & AbuMahfouz, 2018). Sedangkan Reinforcement Learning biasanya berada antara Supervised Learning dan Unsupervised Learning (Board, 2017), teknik ini bekerja dalam lingkungan yang dinamis di mana konsepnya harus menyelesaikan tujuan tanpa adanya pemberitahuan dari komputer secara eksplisit jika tujuan tersebut telah tercapai (Das & Nene, 2017). (Roihan, Sunarya, & Rafika, 2019)

2.4.2. Speech Recognition

Speech recognition merupakan proses identifikasi suara berdasarkan kata yang diucapkan. Parameter yang dibandingkan ialah tingkat penekanan suara yang kemudian akan dicocokkan dengan template database yang tersedia. Sistem pengenalan suara berdasarkan orang yang berbicara inilah yang dinamakan speaker recognition.

Speech recognition (juga dikenal sebagai pengenalan suara otomatis, pengenalan komputer pidato, pidato ke teks) mengkonversi kata yang diucapkan dengan teks. Para "Voice recognition" istilah kadang-kadang digunakan untuk merujuk pada sistem pengenalan yang harus dilatih untuk pembicara tertentu seperti halnya bagi sebagian besar perangkat lunak pengenal desktop. Menyadari pembicara dapat menyederhanakan tugas menerjemahkan pidato. Speech recognition adalah solusi yang lebih luas yang mengacu pada teknologi yang dapat mengenali pidato tanpa ditargetkan pada satu pembicara seperti panggilan sistem yang dapat mengenali suara sewenang-wenangnya. (Nada, Ridhuandi, Santoso, & Apriyanto, 2019)

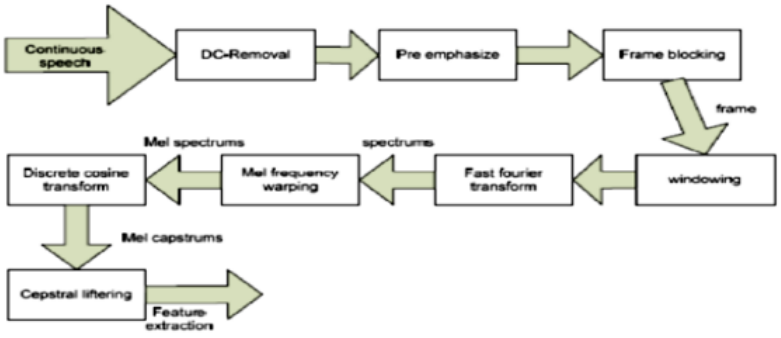
2.4.3. Fast Fourier Transform

FFT adalah suatu algoritma untuk menghitung transformasi fourier diskrit dengan cepat dan efisien. Karena banyak sinyal-sinyal dalam sistem komunikasi yang bersifat kontinyu, sehingga untuk kasus sinyal kontinyu kita gunakan transformasi fourier.

Algoritma FFT berdasarkan atas prinsip pokok dekomposisi perhitungan discrete fourier transform dari suatu sekuen sepanjang N kedalam transformasi diskrit fourier secara berturut-turut lebih kecil. Cara prinsip ini diterapkan memimpin ke arah suatu variasi dari algortima yang berbeda, di mana semuanya memperbandingkan peningkatan kecepatan perhitungan. (Nada, Ridhuandi, Santoso, & Apriyanto, 2019)

2.4.4. Mel-Frequency Ceptrum Coefficient

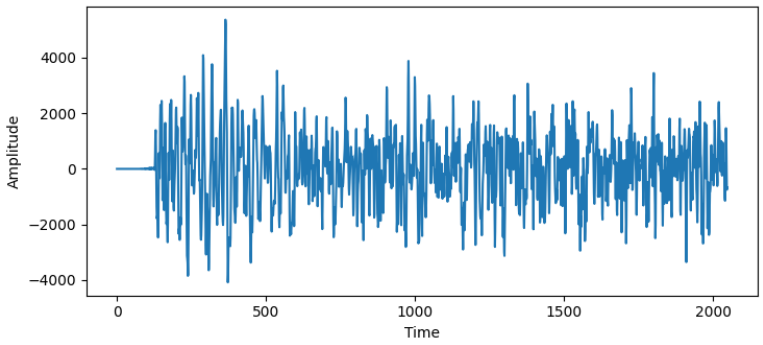
Metode MFCC adalah salah satu metode yang digunakan untuk melakukan feature extraction (ekstraksi ciri), sebuah proses yang mengkorversikan sinyal suara menjadi beberapa parameter. Adapun tahap-tahap pengolahan suara menggunakan metode MFCC ini dapat dilihat gambar berikut.



Berikut adalah tahapan-tahapan MFCC:

1. Input suara dari speaker

   Dalam tahap pertama ini suara diinput ke komputer melalui speaker kemudian dapat disimpan dalam format .wav untuk memudahkan membaca data dari gelombang menjadi format angka. Berikut contoh hasil visualisasi gelombah suara selama 2 detik:



2. DCRemoval

   Tahap ini berfungsi menghitung nilai rata-rata dari data sample suara, dan mengurangkannya untuk setiap sample pada window. Tujuannya adalah untuk memperoleh normalisasi dari data suara input.

   Rumus:

$$D[i] = s[i] - \frac{\sum_{i=1}^{n} s[i]}{n}$$

Keterangan:

D[i] = hasil signal ke –i setelah dilakukan DC Removal
S[i] = signal awal ke-i
N = jumlah sample, n > 0

3. Pre-emphasis (penekanan)

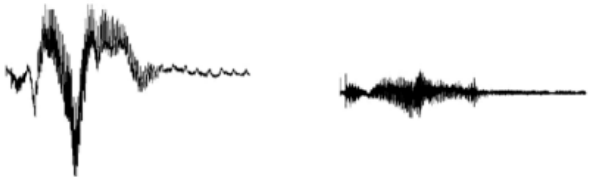
   Merupakan salah satu jenis filter yang sering digunakan sebelum sebuah signal diproses lebih lanjut. Filter ini mempertahankan frekuensi-frekuensi tinggi pada sebuah spektrum, yang umumnya tereliminasi pada saat proses produksi suara.

   Rumus:

$$y(n) = s(n) - \alpha s(n-1)$$

Dimana:
y(n) = Sinyal hasil preemphasis
s(n) = Sinyal sebelum preemphasis

α merupakan konstanta filter preemphasis, biasanya bernilai 0.97. Dalam bentuk dasar operator s sebagai unit filter, persamaan diatas akan memberikan sebuah transfer function filter preemphasis seperti berikut:



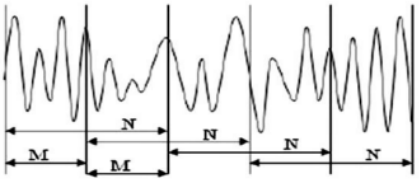
Gambar 2. Sebelum dan sesudah Preemphasis

4. Frame Blocking

   Frame Blocking adalah pembagian suara menjadi beberapa frame yang nantinya dapat memudahkan dalam perhitungan dan analisa suara, satu frame terdiri dari beberapa sampel tergantung tiap berapa detik suara akan disampel dan berapa besar frekuensi samplingnya. Panjang frame yang digunakan, sangat mempengaruhi keberhasilan dalam analisa spektral.

Jumlah frame = ((I-N)/M)+1).....

Dimana: I = Sample rate,
N= Sample point (Sample rate * waktu framing (s))
M = N/2



Gambar 3. Bentuk sinyal yang di frame blocking

5. Windowing

Suara yang di potong-potong menjadi beberapa frame membuat data suara menjadi discontinue, hal ini mengakibatkan kesalahan data proses fourier transform. Agar tidak terjadi kesalahan data pada proses fourier transform maka sampel suara yang telah dibagi menjadi beberapa frame perlu dijadikan suara kontinu dengan cara mengalikan tiap frame dengan window tertentu.

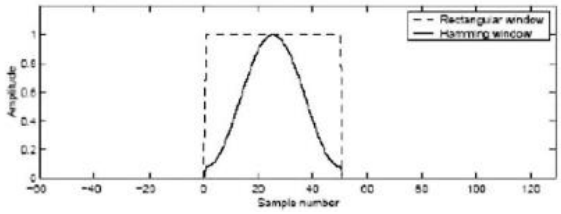
$$x(n) = x\,(n)w(n)$$

Rumus window hamming:

$$\bar{w}[n]= \bar{0}.54-0.46*\cos(2*\pi*n/(N-1))$$

Dimana n = 1,2,3 dan N = panjang frame

Berikut contoh visualisasi window hamming:



6. Fast Fourier Transform

Fast fourier transform adalah suatu metode yang sangat efisien untuk menyelesaikan transformasi fourier diskrit yang banyak dipakai untuk keperluan analisa sinyal seperti pemfilteran, analisa korelasi, dan analisa spektrum. Ada 2 (dua) jenis algoritma FFT yaitu algoritma fast fourier transform decimation in time (FFT DIT) dan algoritma fast fourier transform decimation in frequency (FFT DIF).

Bentuk rumusannya sebagai berikut:

$$X[k]=\sum_{n=1}^{N=1} x(N)w_N^{kn}.$$

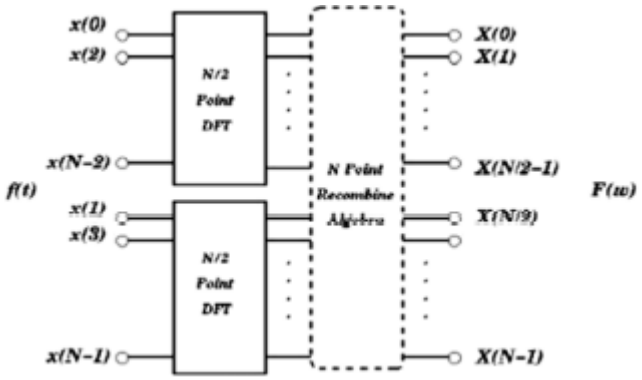
Dimana :
X[k] = Merupakan magnitude frekuensi
x(n) = Nila sampel sinyal
W = cos(2*Pi*k*n/N)-jSin(2*Pi*k*n/N)
N = jumlah sinyal yang akan diproses
k = sinyal yang diproses

Gunakan rumus :

$$|[R^2 + I^2]^{\frac{1}{2}}| \text{ .....}$$

FFT dilakukan dengan membagi N buah titik pada transformasi fourier diskrit menjadi 2, masing-masing (N/2) titik transformasi. Proses memecah

menjadi 2 bagian ini diteruskan dengan membagi (N/2) titik menjadi (N/4) dan seterusnya hingga diperoleh titik minimum.



7. Mel Frequency Wraping

   Dalam mel-frequency wrapping, sinyal hasil FFT dikelompokkan ke dalam berkas filter triangular ini. Proses pengelompokan tersebut adalah setiap nilai FFT dikalikan terhadap filter yang bersesuaian dan hasilnya dijumlahkan. Proses wrapping terhadap sinyal dalam domain frekuensi dilakukan menggunakan persamaan berikut.

$$Y[i]=\sum_{j=1}^{N} s(j).H_i(j) \quad \text{.....}$$

   Dimana: N = jumlah magnitude spectrum
   S[j] = magnitude spectrum pada frekuensi j
   $H_i(j)$ = koefisien *filterbank* pada frekuensi j($1 \leq i \leq M$)
   M= jumlah Channel dalam filterbank

8. Discrete Cosine Transform

   Cepstrum biasa digunakan untuk mendapatkan informasi dari suatu sinyal suara yang diucapkan oleh manusia. Pada tahap terakhir pada MFCC ini, spektrum log mel akan dikonversi menjadi domain waktu menggunakan Discrete Cosine Transform (DCT) menggunakan persamaan berikut.

$$c_j = \sum_{i=1}^{M} X_i.\cos(\frac{j(i-1)}{2}.\frac{\pi}{M}) \quad \text{.....}$$

   Dimana: Ci = nilai koefisien $Ckej$
   j = 1,2,… jumlah koefisien yang diharapkan
   Xi = nilai X hasil *mel-frequency wrapping* pada frekuensi
   i = 1,2,…,n (jumlah wrapping)
   M= jumlah filter
   Hasil dari proses ini dinamakan *Mel-Frequency Cepstrum Coefficients (MFCC).*

9. Cepstral lifting

   Hasil dari fungsi DCT adalah cepstrum yang sebenarnya sudah merupakan hasil akhir dari proses feature extraction. Tetapi, untuk meningkatkan kualitas pengenalan, maka cepstrum hasil dari DCT harus mengalami cepstral liftering.

   Rumus:

$$w[n] = \left\{1 + \frac{L}{2}\sin\left(\frac{n\pi}{L}\right)\right\}$$

   L = jumlah cepstral coefficients
   N = index dari cepstral coefficients

10. Remove Silence

Dengan demikian selesailah proses Mel Frequency Cepstrum Coefficients (MFCC) feature extraction. Namun hasilnya masih mengandung frame-frame silence. Frame semacam ini dapat menggangu pengenalan kata. Oleh karena itu sebaiknya frame-frame ini harus dihilangkan.

$$Noise = \frac{\sum_{i-1}^{n} s[i]}{n} \quad \text{.....}$$

Noise = rata-rata energy frame, S[i] = signal frame ke-i

(Handoko & Kasih, 2018)

2.4.5. Jaringan Saraf Tiruan

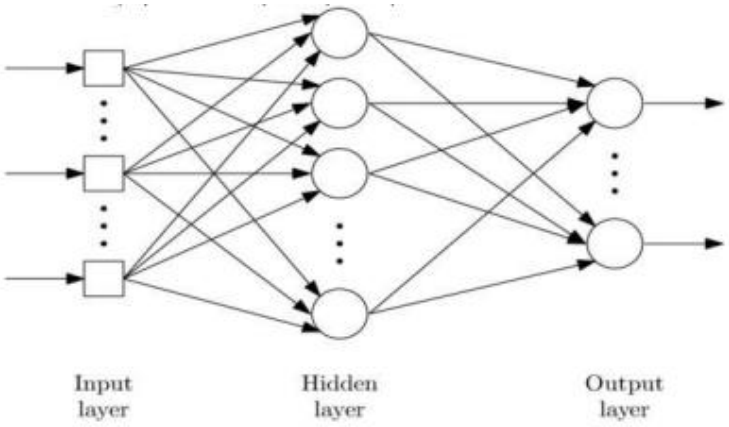
a. Konsep

Jaringan saraf tiruan bisa dibayangkan seperti otak buatan di dalam cerita-cerita fiksi ilmiah. Otak buatan ini dapat berpikir seperti manusia, dan juga sepandai manusia dalam menyimpulkan sesuatu dari potongan-potongan informasi yang diterimanya. Khayalan manusia tersebut mendorong para peneliti untuk mewujudkannya. Komputer diusahakan agar bisa berpikir sama seperti cara berpikir manusia. Caranya adalah dengan melakukan peniruan terhadap aktivitas-aktivitas yang terjadi di dalam sebuah jaringan saraf biologis (Aji , 2016).

Pembagian arsitektur jaringan saraf tiruan bisa dilihat dari kerangka kerja dan skema interkoneksi. Kerangka kerja jaringan saraf tiruan bisa dilihat dari jumlah lapisan (layer) dan jumlah node pada setiap lapisan. Lapisan-lapisan penyusun jaringan saraf tiruan dapat dibagi menjadi tiga, yaitu lapisan input, lapisan tersembunyi, lapisan output:

1. Lapisan input (*input* layer). Node-node di dalam lapisan input disebut unit-unit input. Unit-unit input menerima input dari dunia luar. Input yang dimasukkan merupakan penggambaran suatu masalah.
2. Lapisan tersembunyi (*hidden layer*). Node-node di dalam lapisan tersembunyi disebut unit-unit tersembunyi. Output dari lapisan ini tidak secara langsung dapat diamati.
3. Lapisan output (*output layer)*. Node-node pada lapisan output disebut unit-unit output. Keluaran atau output dari lapisan ini merupakan output jaringan syaraf tiruan terhadap suatu permasalahan.

Gambar dibawah ini merupakan salah satu contoh arsitektur jaringan saraf tiruan multilayer yang terdiri dari sebuah lapisan input, sebuah lapisan tersembunyi, dan sebuah lapisan output. Wij adalah bobot antara lapisan input dengan lapisan tersembunyi, Wjk adalah bobot antara lapisan tersembunyi dengan lapisan output. Sebuah neuron (disebut juga node atau unit) yang terletak di dalam lapisan input akan memiliki fungsi aktivasi dan pola koneksi bobot yang dengan neuron-neuron lainnya yang terletak di dalam lapisan input. Demikian pula halnya sebuah neuron yang terletak di dalam lapisan tersembunyi akan memiliki aktivasi dan pola koneksi bobot yang sama dengan neuron-neuron lainnya yang terletak di dalam lapisan tersembunyi.



b. Fungsi Optimasi

Fungsi Optimasi dalam machine learning adalah fungsi matematika yang digunakan untuk memilih nilai yang dianggap paling optimal dari sejumlah nilai pada proses pembelajaran. Percobaan pada tahap ini dilakukan dengan mengubah ubah parameter fungsi optimasi dan menilai hasilnya.

Salah satu jenis fungsi optimasi adalah Adam, sebuah algoritma untuk optimasi berbasis gradien orde pertama dari fungsi tujuan stokastik, berdasarkan perkiraan adaptif momen orde rendah. Metode ini mudah diterapkan, efisien secara komputasi, memiliki sedikit persyaratan memori, tidak berubah terhadap penskalaan ulang diagonal gradien, dan sangat cocok untuk masalah yang besar dalam hal data dan/atau parameter.

Metode ini juga sesuai untuk tujuan dan masalah non-stasioner dengan gradien yang sangat bising dan/atau jarang. Parameter hiper memiliki interpretasi intuitif dan biasanya memerlukan sedikit penyetelan. Hasil empiris menunjukkan bahwa Adam bekerja dengan baik di berlatih dan membandingkan dengan metode optimasi stokastik lainnya. (Gunawan, 2020)

c. Batch Normalization

Pelatihan Deep Neural Networks diperumit oleh fakta bahwa distribusi input untuk setiap lapisan (*layer*) berubah selama pelatihan, karena parameter lapisan sebelumnya berubah. Hal ini memperlambat pelatihan dengan membutuhkan *learning rate* yang lebih rendah dan penggunaan parameter yang harus tepat, fenomena ini disebut sebagai kovariat internal shift, dan mengatasi masalah dengan menormalkan input lapisan.

Metode kami menarik kekuatannya dari menjadikan normalisasi sebagai bagian dari arsitektur model dan melakukan
normalisasi untuk setiap mini-batch pelatihan. Normalisasi Batch memungkinkan kita untuk menggunakan tingkat pembelajaran yang jauh lebih tinggi dan kurang berhati-hati tentang inisialisasi. Ini juga bertindak sebagai pengatur, dalam beberapa kasus menghilangkan kebutuhan untuk Dropout. Diterapkan pada model klasifikasi gambar mutakhir, Normalisasi Batch mencapai akurasi yang sama dengan 14 kali lebih sedikit langkah pelatihan, dan mengalahkan model aslinya dengan margin yang signifikan.

Menggunakan jaringan yang dinormalisasi batch, meningkatkan hasil pada klasifikasi ImageNet: mencapai 4,9% teratas-5 kesalahan validasi (dan kesalahan uji 4,8%). (Ioffe & Szegedy, 2015)

d. Dropout

Dropout merupakan fungsi matematika yang digunakan untuk menentukan prosentasi pembuangan nilai yang dianggap paling buruk dari sejumlah nilai yang dihasilkan pada satu tahap pembelajaran. Tujuan dari pembuangan ini untuk mengurangi overfitting pada model, yaitu kondisi dimana nilai akurasi yang dihasilkan terlalu sempurna. (Gunawan, 2020)

e. Fungsi Aktivasi

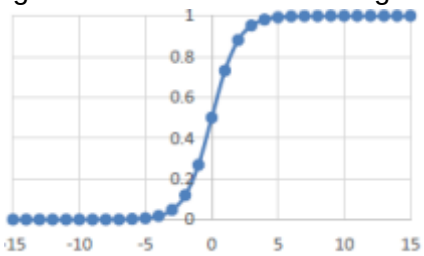
Fungsi aktivasi adalah fungsi non linear yang memungkinkan sebuah JST untuk dapat mentransformasi data input menjadi dimensi yang lebih tinggi sehingga dapat dilakukan pemotongan hyperlane sederhana yang memungkinkan dilakukan klasifikasi. Terdapat banyak fungsi aktivasi yang dapat digunakaan, beberapa diantaranya taitu:

- Sigmoid

  Fungsi sigmoid mentransformasi range nilai dari input x menjadi antara 0 dan 1 dengan rumus:

$$\sigma(x) = \frac{1}{(1 + e^{-x})}$$

  Sehingga fungsi sigmoid memiliki bentuk sebagai berikut:



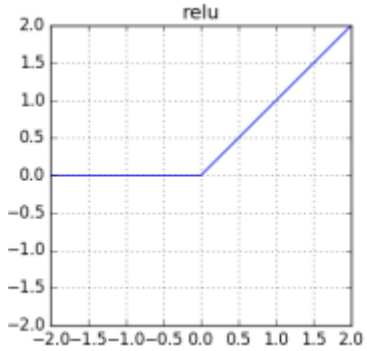
  Fungsi sigmoid sekarang sudah tidak banyak digunakan dalam praktek karena memiliki kelemahan utama yaitu range nilai output dari fungsi sigmoid tidak terpusat pada angka nol.

  Hal tersebut menyebabkan terjadinya proses backpropagation yang tidak ideal, selain itu bobot pada JST tidak terdistribusi rata antara nilai positif dan negatif serta nilai bobot akan banyak mendekati ekstrim 0 atau 1. Dikarenakan komputasi nilai propagasi menggunakan perkalian, maka nilai ekstrim tersebut akan menyebabkan efek saturating gradients dimana jika nilai bobot cukup kecil, maka lama kelamaan nilai bobot akan mendekati salah satu ekstrim sehingga memiliki gradien yang mendekati nol. Jika hal tersebut terjadi, maka neuron tersebut tidak akan dapat mengalami update yang signifikan dan akan nonaktif.

- Relu

  Rectified Linear Unit (ReLU) adalah fungsi aktivasi yang memiliki perhitungan sederhana. Proses forward dan backward melalui ReLU hanya menggunakan kondisi if. Jika elemen bernilai negatif maka nilainya diset menjadi 0, tidak ada operasi eksponensial, perkalian atau pembagian. Dengan karakteristik seperti itu, kelebihan

ReLU akan muncul saat berhadapan dengan jaringan yang memiliki neuron yang banyak sehingga dapat mengurangi waktu training dan testing dengan signifikan.



- Softmax

  Fungsi softmax adalah fungsi yang mengubah vektor nilai real K menjadi vektor nilai real K yang berjumlah 1. Nilai input bisa positif, negatif, nol, atau lebih besar dari satu, tetapi softmax mengubahnya menjadi nilai antara 0 dan 1, sehingga dapat diinterpretasikan sebagai probabilitas. Jika salah satu inputnya kecil atau negatif, softmax mengubahnya menjadi probabilitas kecil, dan jika inputnya besar, maka itu mengubahnya menjadi probabilitas besar, tetapi akan selalu tetap antara 0 dan 1.

  Rumus softmax:

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

  Keterangan:

  e^{z_i} = fungsi eksponensial yang diaplikasikan ke semua input vektor

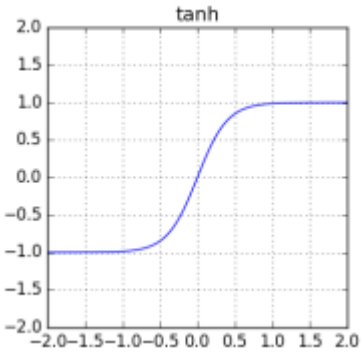
  K = jumlah kelas yang akan diklasifikasikan

  Banyak jaringan saraf multi-layer berakhir di lapisan kedua dari belakang yang menghasilkan skor bernilai nyata yang tidak mudah diskalakan dan yang mungkin sulit untuk dikerjakan. Di sini softmax sangat berguna karena mengubah skor menjadi distribusi probabilitas yang dinormalisasi, yang dapat ditampilkan ke pengguna atau digunakan sebagai input ke sistem lain. Untuk alasan ini biasanya ditambahkan fungsi softmax sebagai lapisan terakhir dari jaringan saraf.

- Tanh

  Hyperbolic tangent function atau yang sering disebut TANH pada umumnya lebih cepat mencapai konvergensi dibandingkan fungsi aktivasi sigmoid dan logistik dan dapat menghasilkan akurasi yang lebih tinggi. Performa yang ditawarkan oleh fungsi aktivasi

TANH hampir sama dengan performa klasifikasi yang dihasilkan oleh fungsi aktivasi RELU.



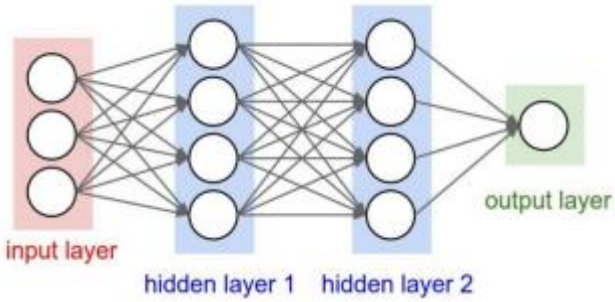
(Wibawa, 2017)

2.4.6. Multi Layer Preceptron

Multilayer Perceptron (MLP) merupakan ANN dari Perceptron. Berupa ANN feedforward dengan satu atau lebih hidden layer. Biasanya, jaringan terdiri dari satu layer neuron komputasi keluaran. Sinyal masukan di propagasikan dengan arah maju pada layer per layer. Contoh arsitektur MLP diberikan pada Gambar 2.1. Setiap layer dalam MLP memiliki fungsi khusus. Layer masukan berfungsi menerima sinyal/vektor masukan dari luar mendistribusikannya ke semua neuron dalam hidden layer. Layer keluaran menerima sinyal keluaran (atau dengan kata lain stimulus pola) dari hidden layer dan memunculkannya sinyal/nilai/kelas keluaran dari keseluruhan jaringan.

Banyak algoritma pelatihan yang tersedia, tetapi yang paling populer adalah Backpropagation. Cara pelatihan yang dilakukan algoritma Backpropagation sama dengan Perceptron. Sejumlah data latih sebagai data masukan diberikan pada jaringan. Jaringan menghitung pada keluaran, jika ada error (perbedaan antara target keluaran yang di inginkan dengan nilai keluaran yang didapatkan) maka bobot dalam jaringan akan diperbaharui untuk mengurangi error tersebut.

Visualisasi arsitektur MLP adalah sebagai berikut:



Sebuah MLP memiliki i layer dengan masing-masing layer berisi j_i neuron. MLP menerima input data satu dimensi dan mempropagasikan data tersebut pada jaringan hingga menghasilkan output. Setiap hubungan antar neuron pada dua layer yang bersebelahan memiliki parameter bobot satu dimensi yang menentukan kualitas mode. Disetiap data input pada layer dilakukan operasi linear dengan nilai bobot yang ada, kemudian hasil komputasi akan ditransformasi menggunakan operasi non linear yang disebut sebagai fungsi aktivasi.

Persamaan yang digunakan untuk pelatihan MLP Backpropagation sebagai berikut:

$v = \sum_{i=1}^{r} x_i \cdot w_i$
v = Nilai keluaran hidden layer.
x_i = Nilai input/fitur.
w_i = Nilai bobot.

Nilai r adalah jumlah masukan (fitur) data masukan, x merupakan nilai fitur/vektor, w adalah bobot vektor. Nilai v tersebut kemudian di aktivasi untuk menghasilkan sinyal keluaran. Fungsi aktivasi yang digunakan adalah sigmoid biner atau sigmoid bipolar. Fungsi aktivasi sigmoid menjadi persamaan:

$y = \frac{1}{1+e^{-v}}$
y = Nilai sigmoid.
e = Eksponen.

Untuk merambatkan sinyal error, dimulai dari layer keluaran dan berjalan kembali ke hidden layer. Sinyal error di neuron k pada iterasi p diberikan pada persamaan:

$e_k(P) = y_{dk}(P) - y_k(P)$
e_k = Nilai selisih / error.
y_{dk} = Nilai Sebenarnya.
y_k = Nilai prediksi.

Prosedur yang digunakan untuk memperbaharui bobot pada koneksi hidden layer ke output layer:

$w_{jk}(p+1) = w_{jk}(p) + \Delta w_{jk}$
Δw_{jk} = Koreksi bobot
w_{jk} = Nilai bobot

Kondisi yang dialami adalah masukan neuron pada output layer berbeda dari input neuron pada input layer xi. Oleh karena itu yang digunakan untuk menghitung koreksi bobot adalah sinyal output neuron j pada hidden layer yj untuk menggantikan xi, Koreksi bobot dalam MLP dihitung dengan persamaan:

$\Delta w_{jk}(p) = \eta * x_i(p) * \delta_j(p)$
η = Learning rate.
$\delta_j(p)$ = Gradien error.
p = Iterasi
Δw_{jk} = Koreksi bobot.

η adalah laju pembelajaran, sedangkan () adalah gradien error pada neuron k dalam output layer pada iterasi ke p. Untuk menghitung gradien error pada fungsi aktivasi sigmoid biner didapatkan:

$\delta(p) = y_k(p) * (1 - y_k(p)) * e_k(p) * w_{jk}$

(Hadimarta, Muhima, & Kurniawan, 2020)

2.4.7. Convolutional Neural Network

a. Pengenalan

Convolutional Neural Network (CNN) adalah pengembangan dari Multilayer Perceptron (MLP) yang didesain untuk mengolah data dua dimensi. CNN termasuk dalam jenis Deep Neural Network karena kedalaman jaringan yang tinggi dan banyak diaplikasikan pada data citra. Pada kasus klasifikasi citra, MLP kurang sesuai untuk digunakan karena tidak menyimpan informasi spasial dari data citra dan menganggap setiap piksel adalah fitur yang independen sehingga menghasilkan hasil yang kurang baik.

CNN pertama kali dikembangkan dengan nama NeoCognitron oleh Kunihiko Fukushima, seorang peneliti dari NHK Broadcasting Science Research Laboratories, Kinuta, Setagaya, Tokyo, Jepang. Konsep tersebut kemudian dimatangkan oleh Yann LeChun, seorang peneliti dari AT&T Bell Laboratories di Holmdel, New Jersey, USA. Model CNN dengan nama LeNet berhasil diterapkan oleh LeChun pada penelitiannya mengenai pengenalan angka dan tulisan tangan. Pada tahun 2012, Alex Krizhevsky dengan penerapan CNN miliknya berhasil menjuarai kompetisi ImageNet Large Scale Visual Recognition Challenge 2012. Prestasi tersebut menjadi momen pembuktian bahwa metode Deep Learning, khususnya CNN. Metode CNN terbukti berhasil mengungguli metode Machine Learning lainnya seperti SVM pada kasus klasifikasi objek pada citra.

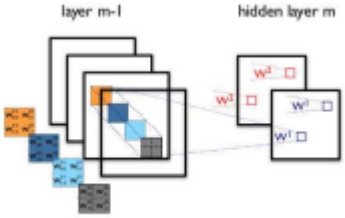
b. Cara kerja

Cara kerja CNN memiliki kesamaan pada MLP, namun dalam CNN setiap neuron dipresentasikan dalam bentuk dua dimensi, tidak seperti MLP yang setiap neuron hanya berukuran satu dimensi.

Pada CNN, data yang dipropagasikan pada jaringan adalah data dua dimensi, sehingga operasi linear dan parameter bobot pada CNN berbeda. Pada CNN operasi linear menggunakan operasi konvolusi, sedangkan bobot tidak lagi satu dimensi saja, namun berbentuk empat dimensi yang merupakan kumpulan kernel konvolusi seperti pada Gambar.2. Dimensi bobot pada CNN adalah:

$$neuron\ input \times neuron\ output \times tinggi \times lebar$$

Karena sifat proses konvolusi, maka CNN hanya dapat digunakan pada data yang memiliki struktur dua dimensi seperti citra dan suara.



c. Arsitektur

JST terdiri dari berbagai layer dan beberapa neuron pada masing-masing layer. Kedua hal tersebut tidak dapat ditentukan menggunakan aturan yang pasti dan berlaku berbeda-beda pada data yang berbeda.

Pada kasus MLP, sebuah jaringan tanpa hidden layer dapat memetakan persamaan linear apapun, sedangkan jaringan dengan satu atau dua hidden layer dapat memetakan sebagian besar persamaan pada data sederhana

Namun pada data yang lebih kompleks, MLP memiliki keterbatasan. Pada permasalahan jumlah hidden layer dibawah tiga layer, terdapat pendekatan untuk menentukan jumlah neuron pada masing-masing layer untuk mendekati hasil optimal. Penggunaan layer diatas dua pada umumnya tidak direkomendasikan dikarenakan akan menyebabkan overfitting serta kekuatan backpropagation berkurang secara signifikan.

Dengan berkembangnya deep learning, ditemukan bahwa untuk mengatasi kekurangan MLP dalam menangani data kompleks, diperlukan

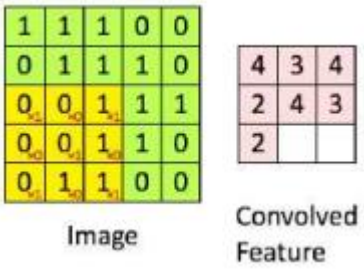
fungsi untuk mentransformasi data input menjadi bentuk yang lebih mudah dimengerti oleh MLP. Hal tersebut memicu berkembangnya deep learning dimana dalam satu model diberi beberapa layer untuk melakukan transformasi data sebelum data diolah menggunakan metode klasifikasi. Hal tersebut memicu berkembangnya model neural network dengan jumlah layer diatas tiga. Namun dikarenakan fungsi layer awal sebagai metode ekstraksi fitur, maka jumlah layer dalam sebuah DNN tidak memiliki aturan universal dan berlaku berbedabeda tergantung dataset yang digunakan.

Karena hal tersebut, jumlah layer pada jaringan serta jumlah neuron pada masing-masing layer dianggap sebagai hyperparameter dan dioptimasi menggunakan pendekatan searching.

Sebuah CNN terdiri dari beberapa layer. Berdasarkan aristektur LeNet5, terdapat empat macam layer utama pada sebuah CNN, dengan ketiga layernya adalah sebagai berikut:

1. Convolutional Layer

   Convolution Layer melakukan operasi konvolusi pada output dari layer sebelumnya. Layer tersebut adalah proses utama yang mendasari sebuah CNN. Konvolusi adalah suatu istilah matematis yang berati mengaplikasikan sebuah fungsi pada output fungsi lain secara berulang. Dalam pengolahan citra, konvolusi berati mengaplikasikan sebuah kernel (kotak kuning) pada citra disemua offset yang memungkinkan seperti yang ditunjukkan pada Dibawah ini.

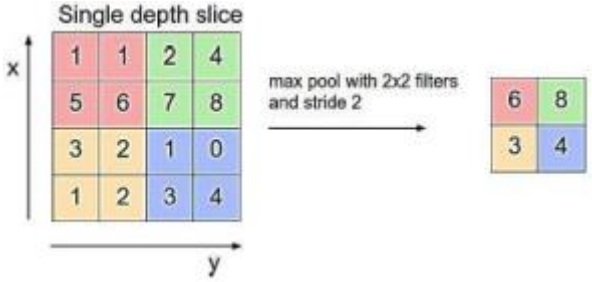


   Kotak hijau secara keseluruhan adalah citra yang akan dikonvolusi. Kernel bergerak dari sudut kiri atas ke kanan bawah. Sehingga hasil konvolusi dari citra tersebut dapat dilihat pada gambar disebelah kanannya. Tujuan dilakukannya konvolusi pada data citra adalah untuk mengekstraksi fitur dari citra input. Konvolusi akan menghasilkan transformasi linear dari data input sesuai informasi spasial pada data. Bobot pada layer tersebut menspesifikasikan kernel konvolusi yang digunakan, sehingga kernel konvolusi dapat dilatih berdasarkan input pada CNN.

2. Subsampling Layer

   Subsampling adalah proses mereduksi ukuran sebuah data citra. Dalam pengolahan citra, subsampling juga bertujuan untuk meningkatkan invariansi posisi dari fitur. Dalam sebagian besar CNN, metode subsampling yang digunakan adalah max pooling. Max pooling membagi output dari convolution layer menjadi beberapa grid kecil lalu mengambil nilai maksimal dari setiap grid untuk menyusun matriks citra yang telah direduksi seperti yang ditunjukkan pada Gambar.4. Grid

yang berwarna merah, hijau, kuning dan biru merupakan kelompok grid yang akan dipilih nilai maksimumnya. Sehingga hasil dari proses tersebut dapat dilihat pada kumpulan grid disebelah kanannya. Proses tersebut memastikan fitur yang didapatkan akan sama meskipun objek citra mengalami translasi (pergeseran).



Penggunaan pooling layer pada CNN hanya bertujuan untuk mereduksi ukuran citra sehingga dapat dengan mudah digantikan dengan sebuah convolution layer dengan stride yang sama dengan pooling layer yang bersangkutan.

3. Fully Connected Layer

Fully Connected Layer Layer tersebut adalah layer yang biasanya digunakan dalam penerapan MLP dan bertujuan untuk melakukan transformasi pada dimensi data agar data dapat diklasifikasikan secara linear.

Setiap neuron pada convolution layer perlu ditransformasi menjadi data satu dimensi terlebih dahulu sebelum dapat dimasukkan ke dalam sebuah fully connected layer. Karena hal tersebut menyebabkan data kehilangan informasi spasialnya dan tidak reversibel, fully connected layer hanya dapat diimplementasikan di akhir jaringan.

Dalam sebuah jurnal oleh Lin et al., dijelaskan bahwa convolution layer dengan ukuran kernel 1 x 1 melakukan fungsi yang sama dengan sebuah fully connected layer namun dengan tetap mempertahankan karakter spasial dari data. Hal tersebut membuat penggunaan fully connectedlayer pada CNN sekarang tidak banyak dipakai.

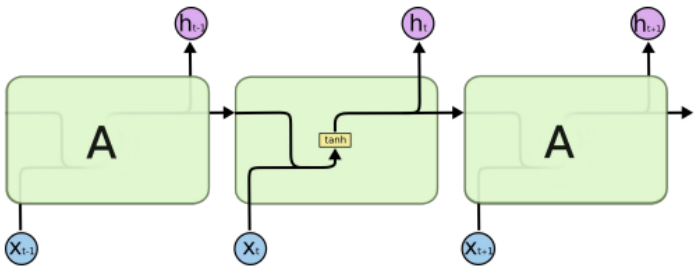
2.4.8. RNN

RNN yang juga disebut jaringan umpan balik adalah jenis jaringan pada neural network dimana terdapat loop sebagai koneksi umpan balik dalam jaringan.

Jaringan RNN adalah jaringan yang mengakomodasi output jaringan untuk menjadi input pada jaringan tersebut yang kemudian digunakan untuk menghasilkan output yang baru. Pembuatan RNN sebenarnya dibuat untuk data-data yang bersifat sequential atau bertahap.

Penggunaan neural network biasanya semua input dan output tidak bergantung satu sama lain, maka akan terjadi penumpukan tugas pada neural network sangat banyak dan bertumpuk, sedangkan jika ingin membangun sebuah aplikasi yang mengelola data ilmiah dengan data sampel berbentuk time series diperlukan agloritma yang dapat menghandle data tersebut dengan cepat dan tepat, maka RNN dapat melakukan hal itu karena RNN melakukan tugas yang sama pada setap elemen data di sebuah urutan, lalu memproses outputnya yang mengacu pada hasil komputasi sebelumnya. Maka secara teori RNN

mampu menggunakan informasi yang telah direkam sebelumnya yang panjang urutannya beragam-ragam.
Visualisasi arsitektur RNN:

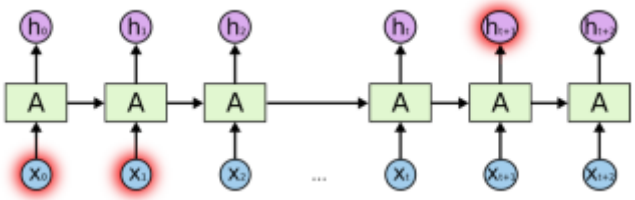


Jaringan node pada RNN dimasukan ke dalam layer yang berurutan. Setiap node dalam lapisan tertentu terhubung dengan koneksi terarah ke setiap node lain dilapisan layer berikutnya secara beturut-turut, setiap node juga memiliki aktivasi. (Tarkus, Sompie, & Jacobus, 2020).

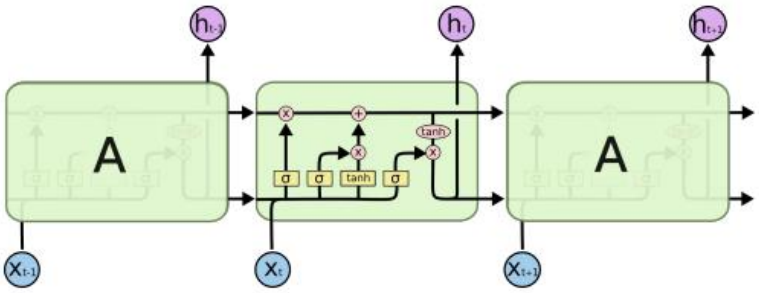
Berikut ini adalah algoritma turunan dari RNN sehingga memiliki konsep dasar yang sama, yaitu:

a. Long Short-Term Memory

Long Short Term Memory networks (LSTM) merupakan sebuah evolusi dari arsitektur RNN, dimana pertama kali diperkenalkan oleh Hochreiter & Schmidhuber (1997). Hingga penelitian ini dilakukan banyak para peneliti yang terus mengembangkan arsitektur LSTM di berbagai bidang seperti dalam bidang speech recognition dan forecasting.



Gambar diatas menjelaskan RNN memiliki kekurangan, kekurangan itu dapat dilihat pada inputan X_0, X_1 memiliki rentang informasi yang sangat besar dengan X_t, X_{t+1} sehingga ketika h_{t+1} memerlukan informasi yang relevan dengan X_0, X_1 RNN tidak dapat untuk belajar menghubungkan informasi karena memori lama yang tersimpan akan semakin tidak berguna dengan seiringnya waktu berjalan karena tertimpa atau tergantikan dengan memori baru, permasalahan ini ditemukan oleh Bengio, et al. (1994). Berbeda dengan RNN, LSTM tidak memiliki kekurangan tersebut karena LSTM dapat mengatur memori pada setiap masukannya dengan menggunakan memory cells dan gate units.



Gambar arsitektur LSTM diatas menjelaskan bagaimana alur kerja memory cells pada setiap neurons LSTM bekerja. Terdapat empat proses fungsi aktivasi pada setiap masukan pada neurons yang selanjutnya disebut sebagai gates units. Gates units tersebut ialah:

- Forget gates

  Pada forget gates informasi pada setiap data masukan akan diolah dan dipilih data mana saja yang akan disimpan atau dibuang pada memory cells. Fungsi aktivasi yang digunakan pada forget gates ini adalah fungsi aktivasi sigmoid. Dimana hasil keluarannya antara 0 dan 1. Jika keluarannya adalah 1 maka semua data akan disimpan dan sebaliknya jika keluarannya 0 maka semua data akan dibuang. Dengan rumus sebagai berikut:

$$f_t = \sigma\left(W_f.[h_{t-1}, x_t] + b_f\right)$$

- Input gates

  Pada input gates terdapat dua gates yang akan dilaksanakan, pertama akan diputuskan nilai mana yang akan diperbarui menggunakan fungsi aktivasi sigmoid. Selanjutnya fungsi aktivasi tanh akan membuat vector nilai baru yang akan disimpan pada memory cell. Dengan rumus sebagai berikut:

$$i_t = \sigma\left(W_i.[h_{t-1}, x_t] + b_i\right)$$
$$\check{c}_t = tanh\left(W_c.[h_{t-1}, x_t] + b_c\right)$$

- Cell gates

  Pada cell gates akan mengganti nilai pada memory cell sebelumnya dengan nilai memory cell yang baru. Dimana nilai ini didapatkan dari menggabungkan nilai yang terdapat pada forget gate dan input gate. Dengan rumus sebagai berikut:

$$c_t = f_t * c_{t-1} + i_t * \check{c}_t$$

- Output gates.

  Pada output gates terdapat dua gates yang akan dilaksanakan, pertama akan diputuskan nilai pada bagian memory cell mana yang akan dikeluarkan dengan menggunakan fungsi aktivasi sigmoid. Selanjutnya akan ditempatkan nilai pada memory cell dengan menggunakan fungsi aktivasi tanh. Terakhir kedua gates tersebut di dikalikan sehingga menghasilkan nilai yang akan dikeluarkan. Dengan rumus sebagai berikut:

$$o_t = \sigma\left(W_o.[h_{t-1}, x_t] + b_o\right)$$
$$h_t = o_t tanh(c_t)$$

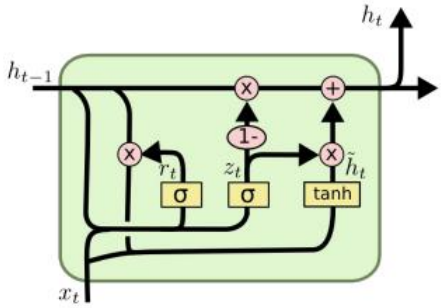
Berikut ini adalah keseluruhan perhitungan yang digunakan dalam LSTM:

$$\tilde{c}_t = \tanh(W_c[a_{t-1}, x_t] + b_c)$$
$$G_u = \sigma(W_u[a_{t-1}, x_t] + b_u)$$
$$G_f = \sigma\left(W_f[a_{t-1}, x_t] + b_f\right)$$
$$G_o = \sigma(W_o[a_{t-1}, x_t] + b_o)$$
$$c_t = G_u * \tilde{c}_t + G_f * c_{t-1}$$
$$a_t = G_o * c_t$$

(Aldi, Jondri, & Aditsania, 2018)

b. Gated Recurrent Unit

GRU adalah arsitektur yang diciptakan oleh Kyunghun Cho pada tahun 2014. Serupa dengan LSTM, GRU juga menggunakan sistem gerbang sebagai berikut:



Arsitektur GRU lebih sederhana daripada LSTM. GRU tidak menggunakan cell state, tetapi memanfaatkan hidden state untuk menyimpan informasi. Reset gate dalam GRU menentukan informasi baru harus dilupakan atau tidak, sedangkan update gate untuk mengingat.

LSTM memiliki tiga gerbang sigmoid dan dua gerbang tanh, sedangkan GRU hanya memerlukan dua sigmoid dan sebuah tanh. Perhitungan yang terlibat untuk setiap update GRU adalah sebagai berikut:

$$\tilde{c}_t = \tanh(W_c[G_r * c_{t-1}, x_t] + b_c)$$
$$G_u = \sigma(W_u[c_{t-1}, x_t] + b_u)$$
$$G_r = \sigma(W_r[c_{t-1}, x_t] + b_r)$$
$$c_t = G_u * \tilde{c}_t + (1 - G_u) * c_{t-1}$$
$$a_t = c_t$$

Karena GRU melibatkan perhitungan yang lebih sedikit daripada LSTM, secara teori GRU dapat dilatih lebih cepat daripada LSTM. Namun, pada praktiknya, untuk masing-masing kasus harus dicoba, agar diperoleh yang lebih cocok untuk menyelesaikan suatu masalah. (Zaman, Sumpeno, & Hariadi, 2019)

BAB 3

PEMBAHASAN

3.2.1. Akuisisi Data

Dataset yang digunakan berasal dati GTZAN music genre clasification yang diunduh melalui website kaggle.com. Dataset tersebut terdiri dari berbagai 10 genre musik yang setiap genrenya memiliki 1000 musik dengan durasi 30 detik, musk tersebut merupakan karya seniman musik dari internasional sehingga pada umumnya menggunakan Bahasa Inggris, namun dalam percobaan yang dilakukan saat ini hanya menggunakan 10 musik disetiap genrenya saja. Berikut adalah daftar genre yang akan diklasifikasikan:

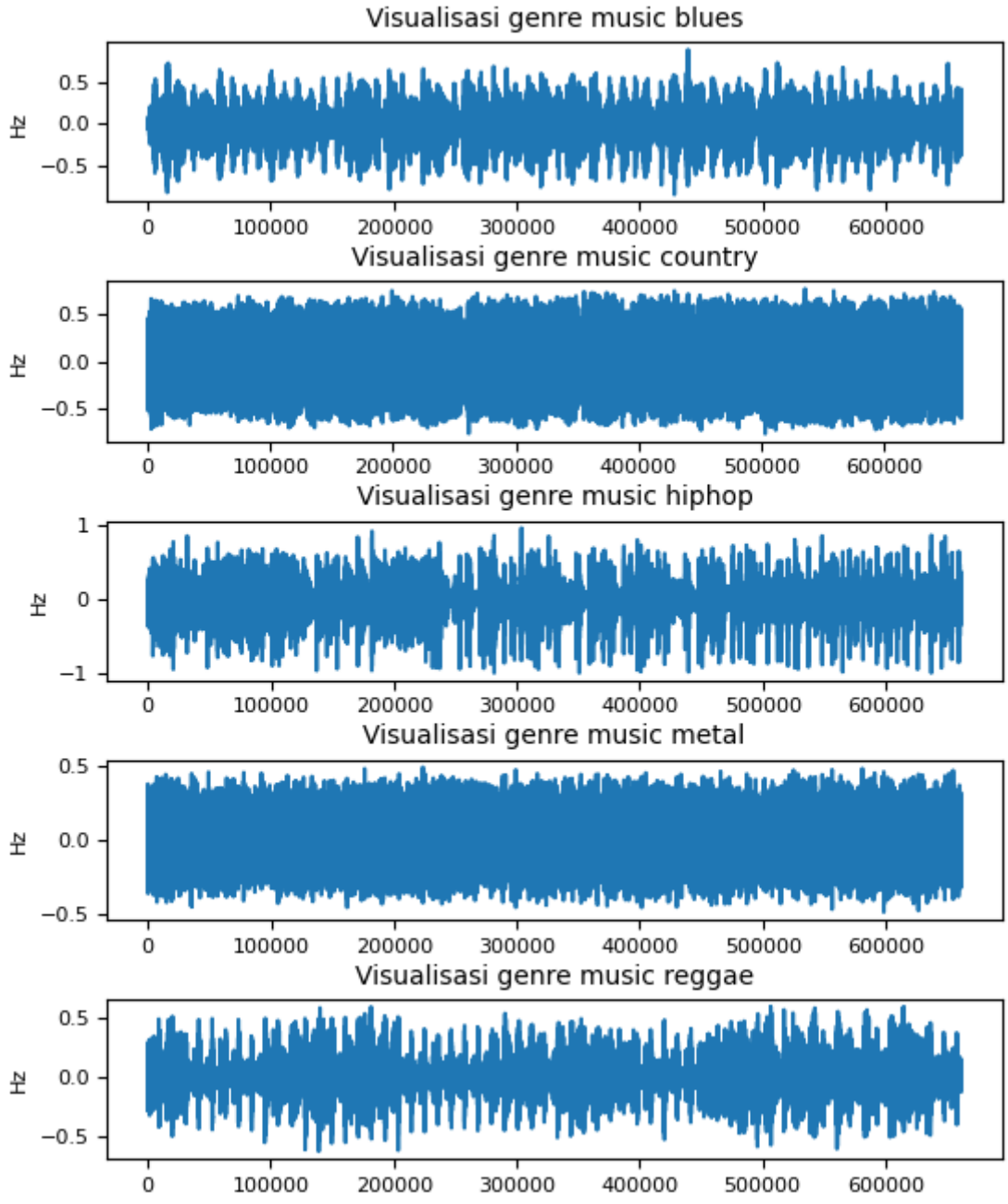
Genre	Jumlah Musik
Blues	10

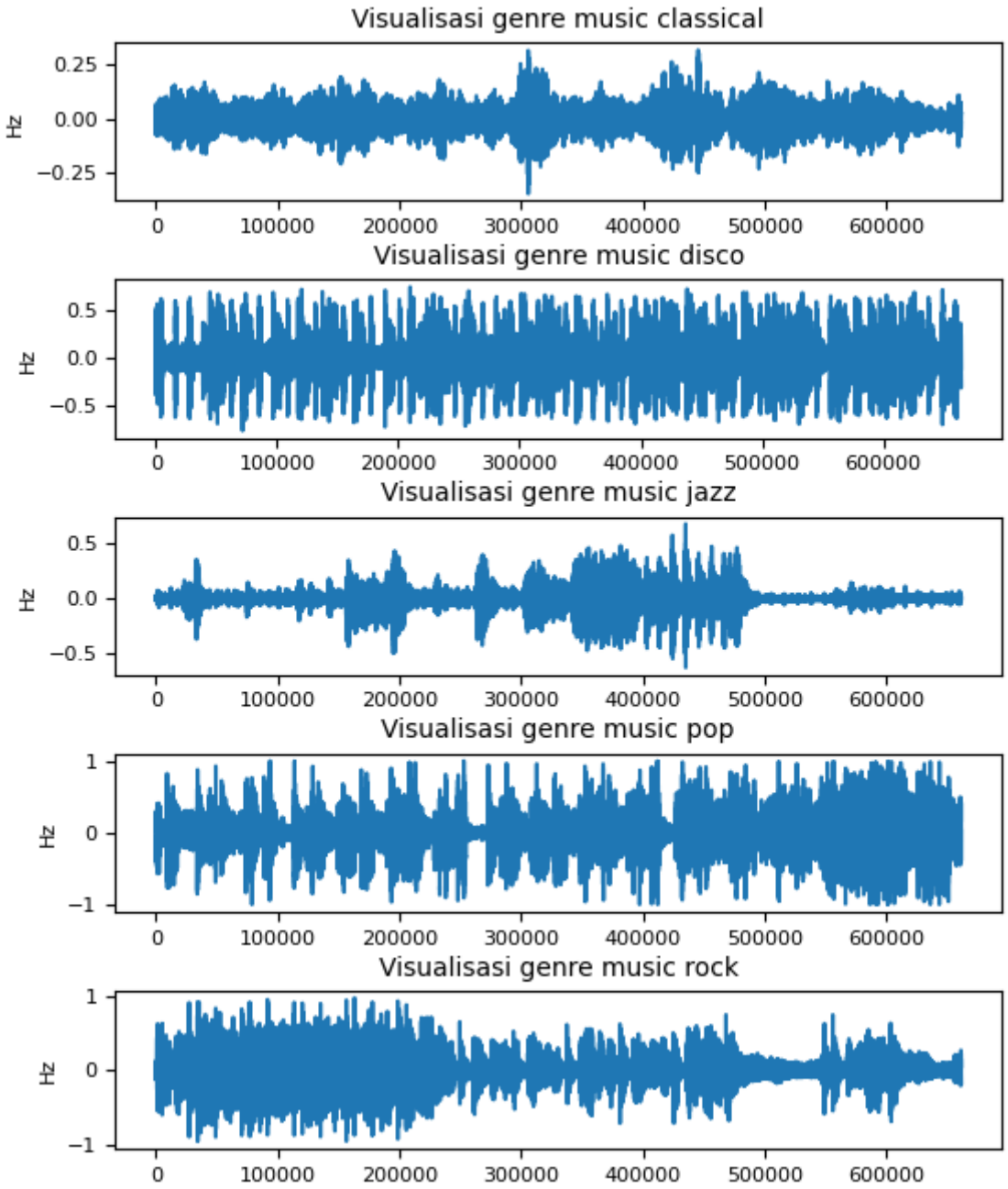
Classic	10
Country	10
Disco	10
Hiphop	10
Jazz	10
Metal	10
Pop	10
Reggae	10
Rock	10

3.2.2. Praproses Data

3.2.1. Visualisasi Data Mentah

Sebelum dilakukan prediksi kelas data mentah yang sudah didapat harus melewati praproses data untuk bisa dilakukan prediksi kelas, berikut ini adalah visualisasi sampel data audio musik yang belum dilakukan praproses data:


Visualisasi genre music blues
Hz
0.5
0.0
−0.5
0
100000
200000
300000
400000
500000
600000
Visualisasi genre music country
Hz
0.5
0.0
−0.5
0
100000
200000
300000
400000
500000
600000
Visualisasi genre music hiphop
Hz
1
0
−1
0
100000
200000
300000
400000
500000
600000
Visualisasi genre music metal
Hz
0.5
0.0
−0.5
0
100000
200000
300000
400000
500000
600000
Visualisasi genre music reggae
Hz
0.5
0.0
−0.5
0
100000
200000
300000
400000
500000
600000



Visualisasi tersebut memperlihat tingkat frekuensi yang berubah-ubah setiap waktu pada suatu lagu dengan 10 genre yang akan diklasifikasi. Seperti yang dapat dilihat setiap genre memiliki perubahan frekuensi yang berbeda-beda dan keunikannya masing-masing, hal itulah yang mendasari untuk memungkinkan dilakukannya klasifikasi, selain itu dengan bantuan deep learning akan muncul pola-pola yang penting yang berbeda setiap genre.

3.2.2. Mel-Frequency Ceptrum Coefficient

Data musik yang disimpan dalam berkas dengan format wav akan dikalkukasi menggunakan MFCC dengan memanfaatkan *package* librosa, hasilnya merupakan array 3 dimensi dengan ukuran 989 x 130 x 13. Data hasil ekstraksi fitur dengan MFCC itu disimpan kedalam berkas dengan format json.

Parameter MFCC:

Parameter	Keterangan	Nilai
Sample rate	Sampel yang diambil tiap detik	22050
Number mfcc	Jumlah koefesien yang di extract	13
Number fft	Interval yang digunakan untuk melakukan fft	2048
Hop length	Jarak pergesaran window saat fft	512
Number segment	Jumlah segmen yang akan dibagi untuk mendapatkan sample track	10

3.2.3. Pembagian Data

Data yang sudah dalam bentuk mfcc kemudian dilakukan pembagian data, dimana untuk masing masing genre akan dibagi menjadi 3 bagian dengan rasio sebagai berikut:

Jenis	Rasio
Data Pelatihan	50%
Data Evaluasi	20%
Data Uji	30%

3.2.4. Multi Layer Preceptron

Prediksi kelas dilakukan dengan menggunakan fungsi MLP dengan menggunakan *package* Keras dari Tensorflow, dalam percobaan ini berkas musik yang sudah melewati tahap ekstraksi fitur dengan MFCC dilakukan prediksi kelas.

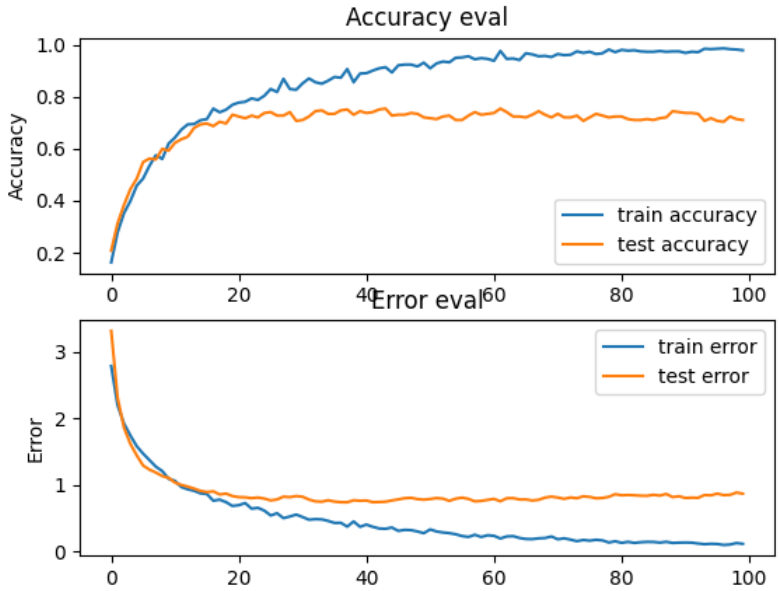
Arsitektur yang digunakan oleh semua algorima yang dibandingan memiliki dasar yang sama yaitu terdiri dari 3 hidden layer dengan *batch normalization* untuk meningkatkan akurasi, *dropout* untuk mencegah terjadinya *overfitting*, dan output layer dengan mengunakan fungsi aktivasi softmax, serta dikompilasi dengan menggunakan pengoptimasi Adam.

Arsitektur yang digunakan:

Hidden Layer Pertama	
Dense	
Neuron units	512
Fungsi Aktivasi	Relu
Dropout	
Rate	0.3
Hidden Layer Kedua	
Dense	
Neuron units	256
Fungsi Aktivasi	Relu
Dropout	
Rate	0.3
Hidden Layer Ketiga	
Dense Layer	
Neuron units	64
Fungsi Aktivasi	Relu
Dropout	
Rate	0.3

Output Layer	
Dense Layer	
Neuron Unit	10
Fungsi Aktivasi	Softmax

Percobaan dilakukan sebanyak 100 kali iterasi (*epoch*) dengan ukuran batch sebesar 32, berikut adalah hasil percobaan dengan menggunakan algoritma MLP:



Dalam grafik tersebut dapat dilakukan penghentian iterasi epoch saat keduanya tidak terpaut jauh yakni saat epoch ke-20 jika diperlukan.

Parameter	Percobaan 1	Percobaan 2	Percobaan 3	Rata-rata
Akurasi	0.7407	0.7778	0.7542	0.7573
Loss	1.1411	1.2127	1.1821	1.1786
Waktu	00:00:34.73	00:00:37.86	00:00:45.90	00:00:39.83

Dari tabel diatas dapat diketahui bahwa algoritma MLP menghasilkan akurasi rata-rata sebesar 75% dan membutuhkan waktu pelatihan selama 39 detik, Karena hanya menghasilkan akurasi sebesar 75% terkadang terjadi kesalahan penentuan genre saat melakukan prediksi, hal tersebut dapat diterima jika melihat waktu pelatihannya yang sangat singkat.

3.2.5. Convolutional Neural Network

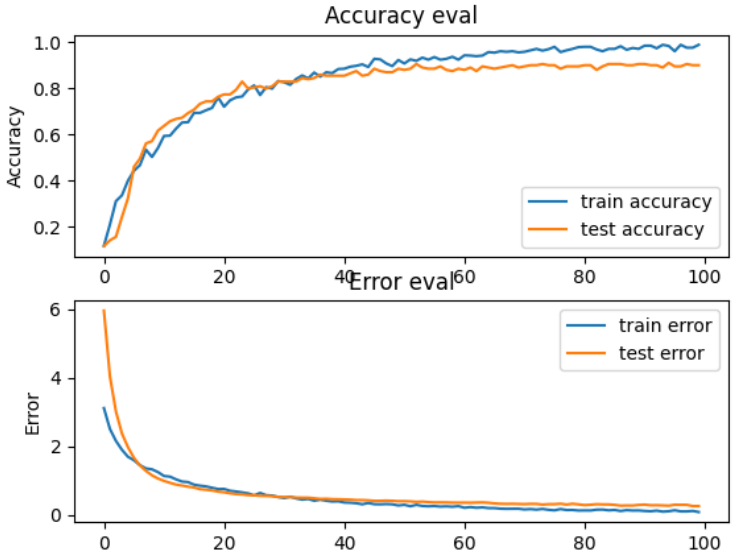
Prediksi kelas dilakukan dengan menggunakan fungsi CNN dilakukan dengan menggunakan *package* Keras dari Tensorflow, karena berkas yang digunakan adalah berkasi audio maka jenis CNN yang digunakan adalah *convolutional* dan *max pooling* 2D, karena akan memanfaatkan 2D *stride* untuk mendapatkan informasi penting dalam berkas audio dengan MFCC yang digunakan.

Sama seperti percobaan MLP, pada percobaan CNN menggunakan arsitektur yang serupa, yaitu terdiri dari 3 hidden layer dengan *batch normalization*, *dropout*, dan output layer serta dikompilasi dengan pengoptimasi Adam.

Arsitektur:

Convolutional Layer Pertama	
Convolutional 2D	
Ukuran filter	32
Ukuran stride	3x3
Fungsi Aktivasi	Relu
Max pooling 2D	
Ukuran pool	3x3
Ukuran strides	2x2
Batch Normalization	
Convolutional Layer Kedua	
Convolutional 2D	
Ukuran filter	32
Ukuran stride	3x3
Fungsi Aktivasi	Relu
Max pooling 2D	
Ukuran pool	3x3
Ukuran strides	2x2
Batch Normalization	
Convolutional Layer Ketiga	
Convolutional 2D	
Ukuran filter	32
Ukuran stride	2x2
Fungsi Aktivasi	Relu
Max pooling 2D	
Ukuran pool	2x2
Ukuran strides	2x2
Batch Normalization	
Dropout	
Rate	0.3
Output Layer	
Neuron	10
Fungsi aktivasi	Softmax

Percobaan dilakukan sebanyak 100 kali iterasi (*epoch*) dengan ukuran batch sebesar 32, berikut adalah hasil percobaan dengan menggunakan algoritma CNN:



Dari grafik diatas akurasi dan loss untuk pelatihan dan validasi cenderung sama dengan perbedaan yang cukup sedikit sehingga model yang dibuat menjadi baik untuk melakukan prediksi.

Parameter	Percobaan 1	Percobaan 2	Percobaan 3	Rata-rata
Akurasi	0.9798	0.9630	0.9731	0.972
Loss	0.0682	0.1435	0.1059	0.1059
Waktu	00:01:26.64	00:01:33.12	00:01:31.92	00:01:30.56

Dari tabel diatas dapat diketahui bahwa algoritma CNN menghasilkan akurasi rata-rata sebesar 97% dan membutuhkan waktu pelatihan selama 1 menit 30 detik, algoritma CNN mampu menghasilkan akurasi yang cukup besar dalam waktu yang relatif singkat.

3.2.6. Recurrent Neural Network

a. LSTM

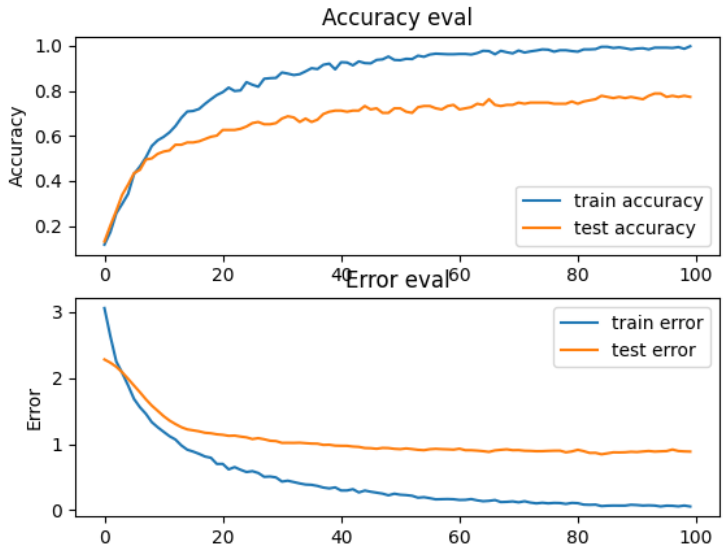
Prediksi kelas dilakukan dengan menggunakan fungsi LSTM dilakukan dengan menggunakan *package* Keras dari Tensorflow. Sama seperti percobaan sebelumnya, pada percobaan LSTM menggunakan arsitektur yang serupa, yaitu terdiri dari 3 hidden layer dengan *batch normalization*, *dropout*, dan output layer serta dikompilasi dengan pengoptimasi Adam.

Arsitektur:

LSTM Layer Pertama	
LSTM	
Jumlah neuron	64
Batch Normalization	
LSTM Layer Kedua	
LSTM	
Jumlah neuron	64
Batch Normalization	
LSTM Layer Ketiga	
LSTM	

Jumlah neuron	64
Batch Normalization	
Dropout	
Rate	0.3
Output Layer	
Neuron	10
Fungsi aktivasi	Softmax

Hasil Percobaan:



Dari hasil grafik diatas akurasi dan error untuk pelatihan dan validasi terpaut cukup jauh sehingga model yang dibuat menjadi kurang baik.

Parameter	Percobaan 1	Percobaan 2	Percobaan 3	Rata-rata
Akurasi	0.9798	0.9556	0.9556	0.9636
Loss	0.0832	0.1927	0.1744	0.1501
Waktu	00:05:36.53	00:05:33.46	00:05:34.07	00:05:34.35

Dari tabel diatas dapat diketahui bahwa algoritma LSTM menghasilkan akurasi rata-rata sebesar 96% dan membutuhkan waktu pelatihan selama 5 menit 34 detik, algoritma LSTM mampu menghasilkan akurasi yang cukup besar meskipun memerlukan waktu yang cukup lama.

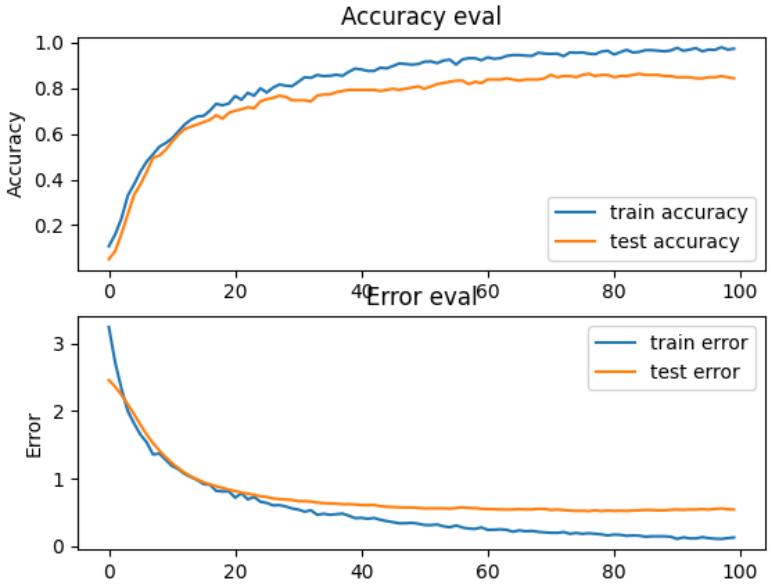
b. GRU

Arsitektur:

GRU Layer Pertama	
GRU	
Jumlah neuron	64
Batch Normalization	
GRU Layer Kedua	
GRU	
Jumlah neuron	64

Batch Normalization	
GRU Layer Ketiga	
GRU	
Jumlah neuron	64
Batch Normalization	
Dropout	
Rate	0.3
Output Layer	
Neuron	10
Fungsi aktivasi	Softmax

Hasil:



Berdasarkan grafik diatas prediksi dan loss untuk pelatihan dan evaluasi memiliki jarak yang sedikit renggang, model yang baik memberikan hasil akurasi/loss pelatihan dan validasi yang sama atau tidak terlampau jauh.

Parameter	Percobaan 1	Percobaan 2	Percobaan 3	Rata-rata
Akurasi	0.9677	0.9718	0.9677	0.969
Loss	0.1149	0.1116	0.1438	0.1234
Waktu	00:05:59.83	00:05:34.54	00:05:34.60	00:05:42.66

Dari tabel diatas dapat diketahui bahwa algoritma GRU menghasilkan akurasi rata-rata sebesar 97% dan membutuhkan waktu pelatihan selama 5 menit 42 detik, algoritma GRU mampu menghasilkan akurasi yang cukup besar meskipun membutuhkan waktu yang cukup lama.

3.2.7. Perbandingan Algoritma

Setelah melalukan prediksi genre musik, selanjutnya kita dapat membandingkan performa yang dihasilkan, berikut adalah tabel perbandingan hasilnya:

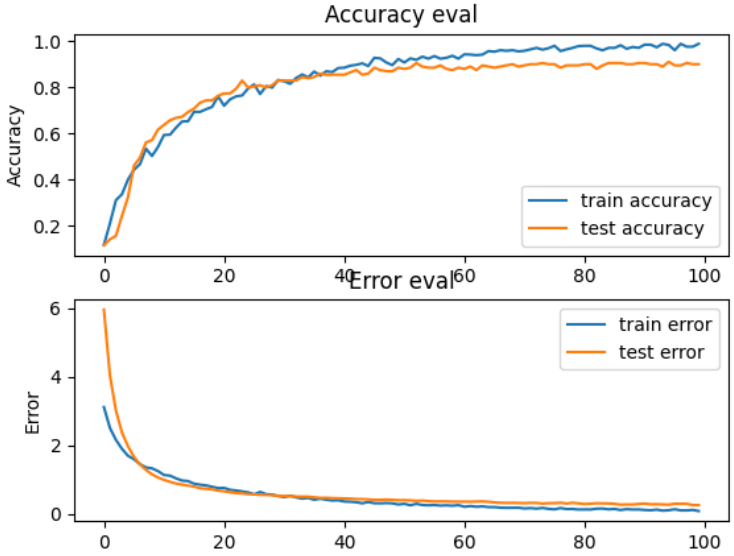
Algoritma	Akurasi	Loss	Waktu

Multi Layer Preceptron	0.7573	1.1786	00:00:39.83
Convolutional Neural Network	0.972	0.1059	00:01:30.56
Long Short-Term Memory	0.9636	0.1501	00:05:34.35
Gated Recurrent Unit	0.969	0.1234	00:05:42.66

Dari tabel diatas dapat diketahui bahwa algoritma dengan tingkat akurasi tertinggi dan loss terendah adalah CNN dan algoritma dengan waktu yang membutuhkan waktu pelatihan tersingkat adalah MLP.

Algoritma MLP meskipun mempunyai waktu pelatihan tersingkat tapi ia menghasilkan akurasi yang relatif kecil jika dibandingkan dengan algorima lainnya, sementara itu algoritma RNN seperti LSTM dan GRU memiliki akurasi yang hapir sama dengan akurasi yang cukup tinggi namun membutuhkan waktu yang cukup lama hingga 5 menit.

Algoritma CNN menjadi algoritma yang paling dapat diandalkan untuk melakukan klasifikasi berkas audio khususnya file musik, karena ia menghasilkan akurasi yang tinggi dengan waktu yang relatif singkat sebesar 1 menit 30 detik.



Jika dibandingkan dengan algoritma lainnya tingkat akurasi dan loss untuk pelatihan dan validasi pada algoritma CNN mempunyai grafik yang sejajar, hal tersebut menandakan CNN menghasilkan model yang baik dan tidak terjadi overfitting.

BAB 4

KESIMPULAN

4.1. Kesimpulan

Dari hasil uji coba yang telah dilakukan didapatkan keputusan bahwa algoritma Convolutional Neural Network menjadi algoritma yang paling dapat diandalkan untuk melakukan klasifikasi genre musik yang sebelumnya dilakukan tahap ekstraksi fitur dengan menggunakan Mel-Frequency Ceptrum Coefficient, dengan akurasi mencapai 97% dalam waktu 1 menit 30 detik.

Sementara itu algoritma Recurrent Neural Network seperti Long Short-Term Memory dan Gated Recurrent Unit menghasilkan akurasi yang tinggi (76%) akan tetapi membutuhkan waktu yang cukup lama yaitu 5 menit 30 detik. Sedangkan algoritma Multi Layer Preceptron memiliki akurasi paling paling sedikit yaitu 75% dengan waktu paling sedikit pula yaitu 39 detik. Algoritma MLP terkadang memberikan hasil prediksi genre yang salah, sehingga tidak disarankan untuk menggunakannya.

4.2. Saran

Berikut adalah saran dari penelitan yang sudah dilakukan:

- Dapat mengubah parameter pada fungsi algoritma seperti ukuran neuron, jumlah epoch dan lain-lain atau mengubah arsitektur seperti jumlah layer untuk menemukan arsitektur dan setelan yang paling tepat untuk algoritma yang diuji.
- Menambahkan algoritma lain seperti SVM dan algoritma lainnya untuk menemukan algoritma yang paling dapat diandalkan.

DAFTAR PUSTAKA

Zaman, L., Sumpeno, S., & Hariadi, M. (2019). Analisis Kinerja LSTM dan GRU sebagai Model Generatif untuk Tari Remo. *JNTETI*, 142-150.

Aldi, M. W., Jondri, & Aditsania, A. (2018). Analisis dan Implementasi Long Short Term Memory Neural Network untuk Prediksi Harga Bitcoin. *e-Proceeding of Engineering*, 2355-9365.

Gunawan, I. (2020). Optimasi Model Artificial Neural Network Untuk Klasifikasi Paket Jaringa. *SIMETRIS*, 1-5.

Hadimarta, T. F., Muhima, R. R., & Kurniawan, M. (2020). Implementasi Multilayer Perceptron Pada Jaringan Saraf Tiruan Untuk . *INTEGER: Journal of Information Technology*, 56-63.

Handoko, D. T., & Kasih, P. (2018). Voice Recognition untuk Sistem Keamanan PC Menggunakan Metode MFCC dan DTW. *Generation Journal*, 57-68.

Ioffe, S., & Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *ArXiv*.

Nada, Q., Ridhuandi, C., Santoso, P., & Apriyanto, D. (2019). Speech Recognition dengan Hidden Markov Model untuk Pengenalan dan Pelafalan Huruf Hijaiyah. *Jurnal AL-AZHAR INDONESIA SERI SAINS DAN TEKNOLOGI*, 19-26.

Roihan, A., Sunarya, P. A., & Rafika, A. S. (2019). Pemanfaatan Machine Learning dalam Berbagai Bidang: . *Indonesian Journal on Computer and Information Technology*, 75-82.

Tarkus, E. D., Sompie, S. R., & Jacobus, A. (2020). Implementasi Metode Recurrent Neural Network pada Pengklasifikasian Kualitas Telur Puyuh. *Jurnal Teknik Informatika*, 137-144.

Wibawa, M. S. (2017). Pengaruh Fungsi Aktivasi, Optimisasi dan Jumlah Epoch. *JURNAL SISTEM DAN INFORMATIKA*, 1-8.