JUSTIN
Jurnal Sistem dan Teknologi Informasi
p-ISSN : 2460-3562 / e-ISSN : 2620-8989
DOI: 10.26418/justin.v11i2.55503
Vol. 11, No. 2, April 2023

Segementasi Gambar pada Dataset MNIST dengan Optimasi Mini Batch dan K-means++ pada Algoritma K-means

Agus Susanto[#1], Rusdianto Roestam[#2]

[#] *Master of Science in Information Technology, Fakultas Komputer, Universitas President*
Jababeka Education Park, Jl. Ki Hajar Dewantara, Cikarang Utara, Bekasi

[1]agus.susanto@student.president.ac.id
[2]rusdianto@president.ac.id

Abstrak

Proses segmentasi gambar tulisan tangan berupa gambar dengan menggunakan *k-means* perlu diawali dengan clusterisasi. Akibat dari proses clusterisasi ini membuat proses secara keseluruhan menjadi lambat. Penelitian ini memanfaatkan *mini batch k-means* untuk mempercepat proses pengenalan tulisan tangan berupa gambar. Data yang digunakan pada penelitian ini adalah *dataset* MNIST. Normalisasi terhadap *dataset* akan dilakukan terlebih dahulu sebelum clusterisasi menggunakan teknik *minibatch k-means* diterapkan. Selanjutnya inisiasi *centroid* dilakukan menggunakan *k-means++*. Hasil penelitian menyatakan bahwa optimasi minibatch dan k-means++ pada algoritma k-means mampu mempercepat waktu komputasi lebih singkat 28 menit 13 detik yaitu 29 menit 39 detik menjadi 1 menit 26 detik, dengan akurasi sebesar 90,13%.
Kata kunci: Segmentasi Gambar, Clustering, k-means, mini batch, k-means++, MNIST

Image Segmentation on MNIST Dataset with Optimization MiniBatch and K-means++ on K-means Algorithm

Abstract

Image segmentation process on handwriting image using k-means algorithm needs to begin with clustering process. As a result of this clustering process makes the overall process slow. This research utilizes MiniBatch k-means to speed up the process of handwriting image recognition. The data used this study is MNIST dataset. Normalization process the dataset will be carried out first before clustering using minibatch k-means. Then centroid initiation will be process using k-means++. The result of the study stated that optimization of minibatch and k-means++ on k-means algorithm was able to speed up computation time with difference 28 minutes 13 second, form 29 minutes 39 seconds to 1 minutes 26 seconds, with an accuracy of 90.13%.

Keywords: image segmentation, clustering, k-menas, minibatch, k-means++, MNIST

I. PENDAHULUAN

Metode dan aplikasi segmentasi gambar telah dijadikan umum untuk digunakan dan terus dikembangkan diseluruh dunia dengan bantuan metode *machine learning*, *deep learning* [1], dan *Computer Vision* [2]. Dalam pengolahan gambar digital dan visi komputer segmentasi gambar adalah proses partisi gambar digital dijadikan beberapa segmen (set dari *pixel*, juga dikenal sebagai objek gambar). Tujuan dari segmentasi adalah untuk menyederhanakan atau mengubah representasi suatu citra dijadikan sesuatu yang lebih bermakna dan lebih mudah dianalisis.

MNIST *dataset* (*Modified National Institute of Standards and Technology database)* adalah kumpulan dari data yang tersedia secara global dari tulisan tangan dan angka. Basis data ini terdiri dari 70000 digit tulisan tangan yang terdiri dari 60000 data training dan 10000 data testing yang digunakan untuk melatih berbagai *system* pengolahan gambar diantaranya melatih teknik *machine learning* [3] dan *deep learning* untuk segmentasi gambar dan pola [4]. Proses pengenalan gambar tulisan tangan pada dataset MNIST menggunakan algoritma *k-means* diawali dengan proses c*lustering*.

Algoritma *k-means* dalam penerapannya memerlukan tiga parameter yang seluruhnya ditentukan pengguna yaitu jumlah *cluster k*, inisiasi *cluster*, dan jarak system, Biasanya, *k-means* dijalankan secara independen dengan inisiasi yang berbeda menghasilkan *cluster* akhir yang berbeda karena algoritma ini secara prinsip hanya mengelompokan data menuju *local minimal*. Salah satu cara untuk mengatasi *local minimal* adalah dengan mengimplementasikan algoritma *k-means*, untuk *K* yang diberikan, dengan beberapa nilai initial partisi yang berbeda dan selanjutnya dipilih partisi dengan kesalahan kuadrat terkecil [5]. Dalam penerapan algoritma *k-means*, jika diberikan sekumpulan data $X = \{x_1, x_2, x_n\}$ di mana $xi = (x_{i1}, x_{i2}, \ldots, x_{in})$ adalah system dalam ruang real *Rn*, maka algoritma *k-means* akan menyusun partisi *X* dalam sejumlah *k* [6]. Setiap *cluster* memiliki titik tengah

(*centroid)* yang merupakan nilai rata rata (*mean)* dari data dalam *cluster* tersebut. Tahapan awal, algoritma *k-means* adalah memilih secara acak *k* buah obyek sebagai *centroid* dalam data. Kemudian, jarak antara obyek dan *centroid* dihitung menggunakan *Euclidian distance*. Rumus yang digunakan untuk menghitung *Euclidian distance* adalah sebagai berikut [7]:

$$D\ (a+b) = \sum_{i=0}^{n} (bi - ai)^2 \quad (1)$$

Algoritma *k-means* secara *iterative* meningkatkan variasi nilai dalam dalam tiap tiap *cluster* di mana obyek selanjutnya ditempatkan dalam kelompok yang terdekat, dihitung dari titik tengah *cluster*. Titik tengah baru ditentukan bila semua data telah ditempatkan dalam *cluster* terdekat. Proses penentuan titik tengah dan penempatan data dalam *cluster* diulangi sampai nilai titik tengah dari semua *cluster* yang terbentuk tidak berubah lagi [8] .

Mini batch k-means adalah varian dari algoritma *k-means* yang menggunakan *mini batch* untuk mengurangi waktu komputerisasi, sambil tetap berusaha untuk mengoptimalkan fungsi tujuan yang sama Pemikiran utama *mini batch k-means* adalah menggunakan *batch* acak kecil dari kumpulan data dengan ukuran tetap sehingga dapat disimpan dalam memori . *Batch mini* ini secara drastis mengurangi jumlah komputerisasi yang diperlukan untuk menyatu ke solusi *local* [9]. Berbeda dengan algoritma lain yang mengurangi waktu konvergensi *k-means*, *mini batch k-means* menghasilkan hasil yang umumnya hanya sedikit lebih buruk daripada algoritma standar [10]. Saat jumlah iterasi meningkat, efek dari contoh baru berkurang, sehingga konvergensi dapat dideteksi ketika tidak ada perubahan dalam cluster yang terjadi dalam beberapa iterasi yang berurutan. Misalkan kumpulan data dan *centroid* sama, ambil sebanyak-banyaknya data acak dari himpunan X, yaitu dicontohkan M, di mana variabel menggambarkan ukuran tumpukan mini. Jadi, fungsi tujuan dari metode ini adalah:

$$n\ f(\mu\ ,x) = \sum_{Xn \in M} \sum_{k} ||\mu k - Xn||^2 \quad (2)$$

Dan proses pembaruan *centroid* akan digunakan

$$\mu ki + 1 = (1 -)\mu ki + \eta kXn \quad (3)$$

dimana ηk adalah kecepatan pembelajaran yang diperbarui di setiap iterasi, hal ini juga bertujuan untuk mengetahui seberapa kecil ukuran *batch* yang membuat hasil sebanding dengan *k-means* asli [11].

Sedangkan untuk permasalah inisiasi *centroid* pada *k-means* untuk data yang berukuran besar, banyak penelitian yang mengatakan bahwa metode *k-means* ++ [12] [13] bisa bekerja dengan baik bila dibandingkan dengan inisiasi *centroid* dengan *k-means* standar. Dengan menambahkan rumus *randomized seeding technique* maka akan lebih menentukan nilai *centroid* pada saat awal perhitungan [12].

$$\frac{D(x')^2}{\sum x \in X\ D(x')^2} \quad (4)$$

Rumus *randomized seeding technique* akan menghasilkan sebuah angka yang akan dijadikan patokan semakin jauh nilai objek maka semakin besar kemungkinan nilai objek akan dijadikan nilai *centriod* berikutnya.

II. METODE PENELITIAN

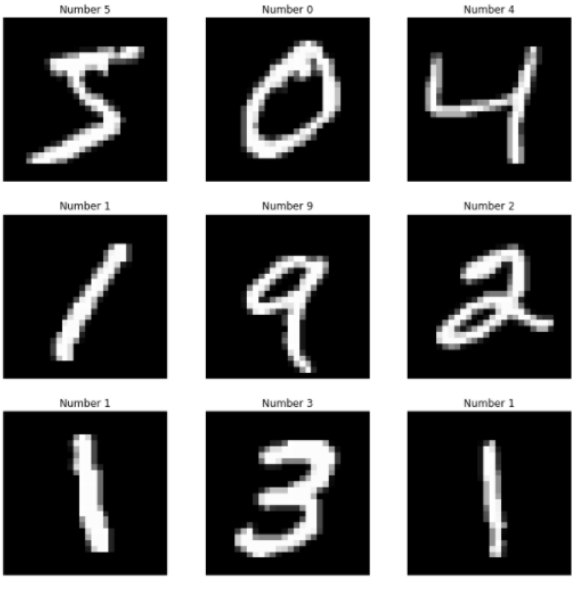
Pada bagian ini akan dijelaskan bagaimana tahapan dan metode yang dilakukan untuk mencapai tujuan dari penelitian yang akan dilakukan.

A. Pembagian data

Pada proses ini dilakukan pembagian data pada bagian ini adalah pengolahan *dataset* yang sudah terkumpul dengan membagi data tersebut menjadi x_train, x_test dan y_train, y_test yang kemudian akan dijadikan sebagai data training dan testing. *Output* nya adalah (60000,28,28), (10000,28,28), (60000,1), (10000,1). 'x_train' dan 'x_test' masing-masing terdiri dari 60000 dan 10.000 gambar monokrom. Setiap masukan gambar memiliki keluaran berupa angka yang ditampilkan pada gambar, 'y_train' dan 'y_test' memiliki ukuran (60000,1) dan (10000,1). Berikut adalah hasil dari pembagian *dataset* tersebut.

B. Scaling

Pada awalnya semua gambar memiliki ukuran pixel yang berbeda, melalui proses s*caling* gambar tersebut di reduksi dijadikan ukuran pixel umum 28 x 28. Pada proses tersebut detail gambar akan hilang karena pengurangan ukuran pixel dan karenanya gambar menjadi kabur. Output nya adalah 5, 0, 4, 1, 9. 'y_train' dan 'y_test' adalah angka 0 hingga 9 yang menunjukan angka yang ditampilkan pada gambar 4.1.



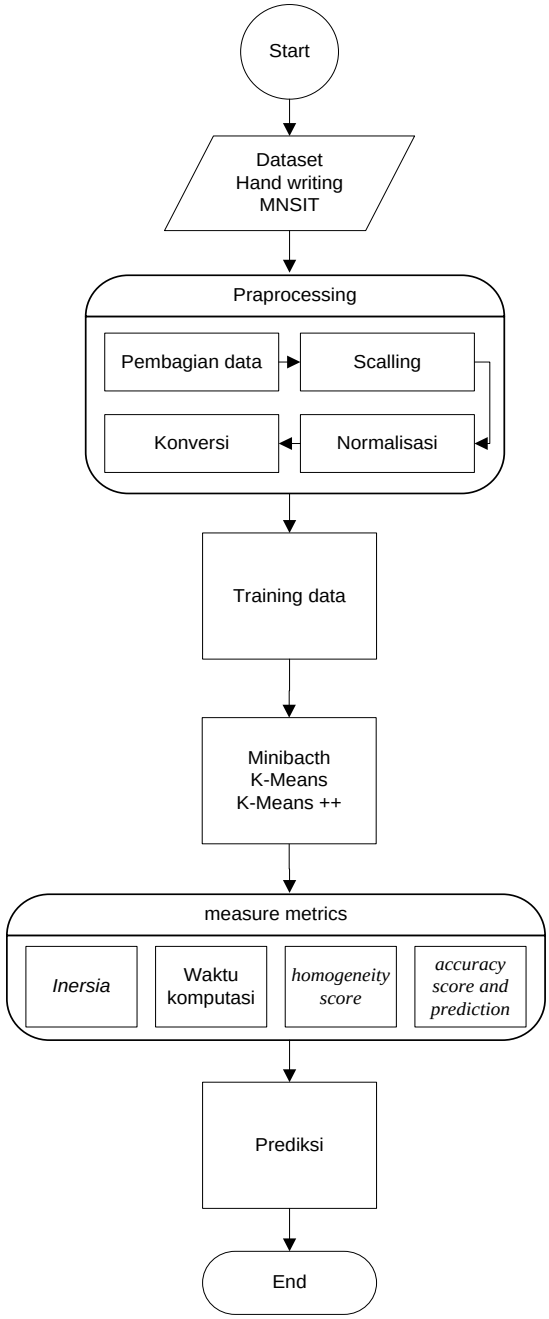
Gambar 1. Contoh hasil *scaling data*

C. Normalisasi

Proses normalisasi yaitu merupakan proses pengubahan data dijadikan bentuk normal. Normalisasi diperlukan ketika data yang ada bernilai sangat besar, sangat kecil, atau memiliki satuan yang berbeda. Pada proses ini dilakukan normalisasi Min-Max. Pada dataset MNIST Nilai minimum dan maksimum masing-masing adalah 0 dan 255. Dalam ruang warna RGB, merah, hijau dan biru menggunakan 8bit yang masing-masing memiliki nilai integer dari 0 sampai 255. Jumlah warna yang memungkinkan adalah 256 * 256 * 256 = 16777216. Karena dataset berisi rentang nilai dari 0 ke 255, dataset harus dilakukan normalisasi. Normalisasi data adalah langkah penting yang memastikan bahwa setiap parameter input (*pixel*, dalam hal ini) memiliki distribusi data yang serupa, ini mempercepat proses menutupi saat melatih model. Normalisasi juga memastikan tidak ada parameter tertentu yang mempengaruhi keluaran secara signifikan. Agar data hasil preprocessing dapat di training dengan algoritma *k-means clustering* data tersebut harus di konversi dari format 3 dimensi ke format 2 dimensi. Oleh karena itu data input harus dibentuk kembali.

D. Model Training

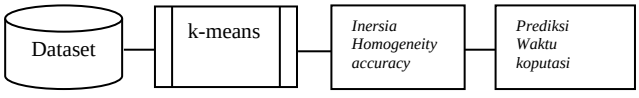
Model yang dibangun menggunakan *mini batch k-means* yang bekerja mirip dengan algoritma *k-means*. Perbedaanya adalah bahwa dalam *mini batch k-means* langkah yang paling lama dilakukan pada saat pengumpulan data acak kecil dengan ukuran tetap. Pendekatan ini dapat secara signifikan mengurangi waktu yang dibutuhkan algoritma. Kemudian untuk meningkatkan hasil akurasi digunakan *k-means* ++ untuk inisiasi *centroid.* Kemudian model training akan diukur dengan 3 *measure metrics inersia, homogeneity score* dan *accuracy score* serta waktu komputerisasi yang dibutuhkan.



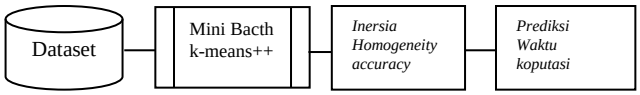
Gambar 2. Model Training Data

E. Aritektur Model

Berikut adalah arsitektur model dengan menggunakan perbandingan model training data *k-means* dan optimasi *mini batch k-means* dan *k-means* ++. Dari model training akan diukur dengan 3 *measure metrics inersia, homogeneity score* dan *accuracy score* serta waktu komputerisasi yang dibutuhkan.



Gambar 3. Arsitektur Model *K-means*



Gambar 4. Arsitektur mini batch k-means dan k-means++

F. Pengukuran

Pada tahap ini adalah melakukan pengukuran untuk setiap eksperimen yang dilakukan guna mencapai *performance* mana yang lebih baik. Pengukuran menggunakan *inertia, homogeneity score* dan *accuracy score*

a) Inersia dalam *cluster* adalah menghitung nilai jumlah kuadrat dalam *cluster* [14].

$$\sum_{i=1}^{n}(xi - Ck)^2 \quad (5)$$

Dimana *n* adalah jumlah sample dalam kumpulan data, dan C adalah pusat clustering. Inersia hanya menghitung jarak kuadrat dari setiap sampel adalah sebuah cluster ke pusat *cluster* yang menjumlahkannya. Ketika cluster ditambahkan sebanyak sampel dalam dataset, maka nilai inersia akan menjadi nol.

b) *Homogeneity* adalah *matrix cluster* yang digunakan untuk menentukan homogenitas titik-titik data dalam satu cluster [15].

$$\mathrm{h} = \frac{H\ (C,K)}{H(C)} \quad (6)$$

Dimana H (C, K) adalah entropi bersyarat dari kelas dari sebuah cluster

$$H\ (C,K) = -\sum_{k=1}^{|K|}\sum_{c=1}^{|C|}\frac{n_{c,k}}{n}\log\frac{n_{c,k}}{n_k} \quad (7)$$

Dan H(C) adalah entropi kelas dan diberikan oleh:

$$H\ (C) = -\sum_{c=1}^{|C|}\frac{n_c}{n}\log\frac{n_c}{n} \quad (8)$$

Dimana n adalah jumlah titik data, nc adalah jumlah titik data yang memiliki class c, nk jumlah titik data masing-masing milik cluster k, dan nc, k adalah perbandingan antara jumlah sampel berlabel c pada *cluster* k dan jumlah sampel pada *cluster* k. jika semua sampel dalam cluster k memiliki label yang sama c, homogenitasnya sama dengan 1.

c) Accuracy

Accuracy adalah proporsi jumlah diprediksikan yang benar.

$$acc = \frac{TP + TN}{(TP + TN + FP + FN)} \quad (9)$$

d) Waktu komputasi adalah waktu yang dibutuhkan pada saat menjalankan model training data.

III. HASIL DAN PEMBAHASAN

Pada bab ini akan dibahas hasil percobaan dari hasil training *dataset* MNIST dan akurasi yang didapat serta kecepatan komputerisasi dengan menggunakan algoritman k-*means* dan metode optimasi *mini batch k-means* dan *k-means* ++. Gambar 3.4 dan Gambar 3.5 membahas mengenai arsitektur model untuk penggunaan algoritma *k-means* tanpa optimasi dan model arsitektur *mini batch k-means* dan *k-means*++, berikut adalah tabel hasil dari model yang telah dilakukan training.

TABEL I
HASIL TRAINING MODEL K-MEANS

Number of Cluster	K-Means		
	Inertia	Homogeneity	Accuracy
10	15299	48,51%	59,07%
16	14201	58,83%	69,25%
36	12561	70,00%	77,78%
64	11587	75,96%	83,63%
144	10378	83,00%	89,71%
256	9586	85,82%	91,14%
Total waktu yang diperlukan	29 menit 39 detik		

TABEL III
HASIL TRAINING MODEL K-MEANS OPTIMASI MINI BATCH DAN K-MEANS ++

Number of Cluster	k-means optimasi mini batch dan k-means++		
	Inertia	Homogeneity	Accuracy
10	0,00056	44,87%	52,81%
16	0,00052	55,56%	62,60%
36	0,00046	69,18%	77,74%
64	0,00042	73,31%	79,66%
144	0,00038	80,39%	86,55%
256	0,00035	84,53%	90,13%
Total waktu yang diperlukan	1 Menit 26 detik		

IV. HASIL DAN PEMBAHASAN

Berdasarkan penelitian yang sudah dilakukan untuk meningkatkan performa *k-means* terutama untuk mempersingkat waktu komputerisasi dalam proses segentasi gambar tulisan pada *dataset* MNIST dengan

cara melakukan optimasi dengan *mini bacth k-means* dan *k-means++* untuk inisiasi *centroid*, maka dapat disimpulkan bahwa *mini batch k-means* dan *k-means++* dapat meningkatkan performa algoritma *k-means* dalam proses pengenalan tulisan tangan pada dataset MNIST dengan akurasi hampir sebanding sebesar 1,01% yaitu 91,14% menjadi 90,13% namun waktu komputerisasi jauh lebih singkat 28 menit 13 detik yaitu 29 menit 39 detik menjadi 1 menit 26 detik.

SARAN

Berdasarkan hasil pengujian terlihat bahwa penambahan algoritma *mini batch* dan *k-means++* dapat mempersingkat waktu komputerisasi pada algoritma *k-means*. Disarankan *research* lanjutan untuk *improve mini batch k-means* dan *k-means++* agar akurasi dapat meningkat lebih baik lagi dari hasil pengujian yang didapat.

Adapun saran untuk penelitian selanjutnya disarankan dapat menggunakan metode clustering lainnya, menggunakan pemilihan fitur *metaheuristic* terbaru lainnya, ataupun menggunakan dataset lainnya agar penggunaan *k-means* dengan optimasi *mini batch* dan *k-means++* memang benar lebih baik dari pada *k-means* tanpa optimasi *mini batch dan k-means++* untuk inisiasi *centroid*.

DAFTAR PUSTAKA

[1] Z. Ge, Z. Song, S. X. Ding, and B. Huang, "Data Mining and Analytics in the Process Industry: The Role of Machine Learning," *IEEE Access*, vol. 5, pp. 20590–20616, 2017, doi: 10.1109/ACCESS.2017.2756872.

[2] J. Gan, W. Wang, and K. Lu, "Compressing the CNN architecture for in-air handwritten Chinese character recognition," *Pattern Recognit. Lett.*, vol. 129, pp. 190–197, 2020, doi: 10.1016/j.patrec.2019.11.028.

[3] A. F. Agarap and A. P. Azcarraga, "Improving k-Means Clustering Performance with Disentangled Internal Representations," *Proc. Int. Jt. Conf. Neural Networks*, no. 2016, 2020, doi: 10.1109/IJCNN48605.2020.9207192.

[4] W. Jiang, "MNIST-MIX: a multi-language handwritten digit recognition dataset," *IOP SciNotes*, vol. 1, no. 2, p. 025002, 2020, doi: 10.1088/2633-1357/abad0e.

[5] M. Vafadar *et al.*, "Unsupervised K-means clustering algorithm," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 8, no. 3, pp. 1–12, 2019, doi: 10.1109/ACCESS.2020.2988796.

[6] N. Dhanachandra, K. Manglem, and Y. J. Chanu, "Image Segmentation Using K-means Clustering Algorithm and Subtractive Clustering Algorithm," *Procedia Comput. Sci.*, vol. 54, pp. 764–771, 2015, doi: 10.1016/j.procs.2015.06.090.

[7] Soeren Bergmann Niclas Feldkamp and S. Strassburger, "APPROXIMATION OF DISPATCHING RULES FOR MANUFACTURING SIMULATION USING DATA MINING METHODS," *Dk*, vol. 53, no. 9, pp. 1689–1699, 2015, doi: 10.1017/CBO9781107415324.004.

[8] V. Kathiresan and P. Sumathi, "An efficient clustering algorithm based on Z-Score ranking method," *2012 Int. Conf. Comput. Commun. Informatics, ICCCI 2012*, pp. 10–13, 2012, doi: 10.1109/ICCCI.2012.6158779.

[9] K. Peng, V. C. M. Leung, and Q. Huang, "Clustering Approach Based on Mini Batch Kmeans for Intrusion Detection System over Big Data," *IEEE Access*, vol. 6, pp. 11897–11906, 2018, doi: 10.1109/ACCESS.2018.2810267.

[10] G. S. Thejas, S. R. Joshi, S. S. Iyengar, N. R. Sunitha, and P. Badrinath, "Mini-Batch Normalized Mutual Information: A Hybrid Feature Selection Method," *IEEE Access*, vol. 7, no. Mi, pp. 116875–116885, 2019, doi: 10.1109/ACCESS.2019.2936346.

[11] S. R. Fitriyani and H. Murfi, "The K-means with mini batch algorithm for topics detection on online news," *2016 4th Int. Conf. Inf. Commun. Technol. ICoICT 2016*, vol. 4, no. c, 2016, doi: 10.1109/ICoICT.2016.7571914.

[12] D. Arthur and S. Vassilvitskii, "K-means++: The advantages of careful seeding," *Proc. Annu. ACM-SIAM Symp. Discret. Algorithms*, vol. 07-09-Janu, pp. 1027–1035, 2007.

[13] S. Ahlawat and A. Choudhary, "Hybrid CNN-SVM Classifier for Handwritten Digit Recognition," *Procedia Comput. Sci.*, vol. 167, no. 2019, pp. 2554–2560, 2020, doi: 10.1016/j.procs.2020.03.309.

[14] Y. Wang, "Research on Moment of Inertia Measurement Method," *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 592, no. 1, 2019, doi: 10.1088/1757-899X/592/1/012078.

[15] O. Faker and E. Dogdu, "Intrusion detection using big data and deep learning techniques," *ACMSE 2019 - Proc. 2019 ACM Southeast Conf.*, pp. 86–93, 2019, doi: 10.1145/3299815.3314439.