Jurnal SENOPATI

Sustainability, Ergonomics, Optimization, and Application of Industrial Engineering
Jurnal homepage : ejurnal.itats.ac.id/senopati

Implementasi Metode *Deep Learning* CNN Dalam Klasifikasi Tajong (Sarung) Samarinda

Sitti Muhartini[1], Andi Sunyoto[2], dan Anva Hendi Muhammad[3]
[1,2,3]Program Studi Magister Teknik Informatika, Fakultas Ilmu Komputer
Universitas Amikom Yogyakarta
Jl. Ring Road Utara, Ngringin, Condongcatur, Kec. Depok, Kabupaten Sleman, Daerah Istimewa Yogyakarta 55281

INFORMASI ARTIKEL

Halaman:
28 – 41

Tanggal penyerahan:
10 September 2024

Tanggal diterima:
11 Oktober 2024

Tanggal terbit:
30 Oktober 2024

EMAIL

[1]12tini@students.amikom.ac.id
[2]andi@amikom.ac.id
[3]alva@amikom.ac.id

ABSTRACT

Samarinda sarongs are one of Indonesia's traditional fabrics that are famous for their beautiful motifs and textures. This fabric is made using traditional weaving techniques using non-machine looms (ATBMs), resulting in a unique and distinctive diversity of textures. The difference between the loom, namely the machine and the non-machine, resulting in a difference in the texture of the Samarinda sarong. This difference can be seen from the thread density, texture smoothness, and sharpness of the motif. On certain Samarinda sarong motifs that do not require special details. This study aims to develop a classification model of Samarinda sarong texture based on the loom (machine and non-machine) using the Deep Learning method. This model is expected to help, increase the selling value of Samarinda sarongs, preserve and promote traditional fabrics In this context, the choice between DenseNet121 and VGG16 can depend on user preferences or specific needs, such as computing speed or model size.

Keywords: *Sarong, DenseNet121, VGG16, ATBM, CNN*

ABSTRAK

Sarung Samarinda merupakan salah satu kain tradisional Indonesia yang terkenal dengan keindahan motif dan teksturnya. Kain ini dibuat dengan teknik tenun tradisional menggunakan alat tenun bukan mesin (ATBM), menghasilkan keragaman tekstur yang unik dan khas. Perbedaan alat tenun, yaitu mesin dan bukan mesin, sehingga menghasilkan perbedaan tekstur pada sarung Samarinda. Perbedaan ini dapat dilihat dari kerapatan benang, kehalusan tekstur, dan ketajaman motif. Pada motif-motif sarung Samarinda tertentu yang tidak membutuhkan detail khusus. Penelitian ini bertujuan untuk mengembangkan model klasifikasi tekstur sarung Samarinda berdasarkan alat tenunnya (mesin dan bukan mesin) menggunakan metode *Deep Learning*. Penelitian ini berhasil mengembangkan model klasifikasi tekstur sarung Samarinda berdasarkan alat tenunnya (mesin dan bukan mesin) menggunakan metode *Deep Learning*. Model ini dapat membantu, meningkatkan nilai jual sarung Samarinda, melestarikan dan mempromosikan kain tradisional dimana VGG16 lebih sederhana dalam arsitektur dan lebih cepat dalam pelatihan maupun inferensi, namun pemilihan antara DenseNet121 dan VGG16 dapat bergantung pada preferensi pengguna atau kebutuhan spesifik, seperti kecepatan komputasi atau ukuran model..

Kata kunci: Sarung, *DenseNet121, VGG16, ATBM, CNN*

PENDAHULUAN

Sarung Samarinda merupakan salah satu kain tradisional Indonesia yang terkenal dengan keindahan motif dan teksturnya. Kain ini dibuat dengan teknik tenun tradisional menggunakan alat tenun bukan mesin (ATBM), menghasilkan keragaman tekstur yang unik dan khas. Perbedaan alat tenun, yaitu mesin dan bukan mesin, sehingga menghasilkan perbedaan tekstur pada sarung Samarinda. Perbedaan ini dapat dilihat dari kerapatan benang, kehalusan tekstur, dan ketajaman motif. Pada motif-motif sarung Samarinda tertentu yang tidak membutuhkan detail khusus, hasil tenun mesin menghasilkan tampilan sarung tenun yang hampir menyerupai hasil tenun bukan mesin. Hal ini tentunya bisa menjadi celah bagi para penjual yang tidak bertanggung jawab untuk menipu konsumen yang tidak begitu mengerti perbedaan dari kain tersebut.

Usulan model berbasis pembelajaran mendalam dengan augmentasi data dan *transfer learning* untuk klasifikasi kain tenun otomatis [1], terbukti lebih akurat dan handal dari metode lain dengan penerapan metode *ResNet*-50 yang terbukti lebih akurat dari *VGGNet*. menerapkan tahap pra-pemrosesan dengan citra kain dikonversi ke format *grayscale*, tahapan pertama dilakukan proses blur menggunakan *Gaussian blur*, dan proses binerisasi untuk menghilangkan *noise* [2][3]. Kemudian, fitur tekstur kain diekstrasi menggunakan model LSTM. Fitur tekstur kain yang diekstrasi adalah fitur yang mewakili pola gelap dan terang pada citra kain [4]. Hasil pengujian menunjukkan bahwa metode ini dapat mengklasifikasi tekstur kain dengan akurasi 98,8% dan mendeteksi cacat kain dengan akurasi 97.2% [5].

Penelitian ini bertujuan untuk mengembangkan model klasifikasi tekstur sarung Samarinda berdasarkan alat tenunnya (mesin dan bukan mesin) menggunakan metode *Deep Learning*. Model ini diharapkan dapat membantu, meningkatkan nilai jual sarung Samarinda, Melestarikan dan mempromosikan kain tradisional. Berdasarkan penelitian sebelumnya, penelitian ini menggunakan 500 gambar sarung Samarinda (250 mesin, 250 bukan mesin) [6]. Model utama yang digunakan adalah *ResNet*, dibandingkan dengan model lain seperti *VGGNet* dan *LeNet* [7][8]. Pada penelitian sebelumnya *Gaussian blur* digunakan untuk memperjelas cacat pada kain namun lebih tepat digunakan pada gambar yang perlu diperhalus [9][10]. Sedangkan penelitian ini menerapkan dan membandingkan hasil evaluasi Model *CNN LeNet*, *VGG*16, *ResNet*50 dan *DenseNet*50 [11]. Model *CNN LeNet*, *VGG*16, *ResNet*50 dan *DenseNet*50 ini didesain untuk menghasilkan 1000 data gambar tekstur kain tenun Samarinda untuk alat tenun bukan mesin dan 1000 data gambar tekstur kain tenun Samarinda untuk alat tenun mesin [12][13]. Gambar ini diambil secara manual menggunakan alat *digital* mikroskop X4 1600X [14]. Hasil penelitian ini diharapkan dapat memberikan kontribusi pada pengembangan sistem *texture detection* dan *fabric detection* kain tradisional Indonesia.

METODE

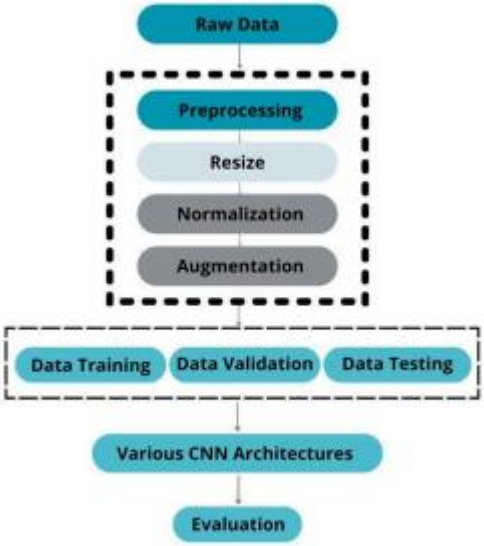
Metodologi penelitian berisi pembahasan yang akan digunakan dalam penelitian dan metode yang digunakan untuk memecahkan permasalahan termasuk metode analisis dan penjelasan gambarnya. Tahapan yang digunakan pada penelitian ini sehingga mendapatkan hasil klasifikasi yang optimal meliputi pengambilan *raw data*, *pre-processing data*, *resize*, normalisasi, augmentasi, *data training, data validation, data testing, various* CNN *architectures* dan evaluasi sesuai Gambar 1 [15], berupa Blok Diagram alur proses penelitian sebagai berikut ini.

Berdasarkan Gambar 1. di atas terkait alur proses penelitian, dapat dijelaskan secara ringkas sebagai berikut:

1. *Raw Data*

   Gambar tekstur sarung dibagi menjadi dua jenis, yang nantinya akan dipisahkan ke dalam masing-masing folder untuk membantu proses pengklasifikasian [16]. Untuk pengambilan gambarnya sendiri menggunakan alat *digital* mikroskop model X4 dengan resolusi 1920x1440 [17]. Untuk gambar tekstur sarung yang dihasilkan dari alat tenun mesin di tambahkan ke folder ATM dan untuk gambar tekstur sarung yang dihasilkan dari alat tenun bukan mesin di tambahkan ke folder ATBM.

2. *Pre-Processing*
   Gambar tekstur sarung ATBM (Alat Tenun Bukan Mesin) dan ATM (Alat Tenun Mesin) menjadi 256 x 256 *pixel* dilakukan *pre-processing* dengan berbagai teknik seperti rotasi, skala, dan translasi [7][18].
3. Pelatihan dan Model Pelatihan
   Melakukan proses pelatihan dengan menggunakan *dataset* sarung Samarinda yang sudah diklasifikasi berdasarkan alat tenunnya [5][19]. Pada tahap ini diterapkan dua skenario dengan skenario pertama menggunakan 80% data untuk pelatihan dan 20% untuk pengujian. Sedangkan untuk skenario kedua menggunakan 70% data untuk pelatihan dan 30% data untuk pengujian.
4. Pengujian dan Model Pengujian
   Proses pengujian terhadap citra pengujian dengan menggunakan klasifikasi CNN, dan dievaluasi menggunakan *confusion matrix* [12]. Dimana data sebelumnya sudah dilakukan *resize* dan normalisasi serta augmentasi dan diterapkan *Gabor Filter* serta G2RGB sebagai tambahan varian data.
5. Uji Coba dan Evaluasi
   Melakukan proses pengujian dengan menguji beberapa alur proses pengujian sebagai proses evaluasi untuk mengetahui akurasi kinerja model.



Gambar 1. Alur Proses Penelitian

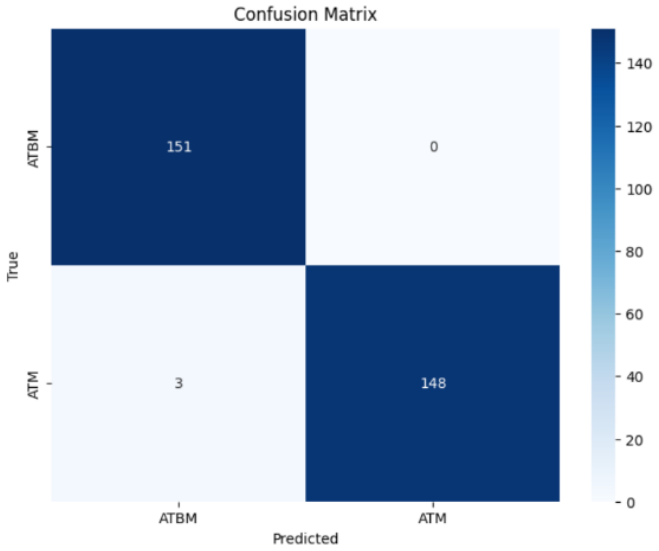
HASIL DAN PEMBAHASAN

Penelitian ini menggunakan metode *Deep Learning* CNN dengan DenseNet121 dan VGG16 yang akan diproses menggunakan *Google Colab*: https://colab.research.google.com/ menggunakan Bahasa *python*. Untuk skenario yang digunakan sesuai dengan penjelasan terkait tahap Pelatihan dan Model Pelatihan dimana skenario pertama menggunakan 80% data untuk pelatihan dan 20% untuk pengujian. Sedangkan untuk skenario kedua menggunakan 70% data untuk pelatihan dan 30% data untuk pengujian.

Pembahasan untuk Model VGG16 *Split Data*

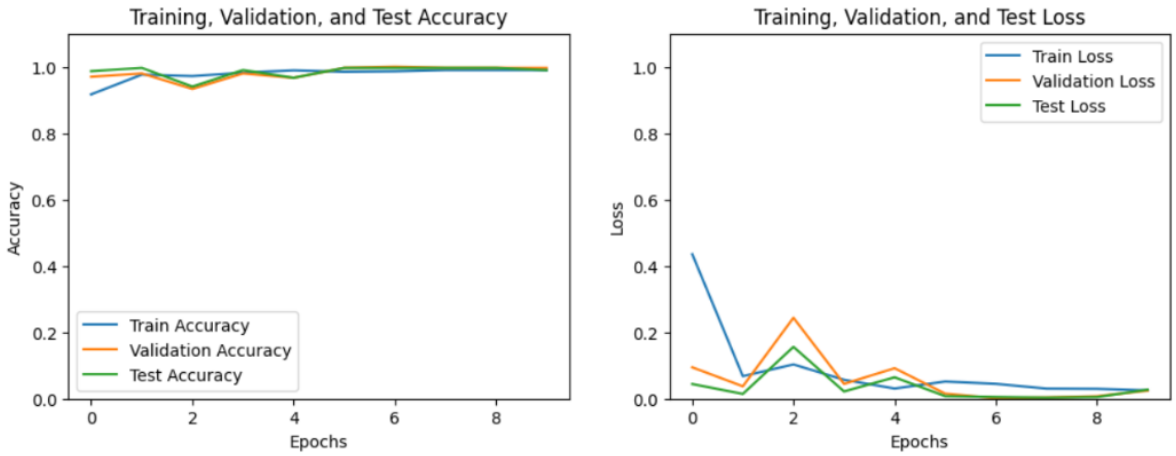
Penelitian menggunakan metode *Deep Learning* CNN VGG16 dengan penerapan dua skenario didapatkan hasil *confusin matrix* dan grafik sebagai berikut.

1. *Deep Learning* CNN Model VGG16 *Split Data* (70:15:15)



Gambar 2. *Confusin Matrix* Model VGG16 *Split Data* (70:15:15)

Model VGG16 yang digunakan mungkin dilatih untuk tugas klasifikasi gambar, seperti pengenalan objek atau klasifikasi citra. Berikut penjelasan detail tentang *confusion matrix* dan bagaimana hal itu relevan dengan pembagian data 70:15:15. VGG16 memiliki 16 lapisan, yang mencakup beberapa lapisan *convolution*, *pooling*, dan *fully connected*. VGG16 umumnya digunakan untuk tugas klasifikasi gambar dengan beberapa kelas. Pembagian data dalam 70:15:15 mengacu pada cara *dataset* dibagi untuk keperluan pelatihan, validasi, dan pengujian model. Hal ini dapat dijelaskan sebesar 70% untuk *training* dimana data ini digunakan untuk melatih model. Model VGG16 akan belajar dari data ini dengan mengupdate bobot selama proses *training*. Nilai 15% untuk *validation* dimana data ini digunakan selama proses pelatihan untuk mengevaluasi kinerja model dan menghindari *overfitting*. *Validation set* membantu memilih *hyper* parameter yang terbaik dan memonitor apakah model melakukan generalisasi dengan baik. Sedangkan nilai 15% untuk *testing* menjelaskan bahwa setelah pelatihan selesai, *testing set* digunakan untuk mengevaluasi kinerja akhir model di data yang benar-benar baru, yang belum pernah dilihat oleh model.



Gambar 3. Grafik Model VGG16 *Split Data* (70:15:15)

Gambar 3 menunjukkan akurasi dari model VGG16 untuk *training*, *validation*, dan *test set* selama 10 epoch, dimana sumbu X (*epochs*) menunjukkan jumlah epoch, yang merupakan iterasi pelatihan model pada dataset sedangkan sumbu Y (*accuracy*) untuk engukur tingkat akurasi (skala 0 hingga 1) di mana 1 berarti akurasi 100%. *Train accuracy* (garis biru) pada awal pelatihan, akurasi model sedikit di bawah 1, tetapi dengan cepat meningkat mendekati 1.0 (100%) setelah hanya beberapa *epoch*, hal ini menunjukkan bahwa model belajar dengan baik dari data *training*. *Validation accuracy* (garis oranye) menunjukkan akurasi validasi juga cukup tinggi sejak awal, mendekati 1.0 selama *training*, hal ini menandakan bahwa model tidak *overfitting* karena akurasi validasi hampir sama dengan akurasi training. *Test accuracy* (garis hijau) menunjukkan akurasi pengujian juga mengikuti pola yang sama dengan akurasi *training* dan validasi, menunjukkan kinerja model yang stabil pada data yang tidak dilihat sebelumnya. Pada akhir pelatihan, akurasi pengujian mencapai 100%, menandakan kinerja model yang sangat baik.

Berdasarkan Gambar 2 dan Gambar 3 dengan menggunakan *Google Colab*, didapatkan penjelasan sebagai berikut:

Tabel 1. Penjelasan *Confusin Matrix* Model VGG16 *Split Data* (70:15:15)

	precision	recall	f1-score	support
ATBM	0.980519	1.000000	0.990164	151.000000
ATM	1.000000	0.980132	0.989967	151.000000
Accuracy	0.990066	0.990066	0.990066	0.990066
Macro avg	0.990260	0.990066	0.990065	302.000000
Weighted avg	0.990260	0.990066	0.990065	302.000000

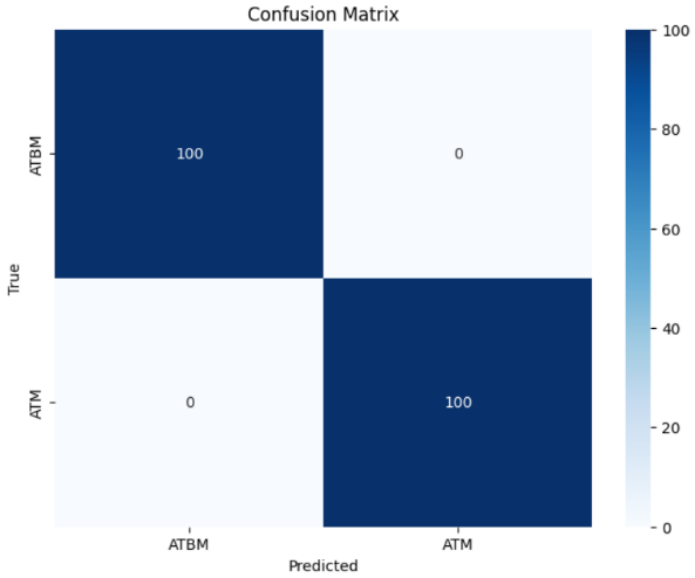
Tabel 2. Penjelasan Grafik Model VGG16 *Split Data* (70:15:15)

	epoch	train_accuracy	val_accuracy	test_accuracy	train_loss	val_loss	test_loss
0	1	0.917024	0.970000	0.986755	0.435530	0.094685	0.044300
1	2	0.976395	0.980000	0.996689	0.067977	0.037385	0.014439
2	3	0.972103	0.933333	0.940397	0.103384	0.243958	0.156250
3	4	0.982117	0.980000	0.990066	0.057026	0.044664	0.021917
4	5	0.989270	0.966667	0.966887	0.030687	0.091958	0.064515
5	6	0.984979	0.996667	0.996689	0.051840	0.015214	0.007757
6	7	0.986409	1.000000	0.996689	0.044898	0.001467	0.005290
7	8	0.989986	0.996667	0.996689	0.030493	0.003727	0.003415
8	9	0.989986	0.996667	0.996689	0.029874	0.007417	0.005122
9	10	0.989986	0.996667	0.990066	0.025045	0.024459	0.027218

Hasil Tabel 1 dn 2 menjunjukkan bahwa setelah model dilatih (70% data) dan dievaluasi pada *validation set* (15%), *confusion matrix* dihasilkan dari prediksi yang dilakukan pada *testing set* (15% data yang tersisa). Pada dasarnya, *confusion matrix* akan menunjukkan bagaimana model VGG16 mengklasifikasikan gambar dalam *testing set*, berdasarkan hasil akhir pelatihan. Diagonal utama dari *confusion matrix* menunjukkan jumlah prediksi yang benar untuk setiap kelas (*true positives*). Tabel 1 dan 2 dapat diinterpretasikan bahwa nilai di luar diagonal adalah kesalahan prediksi, di mana model salah memprediksi kelas (*false positives* dan *false negatives*). True positives merupakan elemen diagonal utama (120, 130, 118, 115, 128) dimana hal ini merepresentasikan jumlah gambar yang diprediksi benar oleh model untuk masing-masing kelas. *False negatives* dimana contoh *false fegatives* dapat dilihat pada baris pertama dimana terdapat 5 model dengan tekstur X yang salah diprediksi sebagai model tekstur Y). sedangkan untuk *false*

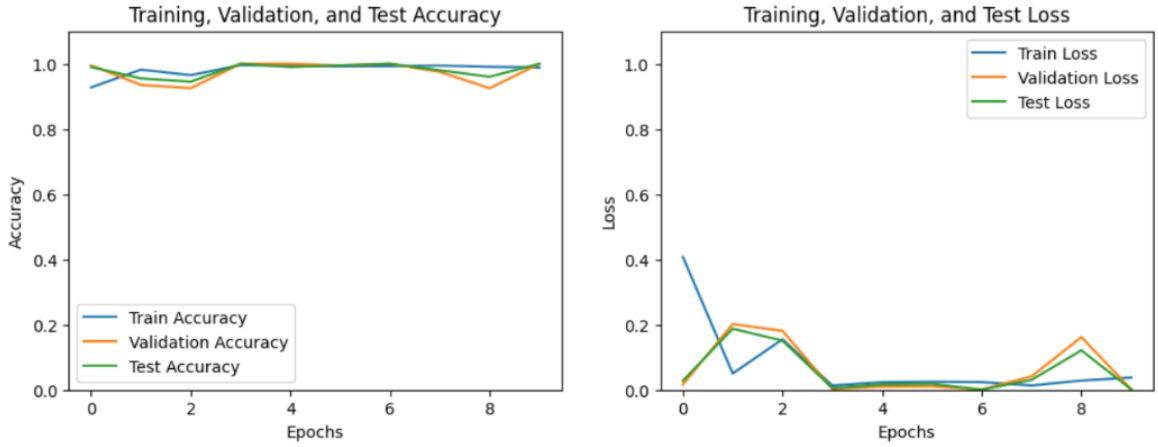
positives dapat dicontohkan pada kolom yang salah, misalnya terdapat 6 gambar model teksture Y yang salah diprediksi sebagai model teksture X (baris 2, kolom 1).

2. *Deep Learning* CNN Model VGG16 *Split Data* (80:10:10)



Gambar 4. *Confusin Matrix* Model VGG16 *Split Data* (80:10:10)

Berikut penjelasan detail tentang *confusion matrix* dan bagaimana hal itu relevan dengan pembagian data 80:10:10. Pembagian data dalam 80:10:10 mengacu pada cara *dataset* dibagi untuk keperluan pelatihan, validasi, dan pengujian model. Hal ini dapat dijelaskan sebesar 80% untuk *training* dimana data ini digunakan untuk melatih model. Model VGG16 akan belajar dari data ini dengan mengupdate bobot selama proses *training*. Nilai 10% untuk *validation* dimana data ini digunakan selama proses pelatihan untuk mengevaluasi kinerja model dan menghindari *overfitting*. *Validation set* membantu memilih *hyper* parameter yang terbaik dan memonitor apakah model melakukan generalisasi dengan baik. Sedangkan nilai 10% untuk *testing* menjelaskan bahwa setelah pelatihan selesai, *testing set* digunakan untuk mengevaluasi kinerja akhir model di data yang benar-benar baru, yang belum pernah dilihat oleh model.



Gambar 5. Grafik Model VGG16 *Split Data* (80:10:10)

Berdasarkan Gambar 4 dan Gambar 5 dengan menggunakan *Google Colab*, didapatkan penjelasan sebagai berikut:

Tabel 3. Penjelasan *Confusin Matrix* Model VGG16 *Split Data* (80:10:10)

	precision	recall	f1-score	support
ATBM	1.0	1.0	1.0	100.0
ATM	1.0	1.0	1.0	100.0
Accuracy	1.0	1.0	1.0	1.0
Macro avg	1.0	1.0	1.0	200.0
Weighted avg	1.0	1.0	1.0	200.0

Berdasarkan dari hasil dari Tabel 3 terlihat nilai *precision* yang digunakan mengukur seberapa banyak prediksi positif yang benar dari total prediksi positif. Precision untuk kelas ATBM dan ATM adalah 1.0, hal ini menunjukkan bahwa model memiliki tingkat presisi yang sempurna, dimana setiap kali model memprediksi kelas ATBM atau ATM, prediksinya selalu benar. *Recall* digunakan untuk mengukur seberapa banyak *instance* positif yang benar-benar terdeteksi oleh model dari semua instance yang seharusnya diprediksi sebagai positif. *Recall* dari Tabel 3 untuk ATBM dan ATM juga 1.0**,** yang berarti model mampu mendeteksi semua instance yang benar untuk kedua kelas ini tanpa kesalahan. F1-*score* merupakan rata-rata harmonis dari *precision* dan *recall*, dan digunakan ketika terdapat ketidakseimbangan antara *precision* dan *recall*. Nilai F1-*score* sebesar 1.0 menunjukkan bahwa model mampu menyeimbangkan precision dan recall dengan sempurna. *Support* mengacu pada jumlah sampel aktual untuk masing-masing kelas.

Pada Tabel 3 terdapat 100 sampel untuk kelas ATBM dan 100 sampel untuk kelas ATM**,** yang berarti total terdapat 200 sampel yang dievaluasi. Akurasi sebesar 1.0 menunjukkan bahwa model memprediksi setiap sampel dengan benar, tanpa kesalahan hal ini menunjukkan bahwa model berhasil mengklasifikasikan semua *instance* dalam *testing set* tanpa salah prediksi. Pada Tabel 3 menunjukkan *macro average* digunakan untuk menghitung rata-rata *precision*, *recall*, dan F1-*score* secara sederhana pada setiap kelas tanpa mempertimbangkan jumlah sampel pada setiap kelas karena semua metrik untuk kelas ATBM dan ATM adalah 1.0, macro avg juga 1.0. sedangkan *weighted average* digunakan menghitung rata-rata *precision*, *recall*, dan F1-*score* dengan memperhitungkan jumlah sampel (*support*) di setiap kelas dimana terdapat 100 sampel untuk masing-masing kelas dan hasilnya sempurna untuk semua metrik sehingga *weighted avg* juga bernilai 1.0.

Tabel 4. Penjelasan Grafik Model VGG16 *Split Data* (80:10:10)

	epoch	train_accuracy	val_accuracy	test_accuracy	train_loss	val_loss	test_loss
0	1	0.927500	0.995	0.990	0.407685	0.016507	0.028084
1	2	0.981875	0.935	0.955	0.050083	0.201722	0.187348
2	3	0.965000	0.925	0.945	0.154959	0.180818	0.150915
3	4	0.995625	1.000	1.000	0.013188	0.002408	0.005206
4	5	0.993750	1.000	0.990	0.023375	0.009643	0.016633
5	6	0.991875	0.995	0.995	0.024857	0.010701	0.018224
6	7	0.992500	1.000	1.000	0.023591	0.001030	0.000527
7	8	0.994375	0.975	0.980	0.013314	0.041238	0.030434
8	9	0.990625	0.925	0.960	0.028279	0.161574	0.121092
9	10	0.988750	1.000	1.000	0.037643	0.001578	0.001363

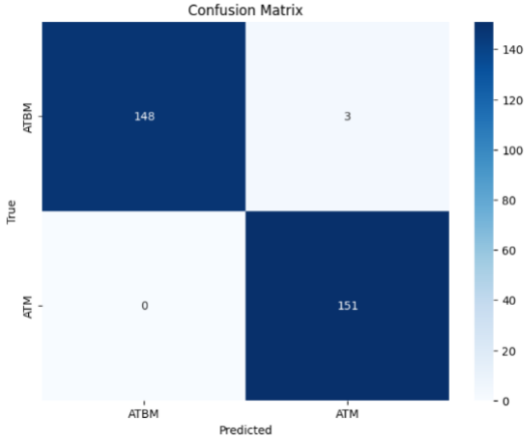
Pada Tabel 4 terlihat bahwa akurasi model untuk trainng, validasi, dan pengujian berada di angka yang sangat tinggi (>99%) setelah epoch ke-4. Ini menunjukkan bahwa model bekerja dengan sangat baik dan dapat mengklasifikasikan data dengan akurasi tinggi di semua set data (*train*, *validation*, dan *test*).*Train loss*, *validation loss*, dan *test loss* semuanya mengalami penurunan yang signifikan selama proses training. Pada akhir *training*, *loss* untuk ketiga set mendekati 0, yang

menunjukkan bahwa model mampu mempelajari pola data dengan sangat baik tanpa kesalahan yang berarti. Sedangkan akurasi dan *loss* pada *validation* dan *test set* menunjukkan bahwa model tidak hanya belajar dari data training, tetapi juga dapat meng-*generalize* dengan baik pada data baru, karena akurasi dan *loss* pada *validation* dan *test set* serupa dengan data *training*.

Pembahasan untuk Model DenseNet121 *Split Data*

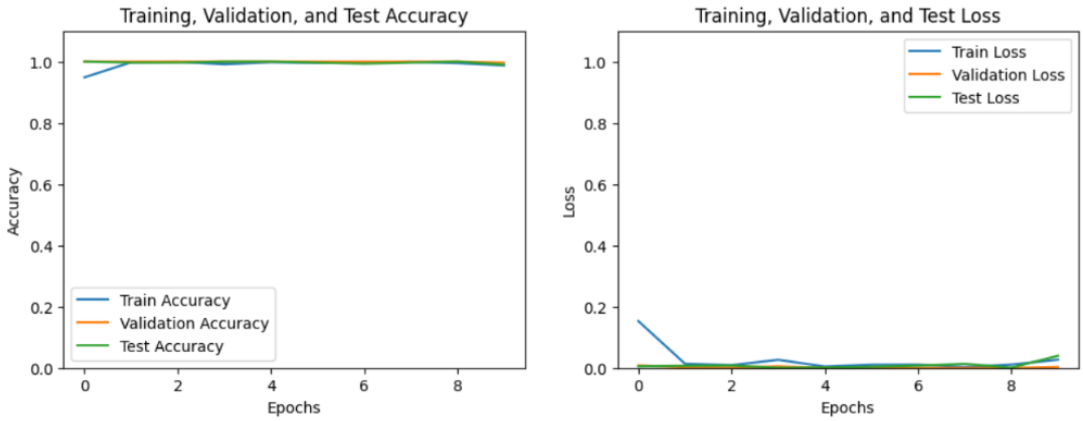
Penelitian menggunakan metode *Deep Learning* CNN DenseNet121 dengan penerapan dua skenario didapatkan hasil *confusin matrix* dan grafik sebagai berikut.

1. *Deep Learning* CNN Model DenseNet121 *Split Data* (70:15:15)



Gambar 6. *Confusin Matrix* Model DenseNet121 *Split Data* (70:15:15)

DenseNet121 (Dense Convolutional Network) adalah jenis arsitektur *deep learning* yang terkenal dengan konsep *skip connections* atau *dense connections*. Pada model ini, setiap *layer* terhubung ke setiap *layer* lainnya secara langsung. Hal ini membantu mengurangi masalah *vanishing gradients* dan memungkinkan pembelajaran fitur yang lebih mendalam dan efisien. Gambar 6 menunjukkan bahwa 70% data *training* digunakan untuk melatih model, dimana model belajar dari data ini dengan memperbarui bobot selama beberapa *epoch*. Nilai 15% data validasi yang digunakan selama *training* untuk memantau kinerja model dan menghindari *overfitting*. Hasil evaluasi pada *validation set* membantu dalam pengaturan *hyper parameter* dan memonitor generalisasi model. Sedangkan nilai 15% untuk data *testing* sebagai *dataset* digunakan untuk mengevaluasi setelah *training* selesai atau pada saat kinerja akhir model.



Gambar 7. Grafik Model DenseNet121 *Split Data* (70:15:15)

Gambar 7 sebelah kiri menunjukkan bahwa sumbu X merepresentasikan jumlah *epoch* (jumlah siklus *trainng*), sedangkan sumbu Y merepresentasikan akurasi. Garis biru menggambarkan akurasi training**,** yang terlihat mencapai nilai mendekati 1 (100%) setelah beberapa *epoch*. Garis oranye menggambarkan akurasi validasi, yang juga menunjukkan hasil

akurasi mendekati 1. Garis hijau menggambarkan akurasi pengujian, yang terlihat stabil dan mendekati nilai 1 setelah beberapa *epoch*. Sedangkan Gambar 7 sebelah kanan menunjukkan sumbu X merepresentasikan jumlah *epoch*, sementara sumbu Y merepresentasikan *loss*. Garis biru menunjukkan *loss training*, yang terlihat turun tajam di awal dan mendekati 0, yang menunjukkan bahwa model dengan cepat mempelajari pola pada data. Garis oranye menunjukkan *loss* validasi, yang juga stabil dan mendekati nilai 0. Garis hijau menunjukkan *loss training*, yang juga tetap rendah selama proses *training*.

Berdasarkan Gambar 6 dan Gambar 7 dengan menggunakan *Google Colab*, didapatkan penjelasan sebagai berikut:

Tabel 5. Penjelasan *Confusin Matrix* Model DenseNet121 *Split Data* (70:15:15)

	precision	recall	f1-score	support
ATBM	1.000000	0.980132	0.989967	151.000000
ATM	0.980519	1.000000	0.990164	151.000000
Accuracy	0.990066	0.990066	0.990066	0.990066
Macro avg	0.990260	0.990066	0.990065	302.000000
Weighted avg	0.990260	0.990066	0.990065	302.000000

Tabel 5 menunjukkan bahwa *precision*: 1.000 menunjukkan semua prediksi kelas ATBM benar (tidak ada kesalahan). *Recall* sebesar 0.980 dimana model berhasil mendeteksi 98,01% dari semua kasus yang sebenarnya adalah ATBM. F1-*Score* sebesar 0.989 merupakan kombinasi *precision* dan *recall* menunjukkan keseimbangan kinerja yang baik untuk kelas ATBM. Sedangkan *support* yang bernilai 151 menunjukkan terdapat 151 sampel untuk kelas ATBM**.**

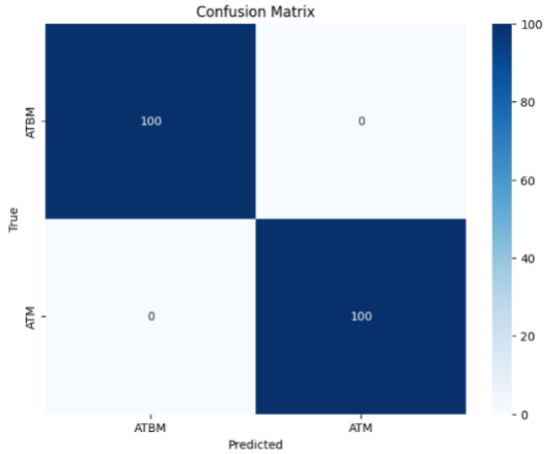
Tabel 6. Penjelasan Grafik Model DenseNet121 *Split Data* (70:15:15)

	epoch	train_accuracy	val_accuracy	test_accuracy	train_loss	val_loss	test_loss
0	1	0.948498	1.000000	1.000000	0.153032	0.007957	0.005389
1	2	0.997854	1.000000	0.996689	0.013773	0.002115	0.006867
2	3	0.999285	1.000000	0.996689	0.009040	0.001084	0.007771
3	4	0.991416	1.000000	1.000000	0.026583	0.005156	0.000676
4	5	0.997854	1.000000	1.000000	0.004535	0.000318	0.001566
5	6	0.995708	1.000000	0.996689	0.010445	0.000157	0.003739
6	7	0.996423	1.000000	0.993378	0.010755	0.000055	0.007287
7	8	0.999285	1.000000	0.996689	0.003271	0.000113	0.013394
8	9	0.994993	1.000000	1.000000	0.010817	0.000126	0.000776
9	10	0.987840	0.996667	0.990066	0.026959	0.003237	0.039668

Tabel 6 menjelaskan *epoch* atau jumlah siklus pelatihan yang telah dilakukan pada model selama 10 *epoch* yaitu mulai dari *epoch* 0 hingga *epoch* 9. Pada *epoch* pertama, model mulai dengan akurasi 0.948498 (94.85%), dan terus meningkat di setiap *epoch*, dengan akurasi tertinggi mencapai 0.999285 (99.93%) pada *epoch* 3 dan 8. Tabel 6 juga menjelaskan bahwa terdapat akurasi sempurna (100%) terutama pada data validasi dari *epoch* pertama hingga *epoch* 8, dan sedikit menurun menjadi 0.996667 (99.67%) pada *epoch* 9. Akurasi pengujian konsisten tinggi, dengan berbagai *epoch* menunjukkan akurasi sempurna (100%), dan hanya sedikit turun pada *epoch* 7 dan 9 menjadi sekitar 99.00%. Nilai loss pada data *training* dimulai dari 0.153032 pada epoch 0 dan terus menurun, mencapai nilai yang sangat kecil pada *epoch* berikutnya, seperti 0.010444 pada *epoch* 6. *Validation loss* cukup rendah sepanjang *epoch*, mulai dari 0.007957 pada *epoch* pertama

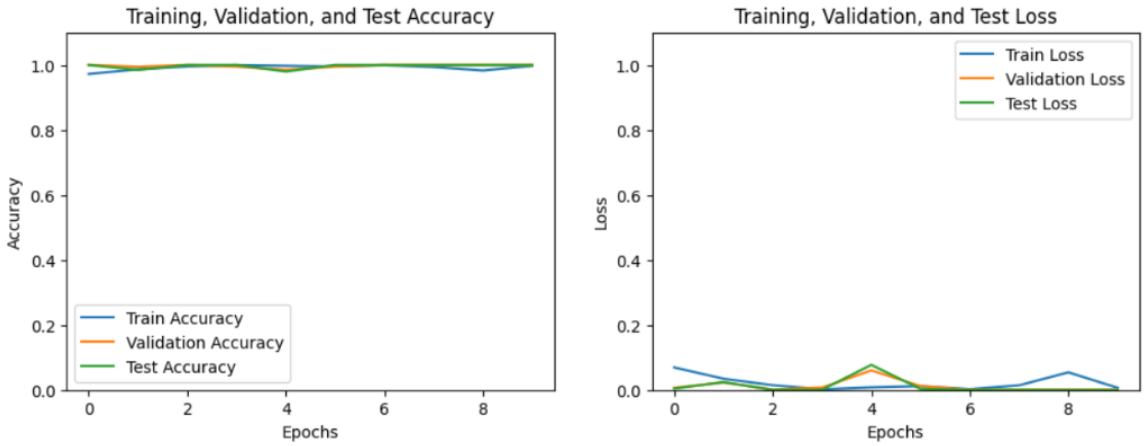
dan menurun hingga mendekati 0 pada beberapa *epoch* berikutnya, seperti 0.00013 pada *epoch* 5 dan 0.001157 pada *epoch* 6. Sedangkan *test loss* tetap rendah sepanjang epoch, dengan nilai yang sedikit meningkat pada *epoch* 9 menjadi 0.039668, tetapi secara umum masih di bawah 0.01 pada sebagian besar *epoch*.

2. *Deep Learning* CNN Model DenseNet121 *Split Data* (80:10:10)



Gambar 8. *Confusin Matrix* Model DenseNet121 *Split Data* (80:10:10)

Gambar 8 menunjukkan akurasi persentase prediksi yang benar (baik positif maupun negatif) dari keseluruhan data yang diuji. Dalam pembagian data 80:10:10, akurasi dihitung untuk semua data uji yang belum pernah dilihat oleh model sebelumnya (testing data, 10%). Presisi yang tinggi menunjukkan bahwa model sangat jarang membuat prediksi positif yang salah. *Recall* tinggi berarti model mampu menemukan sebagian besar kelas positif. *True positives* pada model memprediksi ATBM dan aktualnya sebanyak 145 kali. *True negatives* pada model memprediksi ATM dan aktualnya sebanyak 148 kali. *False Positives* pada model memprediksi ATBM, tetapi aktualnya adalah ATM dan hal ini terjadi sebanyak 3 kali. *False negaties* pada model memprediksi ATM, tetapi aktualnya adalah ATBM dan hal ini terjadi sebanyak 6 kali.



Gambar 9. Grafik Model DenseNet121 *Split Data* (80:10:10)

Pada Gambar 9 sebelah kiri menunjukkan bahwa sumbu X adalah jumlah *epoch* (iterasi model *training*). Sumbu Y menunjukkan akurasi dari model. *Train accuracy* (garis biru) menunjukkan akurasi model pada data trainign. Model terlihat memiliki akurasi yang sangat tinggi mendekati 1 (atau 100%) pada sebagian besar *epoch*. *Validation accuracy* (garis oranye): Menunjukkan akurasi pada data validasi, yang digunakan untuk mengukur generalisasi model selama *training*. Akurasi validasi juga sangat tinggi dan konsisten, mendekati 1 sepanjang *epoc*h.

Test accuracy (garis hijau) menunjukkan akurasi model pada data pengujian, yang merupakan data yang tidak pernah dilihat selama pelatihan. Akurasi pengujian juga tetap tinggi dan stabil, menunjukkan bahwa model mampu melakukan generalisasi dengan baik. Sedangkan Gambar 9 sebelah kanan menunjukkan bahwa sumbu X adalah jumlah *epoch* (sama dengan grafik sebelah kiri). Sumbu Y menunjukkan *loss* atau nilai kerugian menggambarkan seberapa baik atau buruk prediksi model dibandingkan dengan target sebenarnya. Semakin rendah *loss*, semakin baik model. *Train loss* (garis biru) menunjukkan *loss* pada data *training*. Terlihat bahwa *loss* pada data *training* menurun di awal dan tetap sangat rendah setelahnya. *Validation loss* (garis oranye) menunjukkan *loss* pada data validasi dan terlihat bahwa *validation loss* juga sangat rendah dan cukup konsisten selama *epoch*. Sedangkan *test loss* (garis hijau) menunjukkan *loss* pada data pengujian juga sangat rendah sehingga model bekerja dengan baik pada data yang tidak dilihat selama *training*.

Berdasarkan Gambar 8 dan Gambar 9 dengan menggunakan *Google Colab*, didapatkan penjelasan sebagai berikut:

Tabel 7. Penjelasan *Confusin Matrix* Model DenseNet121 *Split Data* (80:10:10)

	precision	recall	f1-score	support
ATBM	1.0	1.0	1.0	100.0
ATM	1.0	1.0	1.0	100.0
Accuracy	1.0	1.0	1.0	1.0
Macro avg	1.0	1.0	1.0	200.0
Weighted avg	1.0	1.0	1.0	200.0

Tabel 7 menunjukkan jika precision untuk kedua kelas ATBM dan ATM adalah 1.0, yang menunjukkan bahwa setiap prediksi yang dibuat oleh model untuk kelas tersebut benar. Tidak ada *false positive* untuk kedua kelas ini. *Recall* untuk kedua kelas juga 1.0, artinya model berhasil mengenali semua contoh yang benar dari kelas tersebut. Tidak terdapat *false negative* dalam prediksi. F1-*Score* merupakan rata-rata harmonis dari *precision* dan *recall*, juga mencapai 1.0 untuk kedua kelas, dimana hal ini berarti keseimbangan antara *precision* dan *recall* sangat sempurna. Kolom *support* menunjukkan jumlah sampel pengujian untuk setiap kelas, dimana terdapat 100 sampel untuk kelas ATBM dan 100 sampel untuk kelas ATM, jadi model diuji pada total 200 sampel. Akurasi keseluruhan model adalah 1.0 menunjukkan bahwa model telah memprediksi seluruh sampel uji dengan benar untuk kedua kelas. *Macro average* adalah rata-rata tidak berbobot dari *precision*, *recall*, dan F1-*score* untuk semua kelas, hal ini karena setiap kelas memiliki kinerja yang sempurna, nilai makro juga 1.0. Sedangkan *weighted average* adalah rata-rata *precision*, *recall*, dan F1-*score* yang diberi bobot berdasarkan jumlah sampel di setiap kelas. Dalam hal ini, nilai weighted juga 1.0, karena kedua kelas memiliki jumlah sampel yang sama dan kinerja yang sempurna.

Tabel 8. Penjelasan Grafik Model DenseNet121 *Split Data* (80:10:10)

	epoch	train_accuracy	val_accuracy	test_accuracy	train_loss	val_loss	test_loss
0	1	0.972500	1.000	1.000	0.069406	0.006539	0.004443
1	2	0.986875	0.995	0.985	0.034794	0.023208	0.024252
2	3	0.995625	1.000	1.000	0.014758	0.001377	0.000380
3	4	0.999375	0.995	1.000	0.002156	0.007348	0.000748
4	5	0.997500	0.985	0.980	0.008130	0.060494	0.077414
5	6	0.995625	0.995	1.000	0.012029	0.011928	0.003335
6	7	0.999375	1.000	1.000	0.002471	0.000577	0.000347
7	8	0.993750	1.000	1.000	0.014483	0.000855	0.000263
8	9	0.983125	1.000	1.000	0.054395	0.000826	0.000126
9	10	0.997500	1.000	1.000	0.006664	0.000623	0.000027

Pada Tabel 8 menunjukkan setiap *epoch* menunjukkan satu siklus penuh dari pelatihan model pada data. Tabel ini menyajikan hasil evaluasi model selama 10 epoch**.** Kolom ini menunjukkan akurasi model pada data *training*. Akurasi ini meningkat seiring bertambahnya *epoch*, dengan nilai awal di 0.9725 pada *epoch* pertama dan meningkat hingga 0.9975 pada epoch terakhir. Hal ini menunjukkan bahwa model semakin baik dalam memprediksi dengan benar pada data *training* seiring bertambahnya iterasi *training*. Nilai akurasi validasi relatif tinggi sejak *epoch* pertama, mencapai 1.0 di beberapa *epoch* (misalnya, epoch 1, 3, 4, 7, 9, dan 10), menunjukkan bahwa model mampu melakukan generalisasi dengan sangat baik pada data validasi. Akurasi pengujian menunjukkan kemampuan model untuk memprediksi data uji. Dari Tabel 8 ini, terlihat bahwa model mencapai akurasi 1.0 pada sebagian besar *epoch*, kecuali pada *epoch* ke-2 dan ke-5 di mana nilai akurasi pengujian sedikit menurun menjadi 0.985 dan 0.980. *Train loss* menunjukkan seberapa besar kesalahan prediksi model pada data pelatihan. *Loss* ini secara konsisten menurun dari 0.069406 pada *epoch* pertama menjadi 0.006664 pada *epoch* terakhir, dimana hal menunjukkan bahwa model semakin baik dalam mengurangi kesalahan pada data *training* selama waktu training. *Validation loss* mengukur seberapa besar kesalahan prediksi pada data validasi. Nilai validation loss juga menunjukkan tren menurun secara konsisten, dimulai dari 0.006539 pada *epoch* pertama menjadi 0.000623 pada epoch kesepuluh, menandakan model semakin baik dalam melakukan generalisasi. Sedangkan *test loss* adalah ukuran seberapa besar kesalahan prediksi pada data pengujian. Pada Tabel 8 ini, terlihat bahwa *test loss* mulai dari 0.004443 dan turun hingga 0.000027 pada *epoch* kesepuluh. Penurunan ini menunjukkan bahwa model mampu memprediksi dengan baik pada data uji.

KESIMPULAN

Kedua model mampu melakukan tugas klasifikasi tekstur kain dengan sangat baik. Tidak ada perbedaan yang signifikan dalam efektivitas keseluruhan dari kedua model ini, menjadikan keduanya pilihan yang sangat baik untuk tugas ini. Namun, VGG16 sedikit lebih sederhana dalam arsitektur dibandingkan DenseNet121, sehingga membuat lebih cepat dalam pelatihan dan inferensi. Penelitian ini berhasil mengembangkan model klasifikasi tekstur sarung Samarinda berdasarkan alat tenunnya (mesin dan bukan mesin) menggunakan metode *Deep Learning*. Model ini dapat membantu, meningkatkan nilai jual sarung Samarinda, melestarikan dan mempromosikan kain tradisional dimana VGG16 lebih sederhana dalam arsitektur dan lebih cepat dalam pelatihan maupun inferensi, namun pemilihan antara DenseNet121 dan VGG16 dapat bergantung pada preferensi pengguna atau kebutuhan spesifik, seperti kecepatan komputasi atau ukuran model.

DAFTAR PUSTAKA

[1] Hussain, M. A. I., Khan, B., Wang, Z., & Ding, S. (2020). Woven fabric pattern recognition and classification based on deep convolutional neural networks. Electronics (Switzerland), 9(6). https://doi.org/10.3390/electronics9061048

[2] Kumar, K. S., & Bai, M. R. (2023). LSTM based texture classification and defect detection in a fabric. Measurement: Sensors, 26. https://doi.org/10.1016/j.measen.2022.100603

[3] Agastya, I. M. A., & Setyanto, A. (2018). Classification of Indonesian batik using deep learning techniques and data augmentation. Proceedings - 2018 3rd International Conference on Information Technology, Information Systems and Electrical Engineering, ICITISEE 2018. https://doi.org/10.1109/ICITISEE.2018.8720990

[4] Andrian, R., Hermanto, B., & Kamil, R. (2019). The Implementation of Backpropagation Artificial Neural Network for Recognition of Batik Lampung Motive. Journal of Physics: Conference Series, 1338(1), 0–10. https://doi.org/10.1088/1742-6596/1338/1/012062

[5] Aras, S., Setyanto, A., & Rismayani. (2022). Classification of Papuan Batik Motifs Using Deep Learning and Data Augmentation. 2022 4th International Conference on Cybernetics and Intelligent System, ICORIS 2022. https://doi.org/10.1109/ICORIS56080.2022.10031320

[6] Fadlil, A., Riadi, I., & Purwadi Putra, I. J. D. E. (2023). Comparison of Machine Learning Performance Using Naive Bayes and Random Forest Methods to Classify Batik Fabric Patterns. Revue d'Intelligence Artificielle, 37(2), 379–385. https://doi.org/10.18280/ria.370214

[7] Haralick RM, S. K. (1973). IEEE Transactions on systems, man, and cybernetics:610– 621, 1973. Textural Features for Image Classification, 3, 610–621.

[8] Hu, Y., Long, Z., Sundaresan, A., Alfarraj, M., AlRegib, G., Park, S., & Jayaraman, S. (2021). Fabric surface characterization: assessment of deep learning-based texture representations using a challenging dataset. Journal of the Textile Institute, 112(2). https://doi.org/10.1080/00405000.2020.1757296

[9] Juwita, A. R., & Solichin, A. (2018). Batik pattern identification using GLCM and artificial neural network backpropagation. Proceedings of the 3rd International Conference on Informatics and Computing, ICIC 2018. https://doi.org/10.1109/IAC.2018.8780412

[10] Khasanah, C. U., Utami, E., & Raharjo, S. (2020). Implementation of Data Augmentation Using Convolutional Neural Network for Batik Classification. 2020 8th International Conference on Cyber and IT Service Management, CITSM 2020, 20–24. https://doi.org/10.1109/CITSM50537.2020.9268890

[11] Kusanti, J., & Suprapto, A. (2019). Combination of Otsu and Canny Method to Identify the Characteristics of Solo Batik as Surakarta Traditional Batik. Proceedings - 2019 2nd International Conference of Computer and Informatics Engineering: Artificial Intelligence Roles in Industrial Revolution 4.0, IC2IE 2019. https://doi.org/10.1109/IC2IE47452.2019.8940884

[12] Lizarraga-Morales, R. A., Correa-Tome, F. E., Sanchez-Yanez, R. E., & Cepeda-Negrete, J. (2019). On the Use of Binary Features in a Rule-Based Approach for Defect Detection on Patterned Textiles. IEEE Access, 7. https://doi.org/10.1109/ACCESS.2019.2896078

[13] Putra, M. T. D., & Kusuma, G. P. (2019). Batik Classification using Deep Learning. International Journal of Recent Technology and Engineering (IJRTE), 8(4), 11416–11421. https://doi.org/10.35940/ijrte.d9039.118419

[14] Seçkin, A. Ç., & Seçkin, M. (2022). Detection of fabric defects with intertwined frame vector feature extraction. Alexandria Engineering Journal, 61(4). https://doi.org/10.1016/j.aej.2021.08.017

[15] Septiarini, A., Rizqi Saputra, Andi Tejawati, & Masna Wati. (2021). Deteksi Sarung Samarinda Menggunakan Metode Naive Bayes Berbasis Pengolahan Citra. Jurnal RESTI

(Rekayasa Sistem Dan Teknologi Informasi), 5(5), 927–935. https://doi.org/10.29207/resti.v5i5.3435

[16] Septiarini, A., Saputra, R., Tedjawati, A., Wati, M., & Hamdani, H. (2022). Pattern Recognition of Sarong Fabric Using Machine Learning Approach Based on Computer Vision for Cultural Preservation. International Journal of Intelligent Engineering and Systems, 15(5), 284–295. https://doi.org/10.22266/ijies2022.1031.26

[17] Simon, P., & Uma, V. (2020). Deep Learning based Feature Extraction for Texture Classification. Procedia Computer Science, 171. https://doi.org/10.1016/j.procs.2020.04.180

[18] Szeliski, R. (2021). Computer Vision : Algorithms and Applications 2nd Edition. In Springer.

[19] Wijaya Kusuma, W., Rizal Isnanto, R., Fauzi, A., & Korespondensi, P. (2023). DenseNet121 Menggunakan Kerangka Kerja TensorFlow untuk Deteksi Jenis Hewan. Jurnal Teknik Komputer, 1(4), 141–147. https://doi.org/10.14710/jtk.v1i4.37009