Implementasi Deep Transfer Learning untuk Klasifikasi Nominal Uang Kertas Rupiah

Bagas Ahmad Sadewa[1], Yuni Yamasari[2]

[1,2] Jurusan Teknik Informatika/Teknik Informatika, Universitas Negeri Surabaya
[1]bagas.20016@mhs.unesa.ac.id
[2]yuniyamasari@unesa.ac.id

***Abstrak*—Penerapan teknologi otomatisasi dilakukan dalam proses transaksi jual beli menggunakan mesin sebagai perantara. Mesin tersebut bertindak sebagai penjual yang memiliki kemampuan serupa dengan otak manusia, termasuk kemampuan membaca dan mengenali nominal uang dengan cepat dan tepat. Penggunaan teknologi otomatisasi ini menjadi solusi kemudahan dalam kegiatan jual beli. Untuk itu, penelitian ini memfokuskan pada pengenalan nominal mata uang kertas rupiah dengan menerapkan teknologi *Deep Transfer Learning* - algoritma *Convolutional Neural Network* (CNN). Lebih jauh, penelitian ini melakukan pemilihan arsitektur *pretrained model* yang sesuai dan penyesuaian nilai *learning rate* yang dapat mempengaruhi kinerja model klasifikasi. Jumlah *epoch* yang optimal juga dipertimbangkan untuk memastikan model memiliki cukup kesempatan untuk belajar dari data pelatihan. Selain itu, penggunaan *batch size* yang dinilai paling ideal juga dieksplorasi karena hal ini dapat mempengaruhi tercapainya performa tinggi dengan waktu komputasi yang efisien.**

Hasil uji coba memperlihatkan bahwa model yang paling optimal dicapai ketika model dibangun dengan menggunakan *pretrained model* VGG16 dengan kombinasi *hyperparameter learning rate* 0.0001, *batch size* 20, dan 15 *epochs*. Kemudian, hasil evaluasi kinerja model saat pelatihan menunjukkan nilai rata-rata akurasi 94.29%, presisi 95.01%, *recall* 94.29%, *f1-score* 94.26%, dan *AUC* 99.84%. Setelah dilakukan *testing* dengan data uji diperoleh rata-rata nilai *accuracy* 92.86%, *precision* 94.07%, *recall* 92.86%, *f1-score* 93.03%, dan *AUC* 99.64%.

Kata Kunci— ***Deep Transfer Learning*, *Convolutional Neural Network*, VGG16, Klasifikasi, Uang Kertas Rupiah.**

I. PENDAHULUAN

Transaksi perdagangan telah ada dan berkembang dari berabad-abad yang lalu, dengan berbagai evolusi dalam proses jual beli. Pada masa lalu, orang-orang menggunakan sistem barter sebagai metode pertukaran barang. Seiring berjalannya waktu, cara melakukan transaksi jual beli telah bergeser ke penggunaan uang sebagai medium resmi pembayaran. Di Indonesia, rupiah dianggap sebagai mata uang yang sah untuk melakukan transaksi. Sesuai dengan Undang-Undang Nomor 7 Tahun 2011 tentang Mata Uang, Bank Indonesia memiliki kewenangan atas pengelolaan uang rupiah. Bank Indonesia mengedarkan dua jenis mata uang, yaitu logam dan uang kertas. Uang memiliki nilai nominal yang penting untuk menentukan nilai dari barang dan jasa yang diperdagangkan. [1].

Kini transaksi jual beli dimudahkan dengan menggunakan mesin sebagai media untuk transaksi jual beli atau biasa disebut *vending machine*. Proses ini dilakukan oleh mesin sebagai penjual dan manusia sebagai pembeli dengan tetap menggunakan uang untuk pembayarannya [2]. Untuk mendukung proses jual beli yang dilakukan oleh mesin dan manusia maka diperlukan sistem yang dapat mendeteksi dan mengklasifikasikan setiap nominal uang dengan akurat dan cepat. Dengan sistem ini maka akan meningkatkan efisiensi dan ketepatan proses transaksi. Serta menciptakan pengalaman transaksi yang lebih baik.

Dalam sistem deteksi nominal uang kertas rupiah ini, proses pengenalan nilai uang dilakukan dengan memeriksa gambar permukaannya, yang dikenal sebagai pengenalan citra. Setelah dikenali langkah berikutnya adalah klasifikasi, yang merupakan proses mencari pola yang dapat memisahkan kelas data dengan tujuan untuk memprediksi kelas atau nilai nominal dari uang tersebut. Proses klasifikasi terdiri dari dua tahap, yaitu tahap pelatihan (*training*) dan tahap pengujian (*testing*). Pada tahap pelatihan, sebagian data yang sudah diketahui kelasnya digunakan untuk membentuk model prediksi. Setelah itu, pada tahap pengujian, model tersebut diuji dengan menggunakan sebagian data lainnya untuk mengevaluasi akurasinya. Jika akurasi sudah optimal, model tersebut dapat digunakan untuk memprediksi kelas data yang belum diketahui [3].

Teknologi yang bisa dimanfaatkan untuk merealisasikan hal tersebut yaitu *Deep Learning*. Salah satu metode yang umum digunakan untuk memproses data gambar adalah dengan menggunakan *Convolutional Neural Network* (CNN). [4]. Algoritma CNN adalah teknik pemrosesan gambar yang menggunakan konvolusi untuk mengenali karakteristik khusus dalam data pelatihan. Karakteristik ini kemudian digunakan sebagai masukan untuk dilatih oleh jaringan saraf atau *neural network* [5]. Oleh karena itu dapat memberikan komputer pengetahuan untuk mengklasifikasi uang berdasarkan fitur unik yang telah dipelajari. Sehingga dapat dikenali sesuai nominalnya.

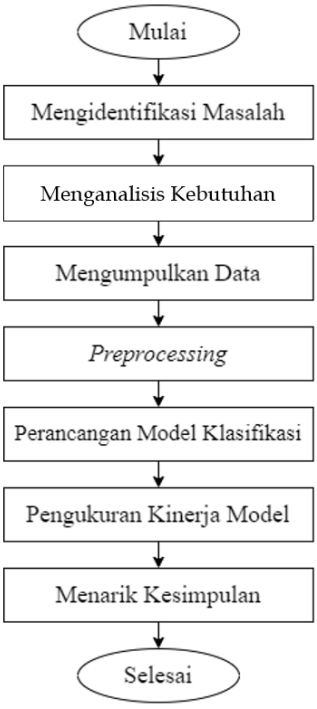
Pada penelitian sebelumnya dengan judul "Klasifikasi Nominal Uang Kertas Rupiah Tahun Emisi 2017 dengan Algoritma CNN menggunakan MXNET" telah dilakukan oleh [6]. Hasil eksperimen membuktikan bahwa menggunakan *kernel* dengan ukuran 3x3 di konvolusi pertama dan 2x2 di konvolusi kedua, dengan 20 *filters*, 50 *epoch*, dan 17 *neuron* dalam *hidden layer* akan menghasilkan akurasi terbaik sebesar 93,57%. Contoh penelitian lainnya dengan judul "Klasifikasi Uang Rupiah Kertas Tidak Layak Edar Menggunakan CNN Xception *Transfer Learning* Berbasis Website" oleh [7]. Metode transfer learning menggunakan model Xception telah terbukti efektif dalam mengklasifikasikan uang dengan tingkat akurasi mencapai 93,75%.

Berdasarkan hasil eksperimen dari dua penelitian tersebut, algoritma *Convolutional Neural Network* (CNN) dapat

dimanfaatkan untuk mengklasifikasikan nominal uang kertas. Dengan demikian penulis membuat penelitian dengan judul "Implementasi *Deep Transfer Learning* Untuk Klasifikasi Nominal Uang Kertas Rupiah". Dengan menerapkan *deep transfer learning*, fitur yang sudah dipelajari dapat dipindahkan ke suatu tugas baru dengan memanfaatkan dataset yang lebih kecil, sehingga mempercepat proses pelatihan. Hal ini disebabkan oleh penggunaan jaringan yang sebelumnya telah dilatih dan mampu mengekstraksi berbagai fitur yang beragam [8]. Maka jaringan yang baru akan lebih tepat dan akurat karena memiliki kemampuan yang lebih baik dalam mengenali fitur-fitur khusus.

II. METODOLOGI PENELITIAN

Metode yang diterapkan untuk membuat sistem klasifikasi mata uang rupiah yaitu menggunakan metode *deep transfer learning*. Berikut ini adalah serangkaian langkah-langkah yang diterapkan dalam penelitian ini:



Gbr. 1 Diagram Alur Metode Penelitian.

A. *Mengidentifikasi Masalah*

Tahap awal dalam proses penelitian adalah mengidentifikasi permasalahan. Permasalahan yang menjadi fokus penyelesaian dalam penelitian ini yaitu bagaimana mengklasifikasikan uang kertas rupiah agar dapat dikenali oleh mesin dengan menggunakan *deep transfer learning*. Peneliti akan melakukan perancangan model dan pengimplementasian pada dataset yang bertujuan untuk membuat model yang dapat mengklasifikasikan nominal mata uang rupiah. Sehingga dapat digunakan untuk mesin penjualan otomatis atau keperluan lainnya.

B. *Menganalisis Kebutuhan*

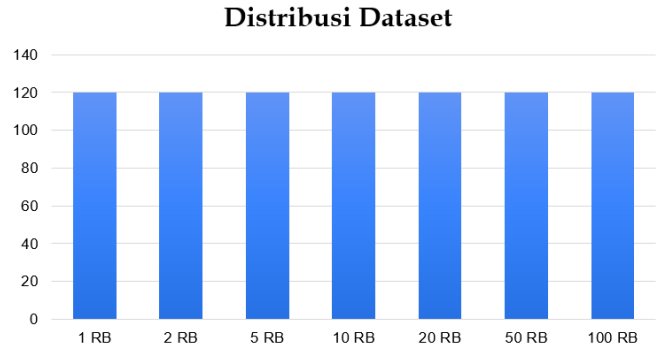
Untuk mencapai tujuan penelitian ini, dibutuhkan perangkat yang dapat digunakan dalam percobaan untuk menghasilkan model yang optimal. Penelitian ini menggunakan Laptop Asus Vivobook X409UJ dengan *Processor 7th Gen Intel Core i3*. *Dual VGA Intel HD Grapichs* 620 ditambah *NVIDIA GeForce MX230*. *RAM* 12 *Gigabyte*, menggunakan sistem operasi *Windows* 10 64-bit. Model *deep transfer learning* dibangun di *Google Colaboratory* dengan *runtime Python 3* dan akselerasi *hardware* menggunakan T4 GPU yang dapat memberikan kecepatan komputasi tinggi, memori besar, dan arsitektur dioptimalkan untuk *deep learning*.

C. *Mengumpulkan Data*

Data citra yang digunakan yaitu gambar uang kertas Rupiah tahun emisi 2022 dengan kondisi baik serta yang menunjukkan tanda-tanda lipatan atau kerusakan. Sumber data primer diperoleh melalui pengambilan gambar langsung menggunakan kamera ponsel. Populasi penelitian mencakup seluruh gambar uang kertas Rupiah tahun emisi 2022, dengan sampel terdiri dari 840 gambar uang kertas Rupiah tahun emisi 2022 dengan nilai nominal Rp1.000,00, Rp2.000,00, Rp5.000,00, Rp10.000,00, Rp20.000,00, Rp50.000,00, Rp100.000,00, . Proses pengambilan sampel melibatkan pemotretan gambar uang kertas dalam berbagai kondisi, seperti terbalik, terlipat, dan dengan pencahayaan yang kurang. Contoh dari sampel data citra yang digunakan dapat dilihat pada Gbr 2.

Gbr. 2 Sampel uang kertas rupiah emisi 2022.

Distribusi dataset yang digunakan dalam penelitian ini dengan total 840 citra yang dibagi menjadi 7 kelas nominal mata uang, dan tiap kelas memiliki 120 data citra. Keseimbangan dataset merupakan aspek krusial dalam pelatihan model [9]. Dengan dataset yang seimbang, model dapat lebih baik menggeneralisasi dan mempelajari fitur-fitur yang merepresentasikan setiap *class*, yang pada akhirnya meningkatkan kinerja model. Gbr. 3 adalah grafik distribusi dataset citra.



Gbr. 3 Distribusi Dataset Citra.

D. *Preprocessing*

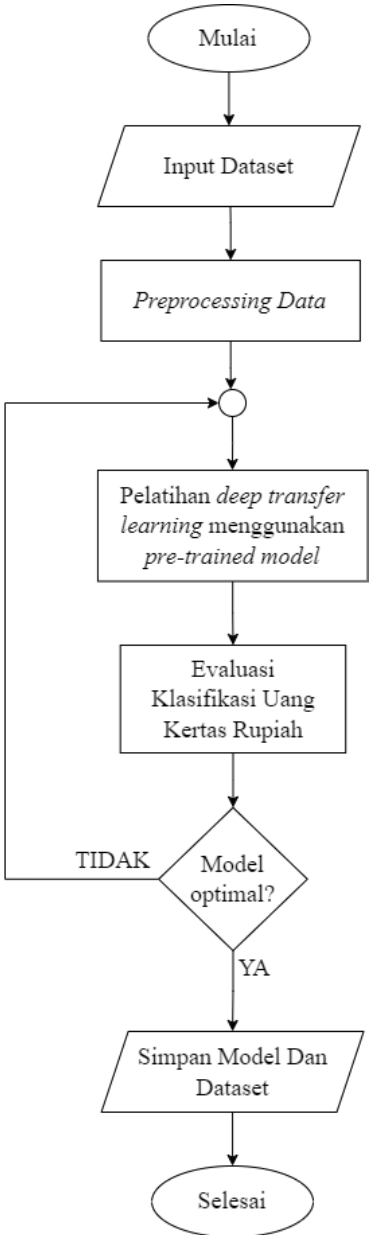
Preprocessing adalah langkah-langkah yang dilakukan pada data sebelum digunakan untuk pelatihan atau analisis lebih lanjut yang bertujuan untuk menyiapkan citra agar siap untuk diproses pada tahap selanjutnya. *Preprocessing* yang baik akan memiliki efek positif pada kinerja model. Diharapkan fitur yang diinginkan dalam citra akan menjadi lebih menonjol [10]. Pada tahap ini citra akan diubah ukuran menjadi 224 x 224 *pixel* dan citra berformat warna RGB, serta mengkategorikan mereka ke dalam label-label kelas yang sesuai. Selanjutnya, data citra tersebut disimpan dalam format *pickle* dan label *class* akan disimpan dalam berkas csv untuk keperluan analisis dan pelatihan model pengenalan nominal uang kertas.

E. *Perancangan Model Klasifikasi*

Model yang akan dirancang dalam penelitian ini menerapkan metode yang tepat dalam klasifikasi citra yaitu dengan mengimplementasikan *deep transfer learning.* Penelitian ini menggunakan *pre-trained model* VGG16. Untuk membangun model klasifikasi dengan kinerja yang optimal, berikut adalah langkah-langkah dari perancangan model klasifikasi :

1) *Input* dataset citra.
2) *Preprocessing* data dengan mengubah ukuran citra menjadi 224 x 224 *pixel* dan pemberian label *class* untuk tiap nominal uang.
3) Dataset berisi 840 citra dibagi menjadi 560 citra sebagai data latih, data validasi 140 citra, dan data uji 140 citra.
4) Data latih akan digunakan untuk *training deep transfer learning* menggunakan *pre-trained model* VGG16.
5) Model akan dievaluasi dengan perbandingan uji kinerja untuk nilai akurasi, presisi, *recall*, *f1-score,* dan *AUC.*
6) Model dengan kinerja terbaik akan disimpan dan dapat dimanfaatkan untuk membuat sistem klasifikasi uang.

Berdasarkan langkah - langkah di atas, berikut adalah alur proses perancangan model klasifikasi yang akan dibangun:



Gbr. 4 *Flowchart* Perancangan Model Klasifikasi.

F. *Pengukuran Kinerja Model*

Model akan dievaluasi untuk mengetahui seberapa baik kinerja dalam memprediksi data latih dan data validasi selama pelatihan. Pengukuran kinerja model adalah suatu proses penting dalam evaluasi seberapa optimal model *machine learning* dalam memprediksi atau mengklasifikasikan data. Pengukuran kinerja model dari eksperimen yang telah dilakukan dievaluasi menggunakan *confusion matrix* untuk menilai tingkat keberhasilannya. Output dari proses klasifikasi

direpresentasikan melalui empat istilah dalam *confusion matrix*, yaitu *true positive* (TP) untuk data yang benar sebagai kelas positif dan berhasil diprediksi sebagai kelas positif, *true negative* (TN) untuk data yang benar sebagai kelas negatif dan berhasil diprediksi sebagai kelas negatif oleh model, *false positive* (FP) untuk data yang sebenarnya salah tetapi diprediksi benar, dan *false negative* (FN) untuk data yang sebenarnya positif tetapi diprediksi sebagai negatif oleh model. [11]. Selanjutnya akan dilakukan perbandingan uji kinerja untuk mengetahui model dengan kinerja untuk nilai akurasi, presisi, *recall*, *f1-score,* dan *AUC.*

Akurasi yaitu ukuran yang menggambarkan perbandingan jumlah prediksi yang tepat dengan total keseluruhan prediksi. Berikut adalah persamaan akurasi.

$$Akurasi = \frac{TP + TN}{TP + TN + FP + FN} \quad (1)$$

Presisi yaitu perbandingan antara jumlah data yang terklasifikasi *true positif* dengan jumlah total data yang diklasifikasikan ke dalam kategori positif.

$$Presisi = \frac{TP}{TP + FP} \quad (2)$$

Recall atau Sensitifitas yaitu ukuran yang bertujuan untuk menilai seberapa baik model dalam menghindari *false negative*. *Recall* dihitung dari rasio *true positive* terhadap *actual positive.*

$$Recall = \frac{TP}{TP + FN} \quad (3)$$

F1-score yaitu gabungan nilai rata-rata dari presisi dan recall. Membantu dalam mencapai titik tengah antara kedua metrik tersebut. Misalnya, dalam suatu tugas di mana kesalahan *false positives* (FP) dan *false negatives* (FN) memiliki konsekuensi yang sebanding, *F1-Score* memberikan pemahaman yang lebih komprehensif tentang kinerja model dibandingkan hanya menggunakan presisi atau *recall* secara terpisah. Berikut adalah persamaan *f1-score.*

$$F1 - score = 2 * \frac{Precision \times Recall}{Precision + Recall} \quad (4)$$

Receiving Operation Characteristic (ROC) adalah kurva representasi dari probabilitas. Kurva ini menggambarkan perbandingan antara Tingkat Positif Benar (*True Positive Rate*) dan Tingkat Positif Salah (*False Positive Rate*). Berikut adalah persamaan ROC :

$$TPR = \frac{TP}{TP + FN} \quad (5)$$

$$FPR = \frac{FP}{FP + TN} \quad (6)$$

Area Under Curve (AUC) adalah ukuran luasan area yang terletak di bawah kurva ROC. Semakin tinggi nilai AUC, semakin baik kemampuan model dalam memisahkan antara kelas positif dan kelas negatif [12]. Berikut ini persamaan AUC :

$$\int_0^1 TPR(FPR), dFPR \quad (7)$$

III. Hasil dan Pembahasan

A. Skenario Pengujian Pretrained Model

Skenario pengujian ini dilakukan untuk menguji pengaruh *pretrained model* terhadap kinerja model. *Pretrained model* yang digunakan dalam penelitian ini yaitu VGG-16. Selain *pretrained model*, diperlukan *fine tuning* dengan menambahkan layer tambahan. Untuk itu perlu diuji untuk mengetahui pengaruh dari *base pretrain model* dan *pretrain model* ditambah *fine tuning*. Skenario pengujian *pretrained model* dapat dilihat di Tabel I.

Tabel I
Skenario Pengujian Pretrained Model

No	*Pre Trained Model*	Parameter
1	VGG16	Tanpa *Tuning*
2	VGG16	Dengan *Tuning*
3	Tanpa *Pretrained model*	Dengan *Tuning*

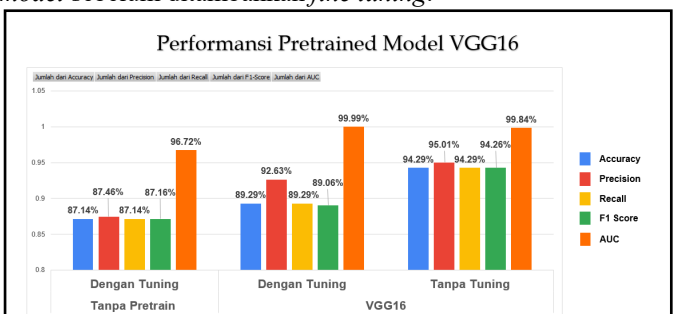
Setelah melalukan 3 percobaan kombinasi *pretrained model* dengan *tuning* dan tanpa *tuning*, didapatkan hasil evaluasi yaitu akurasi, presisi, *recall*, *f1-score,* dan *AUC.* Tabel II adalah hasil metrik evaluasi yang didapatkan setelah melakukan beberapa kali percobaan:

Tabel II
Hasil Pengujian Pretrained Model

No	*Pre Trained Model*	Parameter	*Metrics Evaluation*					Waktu (detik)
			Accuracy	*Precision*	*Recall*	*F1-Score*	*AUC*	
1	VGG16	Tanpa *Tuning*	0.9429	0.9501	0.9429	0.9426	0.9984	167.941
2	VGG16	Dengan *Tuning*	0.8929	0.9263	0.8929	0.8906	0.9999	206.178
3	Tanpa *Pretrained Model*	Dengan *Tuning*	0.8714	0.8746	0.8714	0.8716	0.9672	28.622

B. Analisis Performansi Pretrained Model

Berdasarkan hasil pengujian yang telah dilakukan, performansi kombinasi *pretrained model* diukur dengan metrik evaluasi yaitu akurasi, presisi, *recall, f1 score*, dan *AUC*. Berikut adalah diagram visualisasi performansi *pretrain model* sebelum ditambahkan *fine tuning*.



Gbr. 5 Performansi *Pretrained* Model.

Gbr. 5 Performansi *Pretrained* Model menunjukkan bahwa *pretrain model* VGG16 tanpa *tuning* memiliki performa paling tinggi yang menghasilkan nilai akurasi 94.29%, presisi 96.01%, *recall* 94,29%, *f1-score* 94.26% dan *AUC* 99,84%. Percobaan dengan menambahkan lapisan *fine tuning* yang dilakukan dengan menambah *layer* tambahan berupa *Dense* dengan nilai 200, 100, dan 50 dengan fungsi aktivasi *RelU*. Ditambahkan juga *layer* untuk mencegah terjadinya *overfitting* yaitu *batch normalization* yang berfungsi untuk menormalisasi output dari konvolusi sehingga dapat meningkatkan stabilitas dan generalisasi model. Untuk mengurangi jumlah *neuron* secara acak diperlukan *dropout* dengan nilai 30%. Namun kinerjanya justru lebih rendah setelah ditambahkan *layer tuning* dengan akurasi 89.29%, presisi 92.63%, *recall* 89.29%, *f1-score* 89.06%, dan AUC 99.99%. Percobaan terakhir dilakukan tanpa menggunakan pretrained model VGG16 dan hanya menggunakan lapisan tuning. Percobaan ini menghasilkan nilai akurasi 87.14%, presisi 87.46%, *recall* 87.14%, *f1-score* 87.16% , *AUC* 96.72%.

Berdasarkan hasil dari ketiga percobaan diatas *pretrain model* VGG16 tanpa *fine tuning* memiliki kinerja paling optimal dengan waktu komputasi yang lebih cepat yaitu 167.941 detik. Mengimplementasikan arsitektur *pretrained model* yang cocok serta dataset yang tepat dapat meningkatkan akurasi model klasifikasi yang dihasilkan [13]. Diketahui juga bahwa kinerja *pretrained model* yang ditambahkan *layer fine tuning* tidak selalu menjadi lebih baik dari pada hanya menggunakan *base pretrained model*. Sementara jika tanpa menggunakan *pretrained model* kinerjanya akan lebih rendah karena model belum cukup mempelajari pola-pola dengan dataset yang sedikit. Sehingga diperlukan *transfer learning* menggunakan *pretrained model* yang dapat meningkatkan kinerja model.

C. Skenario Pengujian Learning Rate

Setelah mendapatkan *pretrain model* yang optimal, langkah selanjutnya adalah menguji berbagai nilai *learning rate*. Dalam penggunaan *optimizer* Adam, diperlukan penyesuaian *hyperparameter learning rate*. Pada penelitian ini, dilakukan percobaan dengan 4 variasi yaitu tanpa *learning rate*, menggunakan *learning rate* 0.001, 0.0001, dan 0.00001. Semakin optimal nilai *learning rate* yang dipilih, semakin baik pula kinerja model dalam memprediksi data uji [14]. Berikut adalah skenario pengujiannya:

TABEL III
SKENARIO PENGUJIAN LEARNING RATE

Pre Trained Model	***Hyperparameter***		
	Learning Rate	***Batch Size***	**Jumlah *Epochs***
VGG16	Tanpa LR	20	15
VGG16	0.001	20	15
VGG16	0.0001	20	15
VGG16	0.00001	20	15

D. Analisis Performansi Learning Rate

Berikut adalah hasil dari pengujian learning rate yang telah dilakukan sehingga didapatkan hasil kinerja model yaitu *accuracy, precision, recall, f1-score*, *AUC* serta waktu komputasi yang digunakan.

TABEL IV
SKENARIO PENGUJIAN LEARNING RATE

Learning Rate	**Akurasi**	**Presisi**	***Recall***	***F1-Score***	***AUC***	**Waktu (detik)**
-	0.1000	0.0100	0.1000	0.0182	0.5000	149.217
0.001	0.7500	0.7539	0.7500	0.7500	0.9386	144.085
0.0001	0.9429	0.9501	0.9429	0.9426	0.9984	167.941
0.00001	0.9286	0.9345	0.9286	0.9272	0.9924	260.5

Learning rate adalah faktor yang mengontrol sejauh mana model belajar dari data pelatihan. *Learning rate* sangat berpengaruh pada performa model, apabila tidak menggunakan *learning rate,* model VGG16 menghasilkan kinerja yang buruk yaitu hanya mampu menghasilkan nilai *accuracy* 10%, *precision* 1%, *recall* 1%, *f1-score* 1.82% dan *AUC* 50% dengan waktu komputasi selama 149.217 detik. Sementara *learning rate* 0.001 menghasilkan kinerja yang lebih baik dengan waktu komputasi yang hanya selisih 5,132 detik yaitu 144.085 detik yang mampu menghasilkan nilai *accuracy* 75%, *precision* 75.39%, *recall* 75%, *f1-score* 75%, dan *AUC* 93.86%. Hasil pengujian *learning rate* 0.0001 memiliki performa terbaik yaitu menghasilkan nilai *accuracy* sebesar 94.29%, *precision* 95.01%, *recall* 94.26%, *f1-score* 99.84%, dan *AUC* 99.84% dengan waktu komputasi selama 167,914. Percobaan terakhir dilakukan dengan *learning rate* 0.00001 yang mampu menghasilkan nilai *accuracy* sebesar 92.86%, *precision* 93.45%, *recall* 92.86%, *f1-score* 92.72%, dan *AUC* 99.24% dengan waktu komputasi selama 260.5

detik. Waktu komputasi yang diperlukan lebih lama dari *learning rate* 0.0001 dengan selisih waktu 92,559 detik.

Berdasarkan hasil pengujian learning rate yang telah dipaparkan diatas, dapat diketahui bahwa pemilihan nilai *learning rate* yang sesuai sangat penting dalam proses pelatihan model. Apabila menggunakan *learning rate* terlalu rendah proses pembelajaran model akan menjadi lambat, sementara jika terlalu besar, dapat mengakibatkan divergensi, di mana model tidak dapat konvergen ke solusi yang baik. Oleh karena itu, *learning rate* 0.0001 dipilih untuk diterapkan dalam membangun sistem klasifikasi uang kertas Rupiah, karena memberikan keseimbangan yang baik antara performa dan waktu komputasi.

E. Skenario Pengujian Batch Size

Batch size dalam *deep transfer learning* merujuk pada jumlah sampel data yang diproses secara bersamaan pada setiap iterasi pelatihan model. Memilih *batch size* yang tepat penting untuk mempercepat pelatihan dan mencegah *overfitting*. Penelitian ini menguji berbagai variasi ukuran *batch size* untuk menemukan ukuran *batch size* yang menghasilkan performa paling optimal. Mulai dari tanpa *batch size*, ukuran *batch* 10, 20, dan 30. Setiap konfigurasi akan dianalisis untuk memahami pengaruhnya terhadap kinerja sistem. Dengan memperhatikan faktor-faktor seperti akurasi, presisi, *recall, f1-score, AUC,* serta waktu komputasi yang dibutuhkan. Diharapkan dapat diidentifikasi apakah penggunaan *batch size* tertentu dapat meningkatkan efisiensi dan efektivitas dalam pelatihan model klasifikasi uang kertas rupiah. Berikut adalah tabel skenario pengujian *batch size*:

TABEL V
SKENARIO PENGUJIAN BATCH SIZE

Pre Trained Model	***Hyperparameter***		
	Learning Rate	***Batch Size***	**Jumlah *Epochs***
VGG16	0.0001	-	15
VGG16	0.0001	10	15
VGG16	0.0001	20	15
VGG16	0.0001	30	15

F. Analisis Performansi Batch Size

Berikut adalah hasil dari pengujian *batch size* yang telah dilakukan sehingga didapatkan hasil performa model yaitu akurasi, presisi*, recall, f1-score*, *AUC* serta waktu komputasi yang digunakan.

TABEL VI
SKENARIO PENGUJIAN BATCH SIZE

Batch Size	**Akurasi**	**Presisi**	***Recall***	***F1-Score***	***AUC***	**Waktu (detik)**
-	0.6143	0.6972	0.6143	0.6019	0.9277	160.5
10	0.8429	0.8553	0.8429	0.8417	0.9826	150.4
20	0.9429	0.9501	0.9429	0.9426	0.9984	167.941
30	0.9071	0.9164	0.9071	0.9089	0.9672	170.89

Batch size merujuk pada jumlah sampel data yang diproses dalam satu iterasi dari algoritma pembelajaran mesin. Dalam pelatihan model, data dibagi menjadi beberapa *batch* agar dapat diolah secara efisien oleh komputer. Ukuran batch ini dapat bervariasi tergantung pada kebutuhan. Berdasarkan hasil pengujian pada tabel 4.6, dengan menggunakan *batch size* yang lebih kecil memungkinkan model mencapai nilai optimal lebih cepat daripada menggunakan *batch size* yang lebih besar. Penggunaan *batch size* yang besar dipilih karena dapat meningkatkan kecepatan komputasi hal ini terlihat pada pengujian dengan menggunakan *batch size* 30 membutuhkan waktu komputasi 170,89 detik sedangkan dengan *batch size* yang kecil yaitu 10 membutuhkan waktu komputasi yang lebih cepat 150,4 detik. *Batch size* 10 memiliki performa yang yang lebih rendah yaitu akurasi 84.29%, presisi 85.53%, *recall* 84.29%, *f1-score* 84.17%, dan *AUC* 98.26%. Batch size 30 menghasilkan nilai kinerja yang lebih tinggi yaitu akurasi 90.71%, presisi 91.64%, *recall* 90.71%, *f1-score* 90.89%, dan *AUC* 96.72%. Namun saat melakukan pelatihan model tanpa menggunakan *batch size*, performa model menurun drastis dan hanya memperoleh akurasi 61.43%, presisi 69.72%, *recall* 61.43%, *f1-score* 60.19%, dan *AUC* 92.77%. Saat tidak menggunakan *batch size* waktu komputasi tidak jauh berbeda dengan pengujian menggunakan *batch size* yaitu membutuhkan waktu 160.5 detik. Konfigurasi *batch size* paling ideal untuk penelitian ini yaitu menggunakan *batch size* 20. Menghasilkan performa yang lebih tinggi dari *batch size* 10 dan 30. *Batch size* 20 memperoleh nilai akurasi 94.29%, presisi 96.01%, *recall* 94,29%, *f1-score* 94.26% dan *AUC* 99,84%. Membutuhkan waktu komputasi yang cepat yaitu 167.941 detik.

G. Skenario Pengujian Jumlah Epochs

Epoch adalah iterasi yang dilakukan oleh sistem untuk mempelajari data pelatihan. Jika kemampuan memahami data pelatihan semakin meningkat, kinerja sistem dalam memprediksi data uji juga akan semakin baik [6]. Dalam penelitian ini, dibandingkan berbagai variasi *epochs*, yaitu jumlah epochs 5, 10, 15, dan tanpa menggunakan *epochs*. Berikut adalah skenario pengujiannya:

TABEL VII
SKENARIO PENGUJIAN JUMLAH EPOCHS

Pre Trained Model	***Hyperparameter***		
	Learning Rate	***Batch Size***	**Jumlah *Epochs***
VGG16	0.0001	20	Tanpa Epochs
VGG16	0.0001	20	5
VGG16	0.0001	20	10
VGG16	0.0001	20	15

H. Analisis Performansi Jumlah Epochs

Berikut adalah hasil dari pengujian jumlah *epochs* yang telah dilakukan sehingga didapatkan hasil kinerja model yaitu akurasi, presisi*, recall, f1-score*, *AUC* serta waktu komputasi yang digunakan.

TABEL VIII
SKENARIO PENGUJIAN JUMLAH EPOCHS

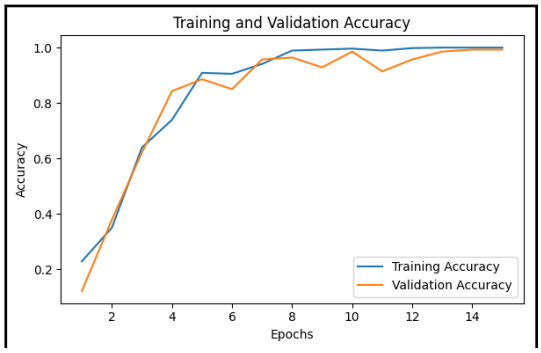
Jumlah *Epochs*	Akurasi	Presisi	*Recall*	*F1 Score*	*AUC*	Waktu (detik)
-	0.1000	0.0100	0.1000	0.0182	0.5000	90.500
5	0.6714	0.6954	0.6714	0.6729	0.9042	130.943
10	0.9143	0.9181	0.9143	0.9103	0.9953	145.905
15	0.9429	0.9501	0.9429	0.9426	0.9984	167.941

Berdasarkan hasil yang telah ditampilkan pada Tabel VIII, maka didapatkan nilai *epochs* ideal dengan kinerja terbaik yaitu 15 *epochs*. Menghasilkan nilai akurasi 94.29%, presisi 96.01%, *recall* 94,29%, *f1-score* 94.26% dan *AUC* 99,84%. Waktu komputasi yang diperlukan tidak terlalu jauh berbeda dengan variasi *epochs* lainnya yaitu 167,914 detik. Selisih 22 detik dari *epochs* 10 yang membutuhkan waktu komputasi 145,905 detik. *Epochs* 10 menghasilkan kinerja lebih rendah yaitu akurasi 91.43%, presisi 91.81%, *recall* 91.43%, *f1-score* 91.03% dan *AUC* 99.53%. Sementara nilai *epochs* terendah yaitu 5 menghasilkan kinerja yang kurang baik dengan akurasi 67.14%, presisi 69.54%, *recall* 67.14%, *f1-score* 67.29% dan *AUC* 90.42% waktu komputasi yang diperlukan lebih cepat yaitu 130,943 detik. Ini terjadi karena sistem belum optimal dalam memahami data latihan. Karena seiring dengan peningkatan jumlah *epochs*, model memiliki lebih banyak kesempatan untuk belajar dari data pelatihan dan menyesuaikan dengan pola-pola yang lebih kompleks, yang pada akhirnya dapat meningkatkan kinerja model.

Pengaturan nilai *epochs* yang tepat penting untuk dilakukan karena apabila tanpa menggunakan *epochs*, model tidak memiliki kesempatan untuk melihat dataset secara menyeluruh atau mempelajari pola-pola yang ada dalam data pelatihan. Karena tidak mengalami pelatihan yang memadai, model tidak dapat menyesuaikan bobotnya atau memperbaiki kinerjanya. Hal ini menyebabkan hasil yang rendah pada metrik evaluasi. Hasil kinerja yang didapatkan sangat rendah dengan nilai akurasi 10%, presisi 1%, *recall* 10%, *f1-score* 1.82% dan *AUC* 50%. Dalam konteks *deep transfer learning*, *epochs* menentukan berapa kali seluruh dataset diberikan ke dalam model untuk pelatihan. Tanpa iterasi ini, model tidak akan mampu belajar dan tidak dapat melakukan klasifikasi dengan baik. Oleh karena itu, penting untuk menentukan nilai *epochs* yang ideal. Sehingga model memiliki kesempatan untuk belajar dari data pelatihan dan dapat meningkatkan kinerjanya.

I. Evaluasi Model Klasifikasi

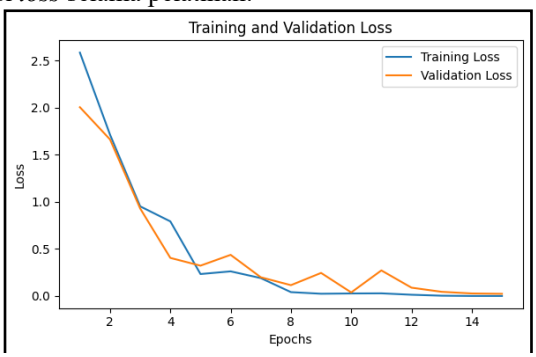
Berdasarkan hasil eksperimen yang telah dilakukan dalam penelitian ini, *pretrained model* VGG16 tanpa *fine tuning* dipilih untuk mencapai kinerja model yang optimal, dengan nilai *learning rate* 0.0001, ukuran *batch size* 20, dan jumlah *epoch* 15. Kemudian dilakukan pelatihan dengan optimasi *Adam*, sehingga menghasilkan diagram hasil evaluasi selama pelatihan sebagai berikut:



Gbr. 6 Grafik Akurasi dan Validasi Selama Pelatihan.

Berdasarkan grafik akurasi dan validasi selama pelatihan pada Gbr.6, menunjukkan performa model yang baik. terlihat bahwa grafik akurasi dan validasi bertahap naik secara konsisten, mendekati atau bahkan mencapai nilai maksimum yang diinginkan, yaitu 1. Grafik yang menunjukkan tren naik yang konsisten dan stabil ini mengindikasikan bahwa model sedang belajar dengan baik dari dataset pelatihan. Selain itu, perbedaan antara skor akurasi dan validasi semakin mengecil seiring berjalannya waktu, menunjukkan bahwa model tidak hanya mengingat informasi dari data pelatihan, tetapi juga mampu menerapkan pengetahuannya dengan baik pada data baru yang belum pernah ditemui sebelumnya. Hal ini menggambarkan keandalan dan keefektifan model dalam melakukan prediksi yang akurat. Dengan grafik yang menunjukkan performa yang konsisten meningkat dan kedua skor akurasi dan validasi yang mendekati nilai maksimum yang diinginkan, dapat disimpulkan bahwa model tidak mengalami *overfitting*.

Selanjutnya, untuk memperoleh pemahaman yang lebih lengkap tentang kinerja model, akan dibahas juga grafik *training* dan validasi *loss*. Berikut adalah grafik training dan validasi *loss* selama pelatihan:

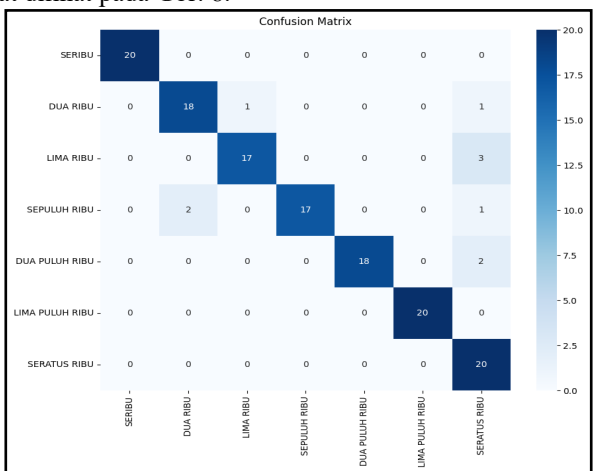


Gbr. 7 Grafik Loss Selama Pelatihan.

Selanjutnya, untuk memperoleh gambaran yang lebih lengkap tentang performa model, perlu diperhatikan grafik training dan validasi loss. Gbr.7, menggambarkan perubahan dalam nilai *loss* (kerugian) selama proses pelatihan model. Dapat diamati bahwa kedua kurva loss menunjukkan kecenderungan penurunan yang stabil seiring dengan meningkatnya jumlah *epochs*, dan kurva perlahan mendekati nilai 0. Terdapat titik temu antara kurva *training loss* dan

validation loss, di mana perbedaan antara keduanya semakin mengecil seiring berjalannya waktu. Hal ini menunjukkan bahwa model memiliki kemampuan yang baik dalam menggeneralisasi data baru yang belum pernah ditemui sebelumnya, serta menunjukkan bahwa tidak terjadi *overfitting*.

Langkah selanjutnya adalah melakukan *testing* data *test* sebanyak 140 citra. *Testing* dilakukan untuk mengukur seberapa baik model dapat mengklasifikasikan citra baru. Hasil klasifikasi ditampilkan dalam *confusion matrix* yang dapat dilihat pada Gbr. 8.



Gbr. 8 Confusion Matrix Hasil Klasifikasi.

Berdasarkan hasil *testing* data citra, didapatkan hasil metrik evaluasi untuk masing-masing nominal mata uang. Model dapat memprediksi benar di nominal seribu, lima puluh ribu, dan seratus ribu. Nominal lainnya model masih terdapat kesalahan dalam memprediksi data baru. Berikut adalah hasil matriks evaluasi testing:

TABEL IX
HASIL MATRIKS EVALUASI TESTING

Nominal	Akurasi	Presisi	*Recall*	*F1-Score*	*AUC*
1 RB	0.9286	1.0000	1.0000	1.0000	0.9964
2 RB	0.9286	0.9000	0.9000	0.9000	0.9964
5 RB	0.9286	0.9444	0.8500	0.8947	0.9964
10 RB	0.9286	1.0000	0.8500	0.9189	0.9964
20 RB	0.9286	1.0000	0.9000	0.9474	0.9964
50 RB	0.9286	1.0000	1.0000	1.0000	0.9964
100 RB	0.9286	0.7407	1.0000	0.8511	0.9964
Rata-Rata	0.9286	0.9407	0.9286	0.9303	0.9964

Berdasarkan Tabel IX, hasil matriks evaluasi pada saat dilakukan pengujian menggunakan data *test* menunjukkan bahwa model dapat memprediksi citra uang kertas rupiah tahun emisi 2022 dengan baik. Hasil prediksi model menghasilkan rata-rata nilai *accuracy* 92.86%, *precision* 94.07%, recall 92.86%, *f1-score* 93.03%, dan *AUC* 99.64%.

IV. KESIMPULAN

Berdasarkan hasil penelitian, pemilihan arsitektur *pretrained model* yang cocok untuk dataset, dapat meningkatkan kinerja model klasifikasi. Selain itu nilai *learning rate* harus disesuaikan karena jika terlalu kecil, model akan mempelajari data pelatihan secara lambat, sedangkan jika terlalu besar dapat menyebabkan divergensi. Semakin optimal *nilai learning rate* yang dipilih, semakin baik pula kinerja model dalam memprediksi data uji.

Jumlah *epoch* yang terlalu sedikit menyebabkan kinerja model kurang optimal karena model tidak dapat memperoleh pemahaman yang memadai terhadap fitur-fitur yang ada dalam data pelatihan. *Epochs* memegang peran penting, karena tanpa epochs model tidak dapat belajar dengan baik sehingga berdampak pada kinerja. Menentukan jumlah *epochs* yang tepat sangat krusial untuk memastikan model memiliki cukup kesempatan untuk belajar dari data pelatihan dan meningkatkan kinerjanya.

Ukuran *batch size* kecil memungkinkan model mencapai hasil optimal lebih cepat daripada *batch size* besar. Namun, performa model tidak mengalami peningkatan yang signifikan dibandingkan dengan *batch size* kecil.

V. SARAN

Berikut adalah saran dapat diberikan untuk penelitian selanjutnya yaitu:

1. Menggunakan dataset yang lebih banyak lagi, karena pada penelitian ini hanya terbatas pada uang kertas emisi 2022, sedangkan uang yang beredar dimasyarakat ada yang masih menggunakan emisi tahun sebelumnya.
2. Apabila jumlah dataset sedikit atau kurang dari 1.000, harap dilakukan augmentasi data untuk memperbanyak varian dataset.
3. Lakukan percobaan dengan arsitektur *transfer learning* lainnya seperti ResNet50, EfficientNet, Xception, Inception, MobileNet, DenseNet, NasNet, dan yang paling baru yaitu ConvNeXt.
4. Model yang dibangun sudah optimal untuk klasifikasi uang kertas rupiah emisi 2022, sehingga penulis berharap dapat dimanfaatkan untuk pengembangan sistem yang dapat berguna bagi masyarakat, seperti dapat digunakan untuk sistem kasir pintar, mesin penjualan otomatis, bahkan dapat digunakan untuk aplikasi peduli sosial seperti alat pendeteksi uang untuk tuna netra.

UCAPAN TERIMA KASIH

Puji syukur kehadirat Allah SWT, karena berkat rahmat-Nya sehingga penulis mampu menyelesaikan penelitian ini. Terimakasih penulis ucapkan kepada orang tua yang telah

mendoakan dan memberikan dukungan. Terimakasih yang sebesar-besarnya kepada dosen pembimbing yang senantiasa memberikan bimbingan terbaiknya. Terimakasih penulis ucapkan kepada teman-teman yang telah membantu dan mendukung penulis dalam menyelesaikan penelitian ini.

REFERENSI

[1] A. M. N. Hidayat and A. Antamil, "Identifikasi Nominal Mata Uang Rupiah Bagi Penyandang Tunanetra Dengan Algoritma Convolutional Neural Network Berbasis Android," *JOURNAL SHIFT VOL*, vol. 3, 2023.

[2] A. C. Catyaningga, "Klasifikasi Uang Kertas Rupiah Berdasarkan Angka Nominal Secara Realtime Menggunakan Metode Single Shot Detector," 2023.

[3] I. Pramudiono, "Pengantar Data Mining: Menambang Permata Pengetahuan di Gunung Data," *Jurnal Ilmu Komputer*, 2003.

[4] N. Dewi and F. Ismawan, "Implementasi Deep Learning Menggunakan CNN untuk Sistem Pengenalan Wajah," *Faktor Exacta*, vol. 14, no. 1, p. 34, Mar. 2021.

[5] I. M. G. I. Mahendra, "Pengembangan Model Pembelajaran Mesin untuk Mengidentifikasi Nominal Uang Kertas Rupiah Menggunakan Algoritma CNN Dengan Library Keras," 2022.

[6] R. N. Izah, "Klasifikasi Nominal Uang Kertas Rupiah Tahun Emisi 2017 dengan Algoritma CNN menggunakan MXNET," *Tugas Akhir, Jurusan Statistika, Universitas Islam Indonesia, Yogyakarta.*, 2018.

[7] M. Albani and R. R. Andhi, "Klasifikasi Uang Rupiah Kertas Tidak Layak Edar Menggunakan CNN Xception Transfer Learning Berbasis Website," *Jurnal Inovtek Polbeng Seri Informatika*, vol. 8, no. 2, pp. 393–406, 2023.

[8] R. Faturahman, Y. S. HARIYANI, and S. HADIYOSO, "Klasifikasi Jajanan Tradisional Indonesia berbasis Deep Learning dan Metode Transfer Learning," *ELKOMIKA: Jurnal Teknik Energi Elektrik, Teknik Telekomunikasi, & Teknik Elektronika*, vol. 11, no. 4, p. 945, Oct. 2023.

[9] G. Gumelar, Q. Ain, R. Marsuciati, S. Agustanti Bambang, A. Sunyoto, and M. Syukri Mustafa, "Kombinasi Algoritma Sampling dengan Algoritma Klasifikasi untuk Meningkatkan Performa Klasifikasi Dataset Imbalance," in *Prosiding SISFOTEK*, 2021.

[10] J. E. Widyaya and S. Budi, "Pengaruh Preprocessing Terhadap Klasifikasi Diabetic Retinopathy dengan Pendekatan Transfer Learning Convolutional Neural Network," *Jurnal Teknik Informatika dan Sistem Informasi*, vol. 7, no. 1, Apr. 2021.

[11] Q. Aina Fitroh, "Deep Transfer Learning untuk Meningkatkan Akurasi Klasifikasi pada Citra Dermoskopi Kanker Kulit," *JURNAL NASIONAL TEKNIK ELEKTRO DAN TEKNOLOGI INFORMASI*, 2023.

[12] S. Yulina and H. Rachmawati, "Analisis Kinerja Model Klasifikasi Pada Multiclass Facial Expression Recognition Berbasis Fitur Eigenface," *Jurnal Sisfokom (Sistem Informasi dan Komputer)*, vol. 12, no. 3, 2023.

[13] A. Muh. A. Siddik, "Comparison of Transfer Learning Algorithm Performance in Hand Sign Language Digits Image Classification," *Jurnal Matematika, Statistika dan Komputasi*, vol. 20, no. 1, pp. 75–89, Sep. 2023, doi: 10.20956/j.v20i1.26503.

[14] N. Rochmawati *et al.*, "Analisa Learning rate dan Batch size Pada Klasifikasi Covid Menggunakan Deep learning dengan Optimizer Adam," *Journal Information Engineering and Educational Technology*, vol. 5, no. 2, 2021.