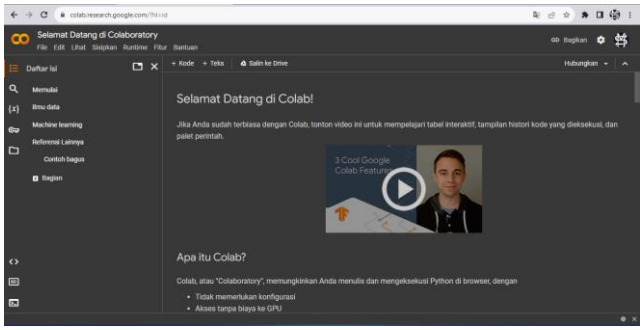
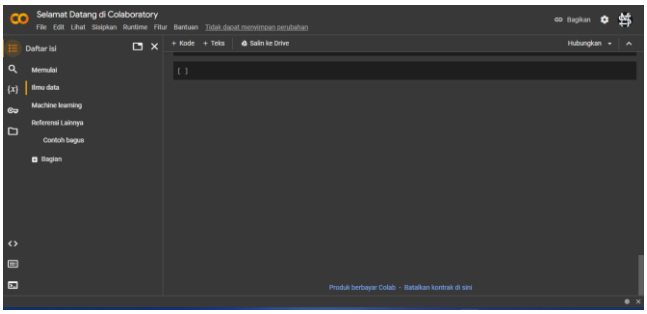
BAB 3
METODE PENELITIAN

3.1 PEMODELAN SISTEM

Penelitian ini membangun sistem yang dapat mengklasifikasikan uang kertas nominal Tahun Emisi 2022. Tujuannya adalah untuk membantu tunanetra atau orang yang mengalami gangguan penglihatan dengan menggunakan algoritma Jaringan *Neural* Konvelusi (CNN) untuk menemukan uang nominal. Simulasi model yang digunakan dalam penelitian ini dilakukan dengan bantuan alat *Google Colab* dan bahasa pemrograman *Python*. Versi saat ini dari *Google Colab* didasarkan pada aplikasi *online open source* yang memungkinkan pengkodean, kolaborasi, visualisasi, dan analisis teks. *Google Colab* menyediakan informasi berikut.



Gambar 3. 1 Tampilan awal *Google Colab*



Gambar 3. 2 Tampilan *New Notebook* pada *Google Colab*

Dalam Gambar 3.1 dan 3.2 yang disajikan di atas, tampilan awal atau File Baru terletak di situs *web Google Colab*, dan dapat digunakan untuk tujuan membangun kode *Script*. Temuan dari penelitian ini terdiri dari dua kode *Script*, yang pertama digunakan untuk membagi dataset menjadi tiga folder: *Train*, validasi, dan Test. Selain itu, data dianalisis untuk menentukan apakah setiap set

sudah selesai atau apakah harus diselesaikan. *Script code* yang digunakan untuk proses klasifikasi pasangan mata uang berdasarkan data pelatihan, validasi, dan data pengujian dengan menggunakan beberapa parameter dan hyperparameter disebut sebagai "dua sisi" *Script code*. Ketika model telah dibangun dan dievaluasi, nilai akurasi 40 dan kerugian yang diperoleh dari model yang telah dibina merupakan titik penting yang perlu dianalisis sesuai dengan banyak *Epoch* yang telah dilakukan.

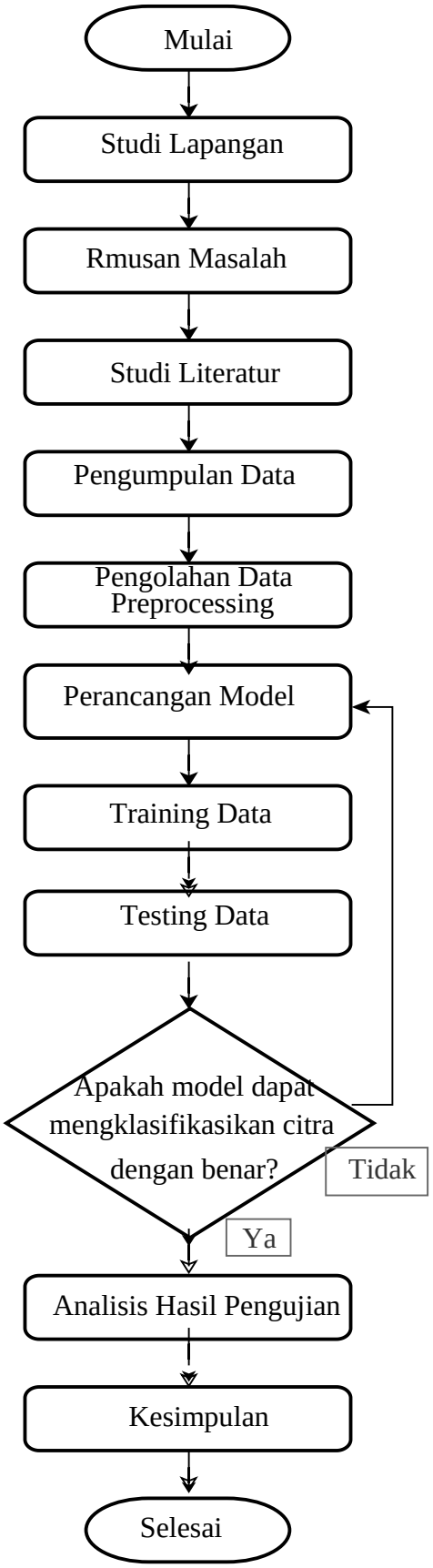
Pada penelitian ini dibutuhkan bantuan laptop untuk menjalankan *Google Colab* dengan spesifikasi sebagai berikut.

1. *Processor* : AMD Athlon Silver 3050U with
2. *RAM* : 4,00 GB
3. *System type* : 64-bit *operating system, x64-based processor*
4. *Graphics* : *Radeon Graphics* 2.30 GHz

Karakteristik laptop juga memiliki dampak pada jumlah waktu yang dibutuhkan untuk meluncurkan program di *Google Colab* atau untuk melakukan tugas biasa. Oleh karena itu, dengan menggunakan laptop dengan tingkat spesifikasi yang tinggi, akan memungkinkan untuk mencapai tingkat pengambilan data yang diinginkan. Namun, jika Anda menggunakan laptop atau komputer pribadi dengan konfigurasi low-end dan sejumlah besar data, maka proses data pelatihan akan memakan waktu lebih lama untuk selesai. Untuk tujuan penyelidikan ini, Sistem Operasi (OS) yang digunakan sebenarnya adalah Windows 11. Selain Windows 11, *Google Colab* juga mampu bekerja pada sistem operasi seperti Windows 7, Ubuntu Linux, dan Debian Linux. Jika spesifikasi laptop tidak sangat tinggi, maka mungkin untuk menginstal sistem operasi berbasis Linux karena sebagian besar pengguna akan menggunakan baris perintah.

Selain itu, selain membutuhkan laptop dan *Google Colab*, proyek penelitian tersebut juga membutuhkan perpustakaan *Python*. Dalam lingkup studi ini, sejumlah perpustakaan *Python* digunakan, yang paling menonjol adalah *TensorFlow*, yang digunakan dalam *code Script*.

3.2 ALUR PENELITIAN



Gambar 3. 3 Diagram Alur Penelitian

Penggambaran Alur penelitian yang harus dilakukan secara bertahap untuk digunakan dalam penelitian digambarkan dalam diagram aliran pada Gambar 3.3, dan penjelasan lengkap aliran penelitian diberikan sebagai berikut:

3.2.1 Studi Lapangan

Studi lapangan adalah salah satu cara untuk mendapatkan pemahaman langsung tentang konteks. Dengan pengamatan langsung terhadap objek yang diteliti dapat merumuskan pertanyaan penelitian yang lebih tepat. Informasi dari studi lapangan juga membantu menentukan lingkup penelitian dan memberikan konteks yang penting untuk interpretasi hasil analisis klasifikasi.

3.2.2 Rumusan Masalah

Rumusan masalah dibuat dengan tujuan memberikan arah dan fokus yang jelas dalam menetapkan ruang lingkup penelitian, mengidentifikasi tujuan penelitian, memberikan dasar bagi hipotesis atau pertanyaan penelitian, dan memberikan alasan perlu dilakukannya penelitian. Perumusan masalah ini dibuat dengan mempertimbngkan beberapa hal yang terdiri dari segi metode, hasil penelitian, dan analisis penelitian yang dilakukan. Tujuan dari penelitian ini ialah untuk mengetahui bagaimana sistem dapat digunakan untuk mendeteksi nominal mata uang kertas rupiah dengan menggunakan metode CNN. Sehingga didapatkan hasil yang sesuai dengan rumusan yang telah dibuat.

3.2.3 Studi Literatur

Langkah pertama dalam proses penelitian adalah meninjau literatur. Setelah itu, informasi dan data akan dikumpulkan untuk penelitian. Untuk melakukan penelitian ini, sejumlah artikel dari berbagai jurnal, buku, dan situs web dibaca. Artikel-artikel ini membahas topik yang mungkin terkait dengan klasifikasi gambar. Beberapa makalah yang dihasilkan dari penelitian sebelumnya termasuk makalah nasional dan internasional. Untuk meningkatkan kinerja penelitian ini, beberapa bahan yang sedang dipelajari termasuk nilai uang kertas rupiah, citra digital, pembelajaran mendalam, algoritma *Convolutional Neural Network* (CNN), metode sequential, peningkatan data, set pelatihan, validasi, dan pengujian, dan kehilangan dan ketepatan.

3.2.4 Pengumpulan Data

Pada Proses pengumpulan data dilakukan dengan mengumpulkan data dalam bentuk representasi grafis dari mata uang rupiah. Nilai uang kertas yang digunakan dalam penelitian ini adalah sebagai berikut: uang kertas dengan nilai nominal berkisar dari Rp 1.000 hingga Rp 100.000 sepanjang tahun 2022. Selama fase pengumpulan data ini, prosesnya dilakukan dengan cara yang sepenuhnya independen, dengan tujuan menghasilkan data untuk tujuan mengidentifikasi kasus plagiarisme. Selain itu, dengan menggunakan dataset yang disediakan oleh penulis, dimungkinkan untuk mengumpulkan informasi dari penelitian yang dilakukan. Alasannya adalah bahwa akan lebih mudah untuk mendapatkan data dari citran itu sendiri.

Gambar 3. 4 Hp Iphone Xr

Data citra uang tersebut diperoleh dengan melakukan pengambilan gambar secara langsung menggunakan Main Kamera dari ponsel Iphone Xr seperti pada Gambar 3.4 dengan bantuan tripod.

Tabel 3. 1 ***Insert file dataset***

Jenis uang	Kelas	Jumlah Data
Rp. 1.000	1k	200 citra
Rp. 2.000	2k	
Rp. 5.000	5k	
Rp. 10.000	10k	
Rp. 20.000	20k	
Rp. 50.000	50k	
Rp. 100.000	100k	

Pada Tabel 3.1 merupakan jumlah dari pengumpulan data yang dimana pada jenis uang kertas dan kelas yang diberikan dengan jumlah data masing-masing kelas sebanyak 200 citra. Seperti jenis uang Rp. 1.000 dengan kelasnya 1k sebanyak 200 citra yang artinya pada angka 1 yaitu nominal uang kertas bernilai Rp.1.000. Data tersebut termasuk ke dalam jenis uang yang akan digunakan. Kemudian dari hasil data yang di dapat yaitu sebanyak 1400 citra sebagai dataset yang terbagi menjadi 7 kelas.

Gambar 3. 5 Citra uang kertas tahun emisi 2022 (tampak depan)

Gambar 3. 6 Citra uang kertas tahun emisi 2022 (tampak belakang)

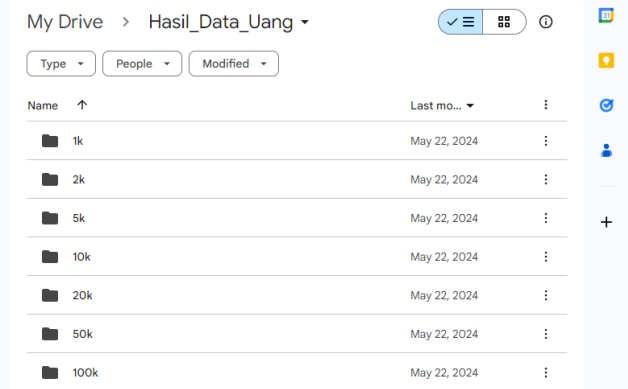
Gambar 3.5 dan Gambar 3.6 adalah contoh data yang digunakan dalam penelitian ini karena signifikansi mereka. Jumlah data yang digunakan dalam penelitian ini adalah 1400 citra, dan terdiri dari tujuh kelas citra yang berbeda. Tanda-tanda tersebut adalah sebagai berikut: Rp 1.000, Rp 2.000, Rp 5.000, Rp 10.000, Rp 20.000, Rp 50.000, dan Rp 100.000. Masing-masing kelas citra terdiri

dari 200 citra, 100 citra sisi uang bagian depan, dan 100 Citra sisi uang bagian belakang di kedua sisi bagian.

Selama fase berikutnya, tugas yang sedang dilakukan adalah mengumpulkan dataset yang akan digunakan sebelum hasil muncul. Proses memperoleh data yang dimasukkan dalam penelitian ini dilakukan dengan menempatkan folder citra di folder yang sama dengan folder lainnya. Sebagai hasilnya, data diatur menjadi tujuh folder, yang merupakan sebagai berikut: folder Rp 1.000, folder Rp 2.000, folder Rp 5.000, folder Rp 10.000, folder Rp 20.000, folder RP 50.000, dan folder Rp 100.000. Terlepas dari fakta bahwa mata uang terdiri dari sisi depan dan sisi belakang, itu masih dikumpulkan bersama menjadi satu folder yang identik dengan nilai nominal mata uang. Sebelum data dibagi, mata uang yang disebutkan di atas harus dinamai ulang secepat mungkin untuk memfasilitasi proses pembagian dataset. Setiap folder, juga dikenal sebagai kelas uang, memiliki label yang unik untuk dirinya sendiri dan berbeda dari label folder lainnya. Dalam contoh ini, folder yang berisi seribu rupiah diberi label seribu pada format nama folder, dan kemudian disertai dengan tujuh jenis mata uang lainnya. Meskipun kelas yang disebutkan di atas tidak memiliki kriteria khusus, penting untuk mempertimbangkan bahwa format kelas harus jelas dan konsisten untuk mencegah masalah apa pun yang terjadi selama proses menjalankan program kode atau membagi dataset.

Setelah proses selesai, data citra uang yang telah diatur menjadi tujuh folder menurut denominasi moneter telah disiapkan dan siap untuk dibawa ke depan. Pada tahap pra-pengolahan ini, data yang sedang diproses dibagi menjadi tiga kategori: data *Train* (data latih), data validasi (untuk tujuan validasi data latih) dan data *Test*. (*for the purpose of evaluating the data*). Validasi dataset ini dilakukan secara otomatis dengan menggunakan kode skrip yang sebelumnya telah validasi oleh validasi dari dataset lain. Dalam proses pembelajaran transfer model, proses pengumpulan data dilakukan secara otomatis oleh sistem. Secara keseluruhan, ada 1400 citra, dan data didistribusikan sebagai berikut: enam puluh persen untuk tujuan pelatihan, dan empat puluh peratus untuk tujuan pengujian. *By the way*, dari 1400 citra, sekitar enam puluh persen atau delapan ratus empat puluh citra digunakan sebagai data *Train*. Sisa empat puluh persen atau lima ratus

enam puluh persen dapat digunakan untuk pengujian data. 560 citra yang disebutkan di atas kemudian dibagi menjadi dua kelompok untuk tujuan validasi data dan pengujian data, dengan masing-masing kelompok diberikan persentase dari lima puluh persen sampai lima puluhan persen. Dengan cara ini, 280 citra digunakan untuk tujuan validasi data, dan 280 citra digunakan untuk keperluan pengujian data. Setiap folder yang berisi data *Train*, validasi data, dan data Test akan memiliki tujuh sub folder yang masing-masing berisi jumlah uang yang berbeda. Jumlah uang yang terkandung dalam masing-masing subfolder ini adalah sebagai berikut: Rp 1.000, Rp 2.000, Rp 5.000, Rp 10.000, Rp 20.000, Rp 50.000, dan Rp 100.000. Akibatnya, proses pengumpulan data dilakukan oleh sistem dengan cara yang jelas dan otomatis untuk memfasilitasi proses *transfer Learning*.



Gambar 3. 7 Hasil data uang masing masing kelas

Gambar 3.7 setiap folder dari hasil data uang berisi 7 sub-folder yang masing-masing mengandung citra uang dengan denominasi Rp 1.000, Rp 2.000, Rp 5.000, Rp 10.000, Rp 20.000, Rp 50.000, dan Rp 100.000. Proses pembagian ini dilakukan secara acak sehingga nomor label dari citra masing-masing folder tidak akan berurutan di dalam folder. Dengan pembagian yang acak ini, model CNN diharapkan dapat dilatih, divalidasi, dan diuji dengan data yang representatif dan tidak terurut, yang membantu meningkatkan kemampuan generalisasi model dalam mengenali citra uang dari berbagai denominasi.

3.2.5 Pengolahan Data (*Preprocessing)*

Proses preprocessing dilakukan dengan tujuan untuk mempersiapkan citra sebelum diproses oleh model. Langkah-langkah dalam *preprocessing* meliputi

pembacaan, konversi, pengubahan ukuran, dan normalisasi citra. Pertama, label-label yang berisi denominasi uang seperti '1k', '2k', '5k', '10k', '20k', '50k', dan '100k' ditentukan bersama dengan ukuran target citra sebesar 224x224 piksel. Fungsi *`get_data`* digunakan untuk membaca dan memproses gambar dari setiap folder yang sesuai dengan labelnya. Setiap gambar dibaca menggunakan *OpenCV* melalui fungsi *`cv2.imread`*, yang memuat citra dalam format BGR. Untuk keperluan pemrosesan lebih lanjut, citra kemudian dikonversi dari format BGR ke format RGB menggunakan fungsi *`cv2.cvtColor`*. Konversi ini penting karena banyak *library* pemrosesan citra dan model *Deep Learning* bekerja lebih optimal dengan citra dalam format RGB. Setelah konversi, citra diubah ukurannya menjadi 224x224 *piksel* menggunakan fungsi *`cv2.resize`*. Ukuran ini dipilih untuk memastikan bahwa semua citra memiliki dimensi yang konsisten saat dimasukkan ke dalam model.

Setiap gambar yang berhasil diproses kemudian ditambahkan ke dalam list bersama dengan labelnya dalam bentuk indeks dari daftar label tersebut. Proses ini memastikan bahwa setiap gambar dipasangkan dengan label yang sesuai, memungkinkan model untuk mempelajari hubungan antara citra dan labelnya selama *Training*. Untuk memastikan bahwa gambar telah di *resize* dan dikonversi dengan benar, gambar-gambar tersebut divisualisasikan menggunakan *`matplotlib`*. Setiap gambar ditampilkan dalam *subplot* untuk memberikan visualisasi yang jelas. Hal ini membantu dalam melakukan verifikasi visual bahwa *preprocessing* telah dilakukan dengan benar sebelum data digunakan dalam model. Setelah semua gambar diproses, list gambar tersebut dikonversi menjadi *Numpy array*. Konversi ini penting karena banyak operasi pemrosesan data dalam *TensorFlow* dan *Keras* memerlukan input dalam format *Numpy array*. Normalisasi dilakukan dengan membagi setiap *piksel* dengan 255.0. Proses normalisasi ini mengubah rentang piksel dari [0, 255] menjadi [0, 1]. Normalisasi sangat penting karena model *Neural Network* cenderung bekerja lebih baik dan lebih cepat pada data yang memiliki rentang nilai yang lebih kecil dan seragam.

Dengan preprocessing ini, semua gambar telah diubah ke ukuran yang sama dan dinormalisasi, sehingga siap untuk diinputkan ke dalam model untuk proses *Training*, *Validation*, dan *Testing*. Langkah-langkah preprocessing ini

memastikan bahwa data yang diinputkan ke dalam model adalah konsisten dan terstruktur, yang merupakan faktor penting untuk mendapatkan hasil yang akurat dan andal dari model *Machine Learning* atau *Deep Learning.*

3.2.6 Perancangan Model Dengan Klasifikasi Algoritma CNN

Salah satu aspek yang paling penting dari penelitian ini adalah klasifikasi gambar, yang sering dikenal sebagai kualifikasi gambar. Tujuan dari jadwal ini adalah untuk memungkinkan Anda untuk secara akurat mendeteksi nilai mata uang yang Anda gunakan. Sebuah penjelasan untuk penggunaan CNN dalam klasifikasi data karena CNN dirancang untuk mengatur dan menganalisis data dalam dua dimensi, dan banyak dari itu digunakan untuk mengklasifikasikan data. CNN akan menggunakan proses yang disebut "kode konvolusi" untuk mengubah satu gambar, dan kemudian, sebagai hasilnya, komputer akan dapat memperoleh informasi dari sumber gambar dengan menerapkan filter pada gambar. Sistem klasifikasi ini sangat penting karena digunakan untuk tujuan mendapatkan tingkat akurasi yang tinggi pada model yang sedang dievaluasi. Klasifikasi tersebut menggunakan beberapa lapisan dan *kernel* untuk memastikan bahwa model dapat memahami data tersebut dan memberikan penjelasan untuknya. Data dari *Train* kemudian digunakan untuk melakukan proses pelatihan CNN, yang melibatkan penggunaan parameter dan dilakukan dengan cara yang berkelanjutan dan berlanjut sampai jumlah maksimum perulangan batas yang telah ditentukan. (*Epoch*).

Dalam penyelidikan saat ini, parameter yang sedang dipertimbangkan adalah optimasi, jumlah *Epoch* yang dilakukan, jumlah lapisan konvolusi, ukuran kernel yang digunakan, dan ukuran *batch*. Baik tingkat belajar dan langkah-langkah per era. Istilah "*hyperparameter*" mengacu pada fakta bahwa parameter tertentu ini dapat diubah oleh peneliti untuk tujuan mereka sendiri. Sejak awal Zaman, berat besar akan selalu diperbarui sesuai dengan tingkat pembelajaran besar yang sedang digunakan. Selama pelatihan data, setiap dan setiap dataset akan dapat dianalisis dan dieksekusi dalam batch masing-masing. Dalam proses belajar dataset yang sekarang digunakan, model akan dapat mempelajari dan menganalisis data yang sedang digunakan, dan jika ada dataset baru yang

digunakan, maka proses validasi data akan dilakukan untuk menentukan apakah model masih mampu belajar dan menganalisa data dengan data yang berbeda dari data yang digunakan untuk pelatihan. Setelah menyelesaikan proses pelatihan atau pelatihan data, Anda akan dapat mendapatkan model yang cocok untuk proses klasifikasi. Model yang akan digunakan dalam penyelidikan ini terdiri dari tiga model arsitektur yang berbeda: model aritektur yang menggunakan EvvicientNetV2m, model arsitektur yang menggunakan VGG16, dan model arsitektur yang menggunakan *EfficientNet*B0. Akurasi dan kerugian yang diperoleh dari empat model yang sedang dibahas dapat dibandingkan dengan hasil yang dicapai ketika data pelatihan dan validasi disesuaikan sesuai dengan hyperparameter yang sebelumnya ditentukan oleh penulis. Ini memungkinkan untuk menghasilkan nilai akurasi dan kerugian yang lebih unggul dari yang diperoleh dari model lain yang sedang dibangun. Nilai Hyperparameter yang digunakan dengan cara ini juga secara signifikan berkontribusi pada perbaikan model dalam hal analisis data. Selama proses belajar ini, setelah data telah diproses, itu akan diubah menjadi bentuk data pixel dan data label, yang kemudian akan berubah menjadi input untuk fase berikutnya dari proses belajar.

3.2.7 *Training* Model

Pada tahap pelatihan atau *Training*, model *Convolutional Neural Network* (CNN) akan mempelajari data *Training* yang terdiri dari citra uang kertas dengan 7 kelas yang berbeda (1k, 2k, 5k, 10k, 20k, 50k, dan 100k). Tujuan dari pelatihan ini adalah untuk melatih model agar dapat mengenali dan mengklasifikasikan citra uang kertas dengan benar. Pelatihan model dilakukan dengan memanfaatkan *transfer Learning*, di mana terdapat 3 model yang digunakan pada penelian ini diantaranya adalah *EfficientNetV2M*, VGG 16 dan VGG 19 yang telah dilatih sebelumnya pada dataset *ImageNet*.

Selama proses pelatihan, lapisan-lapisan tambahan ditambahkan di atas model dasar untuk menyesuaikannya dengan tugas klasifikasi yang spesifik. Lapisan-lapisan ini termasuk lapisan *Flatten*, beberapa lapisan *Dense* dengan fungsi aktivasi *ReLU*, lapisan *BatchNormalization* untuk menstabilkan dan mempercepat pelatihan, serta lapisan *Dropout* untuk mencegah *Overfitting*.

Lapisan *Output* menggunakan fungsi aktivasi *softmax* untuk menghasilkan probabilitas klasifikasi untuk setiap kelas.

Model kemudian dikompilasi menggunakan optimizer Adam dengan *Learning Rate* yang disesuaikan, serta *Loss* function categorical *Cross*entropy yang sesuai untuk tugas klasifikasi multi-kelas. Proses pelatihan model dilakukan dengan beberapa parameter penting seperti jumlah *Epoch*, *Batch size*, dan *callbacks* untuk menyimpan model terbaik berdasarkan performa validasi.

Pelatihan dilakukan dengan variasi *Epoch* yang beragam menyesuaikan model yang diberikan. Data *Training* tidak dijalankan sekaligus melainkan dibagi per *batch* menggunakan *Batch size* yang telah ditentukan. Selama proses pelatihan, nilai akurasi dan *Loss* dihitung pada setiap *Epoch* untuk memantau performa model. Nilai akurasi menunjukkan seberapa baik model mempelajari pola-pola dalam data pelatihan, sedangkan nilai *Loss* menunjukkan seberapa besar kesalahan model dalam mengklasifikasikan data.

Dengan memantau perubahan nilai akurasi dan *Loss* pada setiap model dan nilai *Hyperparameter* yang diberikan, peneliti dapat mengevaluasi seberapa baik model dalam mempelajari data dan memperbaiki performanya secara bertahap. Setelah proses pelatihan selesai, model terbaik disimpan berdasarkan performa validasinya, yang memungkinkan model untuk digunakan pada data pengujian atau data baru yang belum pernah dilihat sebelumnya. Proses ini memastikan bahwa model dapat menggeneralisasi dengan baik dan memberikan hasil klasifikasi yang akurat pada citra uang kertas.

3.2.8 *Testing* Model

Tahap Untuk memodifikasi program yang sudah dikembangkan, proses review model dilakukan. Selama fase proses ini, data pelatihan dibandingkan dengan data kereta dengan tujuan memastikan bahwa model dapat belajar dari data tersebut, mengakibatkan akurasi dan hilangnya data pelatih. Untuk menentukan apakah model ini mampu menganalisis data citra ketika ada data Citra baru yang telah dimasukkan, perlu untuk melakukan validasi data. Berkaitan dengan tahap yang dikenal sebagai "*Testing*" atau "pengujian", tahap ini dilakukan dua kali seminggu, dengan tahap pertama terdiri dari pengujian, juga dikenal

sebagai tes per gambar dan tes per folder. Dalam pengujian masing-masing gambar, ini digunakan untuk menentukan kelas yang diprediksi dari citra yang disebutkan di atas, dan kemudian dibandingkan dengan tes setiap folder untuk melihat apakah kelas citra nilai nominal pada kertas menunjukkan hasil yang sama atau berbeda.

3.2.9 Analisis

Analisis data didapatkan dari tahapan pengumpulan data yang kemudian diproses dalam sistem yang diikuti dengan pengujian untuk menentukan nominal pada mata uang kertas rupiah. Sehingga sistem memungkinkan untuk mengkategorikan apakah mata uang kertas tersebut sudah mencapai tahap siap menentukan nominal uang.

3.2.10 Kesimpulan Dan Saran

Pada tahap ini yaitu menentukan kesimpulan dari penelitian yang telah dilakukan dengan menggunakan metode CNN untuk menentukan nilai mata uang kertas Rupiah. Serta evaluasi dilakukan untuk menentukan sejauh mana implementasi metode CNN optimal dalam menilai untuk menetukan nominal mata uang kertas Rupiah.

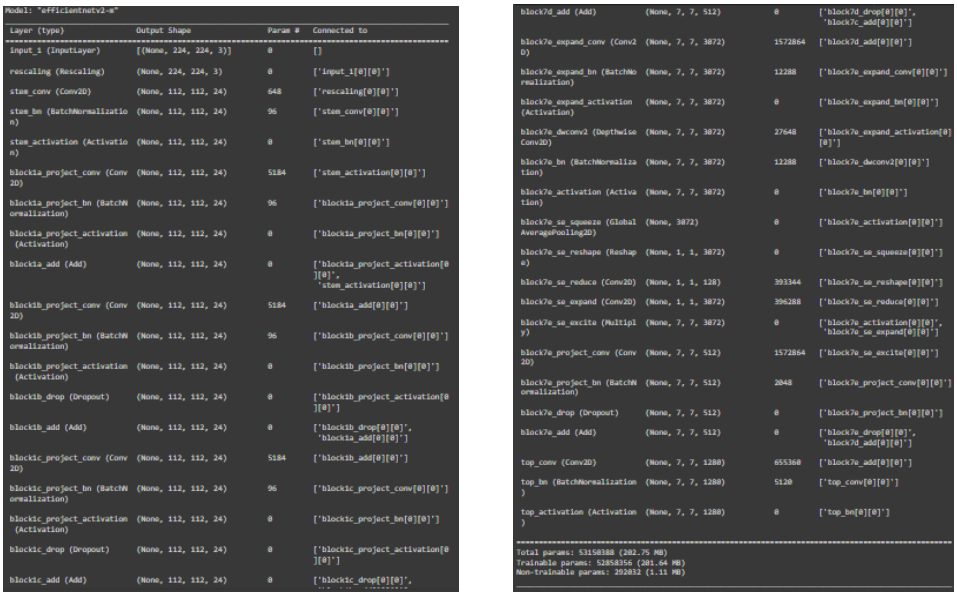
3.3 DESAIN ARSITEKTUR

Penelitian ini menggunakan arsitektur *Convolutional Neural Network* (CNN) dengan model *transfer Learning* dan terdapat beberapa jenis layer yang umum digunakan untuk ekstraksi fitur dan klasifikasi. Layer-layer tersebut termasuk *Convolution* untuk mengekstraksi fitur dari input, *BatchNormalization* untuk meningkatkan stabilitas dan kecepatan konvergensi model, *Dense* untuk melakukan klasifikasi atau regresi, *Pooling* untuk mengurangi dimensi spasial dari fitur yang ditemukan oleh layer konvolusi, dan Flatten untuk mengubah struktur data menjadi satu dimensi sebelum masuk ke layer *Dense*. Fungsi aktivasi *ReLU* digunakan di setiap layer kecuali pada layer *Dense Output* yang menggunakan fungsi aktivasi *softmax* untuk klasifikasi multinomial.

Model menerima input gambar berukuran 224x224 piksel dengan 3 saluran warna (RGB). Pada layer konvolusi, digunakan filter dengan ukuran 64, 128, 256, dan 512 dengan *kernel* berukuran 3x3 untuk mendeteksi pola seperti tepi dan tekstur. Kombinasi dari filter, *BatchNormalization*, *MaxPooling*, dan *Dropout* diterapkan dalam setiap lapisan konvolusi untuk membantu model mengekstraksi dan menggeneralisasi fitur yang lebih kompleks secara bertahap, sehingga dapat melakukan klasifikasi dengan lebih akurat. Setelah proses konvolusi, model melanjutkan ke layer *pooling*, flatten, dan akhirnya ke layer *Dense* untuk klasifikasi.

Untuk mengevaluasi kinerja sistem dalam mengklasifikasikan nominal mata uang kertas rupiah dengan model *transfer Learning*, nilai *Accuracy* dan *Loss* diukur setelah proses *Learning* menggunakan tiga model arsitektur yang berbeda: *EfficientNetV2M*, *VGG16*, dan *EfficientNetB0*. Setelah data preprocessing, yang mencakup pengumpulan, *resizing*, dan normalisasi gambar mata uang kertas, setiap model dilatih dengan dataset yang sama. Selama tahap *Training*, *Accuracy* dan *Loss* dihitung pada setiap *Epoch* untuk memantau performa model. *Accuracy* menunjukkan persentase gambar mata uang yang diklasifikasikan dengan benar, sedangkan *Loss* mengukur seberapa baik model memprediksi nilai nominal mata uang berdasarkan *Output* yang diharapkan. Setelah proses *Learning* selesai, nilai *Accuracy* dan *Loss* pada dataset validasi digunakan untuk mengevaluasi kinerja masing-masing model. Hasil evaluasi menunjukkan model mana yang paling efektif dan efisien dalam mengklasifikasikan nominal mata uang kertas, dengan model yang memiliki *Accuracy* tinggi dan *Loss* rendah dianggap sebagai yang terbaik. Perbandingan kinerja ini memungkinkan pemilihan model yang optimal untuk digunakan dalam aplikasi nyata, memastikan bahwa sistem dapat mengenali nominal mata uang kertas rupiah dengan tingkat akurasi yang tinggi dan kesalahan prediksi yang minimal.

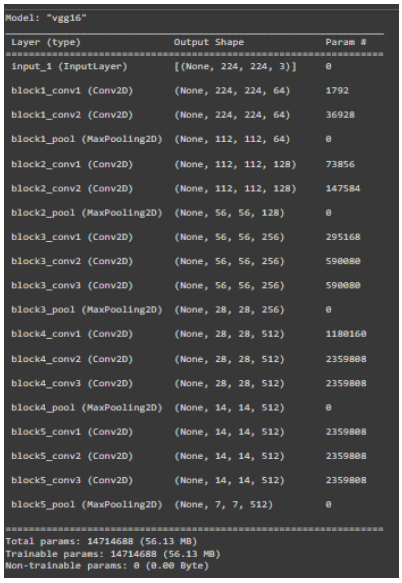
Maka berikut adalah suatu gambar yang menampilkan *output* dari setiap arsitektur dengan arsitektur yang digunakan pada penelitian yaitu arsitektur CNN dengan model *EfffficientNeetV2M*, arsitektur CNN dengan model *VGG16*, arsitektur CNN dengan model *EfficientNetB0*.



```
Model: "efficientnetv2-m"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 224, 224, 3)]     0

 rescaling (Rescaling)       (None, 224, 224, 3)       0

 stem_conv (Conv2D)          (None, 112, 112, 24)      648

 stem_bn (BatchNormalizatio  (None, 112, 112, 24)      96
 n)

 stem_activation (Activatio  (None, 112, 112, 24)      0
 n)

 block1a_project_conv (Conv  (None, 112, 112, 24)      5184
 2D)

 block1a_project_bn (BatchN  (None, 112, 112, 24)      96
 ormalization)

 block1a_project_activation  (None, 112, 112, 24)      0
  (Activation)

 block1a_add (Add)           (None, 112, 112, 24)      0

 block1b_project_conv (Conv  (None, 112, 112, 24)      5184
 2D)

 block1b_project_bn (BatchN  (None, 112, 112, 24)      96
 ormalization)

 block1b_project_activation  (None, 112, 112, 24)      0
  (Activation)

 block1b_drop (Dropout)      (None, 112, 112, 24)      0

 block1b_add (Add)           (None, 112, 112, 24)      0

 block1c_project_conv (Conv  (None, 112, 112, 24)      5184
 2D)

 block1c_project_bn (BatchN  (None, 112, 112, 24)      96
 ormalization)

 block1c_project_activation  (None, 112, 112, 24)      0
  (Activation)

 block1c_drop (Dropout)      (None, 112, 112, 24)      0

 block1c_add (Add)           (None, 112, 112, 24)      0
```

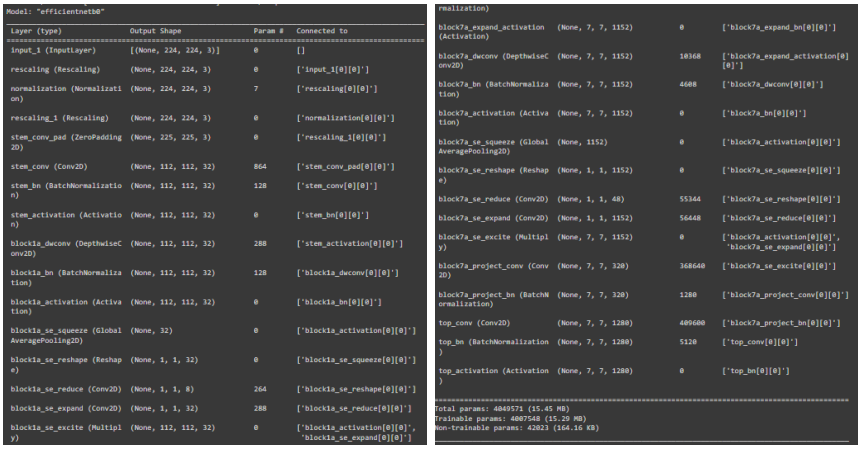
```
 block7d_add (Add)           (None, 7, 7, 512)         0

 block7e_expand_conv (Conv2  (None, 7, 7, 3072)        1572864
 D)

 block7e_expand_bn (BatchNo  (None, 7, 7, 3072)        12288
 rmalization)

 block7e_expand_activation   (None, 7, 7, 3072)        0
 (Activation)

 block7e_dwconv2 (Depthwise  (None, 7, 7, 3072)        27648
 Conv2D)

 block7e_bn (BatchNormaliza  (None, 7, 7, 3072)        12288
 tion)

 block7e_activation (Activa  (None, 7, 7, 3072)        0
 tion)

 block7e_se_squeeze (Global  (None, 3072)              0
 AveragePooling2D)

 block7e_se_reshape (Reshap  (None, 1, 1, 3072)        0
 e)

 block7e_se_reduce (Conv2D)  (None, 1, 1, 128)         393344

 block7e_se_expand (Conv2D)  (None, 1, 1, 3072)        396288

 block7e_se_excite (Multipl  (None, 7, 7, 3072)        0
 y)

 block7e_project_conv (Conv  (None, 7, 7, 512)         1572864
 2D)

 block7e_project_bn (BatchN  (None, 7, 7, 512)         2048
 ormalization)

 block7e_drop (Dropout)      (None, 7, 7, 512)         0

 block7e_add (Add)           (None, 7, 7, 512)         0

 top_conv (Conv2D)           (None, 7, 7, 1280)        655360

 top_bn (BatchNormalization  (None, 7, 7, 1280)        5120
 )

 top_activation (Activation  (None, 7, 7, 1280)        0
 )

=================================================================
Total params: 53150388 (202.75 MB)
Trainable params: 52858356 (201.64 MB)
Non-trainable params: 292032 (1.11 MB)
_________________________________________________________________
```

Gambar 3. 8 *Output* Arsitektur CNN model *EfficientNetV2M*



```
Model: "vgg16"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 224, 224, 3)]     0

 block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792

 block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928

 block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0

 block2_conv1 (Conv2D)       (None, 112, 112, 128)     73856

 block2_conv2 (Conv2D)       (None, 112, 112, 128)     147584

 block2_pool (MaxPooling2D)  (None, 56, 56, 128)       0

 block3_conv1 (Conv2D)       (None, 56, 56, 256)       295168

 block3_conv2 (Conv2D)       (None, 56, 56, 256)       590080

 block3_conv3 (Conv2D)       (None, 56, 56, 256)       590080

 block3_pool (MaxPooling2D)  (None, 28, 28, 256)       0

 block4_conv1 (Conv2D)       (None, 28, 28, 512)       1180160

 block4_conv2 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_conv3 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_pool (MaxPooling2D)  (None, 14, 14, 512)       0

 block5_conv1 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv2 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv3 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_pool (MaxPooling2D)  (None, 7, 7, 512)         0

=================================================================
Total params: 14714688 (56.13 MB)
Trainable params: 14714688 (56.13 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Gambar 3. 9 *Output* Arsitektur CNN model *VGG16*



```
Model: "efficientnetb0"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   Connected to
=================================================================
 input_1 (InputLayer)        [(None, 224, 224, 3)]     0         []

 rescaling (Rescaling)       (None, 224, 224, 3)       0         ['input_1[0][0]']

 normalization (Normalizati  (None, 224, 224, 3)       7         ['rescaling[0][0]']
 on)

 rescaling_1 (Rescaling)     (None, 224, 224, 3)       0         ['normalization[0][0]']

 stem_conv_pad (ZeroPadding  (None, 225, 225, 3)       0         ['rescaling_1[0][0]']
 2D)

 stem_conv (Conv2D)          (None, 112, 112, 32)      864       ['stem_conv_pad[0][0]']

 stem_bn (BatchNormalizatio  (None, 112, 112, 32)      128       ['stem_conv[0][0]']
 n)

 stem_activation (Activatio  (None, 112, 112, 32)      0         ['stem_bn[0][0]']
 n)

 block1a_dwconv (DepthwiseC  (None, 112, 112, 32)      288       ['stem_activation[0][0]']
 onv2D)

 block1a_bn (BatchNormaliza  (None, 112, 112, 32)      128       ['block1a_dwconv[0][0]']
 tion)

 block1a_activation (Activa  (None, 112, 112, 32)      0         ['block1a_bn[0][0]']
 tion)

 block1a_se_squeeze (Global  (None, 32)                0         ['block1a_activation[0][0]']
 AveragePooling2D)

 block1a_se_reshape (Reshap  (None, 1, 1, 32)          0         ['block1a_se_squeeze[0][0]']
 e)

 block1a_se_reduce (Conv2D)  (None, 1, 1, 8)           264       ['block1a_se_reshape[0][0]']

 block1a_se_expand (Conv2D)  (None, 1, 1, 32)          288       ['block1a_se_reduce[0][0]']

 block1a_se_excite (Multipl  (None, 112, 112, 32)      0         ['block1a_activation[0][0]',
 y)                                                              'block1a_se_expand[0][0]']
```

```
rmalization)

 block7a_expand_activation   (None, 7, 7, 1152)        0         ['block7a_expand_bn[0][0]']
 (Activation)

 block7a_dwconv (DepthwiseC  (None, 7, 7, 1152)        10368     ['block7a_expand_activation[0]
 onv2D)                                                          [0]']

 block7a_bn (BatchNormaliza  (None, 7, 7, 1152)        4608      ['block7a_dwconv[0][0]']
 tion)

 block7a_activation (Activa  (None, 7, 7, 1152)        0         ['block7a_bn[0][0]']
 tion)

 block7a_se_squeeze (Global  (None, 1152)              0         ['block7a_activation[0][0]']
 AveragePooling2D)

 block7a_se_reshape (Reshap  (None, 1, 1, 1152)        0         ['block7a_se_squeeze[0][0]']
 e)

 block7a_se_reduce (Conv2D)  (None, 1, 1, 48)          55344     ['block7a_se_reshape[0][0]']

 block7a_se_expand (Conv2D)  (None, 1, 1, 1152)        56448     ['block7a_se_reduce[0][0]']

 block7a_se_excite (Multipl  (None, 7, 7, 1152)        0         ['block7a_activation[0][0]',
 y)                                                              'block7a_se_expand[0][0]']

 block7a_project_conv (Conv  (None, 7, 7, 320)         368640    ['block7a_se_excite[0][0]']
 2D)

 block7a_project_bn (BatchN  (None, 7, 7, 320)         1280      ['block7a_project_conv[0][0]']
 ormalization)

 top_conv (Conv2D)           (None, 7, 7, 1280)        409600    ['block7a_project_bn[0][0]']

 top_bn (BatchNormalization  (None, 7, 7, 1280)        5120      ['top_conv[0][0]']
 )

 top_activation (Activation  (None, 7, 7, 1280)        0         ['top_bn[0][0]']
 )

=================================================================
Total params: 4049571 (15.45 MB)
Trainable params: 4007548 (15.29 MB)
Non-trainable params: 42023 (164.16 KB)
_________________________________________________________________
```

Gambar 3. 10 *Output* Arsitektur CNN model *EfficientNetB0*

Pada Gambar 3.8 sampai 3.10 merupakan hasil dari *Output* 3 model yang didapatkan. Jika dilihat gambar diatas merupakan *summary* dari setiap layer dengan hasil *Output shapenya* dan parameter yang dihasilkan pada setiap model. Setelah model selesai dibuat, maka langkah selanjutnya adalah melakukan percobaan *Training*, *Validation* dan *Testing* pada model. Dalam tahap percobaan ini terdapat beberapa parameter yang dapat diubah nilainya atau dalam kata lain dapat ditentukan oleh peneliti yang biasa disebut dengan *Hyperparameter*. Perubahan nilai pada *Hyperparameter* dilakukan untuk memberikan variasi pada setiap model, dengan tujuan untuk melihat pengaruh perubahannya dalam menghasilkan *Accuracy* dan *Loss* yang terbaik pada setiap model. Berikut adalah *Hyperparameter* yang digunakan dalam penelitian ini beserta dengan nilainya.

3.4 METODE PENGUJIAN

Dengan memanfaatkan metode CNN Jika Sistem model yang dibangun telah berhasil mengenali nominal uang kertas rupiah dengan sample sebanyak 7 kelas dan data sebanyak 1400 citra kemudian hasil akurasi data pada data *Train* dan data *Validation* telah optimal. Untuk menentukan nominal mata uang kertas rupiah maka dilakukannya proses klasifikasi nominal uang kertas menggunakan algoritma CNN dengan memasukan data citra yang telah ditentukan.

Sistem pengenalan nominal mata uang kertas Rupiah menggunakan metode klasifikasi dengan fokus pada akurasi dan *Loss* yang telah dirancang dan diuji. Proses dimulai dengan pemilihan CNN sebagai model utama untuk penelitian ini. Dataset yang terdiri dari gambar-gambar nominal mata uang Rupiah dibagi menjadi dua bagian, dengan 60% digunakan untuk pelatihan dan 40% untuk pengujian.. Model yang dibangun kemudian dilatih menggunakan data pelatihan dengan memonitor akurasi dan *Loss* pada set pelatihan dan validasi selama beberapa *Epoch* yang akan dilakukan. Setelah pelatihan selesai, kinerja model diuji menggunakan dataset pengujian, dan metrik evaluasi seperti *Accuracy* dan *Loss* dianalisis. Jika kinerja belum memuaskan, dilakukan *fine-tuning* pada model dengan penyesuaian parameter atau struktur untuk memastikan hasil yang lebih konsisten. Hasil pengujian, termasuk *Accuracy* dan *Loss*, didokumentasikan dengan cermat untuk membantu pemahaman dan pengembangan lebih lanjut pada sistem pengenalan nilai nominal mata uang kertas Rupiah ini.