Rancang Bangun Klasifikasi Citra Dengan Teknologi Deep Learning Berbasis Metode Convolutional Neural Network

Ari Peryanto[1], Anton Yudhana[2] dan Rusydi Umar[3]
Program Studi Teknik Elektro, Universitas Ahmad Dahlan[,2]
Program Studi Magister Teknik Informatika, Universitas Ahmad Dahlan [1,3]
Jl. Prof. Dr. Soepomo, Janturan, Umbulharjo, Yogyakarta
E-mail : ari1907048002@webmail.uad.ac.id[1], eyudhana@ee.uad.ac.id[2], rusydi.umar@tif.uad.ac.id[3]

Abstract – With the rapid development of technology today, resulting in Deep Learning to become one of the most popular machine learning methods. GPU Acceleration technology is one reason for the rapid development of Deep Learning. Deep learning is very suitable to be used to solve classical problems in Computer Vision, namely in the classification of images. One method in deep learning that is often used in image processing is the Convolutional Neural Network and is a development of the Multi Layer Perceptron. In this research the implementation of this method is done using hard libraries with Python programming language. In the training process using the Convolutional Neural Network, setting the number of epochs and increasing the size of the training data to improve accuracy in image classification. The sizes used are 32x32, 64x64 and 128x128. The training process with the number of epoch 40 and size
32x32 obtained the highest accuracy value which reached 98.02% and the highest average accuracy was 97.56%, and the system accuracy was 96.64%.

***Keywords*: Deep Learning, Convolution Neural Network, Image, Clasification**

***Abstrak* – Dengan berkembang pesatnya teknologi saat ini, mengakibatkan *Deep Learning* menjadi salah satu metode *machine learning* yang sangat diminati. Teknologi GPU Acceleration menjadi salah satu sebab dari pesatnya perkembangan *Deep Learning*. *Deep learning* sangat cocok digunakan untuk memecahkan permasalahan klasik dalam *Computer Vision*, yaitu dalam pengklasifikasian citra. Salah satu metode dalam *deep learning* yang sering digunakan dalam pengolah citra adalah *Convolutional Neural Network* dan merupakan pengembangan dari *MultiLayer Perceptron*. Pada penelitian ini pengimplementasian metode ini dilakukan menggunakan library keras dengan bahasa pemrograman phyton. Pada proses *training* menggunakan *Convolutional Neural Network*, dilakukan setting jumlah epoch dan memperbesar ukuran data training untuk meningkatkan akurasi dalam pengklasifikasian citra. Ukuran yang digunakan adalah 32x32, 64x64 dan 128x128. Proses *training* dengan jumlah epoch 40 dan ukuran 32x32 didapat nilai akurasi tertinggi yang mencapai 98,02% dan rata-rata akurasi tertinggi yaitu 97,56 %, serta akurasi sistem sebesar 96,64%.**

***Kata Kunci*: Deep Learning, Convolution Neural Network, Citra, Klasifikasi**

I. PENDAHULUAN

Klasifikasi objek pada citra secara umum adalah masalah utama dalam *Computer Vision* yang sejak dahulu dicari solusinya. Pada saat ini tidak dapat dipungkiri bahwa perkembangan teknologi informasi sangat cepat. Selain perkembangan hardware dalam meningkatkan performa komputer banyak pula berkembang software yang mampu meniru kecerdasan manusia (kecerdasan buatan). Kini komputer dituntut untuk bisa membuat manusia dalam menyelesaikan pekerjaan dengan lebih cepat dan dalam waktu yang singkat. Dengan berkembangnya dunia komputasi dan dengan semakin meningkatnya kapasitas dan kecerdasan proses komputer saat ini muncul ilmu-ilmu komputasi yang memungkinkan komputer dapat mengambil informasi dari suatu citra untuk keperluan pengenalan objek secara otomatis.

Metode yang paling banyak digunakan dalam pengolah citra adalah metode *Convolutional Neural Network* (CNN), dari berbagai macam metode yang ada dalam pengolahan citra. CNN merupakan pengembangan dari Multi Layer Perceptron (MLP) dan merupakan salah satu algoritma dari *Deep Learning*. Metode CNN memiliki hasil paling signifikan dalam pengenalan citra, hal tersebut dikarenakan CNN berusaha meniru sistem pengenalan citra pada visual cortex manusia, sehingga memiliki kemampuan mengolah informasi citra [1].

Penelitian sebelumnya yang dilakukan oleh [1] pada basis data caltech 101 mendapatkan hasil bahwa klasifikasi citra objek dengan tingkat *confusion* yang berbeda menghasilkan nilai akurasi sebesar 20% - 50 %, sehingga disimpulkan bahwa metode CNN relatif handal. Penelitian yang dilakukan oleh [2] berhasil mengimplementasikan metode CNN dengan tingkat kecocokan data sebesar 98,57%. Selain itu disimpulkan pula bahwa dengan ukuran gambar yang semakin besar dan penambahan layer pada saat proses training membuat proses training menjadi lama, walaupun tingkat akurasi menjadi bertambah. Penelitian yang dilakukan oleh [3] Untuk mendapatkan hasil yangmaksimal dalam prediksi, jumlah data training juga harus banyak, dan itu berimplikasi dengan peralatan komputasi yang dibutuhkan, untuk mempercepat proses pembelajaran. Penelitian yang dilakukan oleh [4] mendapatkan hasil bahwa keakuratan sistem untuk mengidentifikasi dan mengenali citra jamur sangat dipengaruhi oleh parameter jaringan internal, yaitu laju pembelajaran, ju mlah neuron layar tersembunyi, dan iterasi. Persentase akurasi tertinggi mencapai 93% itu berarti hasilnya bagus, dan untuk penelitian lebih lanjut dapat dicoba tanpa mengubah ke skala abu-abu terlebih dahulu.

II. LANDASAN TEORI DAN METODE

A. *Deep Learning*

Deep Learning merupakan salah satu cabang dari *machine learning* [5] [6] [7] [8]. Model *deep learning* dapat mempelajari komputasinya sendiri dengan menggunakan otaknya sendiri. *Deep learning* dirancang untuk terus menganalisa data seperti pada otak manusia dalam mengambil keputusan. Agar kemampuan *deep learning* semakin mempuni maka *deep learning* menggunakan algoritma *artifical neural network* (ANN), yang terinspirasi dari jaringan biologis otak manusia.

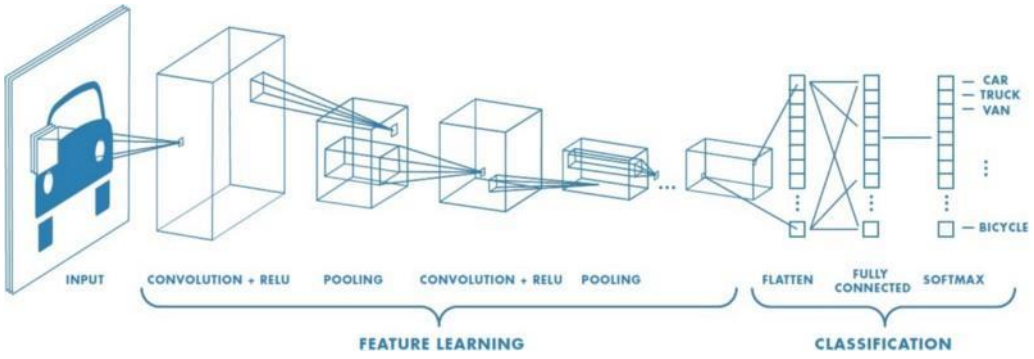
B. *Convolutional Neural Network*

Salah satu jenis neural network yang biasa digunakan pada data image adalah CNN. Karena dalamnya tingkat jaringan maka CNN termasuk dalam jenis *Deep Neural Network* dan sering digunakan dalam data citra. Ada dua metode yang dimiliki oleh CNN, yaitu klasifikasi menggunakan *feedforward* dan tahap pembelajaran menggunakan *backpropagation*.

Sekitar tahun 1988 Yann LeCun mulai mengenalkan CNN. *Deep learning* mengalami peningkatan dan menjadi sukses semenjak CNN diperkenalkan dan menjadi salah satu metode pada *Deep Learning*. Jauh sebelumnya tahun 1950-an Hubel dan Wisel melakukan penelitian visual cortex yaitu merupakan bagian pada otak kucing. Mereka menemukan bahwa ada bagian kecil berupa sel-sel yang sensitif terhadap area tertentu pada pandangan mata yang dimiliki visual cortex. 2 tipe visual cortex yang ditemukan oleh Hubel dan Wisel, yaitu simple cell dan complex cell. Dari hasil pengamatannya, Kunihiko [9] merancang *Neocognitron* yang merupakan model *Hierarchical Multilayered Neural Network* pada tahun 1980-an. Model ini kemudian digunakan untuk beberapa kasus seperti klasifikasi karakter dari tulisan tangan (*Handwritten Character Recognition*).

Ada kesamaan struktur yang dimiliki CNN dengan *artificial neural network*. Pada klasifikasi citra, CNN mendapat masukan atau citra masukan untuk diproses dan diklasifikasi ke kategori tertentu. Perbedaan CNN dan ANN adalah pada arsitektur tambahan pada CNN yang dioptimisasi untuk fitur yang ada pada citra masukan. Ada beberapa komponen utama yang ada dalam CNN yaitu diantaranya adalah:

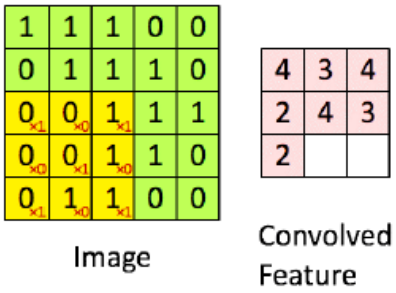
1. *Convolution Layer*
2. *Pooling Layer*
3. *Fully Connected Layer*
4. *Dropout*



Gambar 1. Proses *Convolutional Neural Network*
(Sumber: https://www.mathworks.com/discovery/convolutional-neural-network.html)

1. *Convolutional Layer*

Convolution layer adalah lapisan utama yang paling penting di metode CNN [10]. Hasil dari *convolution layer* adalah citra baru yang menunjukkan fitur dari citra masukan. Setiap citra yang menjadi masukan, menggunakan filter *convolution layer* dalam proses tersebut. *Convolutional layer* terdiri dari *neuron* yang tersusun sedemikian rupa sehingga membentuk sebuah filter dengan panjang dan tinggi (pixel). Sebagai contoh, layer pertama pada *feature extraction layer* adalah *convolutional layer* dengan ukuran 6x6x3. Panjang 6 pixel, tinggi 6 pixel, dan tebal/jumlah 3 buah sesuai dengan channel dari gambar tersebut. Ketiga filter ini akan digeser keseluruhan bagian dari gambar. Setiap pergeseran akan dilakukan operasi "*dot*" antara input dan nilai dari filter tersebut sehinga menghasilkan sebuah output atau biasa disebut sebagai *activation map* atau *feature map*.



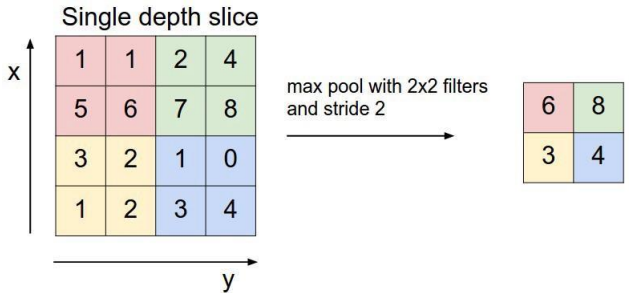
Gambar 2. Operasi Konvulasi

Kotak hijau secara keseluruhan adalah citra yang akan dilakukan konvolusi. Kernel bergerak dari sudut kiri atas ke kanan bawah. Sehingga hasil konvolusi dari citra tersebut dapat dilihat dari gambar di sebelah kanan. Tujuannya dilakukan konvolusi pada data citra yaitu untuk mengekstraksi fitur dari citra input. Konvolusi akan menghasilkan transformasi linear dari data input sesuai informasi spasial pada data. Bobot pada lapisan tersebut mengspesifikasi kernel konvolusi yang digunakan, sehingga kernel konvolusi dapat dilatih berdasarkan input pada CNN.

2. *Pooling Layer*

Pooling layer adalah lapisan fungsi untuk *Feature Maps* sebagai masukan dan mengolahnya dengan berbagai operasi statistik nilai piksel terdekat. Pada model CNN, lapisan pooling biasanya disisipkan secara teratur setelah beberapa lapisan konvolusi. Lapisan pooling yang dimasukkan di antara lapisan konvolusi secara berturut-turut dalam susunan arsitektur model CNN dapat secara progresif mengurangi ukuran volume output pada *Feature Maps*, sehinga dapat mengurangi jumlah parameter dan perhitungan di jaringan, dan untuk mengendalikan *overfitting*. Hal penting dalam pembuatan model CNN adalah dengan memilih banyak jenis lapisan *pooling* yang dalam hal ini dapat menguntungkan kinerja model [11]. Lapisan *pooling* bekerja di setiap susunan *Feature Maps* dan mengurangi ukurannya. Bentuk lapisan *pooling* yang paling umum adalah dengan menggunakan filter berukuran 2x2 yang diaplikasikan dengan langkah sebanyak dua dan kemudian beroperasi pada setiap irisan dari input. Bentuk seperti ini akan mengurangi *Feature Maps* hingga 75% dari ukuran aslinya.

Pada *prinsipnya pooling layer* terdiri dari sebuah filter dengan ukuran dan *stride* tertentu yang akan bergeser pada seluruh area *feature map*. *Pooling* yang biasa digunakan adalah *Max Pooling* dan *Average Pooling*. Sebagai contoh jika kita menggunakan *Max Pooling* 2x2 dengan *stride* dua, maka pada setiap pergeseran filter, nilai maximum pada area 2x2 pixel tersebut yang akan dipilih, sedangkan *Average Pooling* akan memilih nilai rata-ratanya.



Gambar 3. *Max Pooling*

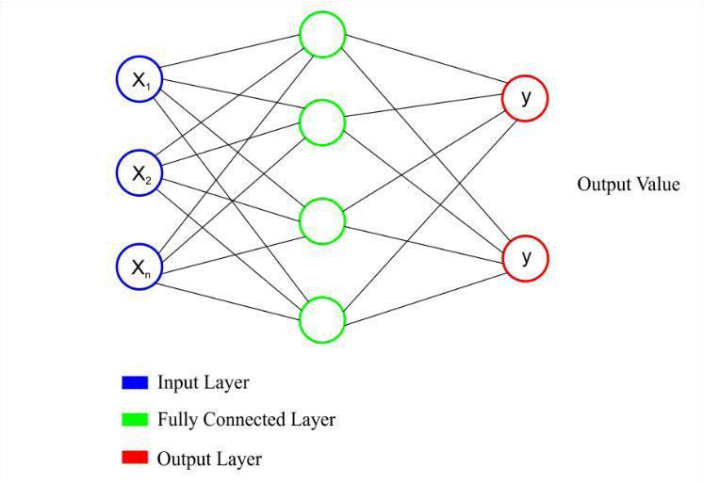
Tujuan dari penggunaan *pooling layer* adalah mengurangi dimensi dari *feature map* (*downsampling*), sehingga mempercepat komputasi karena parameter yang harus diupdate semakin sedikit dan mengatasi *overfitting*.

3. *Fully Connected Layer*

Feature map yang dihasilkan dari *feature extraction* masih berbentuk *multidimensional array*, sehingga harus melakukan "*flatten*" atau *reshape feature map* mejadi sebuah *vector* agar bisa digunakan sebagai input dari *fully-connected layer*.

Lapisan Fully-connected adalah lapisan dimana semua *neuron* aktivitas dari lapisan sebelumnya terhubung semua dengan *neuron* di lapisan selanjutnya seperti hal nya jaringan syaraf tiruan bisa. Setiap aktivitas dari lapisan sebelumnya perlu diubah menjadi data satu dimensi sebelum dapat dihubungkan ke semua *neuron* di lapisan *Fully-Connected*.

Lapisan Fully-Connected biasanya digunakan pada metode Multi lapisan *Perceptron* dan bertujuan untuk mengolah data sehingga bisa diklasifikasikan. Perbedaan antar lapisan *Fully-Connected* dan lapisan konvolusi biasa dalah *neuron* di lapisan konvolusi terhubung hanya ke daerah tertentu pada input. Sementara lapisan *Fully-Connected* memiliki *neuron* yang secara keseluruhan terhubung. Namun, kedua lapisan tersebut masih mengoprasikan produk *dot*, sehinga fungsinya tidak begitu berbeda.

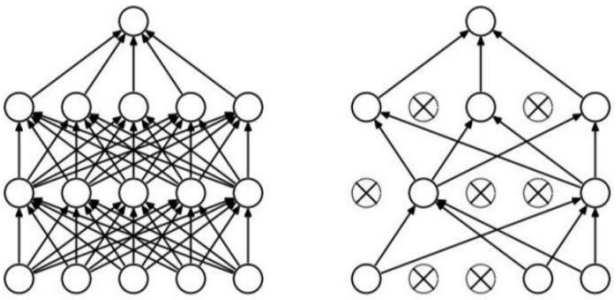


Gambar 4. *Fully Connected Layer*

4. *Dropout*

Dropout adalah teknik *regularisasi* jaringan syaraf dimana beberapa *neuron* akan dipilih secara acak dan tidak dipakai selama pelatihan. *Neuron-neuron* ini dapat dibilang dibuang secara acak. Hal ini berarti bahwa kontribusi *neuron* yang dibuang akan diberhentikan sementara jaringan dan bobot baru juga tidak diterapkan pada *neuron* pada saat melakukan *backpropagation*.

Dropout merupakan proses mencegah terjadinya *overfitting* dan juga mempercepat proses *learning*. *Dropout* mengacu kepada menghilangkan *neuron* yang berupa *hidden* mapun *layer* yang *visible* didalam jaringan. Dengan menghilangkan suatu *neuron*, berarti menghilangkannya sementara dari jaringan yang ada. *Neuron* yang akan dihilangkan akan dipilih secara acak. Setiap *neuron* akan diberikan *probabilitas* yang bernilai antara nol dan satu.



(a) Jaringan Syaraf Biasa

(b) Jaringan Syaraf Setelah dikenakan Teknik *Dropout*

Gambar 5. Contoh Implementasi Dropout

Pada gambar 5 jaringan syaraf (a) merupakan jaringan syaraf biasa dengan dua lapisan tersembunyi. Sedangkan pada bagian (b) jaringan syaraf sudah diaplikasikan teknik *regularisasi dropout* dimana ada beberapa *neuron* aktivasi yang tidak dipakai lagi. Teknik ini sangat mudah diimplementasikan pada model CNN dan akan berdampak pada performa model dalam melatih serta mengurangi *overfitting*.

C. *Pengolahan Data Awal (Preprocessing)*

Dataset yang digunakan pada penelitian ini, didapatkan dari pencarian acak di internet. Ada dua kelas yang akan digunakan yaitu kelas data mobil dan kelas data motor. Kelas data motor berjumlah total sebanyak 35 data, yang terdiri dari 30 data atau 85 persen sebagai data training dan 5 atau 15 persen sebagai data testing. Kemudian untuk data kelas mobil juga berjumlah total 35 data dengan 30 data atau 85 persen sebagai data training dan 5 data atau 15 persen sebagai data tes. Untuk lebih jelasnya bisa dilihat pada tabel prosentase dataset di tabel 1.

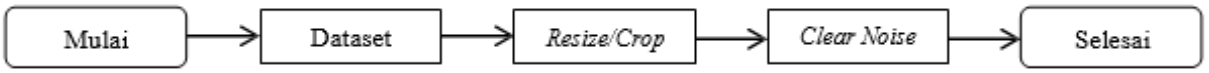
Tabel 1. Prosentase Dataset

Kategori	Jumlah Data	Training	Prosentase %	Testing	Prosentase %
Mobi	35	30	85	5	15
Moto	35	30	85	5	15

Setelah data terkumpul maka proses selanjutnya dilakukan pengolahan data awal terlebih dahulu, agar didapat data awal yang baik sehingga didapatkan hasil yang baik pula. Adapun tahapan dalam pengolahan data awal adalah :

1. *Croping*
   Agar nantinya mendapat proses yang maksimal dalam proses uji dataset terlebih dilakukan *croping* agar dataset bisa lebih fokus dan juga seragam. Proses *croping* dilakukan secara manual dengan bantuan software pengolah citra.
2. Pembersihan *Noise*
   Pembersihan *noise* dilakukan untuk meningkatkan kualitas dari dataset sehingga nantinya saat dataset digunakan dalam proses uji akan menghasilkan tingkat akurasi yang tinggi. *Noise* merupakan pixel yang mengganggu kualitas gambar sehingga menyebabkan penurunan kualitas dari dataset. Proses pembersihan *Noise* ini juga dilakukan secara manual menggunakan bantuan software pengolah citra.

Dari tahapan dalam pengolahan data awal dapat digambarkan alur pengolahan data awal, seperti terlihat pada gambar 6 dibawah ini.



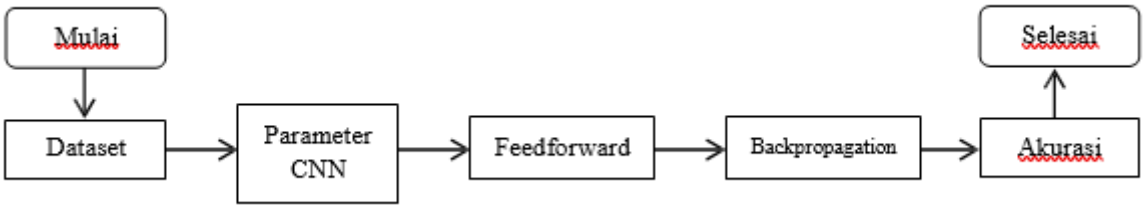
Gambar 6. Alur Pengolahan Dataset

Semakin jelas dataset yang akan digunakan sebagai data *training* juga akan sangat mempengaruhi akurasi. Sehingga proses pengolahan data awal ini merupakan proses yang sangat vital dan sangat penting. Contoh dataset setelah dilakukan proses pengolahan data awal dapat dilihat pada gambar 6.

Gambar 7. Contoh Dataset

D. *Training Dataset*

Proses ini merupakan tahapan dimana CNN melakukan training untuk mendapatkan nilai akurasi yang tinggi. Ini merupakan tahap awal dalam proses CNN dimana proses ini digunakan untuk membentuk suatu model yang nantinya model tersebut digunakan dalam proses pengujian. Pada tahap ini terdapat dua proses yaitu klasifikasi dengan *feedforward* dan pembelajaran dengan *backpropagation*. Proses *feedforward* dimulai setelah dilakukan proses pengolahan dataset. Proses kerja *feedforward* adalah citra vektor akan melalui proses konvolusi dan *max pooling* untuk mereduksi ukuran citranya dan memperbanyak neuronnya. Sehingga terbentuk banyak jaringan yang mana menambah variant data untuk dipelajari. Hasil dari proses *feedforward* berupa bobot yang akan digunakan untuk mengevaluasi proses *neural network* tadi. Alur dari training dataset dapat dilihat pada gambar 8.



Gambar 8. Alur Proses Training

E. Pengujian Dataset

Proses ini sebenarnya tidak ada bedanya dengan proses training, namun pada proses ini tidak terdapat proses pembelajaran dengan *backpropagation* setelah proses *feedforward*. Hasil akhir dari proses ini adalah tingkat akurasi atau kecocokan dari hasil pembelajaran yang telah dilakukan.

III. HASIL DAN PEMBAHASAN

Uji coba dalam penelitian ini menggunakan dataset sample acak yang didapatkan dari google image sejumlah total 70, dimana 35 data adalah kategori mobil dan 35 data yang lain adalah kategori motor.

A. Hasil Klasifikasi

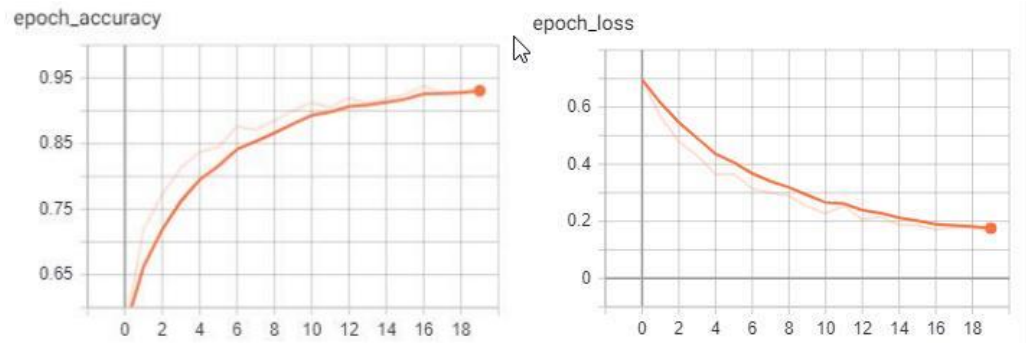
Hasil klasifikasi yang telah dilakukan dalam penelitian ini mencari parameter paling efisien yang bisa digunakan dalam pengklasifikasian.

1. Pengujian Jumlah Parameter Epoch

Percobaan yang pertama dilakukan dengan data gambar berukuran 32 x 32 dengan epochs sebanyak 20, akurasi yang dihasilkan mencapai 93%. Hal ini seperti yang ditunjukkan pada gambar 9 dan 10.

```
Epoch 17/20
31/31 [==============================] - 15s 498ms/step - loss: 0.1699
- accuracy: 0.9372
Epoch 18/20
31/31 [==============================] - 16s 506ms/step - loss: 0.1795
- accuracy: 0.9283
Epoch 19/20
31/31 [==============================] - 16s 525ms/step - loss: 0.1751
- accuracy: 0.9287
Epoch 20/20
31/31 [==============================] - 16s 532ms/step - loss: 0.1656
- accuracy: 0.9352
Execution Time: 4.840581293900808 minutes
```

Gambar 9. Proses Training 32x32 – 20 epoch

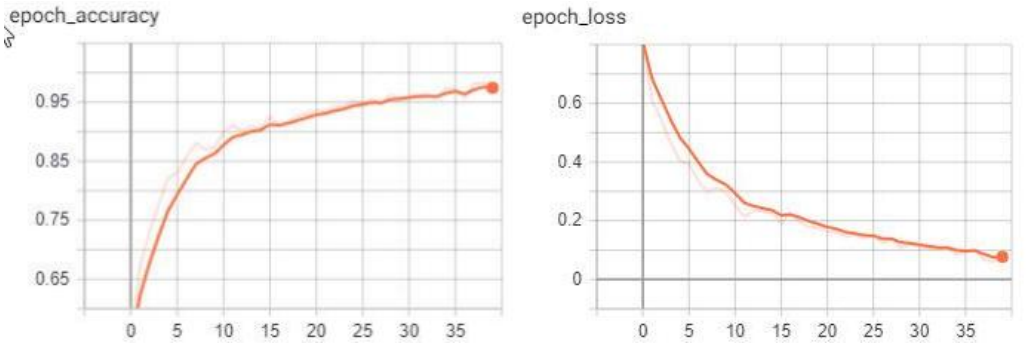


Gambar 10. Grafik Training Akurasi dan Loss 32x32 – 20 epoch

Pada gambar 11 dan 12 adalah hasil dari percobaan yang kedua dilakukan dengan mencoba parameter berikutnya yaitu menaikkan jumlah epoch dari sebelumnya sebanyak 20 menjadi 40, dengan ukuran gambar masih sama yaitu 32 x 32. Terjadi peningkatan jumlah akurasi dari sebelumnya 94% menjadi 97%. Dengan penambahan jumlah epoch ternyata tidak terlalu signifikan meningkatkan jumlah akurasi. Dengan penambahan epoch juga menjadikan proses training menjadi lama.

```
Epoch 37/40
31/31 [==============================] - 23s 745ms/step - loss: 0.0998
- accuracy: 0.9574
Epoch 38/40
31/31 [==============================] - 23s 755ms/step - loss: 0.0666
- accuracy: 0.9815
Epoch 39/40
31/31 [==============================] - 24s 779ms/step - loss: 0.0600
- accuracy: 0.9825
Epoch 40/40
31/31 [==============================] - 24s 763ms/step - loss: 0.0790
- accuracy: 0.9722
Execution Time: 13.783737583955128 minutes
```

Gambar 11. Proses Training 32x32 – 40 epoch



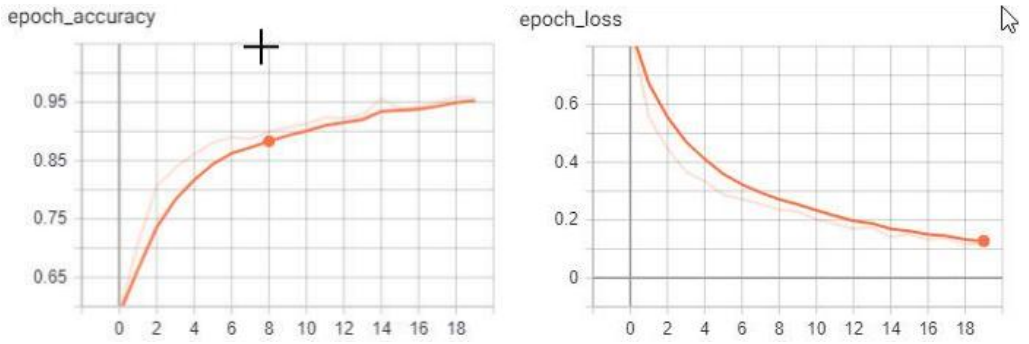
Gambar 12. Grafik Training Akurasi dan Loss 32x32 – 40 epoch

2. Pengujian Ukuran Data Training

   Pengujian pertama dilakukan dengan cara merubah parameter ukuran data training menjadi 64 x 64, dimana data train yang digunakan berjumlah sama dengan sebelumnya. Dari pengujian yang telah dilakukan dengan epoch sejumlah 20, maka didapatkan nilai akurasi sebanyak 95%. Untuk lebih jelas dapat dilihat pada gambar 13 dan 14.

```
Epoch 17/20
31/31 [==============================] - 10s 336ms/step - loss: 0.1323
- accuracy: 0.9414
Epoch 18/20
31/31 [==============================] - 10s 331ms/step - loss: 0.1377
- accuracy: 0.9496
Epoch 19/20
31/31 [==============================] - 10s 338ms/step - loss: 0.1128
- accuracy: 0.9588
Epoch 20/20
31/31 [==============================] - 11s 359ms/step - loss: 0.1189
- accuracy: 0.9578
Execution Time: 3.500264330705007 minutes
```

Gambar 13. Proses Training 64x64 – 20 epoch

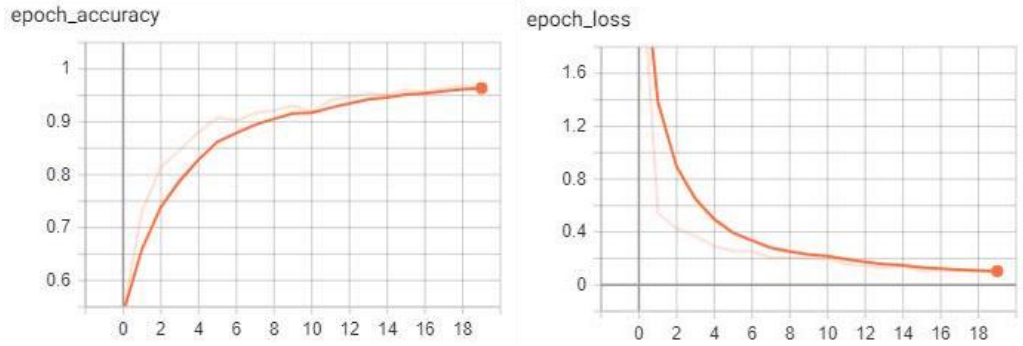


Gambar 14. Grafik Training Akurasi dan Loss 64x64 – 20 epoch

Percobaan yang kedua dilakukan dengan mencoba parameter berikutnya yaitu menaikkan ukuran data train dari 4 x 64 menjadi 128 x 128, dengan jumlah epoch masih sama yaitu sebanyak 20. Tidak terjadi peningkatan jumlah akurasi dari sebelumnya 95,5% menjadi tetap 95,5%. Dengan menaikkan ukuran gambar data train ternyata tidak terlalu menambah akurasi. Dengan peningkatan ukuran gambar train membuat proses train menjadi lebih lama, pada proses train dengan ukuran data 64 x 64 waktu yang dibutuhkan dalam setiap st ep berkisar antara 10-12 detik, sedangkan saat proses train dengan ukuran data 128 x 128 rata-rata dibutuhkan waktu 37-38 detik. Untuk lebih jelas dapat dilihat digambar 15 dan 16.

```
accuracy: 0.9506
Epoch 18/20
31/31 [==============================] - 37s 1s/step - loss: 0.1257 -
accuracy: 0.9496
Epoch 19/20
31/31 [==============================] - 37s 1s/step - loss: 0.0968 -
accuracy: 0.9647
Epoch 20/20
31/31 [==============================] - 38s 1s/step - loss: 0.1007 -
accuracy: 0.9582
Execution Time: 13.224437689781189 minutes
```

Gambar 15. Proses Training 128x128 – 20 epoch



Gambar 16. Grafik Training Akurasi dan Loss 128x128 – 20 epoch

Dari perbandingan diatas dapat ditarik kesimpulan sederhana bahwa penambahan jumlah epoch menjadikan akurasi menjadi naik dan nilai prediksi menjadi lebih bagus, walaupun juga terdapat kekurangan pada saat proses training dengan menggunakan CNN menjadi lebih lama dan berat untuk proses non GPU (CPU). Sedangkan untuk peningkatan ukuran data training untuk proses *training* tidak menyebabkan akurasi meningkat secara signifikan. Peningkatan akurasi hanya terjadi 0,0274 % sehingga tidak terlalu efektif. Pada saat proses train dilakukan menggunakan ukuran gambar yang lebih besar proses train menjadi sangat lama.

B. Validasi Hasil

Pada penelitian ini digunakan algoritma untuk menguji validitas hasil akurasi menggunakan *cross validation*. Algoritma ini digunakan untuk mengevaluasi dan membandingkan dengan cara membagi data menjadi dua segmen. Satu segmen nantinya digunakan untuk *training*, dan yang lain digunakan untuk memvalidasi (*testing*). Dalam *cross*

validation set *training* dan *testing* harus *crossover* berturut-turut sehingga setiap data memiliki kesempatan tervalidasi.

Pengujian ini membagi dataset dalam 5 folder yang didalamnya terdapat 35 data yang dibagi menjadi dua yaitu dataset *training* sebanyak 30 dan 5 dataset sebagai data *testing*. Penentuan dataset sebagai data *training* maupun data *testing* dilakukan secara acak. Setiap folder akan dilakukan proses *training* sebanyak 3 kali dan dihitung tingkat akurasinya secara rata-rata. Proses training yang dilakukan menggunakan parameter hasil tertinggi yaitu ukuran gambar 32x32 dengan epoch sebanyak 40. Hasil pengujian menggunakan *cross validation* dapat dilihat pada tabel 3.

Tabel 2. Akurasi Hasil *Training* dengan Metode K-Fold *Cross Validation*

Fold	Training n-1 %	Training n-2 %	Training n-3 %	Rata-rata %
1	96,3	95,1	96,4	95,97
2	97,3	97,4	97,9	97,56
3	96,4	95,9	98,0	96,81
4	94,5	95,8	97,5	95,99
5	96,7	97,3	96,5	96,89
Rata-rata	96,2	96,3	97,2	96,64

Dari 3 kali *training* yang dilakukan, *Fold* 2 mendapatkan rata-rata akurasi tertinggi yaitu 97,56 %. Kemudian untuk rata-rata nilai akurasi terendah didapat pada *Fold* 1 yaitu 95,97 %. Untuk nilai akurasi tertinggi didapat pada *Fold* 3 pada *training* ke-3 dari 3 kali *training*. Dari metode *K-Fold Cross Validation* didapatkan nilai akurasi sistem sebesar 96,64 yang didapat dari hasil rata-rata seluruh akurasi *Fold* pada setiap *training*. *Fold* 2 sebagai *fold* yang mempunyai nilai akurasi rata-rata tertinggi digunakan sebagai data training pada skenario pengujian berikutnya.

Pada skenario pengujian selanjutnya yaitu *testing* dengan dataset sebanyak 30 data, dan data tersebut sama sekali belum pernah digunakan dalam sistem. Dengan pembagian yaitu 15 data kelas mobil dan 15 data kelas motor. Pengujian dilakukan sebanyak 3 kali percobaan dan menghasilkan prediksi yang sangat tinggi. Hasil dari percobaan tersebut dapat dilihat seperti terlihat pada tabel 3.

Tabel 3. Hasil Klasifikasi Dengan Mengubah Epoch

Percobaan	Prediksi Benar	Prediksi Salah	Prosentase %
1	28	2	93,33
2	25	5	83,33
3	29	1	96,66
Rerat			91,10

Dari tabel 3 dapat kita lihat bahwa percobaan pertama menghasilkan prediksi benar sebanyak 28 data dan salah sebanyak 2 data sehingga mendapatkan nilai akurasi sebesar 93,33%. Pada percobaan kedua terdapat 5 prediksi salah dan hanya mampu memprediksi benar sebanyak 25 data sehingga mendapatkan nilai akurasi sebesar 83,33%. Sedangkan pada percobaan ketiga didapati 29 prediksi benar dan 1 prediksi salah denga n nilai akurasi sebesar 96,66%, sehingga dari ketiga percobaan tersebut didapatkan nilai akurasi sistem sebesar 91,10%.

IV. KESIMPULAN

Penelitian ini berhasil mengimplementasikan metode CNN untuk pengklasifikasian citra menggunakan *library* keras dengan bahasa pemrograman phyton, dan didapatkan tingkat kecocokan / akurasi tertinggi sebesar 98,02% dan rata-rata akurasi tertinggi yaitu 97,56 %, serta akurasi sistem sebesar 96,64%. Sistem yang telah dibuat juga telah dapat memprediksi dengan prosentase yang sangat tinggi yaitu sebesar 91,10%. Selain itu dapat disimpulkan juga bahwa penambahan jumlah epoch menjadikan akurasi menjadi naik dan nilai prediksi menjadi lebih bagus, walaupun juga terdapat kekurangan pada saat proses training menjadi lebih lama dan berat untuk proses non GPU (CPU). Sedangkan untuk peningkatan ukuran data training untuk proses training tidak menyebabkan akurasi meningkat secara signifikan

. Peningkatan akurasi hanya terjadi 0,0274 % sehingga tidak terlalu efektif. Pada saat proses train dilakukan menggunakan ukuran gambar yang lebih besar proses train menjadi sangat lama.

V. REFERENSI

[1] W. S. Eka Putra, "Klasifikasi Citra Menggunakan Convolutional Neural Network (CNN) pada Caltech 101," *J. Tek. ITS*, vol. 5, no. 1, 2016.

[2] A. Santoso and G. Ariyanto, "Implementasi Deep Learning Berbasis Keras Untuk Pengenalan Wajah," Emit. J. Tek. Elektro, vol. 18, no. 01, pp. 15–21, 2018.

[3] A. Nur and G. B. Hertantyo, "Implementasi Convolutional Neural Network untuk Klasifikasi Pembalap MotoGP Berbasis GPU," pp. 50–55, 2018.

[4] A. Fadlil, R. Umar, and S. Gustina, "Mushroom Images Identification Using Orde 1 Statistics Feature Extraction with Artificial Neural Network Classification Technique Mushroom Images Identification Using Orde 1 Statistics Feature Extraction with Artificial Neural Network Classification Techn," 2019.

[5] A. Ahmad, "Mengenal Artificial Intelligence, Machine Learning, Neural Network, dan Deep Learning," J. Teknol. Indones., no. October, p. 3, 2017.

[6] B. J. Erickson, P. Korfiatis, Z. Akkus, T. Kline, and K. Philbrick, "Toolkits and Libraries for Deep Learning," J. Digit. Imaging, vol. 30, no. 4, pp. 400–405, 2017.

[7] P. Jan and W. Gotama, "Pengenalan Pembelajaran Mesin dan Deep Learning Jan Wira Gotama Putra Pengenalan Konsep Pembelajaran Mesin dan Deep Learning," no. July, pp. 1–199, 2018.

[8] K. Chauhan and S. Ram, "International Journal of Advance Engineering and Research Image Classification with Deep Learning and Comparison between Different Convolutional Neural Network Structures using Tensorflow and Keras," pp. 533–538, 2018.

[9] H. Darmanto, D. Learning, T. Learning, and G. Descent, "Pengenalan Spesies Ikan Berdasarkan Kontur Otolith," vol. 2, 2019.

[10] K. O'Shea and R. Nash, "An Introduction to Convolutional Neural Networks," pp. 1–11, 2015.

[11] C. Y. Lee, P. W. Gallagher, and Z. Tu, "Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree," Proc. 19th Int. Conf. Artif. Intell. Stat. AISTATS 2016, pp. 464–472, 2016.

[11] Erickson, B. J., Korfiatis, P., Akkus, Z., Kline, T., & Philbrick, K. (2017). Toolkits and Libraries for Deep Learning. Journal of Digital Imaging, 30(4), 400–405.

[12] Arrofiqoh, E. N., & Harintaka, H. (2018). Implementasi Metode Convolutional Neural Network Untuk Klasifikasi Tanaman Pada Citra Resolusi Tinggi. Geomatika, 24(2), 61.

[13] Saifullah, S., -, S., & Yudhana, A. (2016). Analisis Perbandingan Pengolahan Citra Asli Dan Hasil Croping Untuk Identifikasi Telur. Jurnal Teknik Informatika Dan Sistem Informasi, 2(3), 341–350.

[14] Riadi, I., Umar, R., & Aini, F. D. (2019). Analisis Perbandingan Detection Traffic Anomaly Dengan Metode Naive Bayes Dan Support Vector Machine (Svm). ILKOM Jurnal Ilmiah, 11(1), 17.

[15] Faza, S. (2018). Peningkatan Kinerja Dalam Pengklasifikasian Menggunakan Deep Learning.

[16] Harjoseputro, Y. (2018). Convolutional Neural Network (Cnn) Untuk Pengklasifikasian Aksara Jawa. Buana Informatika, 23.

[17] Maha, V., Salawazo, P., Putra, D., Gea, J., Teknologi, F., & Indonesia, U. P. (2019). Implementasi Metode Convolutional Neural Network (Cnn) Pada Peneganalan Objek Video Cctv. 3(1), 74–79.

[18] Marifatul Azizah, L., Fadillah Umayah, S., & Fajar, F. (2018). Deteksi Kecacatan Permukaan Buah Manggis Menggunakan Metode Deep Learning dengan Konvolusi Multilayer. Semesta Teknika, 21(2), 230–236.

[19] Ilahiyah, S., & Nilogiri, A. (2018). Implementasi Deep Learning Pada Identifikasi Jenis Tumbuhan Berdasarkan Citra Daun Menggunakan Convolutional Neural Network. 3(2), 49–56.

[20] Abhirawan, H., Jondri, & Arifianto, A. (2017). Pengenalan Wajah Menggunakan Convolutional Neural Networks (CNN). Universitas Telkom, 4(3), 4907–4916.

[21] Rachmadi, R. F., & Purnama, I. K. E. (2018). Paralel Spatial Pyramid Convolutional Neural Network untuk Verifikasi Kekerabatan berbasis Citra Wajah. Jurnal Teknologi Dan Sistem Komputer, 6(4), 152.

[22] Pangestu, M. A., & Bunyamin, H. (2018). Analisis Performa dan Pengembangan Sistem Deteksi Ras Anjing padaGambar dengan Menggunakan Pre-Trained CNN Model. Jurnal Teknik Informatika Dan Sistem Informasi, 4, 337–344.

[23] Dewa, C. K., Fadhilah, A. L., & Afiahayati, A. (2018). Convolutional Neural Networks for Handwritten Javanese Character Recognition. IJCCS (Indonesian Journal of Computing and Cybernetics Systems), 12(1), 83.

[24] Putu Aryasuta Wicaksana, I Made Sudarma, D. C. K. (2019). Pengenalan Pola Motif Kain Tenun Gringsing Menggunakan Metode Convolutional Neural Network Dengan Model Arsitektur. 6(3), 159–168.

[25] Madhu Latha, M., & Krishnam Raju, K. V. (2019). Transfer learning based face recognition using deep learning. International Journal of Recent Technology and Engineering, 8(1), 38–44.