PENERAPAN *DEEP LEARNING* MENGGUNAKAN *CONVOLUTIONAL NEURAL NETWORK* UNTUK KLASIFIKASI CITRA WAYANG PUNAKAWAN

TUGAS AKHIR

Disusun Oleh:

Salsabila

14 611 156

JURUSAN STATISTIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS ISLAM INDONESIA

YOGYAKARTA

2018

PENERAPAN DEEP LEARNING MENGGUNAKAN *CONVOLUTIONAL NEURAL NETWORK* UNTUK KLASIFIKASI CITRA WAYANG PUNAKAWAN

TUGAS AKHIR

(Diajukan Sebagai Salah Satu Syarat untuk Memperoleh Gelar Sarjana Jurusan Statistika)

Salsabila

14 611 156

JURUSAN STATISTIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS ISLAM INDONESIA

YOGYAKARTA

2018

HALAMAN PERSETUJUAN PEMBIMBING

TUGAS AKHIR

Judul : Penerapan *Deep Learning* Menggunakan *Convolutional Neural Network* untuk Klasifikasi Citra Wayang Punakawan

Nama : Salsabila

Nomor Mahasiswa : 14 611 156

TUGAS AKHIR INI TELAH DIPERIKSA DAN DISETUJUI UNTUK DIUJIKAN

Yogyakarta, 21 Maret 2018

Mengetahui,
Dosen Pembimbing

Dr. RB Fajriya Hakim, S.Si., M.Si

HALAMAN PENGESAHAN

TUGAS AKHIR

PENERAPAN *DEEP LEARNING* MENGGUNAKAN *CONVOLUTIONAL NEURAL NETWORK* UNTUK KLASIFIKASI CITRA WAYANG PUNAKAWAN

Nama Mahasiswa : Salsabila

Nomor Mahasiswa : 14 611 156

TUGAS AKHIR INI TELAH DIUJIKAN

PADA TANGGAL 18 APRIL 2018

Nama Penguji :	**Tanda Tangan**
1. Andrie Pasca Hendradewa, S.T., M.T	.....
2. Ayundyah Kesumawati, S.Si., M.Si	.....
3. Dr. RB Fajriya Hakim, S.Si., M.Si	.....

Mengetahui,

Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam

Drs. Allwar, M.Sc., Ph.D

KATA PENGANTAR

Assalamu'alaikum warahmatullah hi wa barakatuh.

Puji syukur penulis ucapkan kepada Allah Subhanawataala atas segala limpahan rahmat, hidayah dan karunia-Nya penulis dapat menyelesaikan Tugas Akhir yang berjudul "**Penerapan Deep Learning Menggunakan Convolutional Neural Network Untuk Klasifikasi Citra Wayang Punakawan**" ini dengan baik. Shalawat serta salam penulis haturkan kepada Nabi Muhammad Salallahualaihiwassalam beserta keluarga, sahabat, dan umatnya.

Tugas akhir ini disusun sebagai salah satu syarat untuk memperoleh gelar Sarjana Statistika, Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Islam Indonesia. Penulis menyadari bahwa penulisan tugas akhir ini banyak mendapatkan bantuan dari berbagai pihak, baik berupa saran, kritik, bimbingan maupun bantuan lainnya. Oleh karena itu, pada kesempatan ini penulis menyampaikan ucapan terima kasih kepada:

1. Bapak dan Ibu penulis, orang tua yang luar biasa dan selalu memberikan doa dan dukungan kepada penulis atas terselesaikannya tugas akhir ini.
2. Diajeng Camila, adik perempuan penulis yang turut mendoakan serta memberikan dukungan.
3. Keluarga penulis lainnya yang telah mendoakan dan memberi dukungan.
4. Bapak Drs. Allwar, M.Sc., Ph.D., selaku Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Islam Indonesia.
5. Dr. RB. Fajriya Hakim, S.Si., M.Si, selaku Ketua Jurusan Statistika Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Islam Indonesia serta dosen pembimbing tugas akhir atas segala waktu dan bimbingannya sehingga tugas akhir ini dapat dimudahkan.
6. Dosen-Dosen di Prodi Statistika FMIPA UII yang telah membantu dan membimbing penulis dari semester awal hingga sekarang.

7. Sahabat-sahabat penulis yang juga seperjuangan Leni, Ridha, Sita, Panji, Aulia baik yang telah membantu, menemani maupun memberi dukungan kepada penulis selama pengerjaan tugas akhir ini.
8. Sahabat penulis Athar, Yuma, Devira yang memberikan dukungan berupa semangat dan doa.
9. Teman-teman KKN angkatan 55 unit 77 Fira, Fadjar, Elita, Rain, Harnum, Hamam, Kholik, dan Dimas, yang turut memberikan dukungan.
10. Teman-teman satu bimbingan tugas akhir yang membantu penulis.
11. Teman-teman seperjuangan jurusan Statistika angkatan 2014 atas dukungannya yang tidak bisa disebutkan satu persatu.
12. Segenap staf dan karyawan Fakultas Matematika dan Ilmu Pengetahuan Alam yang telah membantu kelancaran proses perkuliahan dalam bidang administrasi, akademik, dan lain-lain.
13. Semua pihak yang telah membantu yang tidak dapat penulis sebutkan satu per satu.

Demikian tugas akhir ini, penulis menyadari bahwa dalam tugas akhir ini masih banyak kekurangan karena keterbatasan pengetahuan dan kemampuan yang dimiliki oleh penulis. Oleh karena itu, kritik dan saran yang bersifat membangun diharapkan demi lebih baiknya penulisan tugas akhir ini. Semoga tugas akhir ini bermanfaat bagi semua pihak.

Wassalamu'alaikum warahmatullah hi wa barakatuh.

Yogyakarta, 18 April 2018

Salsabila

DAFTAR ISI

DAFTAR TABEL

Nomor Judul Halaman

DAFTAR GAMBAR

Nomor Judul Halaman

DAFTAR LAMPIRAN

PERNYATAAN

Dengan ini penulis menyatakan bahwa dalam tugas akhir ini tidak terdapat karya yang sebelumnya pernah diajukan untuk memperoleh gelar kesarjanaan di suatu perguruan tinggi dan sepengetahuan penulis juga tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang diacu dalam naskah ini dan disebutkan dalam daftar pustaka.

Yogyakarta, 18 April 2018

Salsabila

PENERAPAN *DEEP LEARNING* MENGGUNAKAN *CONVOLUTIONAL NEURAL NETWORK* UNTUK KLASIFIKASI CITRA WAYANG PUNAKAWAN

Salsabila

Program Studi Statistika Fakultas Matematika dan Ilmu Pengetahuan Alam

Universitas Islam Indonesia

INTISARI

Wayang merupakan budaya Indonesia yang harus dilestarikan keberadaannya agar dapat terus dikenal oleh generasi-generasi berikutnya. Terdapat banyak jenis wayang di Indonesia, salah satunya adalah wayang Punakawan. Wayang Punakawan merupakan salah satu wayang yang berasal dari sejarah Mahabrata. Wayang punakawan terdiri dari empat tokoh wayang, yaitu Semar, Gareng, Petruk, dan Bagong. Berdasarkan banyaknya jenis wayang di Indonesia, peneliti ingin membuat sebuah sistem yang mampu mengenal hanya untuk keempat wayang tersebut. Dalam hal ini dibutuhkan sebuah metode yang mampu mengklasifikasikan wayang-wayang menjadi keempat tokoh tersebut. Sebuah sistem diharapkan dapat dibangun untuk dapat mengenal keempat objek tersebut. Dalam hal ini, digunakan sebuah cabang ilmu *machine learning* yang mampu mengenal kumpulan citra dan mengklasifikasikan yaitu *deep learning.* Salah satu metode *deep learning* yang digunakan adalah *Convolutional Neural Network* (CNN). Hal yang membedakan *CNN* dengan metode *neural network* lainnya adalah jumlah *hidden layer* yang banyak pada proses klasifikasi. Proses klasifikasi *CNN* yang digunakan pada penelitian ini menggunakan dukungan *Mxnet* sebagai *framework* dari metode *deep learning.* Dalam penelitian ini digunakan 1200 *dataset* wayang Punakawan dengan jumlah data *train* dan data *test* masing-masing sebanyak 1080 dan 120 citra. *Preprocessing* data, klasifikasi *CNN,* dan pembuatan model dapat dilewati dengan baik. Hasil yang diperoleh adalah model dapat mengenali dan mengklasifikasikan data citra uji dengan akurasi sebesar 91.6 %.

Kata Kunci: *Deep Learning, CNN, Mxnet, Akurasi, Klasifikasi, Punakawan*

IMPLEMENTATION OF DEEP LEARNING USING CONVOLUTIONAL NEURAL NETWORK FOR IMAGE CLASSIFICATION OF PUNAKAWAN PUPPET

Salsabila

Department of Statistics, Faculty of Mathematics and Natural Sciences

Islamic University of Indonesia

ABSTRACT

Puppet is an Indonesian culture that must be preserved for its existence to be recognized by the next generations. There are many types of puppet in Indonesia, one of them is Punakawan puppet. Punakawan is one of the puppets from Mahabrata history. Punakawan puppet consists of four puppet figures, namely Semar, Gareng, Petruk, and Bagong. Based on the many types of puppets in Indonesia, researchers want to create a system that is able to recognize only for the four puppet. In this case, required a method that can classify the puppet into the fourth character. A system is expected to be built to be able to recognize the four objects. In this case, used a branch of machine learning that is able to recognize the collection of images and classify deep learning. One of the deep learning methods used is Convolutional Neural Network (CNN). What distinguishes CNN from other neural network methods is the number of hidden layers in the classification process. CNN classification process is using Mxnet support as a framework of deep learning method. In this study, used 1200 puppet dataset Punakawan with the number of data train and test data as much as 1080 and 120. Preprocessing data, CNN classification, and modeling can be passed well. The result obtained is the model can recognize and classify the test data with an accuracy of 91.6%.

Keywords: *Deep Learning, CNN, Mxnet, Accuracy, Classification, Punakawan*

BAB I
PENDAHULUAN

1.1. Latar Belakang

Keragaman budaya yang dimiliki Indonesia telah menjadi kebanggaan tersendiri sejak jaman dahulu. Salah satu kebudayaan Indonesia yang masih dikenal hingga kini ialah budaya wayang. Wayang merupakan budaya wiracarita yang mengisahkan kehidupan para tokoh-tokoh terdahulu dengan berbagai macam watak dan khasnya tersendiri. Satu hal yang khas dari budaya satu ini yaitu wayang merupakan budaya yang dapat menjadi media untuk menyampaikan dakwah, pendidikan, hiburan, dan pemahaman filsafat. "Wayang merupakan sastra tradisional yang memenuhi kualifikasi karya *master piece*, karya sastra dan atau budaya *adiluhung*" (Burhan, 2011). Budaya satu ini bahkan telah dikenal hingga di kalangan internasional, baik karena kekhasannya yang memikat maupun dilestarikan oleh anak bangsa. Wayang merupakan media untuk menceritakan tokoh-tokoh dalam sejarah yang dimainkan oleh seorang atau lebih yang biasanya disebut sebagai dalang. Salah satu jenis wayang yang terkenal di kalangan masyarakat yaitu wayang Punakawan. Menurut sejarah, Punakawan atau juga disebut Panakawan lahir di bumi Indonesia. Sedangkan tokoh-tokoh Punakawan yang menjadi topik bahasan pada penelitian ini berfokus pada wayang purwa (Jawa). Wayang Punakawan merupakan tokoh yang khas dalam wayang Indonesia, mempunyai karakter yang unik dan bisa menjalankan berbagai macam peran, seperti pengasuh dan penasehat para ksatria, penghibur, kritikus, pelawak bahkan sebagai penutur kebenaran dan kebajikan. Tidak semua jenis wayang terdapat tokoh-tokoh Punakawan, Semar, Gareng, Bagong, Petruk, hanya wayang yang mengambil cerita dari kisah Mahabarata saja yang terdapat tokoh Punakawan.

Gambar 1.1 Wayang Punakawan

Sumber : flickr.com

Banyaknya jenis wayang yang ada di Indonesia membuat peneliti tertarik untuk membuat suatu program pengenalan jenis wayang khusunya untuk wayang Punakawan berdasarkan dataset foto atau citra wayang Punakawan. Pengenalan gambar wayang Punakawan tersebut nantinya akan diklasifikasikan sesuai dengan empat jenis punakawan yang ada. Pembuatan program pengenalan ini adalah sebagai salah satu langkah penulis dalam melestarikan budaya yang ada di Indonesia. Kenekaragaman budaya yang ada harus selalu dilestarikan sampai kepada generasi-generasi berikutnya. Penggunaan objek wayang Punakawan sebagai bahan penelitian adalah salah satu media penulis sebagai pelajar dalam proses pelestarian budaya Indonesia. Melihat berkembangnya zaman dimana masyarakat Indonesia bergantung sekali dengan kecanggihan teknologi, memungkinkan sekali apabila budaya yang ada mulai dilupakan. Kebiasaan hidup yang serba modern dan kesibukan sehari-hari masyarakat juga sangat memungkinkan budaya tradisional mulai dilupakan. Oleh karena itu, penggunaan objek Punakawan dalam penelitian ini diharapkan mampu memberikan kontribusi dalam pelestarian budaya Indonesia. Dalam hal ini kumpulan data objek wayang Punakawan akan diklasifikasikan sesuai dengan jenisnya menggunakan sebuah program.

“Klasifikasi adalah proses untuk menemukan model atau fungsi yang menjelaskan atau membedakan konsep atau kelas data, dengan tujuan untuk dapat

memperkirakan kelas dari suatu objek yang labelnya tidak diketahui, proses klasifikasi biasanya dibagi menjadi dua fase yaitu fase *learning* dan fase *test*, pada fase *learning*, sebagian data yang telah diketahui kelas datanya diumpankan untuk membentuk model perkiraan, kemudian pada fase *test* model yang sudah terbentuk diuji dengan sebagian data lainnya untuk mengetahui akurasi dari model tersebut, bila akurasinya mencukupi model ini dapat dipakai untuk prediksi kelas data yang belum diketahui" (Pramudiono, 2003).

Seiring berkembangnya zaman, ada saja cara dan teori-teori baru yang ditemukan sehingga pekerjaan yang sebelumnya sulit atau lama dilakukan menjadi lebih mudah dan cepat. *Machine Learning* merupakan salah satu teori baru di dunia pemrograman komputer. .

Deep Learning adalah area baru dalam penelitian *Machine Learning*, yang telah diperkenalkan dengan tujuan menggerakkan *Machine Learning* lebih dekat dengan salah satu tujuan aslinya yaitu *Artificial Intelligence*. *Convolutional Neural Network* (CNN) merupakan salah satu metode *Deep Learning* yang dapat diterapkan untuk melakukan klasifikasi gambar. Metode ini telah digunakan antara lain dalam pengenalan citra, *computer vision*, serta *Natural Language Processing* (NLP). Penelitian yang dilakukan oleh Rismiyati (2016) menggunakan metode *Convolutional Neural Network* untuk membedakan salak yang lulus ekspor dan tidak memperoleh hasil akurasi sebesar 81,5 %.

Metode *CNN* merupakan salah satu metode dalam *Deep Learning* yang memiliki beberapa kelebihan, diantaranya adalah *CNN* merupakan bidang terbaru dalam *Deep Learning* dan juga telah banyak penelitian terdahulu yang menggunakan *CNN* sebagai metode pengenalan pola khususnya klasifikasi gambar dengan hasil yang sangat baik. Mengingat belum ada pula yang melakukan penelitian klasifikasi wayang Punakawan menggunakan metode *CNN*, oleh karena itu, *Convolutional Neural Network* digunakan dalam penelitian ini guna dapat mengenali serta mengklasifikasikan wayang berdasarkan empat jenis wayang, yaitu Petruk, Bagong, Semar, dan Gareng.

1.2. Rumusan Masalah

Berdasarkan latar belakang yang telah disampaikan sebelumnya maka rumusan masalah dalam penelitian ini adalah :

1. Bagaimanakah penerapan *preprocessing* pada sampel citra yang kemudian dijadikan sebagai *dataset* proses *training CNN*?
2. Bagaimanakah penentuan jumlah *dataset* yang tepat guna dapat memperoleh akurasi pengklasifikasian yang terbaik ?
3. Bagaimanakah hasil klasifikasi wayang Punakawan dari model yang diperoleh ?

1.3. Batasan Masalah

Agar permasalahan yang diteliti tidak meluas, maka diberikan batasan-batasan sebagai berikut :

1. Penelitian ini berfokus pada *Convolutional Neural Network* yang merupakan salah satu teknik dalam *deep learning.*
2. Data *input* dan data uji merupakan citra wayang Punakawan yang bersumber dari *internet* dengan berbagai sumber dan foto manual.
3. Data *input* berupa citra untuk setiap tokoh dari wayang Punakawan.
4. Alat analisis yang digunakan adalah *Convolutional Neural Network* yang didukung *framework Mxnet*.
5. Data diolah menggunakan bantuan *software R Studio* dan *Microsoft Excel 2013.*

1.4. Tujuan Penelitian

Berdasarkan rumusan masalah yang telah disebutkan sebelumnya, tujuan penelitian ini adalah :

1. Menerapkan *preprocessing* pada sampel citra yang kemudian dijadikan sebagai *dataset* proses *training CNN.*
2. Mengetahui jumlah data yang tepat guna dapat menghasilkan akurasi pengklasifikasian wayang Punakawan yang terbaik.
3. Mengetahui hasil klasifikasi wayang Punakawan dari model yang diperoleh.

1.5. Manfaat Penelitian

Manfaat yang diharapkan dari penelitian ini yaitu :

1. Mengetahui hasil penerapan *preprocessing* pada sampel citra yang kemudian menjadi *dataset* proses *training CNN*.
2. Dapat mengetahui jumlah data yang tepat guna dapat menghasilkan akurasi pengklasifikasian wayang Punakawan yang terbaik.
3. Dapat mengetahui hasil klasifikasi wayang Punakawan dari model yang diperoleh.

BAB II
TINJAUAN PUSTAKA

Setelah penulis melakukan telaah terhadap beberapa penelitian sebelumnya, terdapat beberapa penelitian yang memiliki keterkaitan dengan penelitian yang dilakukan oleh penulis.

Penelitian pertama yang berhasil peneliti temukan yaitu penelitian yang dilakukan oleh Razi (2017) dengan judul "Klasifikasi Artikel Berita Berbahasa Indonesia Menggunakan Convolutional Neural Network". Dalam penelitian tersebut digunakan metode *Convolutional Neural Network* (CNN) untuk mengklasifikasikan artikel berita berbahasa indonesia. Terdapat lima kelas dalam pengklasifikasian, yaitu entertainment, kesehatan, olahraga, teknologi, dan ekonomi. Sebelum dilakukan klasifikasi, terlebih dahulu dilakukan pengubahan kata-kata ke dalam bentuk vektor dengan menggunakan *word2vec* sehingga hasil dari pengubahan tersebut dapat diinputkan ke dalam *CNN*. Hasil pengujian menunjukkan bahwa kombinasi penggunaan metode *Convolutional Neural Network* dan *word2vec* memberikan hasil nilai akurasi sebesar 96.70 %, dan presisi, *recall* serta *f-measure* mencapai 96,60 %.

Penelitian kedua yaitu penelitian yang dilakukan oleh Zufar dan Setiyono (2016) dengan judul "Convolutional Neural Networks untuk Pengenalan Wajah Secara Real-Time". Penelitian ini menggunakan mengimplementasikan metode *CNN* dan bantuan *library OpenCV* untuk deteksi multi wajah dan perangkat *Web Cam M-Tech* 5MP. Hasil uji coba dengan munggunakan konstruksi model *CNN* sampai kedalaman 7 lapisan dengan input dari hasil ekstraksi *Extended Local Binary Pattern* dengan radius 1 dan *neighbor* 15 memperlihatkan kinerja pengenalan wajah meraih rata-rata tingkat akurasi lebih dari 89% dalam $\mp$ 2 *frame* per detik.

Penelitian ketiga yaitu oleh Rismiyati (2016) dalam tesisnya dengan judul "Implementasi Convolutional Neural Network Untuk Sortasi Mutu Salak Ekspor Berbasis Citra Digital". Citra salak digunakan sebagai data input dalam menentukan

salak yang lolos ekspor atau tidak dengan beberapa kelas mutu yaitu bagus, matang, cacat atau sobek. Hasil penelitian ini menunjukkan bahwa akurasi terbaik untuk model dua kelas didapatkan dengan metode *CNN* dengan menggunakan *learning rate* 0.0001, satu lapisan konvolusi dengan jumlah *filter* lima belas ukuran 3x3x3, dan jumlah neuron pada lapisan tersembunyi 100. Akurasi yang didapatkan adalah 81,5%. Model empat kelas mendapat akurasi 70,7% dengan dua lapisan konvolusi.

Penelitian keempat yaitu oleh Pudi (2017) dalam tesisnya dengan judul "Implementasi Deep Learning Menggunakan Convolutional Neural Network Untuk Klasifikasi Citra Candi Berbasis Gpu". Penelitian ini menggunakan citra candi sebagai data *input.* Hasil pengujian menunjukan akurasi sebesar 98,99% pada *training set* dan 85,57% pada test set dengan waktu pelatihan mencapai 389,14 detik. Sehingga dapat disimpulkan bahwa klasifikasi citra candi dapat dilakukan dengan sangat baik menggunakan teknik *Deep Learning* dengan *CNN*.

Penelitian kelima yaitu oleh Mulyani (2016) dalam tugas akhirnya dengan judul "Pengenalan Suara Pada Sistem Notulen Rapat Menggunakan Convolutional Neural Network (CNN)". Data *input* berupa suara dalam rapat dikonversi dalam bentuk *spectrogram image.* Setelah diperoleh data *spectrogram image*, kemudian akan dilakukan *scaling* serta *thresholding*. Proses perhitungan akurasi pada setiap data dilakukan menggunakan *confission matrix*. Hasil eksperimen terhadap 120 data rekaman suara dengan format .wap yang dikumpulkan memperlihatkan bahwa akurasi tertinggi yang dihasilkan berdasarkan *voice recognition* menggunakan model *CNN* adalah 92,5% dengan rata-rata akurasi adalah 80,33%.

Penelitian keenam yaitu oleh Rajagede (2016) dalam tesisnya yang berjudul "Deep Learning untuk Pengenalan Pelafalan Huruf Hijaiyah Berharakat". Arsitektur *Convolutional Neural Network* (CNN) digunakan untuk menyelesaikan kasus pengenalan pelafalan huruf Arab (huruf Hijaiyah berharakat). Digunakan pula beberapa metode optimasi regularisasi untuk meningkatkan akurasi dan mengurangi *overfitting*, seperti *dropout* dan L2 *regularization.* Penelitian ini menguji data rekaman suara dan mengklasifikasikan ke 10 kelas huruf hijaiyah berharakat. Hasil penelitian ini diperoleh akurasi 78.75% ketika dilakukan tanpa regularisasi dan mencapai 80.75% ketika menggunakan regularisasi.

Beberapa paragraf diatas telah diulas beberapa penelitian yang membahas terkait metode penelitian menggunakan *Convolutional Neural Network.* Selanjutnya penelitian terdahulu yang berkaitan dengan data penelitian dilakukan oleh Nugraha, Santoso, dan Suselo (2013) dengan judul "Algoritma Backpropagation pada Jaringan Saraf Tiruan untuk Pengenalan Pola Wayang Kulit". Adapun data input yang digunakan adalah wayang kulit pandawa, yaitu wayang kulit Bima, Yudhistira, Arjuna, Nakula, dan Sadewa. Algoritama *Backpropagation* dalam *Neural Network* digunakan untuk mengetahui apakah pola wayang tersebut dapat dikenali dengan baik serta tingkat akurasi yang baik pula. Hasil penelitian menunjukkan bahwa pengenalan pola dapat berlangsung dengan baik untuk gambar wayang kulit yang sudah dikenali sebelumnya dengan nilai akurasi sebesar 100%, sedangkan untuk gambar yang belum dikenali bergantung pada banyak sedikitnya jumlah gambar pada *training set*.

Penelitian yang dilakukan dalam tugas akhir ini adalah bagaimana penerapan model *deep learning* metode *Convolutional Neural Network* (CNN) pada wayang Punakawan yang terdiri dari wayang Bagong, Semar, Petruk, Dan Gareng. Data dalam penelitian ini merupakan kumpulan citra wayang Punakawan yang bersumber dari *internet* dan foto manual yang dilakukan oleh peneliti. Tokoh wayang Punakawan memiliki ciri atau pola tubuh yang berbeda-beda. Semar misalnya, ciri yang paling khas adalah tubuhnya sangat gemuk dengan telunjuk yang selalu mengarah kedepan. Kemudian petruk dengan tubuh kurus, hidung panjang, dan kuncir di kepala. Pola-pola unik seperti inilah yang dapat menjadi perbedaan dengan pola pada objek lainnya. Perbedaan inilah yang membuat sebuah sistem yang menggunakan spesifikasi parameter pada penelitian terdahulu belum tentu dapat bekerja pada sistem pengenalan objek Punakawan. Oleh karena itu, dibutuhkan pengujian-pengujian parameter guna memperoleh hasil yang terbaik. Adapun tujuan penelitian ini untuk mengetahui tingkat keakuratan metode *CNN* untuk mengenali wayang Punakawan.

BAB III
LANDASAN TEORI

3.1. Wayang Punakawan

Salah satu wayang yang terkenal sebagai budaya wayang di Indonesia yaitu wayang Punakawan. Wayang punakawan merupakan salah satu wayang yang berasal dari cerita mahabrata. Terdapat empat tokoh dalam wayang Punakawan, yaitu Semar, Petruk, Gareng dan Bagong dengan berbagai karakter pada masing-masing wayang. Masing-masing tokoh Punakawan dalam wayang kulit purwa khusunya punakawan jawa memiliki karakter yang khas dan bermakna, berikut ini merupakan karakter dalam wayang Punakawan :

a) Semar, pengasuh para Pandawa. Meskipun semar berwujud manusia jelek, Semar memiliki kesaktian yang sangat tinggi bahkan dapat melebihi para dewa.
b) Gareng, putra Semar. Gareng adalah seseorang yang tidak pandai bicara dan apa yang dikatakannya terkadang serba salah tetapi sangat lucu.
c) Petruk, Putra Semar yang bermuka manis dengan senyuman yang menarik hati, pandai berbicara dan juga sangat lucu. Dikisahkan bahwa tidak ada yang dapat mengalahkan Petruk selain Gareng.
d) Bagong, berarti bayangan Semar. Menurut sejarahnya, ketika diturunkan ke dunia, Dewa bersabda pada Semar bahwa bayangannyalah yang akan menjadi temannya. Seketika itu juga bayangannya berubah wujud menjadi Bagong, yang memiliki sifat lancang dan berlagak bodoh akan tetapi sangat lucu.

Gambar 3.1 Tokoh wayang punakawan

Sumber : http://kelas3sdn1kebumen.blogspot.co.id

3.2. Klasifikasi

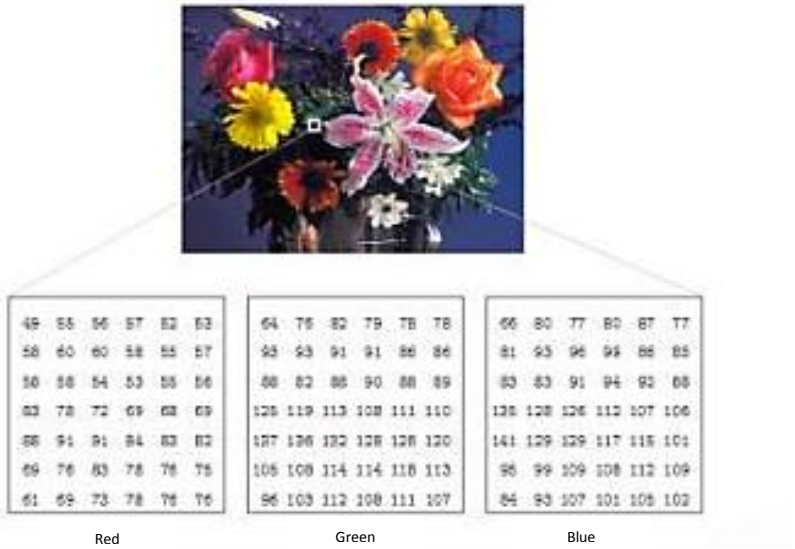
"Klasifikasi diartikan sebagai proses untuk memperoleh model ataupun fungsi yang melukiskan dan membedakan kelas data maupun konsep yang mempunyai tujuan memprediksikan kelas untuk data yang tidak dikenali kelasnya" (J. Han, M. Kamber, 2006). "Ada banyak teknik yang dapat dilakukan untuk mengklasifikasikan data, diantaranya adalah decision tree, naive bayesian classifier, bayesian belief network dan rule based classifier" (Han dan Kamber, 2006). "Setiap algoritma klasifikasi tersebut memiliki kelebihan dan kekurangan, tetapi prinsip dari masing-masing algoritma tersebut sama, yaitu melakukan suatu pelatihan sehingga di akhir pelatihan, model dapat memprediksi setiap vektor masukan ke label kelas output dengan tepat" (Prasetyo, 2012).

3.3. Model Warna

"Model warna merupakan sebuah cara atau metode untuk mengatur, membuat dan memvisualisasikan warna" (Ford and Roberts, 1998). Bagi aplikasi yang berbeda maka ruang warna yang dipakai bisa juga berbeda, hal ini disebabkan oleh beberapa peralatan tertentu yang membatasi secara ketat ukuran dan jenis ruang warna yang dapat digunakan. Model warna biasanya digunakan untuk menganalisis citra. Model warna yang umum digunakan adalah model *rgb* (red,green,blue), *cmy* (cyan, magenta, yellow), *cmyk* (cyan, magenta, yellow, black) dan *hsi* (hue, saturation, intensity). Citra *rgb* merupakan salah satu model warna yang dimiliki oleh citra hasil dari kamera digital dan *display* monitor.

3.3.1. Model Warna RGB

Pada *color image* ini masing-masing piksel memiliki warna tertentu, warna tersebut adalah merah (Red), hijau (Green) dan biru (Blue). Jika masing-masing warna memiliki *range* 0 - 255, maka totalnya adalah 255 x 3 = 16.581.375 variasi warna berbeda pada gambar, dimana variasi warna ini cukup untuk gambar apapun. Karena jumlah bit yang diperlukan untuk setiap piksel, gambar tersebut juga disebut gambar-bit warna. *Color image* ini terdiri dari tiga matriks yang mewakili nilai-nilai merah, hijau dan biru untuk setiap pikselnya, seperti yang ditunjukkan pada **Gambar 3.2** dibawah ini.



Gambar 3.2 Color image (Mcandrew, 2003)

3.3.2. Konversi Warna *RGB* ke *Greyscale*

Perubahan model warna *RGB* ke nilai *greyscale* biasanya dilakukan untuk mempermudah saat melakukan pemrosesan citra digital. Terdapat beberapa rumus yang dapat digunakan untuk mengubah model warna *RGB* ke nilai *greyscale*. "Dua persamaan yang umum dipakai untuk melakukan konversi ini antara lain" (Kanan dan Cottrell, 2012) :

Averaging

Metode *average* merupakan metode yang paling sederhana dalam hal mengubah nilai *RGB* menjadi nilai *greyscale*. Dengan metode ini, nilai *greyscale* didapatkan dari mencari rata-rata nilai dari *R, G,* dan *B* atau sebagaimana ditunjukkan pada Persamaan (3.1)

$$G_{Average} = \frac{R+G+B}{3} \tag{3.1}$$

Luminance

Metode *luminance* dirancang untuk menyamakan dengan kecerahan yang diterima manusia dengan menggunakan kombinasi berbobot dari nilai komponen *RGB.* Metode ini telah umum digunakan dalam ilmu *computer vision*. Persamaan untuk metode ini ditunjukkan pada Persamaan (3.2).

$$G_{Average} = 0.299 * R + 0.587 * G + 0.114 * B \tag{3.2}$$

3.4. *Deep Learning*

Teknologi *Machine Learning* menguatkan banyak aspek dari masyarakat yang modern : mulai dari pencarian sebuah *website* hingga penyaringan konten pada jaringan sosial hingga rekomendasi di situs *web e-commerce*, dan semakin hadir dalam produk konsumen seperti kamera dan *smartphone*. Sistem *machine learning* digunakan untuk mengidentifikasi objek dalam gambar, mencocokkan item berita, mengubah ucapan menjadi kata, memilih hasil pencarian yang relevan, dan *posting* atau produk sesuai dengan minat pengguna. Semakin banyaknya, pengaplikasian tersebut memanfaatkan jenis teknik yang disebut *deep learning.*

"Deep learning adalah salah satu teknik pada machine learning yang memanfaatkan banyak layer pengolahan informasi nonlinier untuk melakukan ekstraksi fitur, pengenalan pola, dan klasifikasi" (Deng dan Yu, 2014).

3.5. Neural Network

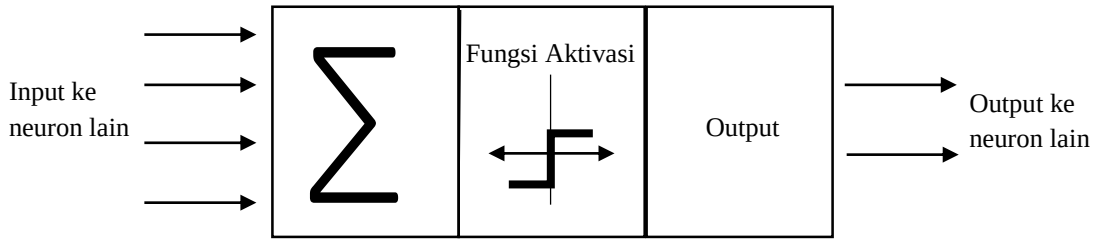
"*Neural network* adalah salah satu representasi buatan dari otak manusia yang selalu mencoba untuk mensimulasikan proses pembelajaran pada otak manusia tersebut" (Kusumadewi, 2004). Otak manusia terdiri dari ratusan juta sel syaraf yang disebut neuron. Otak digambarkan sebagai sebuah mesin yang mengubungkan neuron satu dengan lainnya dalam bentuk impuls saraf sehingga dapat mengkoordinasikan berbagai fungsi tubuh.

Cara kerja neuron tersebut ditiru oleh *Neural Network* sebagai langkah membuat *machine* yang pintar. Dalam hal ini, *Neural Network* tidak diprogram untuk menghasilkan keluaran tertentu. Semua keluaran yang ditarik oleh jaringan didasarkan pada pengalamannya selama mengikuti proses pembelajaran. Pada proses pembelajaran, pola-pola *input* dimasukkan ke *Neural Network* untuk dan kemudian jaringan diajari untuk menentukan jawaban yang bisa diterima. Pembelajaran dilakukan dengan menentukan bobot masing-masing node dalam jaringan.

3.5.1. Komponen Neural Network

Neural network terdiri dari kumpulan beberapa neuron unit yang saling terhubung. Masing-masing neuron mentransformasikan informasi yang telah

diterima menuju neuron lain melalui sambungan atau *link*. Pada *neural network*, hubungan ini disebut dengan bobot. **Gambar 3.3** menunjukkan struktur neuron pada *neural network*. Informasi masukan bagi neuron, atau *input*, dikirim ke neuron dengan bobot kedatangan tertentu. *Input* ini diproses dengan suatu fungsi yang menjumlahkan nilai berbobot dari semua *input* yang datang. Hasil penjumlahan kemudian dikenakan fungsi aktivasi untuk menentukan apakah neuron tersebut akan diaktifkan atau tidak. Biasanya hal ini dilakukan dengan membandingkan dengan *threshold* atau ambang nilai tertentu. Apabila neuron tersebut diaktifkan, maka neuron tersebut mengirimkan *output* melalui bobot-bobotnya ke semua neuron selanjutnya yang berhubungan.



Gambar 3.3 Satu unit neuron pada JST (Kusumadewi, 2004)

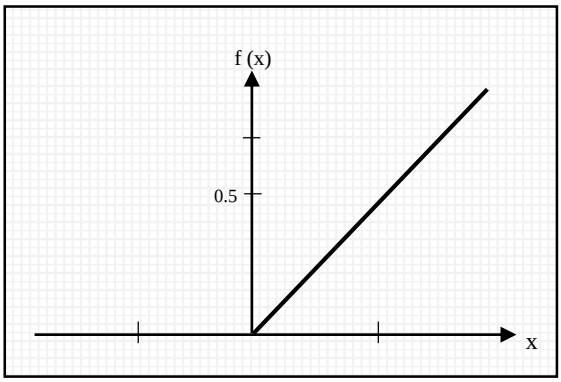
3.5.2. Fungsi Aktivasi

Fungsi aktivasi merupakan sebuah fungsi yang menentukan aktif tidaknya neuron. Fungsi yang dipakai bisa berupa fungsi linear dengan nilai ambang atau ReLu, atau fungsi non-linear seperti fungsi sigmoid dan fungsi tanh.

Fungsi ReLu (Rectified Linier Unit)

Fungsi yang digunakan untuk aktivasi pada reLu adalah sebagaimana ditunjukkan pada **Gambar 3.4**. Secara umum fungsi ReLU dinyatakan dalam persamaaan (3.3). Dari gambar dan persamaan tersebut, maka nilai *output* dari neuron bisa dinyatakan sebagai 0 jika *input*nya adalah negatif. Jika nilai *input* dari fungsi aktivasi adalah positif, maka *output* dari neuron adalah nilai *input* aktivasi itu sendiri.

$$f(x) = \max(0, x) \tag{3.3}$$

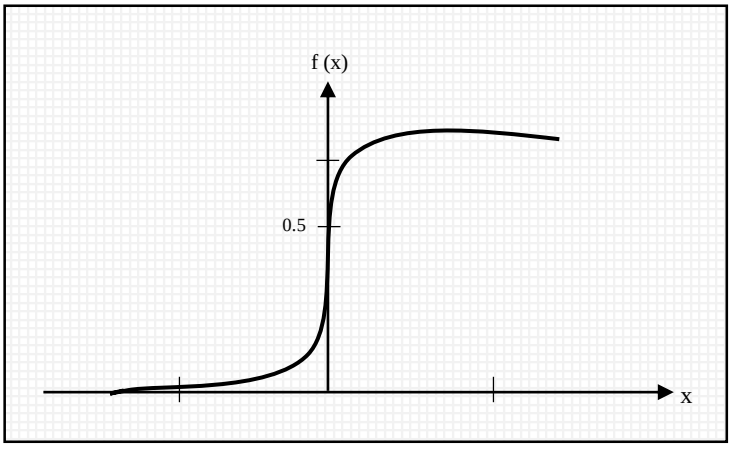


Gambar 3.4 Fungsi Aktivasi ReLu

Fungsi Sigmoid

Fungsi sigmoid adalah fungsi nonlinear yang mempunyai persamaan matematika sebagaimana ditunjukkan pada Persamaan (3.4). Masukan untuk fungsi aktivasi tersebut adalah nilai real dan keluaran dari fungsi tersebut adalah nilai antara 0 dan 1. Jika masukannya sangat negatif, maka keluaran yang didapatkan adalah 0, sedangkan jika masukan sangat positif maka nilai keluaran yang didapatkan adalah 1. Nilai masukan dan keluaran dari fungsi sigmoid dapat dinyatakan dalam grafik pada **Gambar 3.5**.

$$f(x) = \frac{1}{1+e^{-x}} \qquad (3.4)$$



Gambar 3.5 Fungsi aktivasi sigmoid

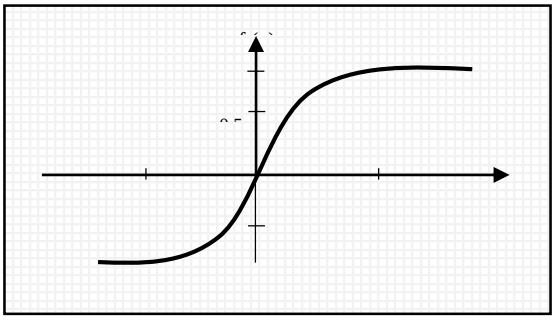
Fungsi sigmoid mempunyai dua kekurangan utama yaitu pertama,bahwa fungsi ini mempunyai gradien mendekati nilai nol jika nilai *input* sangat negatif atau sangat positif. Hal ini tidak diharapkan karena nilai gradien digunakan dalam

proses pelatihan. Kelemahan kedua adalah bahwa fungsi ini tidak terpusat pada nilai 0 (zero centered).

Fungsi Tanh

Fungsi tanh mengubah masukan yang bernilai real menjadi nilai antara -1 dan 1 sebagaimana ditunjukkan pada **Gambar 3.6**. Pada gambar tersebut ditunjukkan bahwa nilai yang sangat negatif dirubah menjadi nilai -1 dan nilai yang sangat positif dirubah menjadi nilai 1. Fungsi ini merupakan fungsi yang nilainya terpusat pada nilai 0 sehingga lebih dipilih dibandingkan dengan fungsi sigmoid. Persamaan untuk fungsi tanh ditunjukkan pada Persamaan (3.5) dimana nilai fungsi tanh merupakan dua kali nilai sigmoid dikurangi satu.

$$\tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \tag{3.5}$$



Gambar 3.6 Fungsi aktivasi tanh

3.5.3. *Backpropagation Learning*

Neural network menggunakan cara kerja otak manusia dalam prosesnya, oleh karena itu diperlukan sebuah pembelajaran atau *learning*. Pembelajaran dilakukan untuk menentukan nilai bobot yang tepat untuk masing-masing *input*. Bobot akan bertambah jika informasi yang diberikan oleh neuron yang bersangkutan dapat tersampaikan. Dan sebaliknya, bobot akan berubah secara dinamis sehingga dicapai suatu nilai yang seimbang jika informasi tidak disampaikan. Proses pembelajaran dapat dihentikan apabila nilai-nilai ini mampu mengidentifikasikan hubungan antara *input dan output.* Terdapat dua metode utama dalam melakukan pembelajaran, yaitu :

1. Pembelajaran tak terawasi (*Unsupervised learning*)

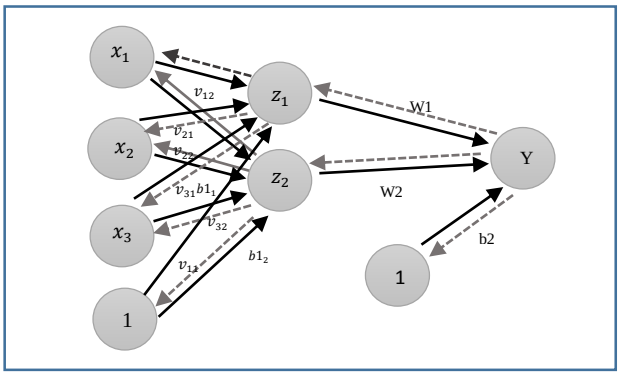
Pada metode ini target *output* tidak diperlukan dan hasil yang diinginkan tidak dapat ditentukan dari awal. Tujuan dari pembelajaran metode ini adalah mengelompokkan unit-unit yang hampir sama dalam suatu kelompok tertentu.

2. Pembelajaran terawasi (*Supervised learning*)

Metode pembelajaran pada *neural network* disebut terawasi jika *output* yang diharapkan telah diketahui sebelumnya. Pada proses pembelajaran, satu pola *input* diberikan ke neuron pada lapisan *input*. Pola ini dirambatkan sepanjang lapisan neual network tersebut hingga sampai pada neuron pada lapisan *output*. Lapisan *output* membangkitkan pola *output* yang nantinya dicocokkan dengan pola *output* targetnya. Contoh metode ini adalah *backpropagation*.

"*Backpropagation* merupakan algoritma *supervised learning* dan biasanya digunakan oleh perceptron dengan banyak lapisan untuk mengubah bobot-bobot yang terhubung dengan neuron-neuron yang ada pada lapisan tersembunyi. Algoritma ini menggunakan *error output* untuk mengubah nilai bobot-bobotnya dalam arah mundur (backward). Untuk mendapatkan *error* ini, tahap perambatan maju (forward propagation) harus dikerjakan terlebih dahulu. Pada saat perambatan maju, neuron-neuron diaktifkan dengan menggunakan fungsi aktivasi yang dapat diturunkan, seperti fungsi sigmoid" (Kusumadewi, 2004).

Arsitektur jaringan *backpropagation* seperti ditunjukkan pada **Gambar 3.7**. Gambar tersebut menunjukkan *neural network* yang terdiri dari tiga unit neuron pada lapisan *input* (x1, x2, dan x3), dua neuron pada lapisan tersembunyi (Z1dan Z2), dan 1 unit neuron pada lapisan *output* (Y). Bobot yang menghubungkan x1, x2, dan x3 dengan neuron pertama pada lapisan tersembunyi adalah V11, V21, dan V31. b11 dan b12 adalah bobot bias yang menuju neuron pertama dan kedua pada lapisan tersembunyi. Bobot yang mengubungkan Z1 dan Z2 dengan neuron pada lapisan *output* adalah w1 dan w2. Bobot bias b2 menghubungkan lapisan tersembunyi dengan lapisan *output*.



Gambar 3.7 Jaringan backpropagation

Algoritma *backpropagation* untuk jaringan dengan satu lapisan tersembunyi sebagaimana pada **Gambar 3.7** bekerja sebagai berikut (Kusumadewi,2004):

a) Inisialisasi bobot (menetapkan nilai bobot awal untuk semua parameter).

b) Tetapkan kondisi berhenti yang berupa maksimum epoch, iterasi, atau target *error*. Satu *epoch* adalah satu putaran training untuk semua data latih yang ada.

c) Tetapkan *learning rate*(α).

d) Inisialisasi epoch=0.

e) Kerjakan selama kondisi berhenti belum terpenuhi (epoh < maksimum epoh dan error < target error):

1. epoch=epoch+1.
2. Untuk tiap-tiap pasangan elemen yang dilakukan pembelajaran, lakukan feedforward.
3. Hitung kesalahan (error) antara hasil klasifikasi dan label kelas, dan gunakan informasi ini untuk mencari gradien kesalahan terhadap parameter-parameter yang ada.
4. Lakukan update bobot dengan informasi gradien yang didapatkan sebelumnya dengan Persamaan (3.6).

$$w(t+1) = w(t) - \eta \nabla E(w(t)) \qquad (3.6)$$

Pada persamaan tersebut, w(t+1) adalah bobot yang baru, w(t) adalah bobot lama, η adalah *learning rate* dan E(w (t) adalah gradien dari ∇ *error*.

3.5.1. *Gradient Descent Learning*

Pada jaringan *feedforward*, pelatihan dilakukan dalam rangka melakukan pengaturan bobot, sehingga pada akhir pelatihan diperoleh bobot-bobot yang terbaik. Selama proses pelatihan, bobot-bobot diatur secara iteratif untuk meminimalkan kesalahan fungsi jaringan. Fungsi kesalahan yang sering digunakan adalah minimum *square error* (MSE), yang mengambil rata-rata kuadrat kesalahan yang terjadi antara *output* jaringan dan target. Sebagian besar algoritma pelatihan untuk jaringan *feedforward* menggunakan gradien dari fungsi kesalahan untuk menentukan perubahan bobot dalam rangka meminimalkan fungsi kesalahan tersebut. Gradien ini ditentukan dengan teknik *backpropagation*. Prinsip dasar dari algoritma *backpropagation* sederhana adalah memperbaiki bobot-bobot jaringan dengan arah yang membuat fungsi kinerja menjadi turun dengan cepat. Dengan kata lain, algoritma *backpropagation* menggerakkan bobot dengan arah gradien negatif.

Pada Persamaan (3.6), E(w (t)) adalah gradien dari fungsi kesalahan ∇ pada waktu t. Dalam hal ini, fungsi kesalahan didefinisikan untuk satu kesatuan data latih, sehingga masing-masing langkah membutuhkan seluruh data latih untuk diproses untuk mencari E. Teknik yang menggunakan seluruh data set ∇ untuk melakukan perubahan terhadap bobot disebut dengan *batch mode gradient descent* atau *gradient descent*.

3.6. *Dropout*

Salah satu teknik untuk mengurangi *overfitting* adalah menggunakan *dropout. Overfitting* merupakan kondisi dimana sistem jaringan syaraf tiruan mampu belajar dengan baik dengan data pelatihan, namun tidak bisa menggeneralisasi pada data tes (Rajagede, 2016). Dengan menggunakan dropout, maka secara acak akan dipilih beberapa neuron lalu dihapus sementara ketika sedang proses pelatihan. Proses ini dilakukan secara berulang, setelah pembaruan nilai bobot dan bias, neuron yang tadinya dihapus dikembalikan lagi, lalu akan dipilih lagi secara acak untuk dihapus dan dilakukan pelatihan lagi, dan seterusnya. Pada saat pelatihan peluang sebuah neuron dihapus adalah p. Ketika proses evaluasi atau tes, semua neuron tidak ada yang dihapus. Namun, karena saat pelatihan bobot

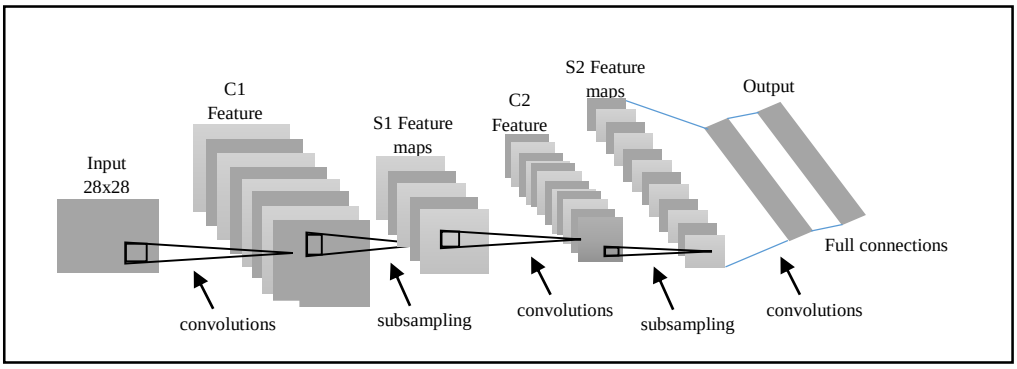
dan bias pada sistem sudah terlatih dengan hilangnya p neuron, maka saat proses evaluasi, bobot yang keluar dari sebuah neuron harus dikali dengan p. Ini untuk memastikan output saat proses tes sama dengan saat proses pelatihan, dimana ada kemungkinan p neuron hilang (Srivastava, dkk. 2014).

3.7. *Convolutional Neural Network* (CNN)

Convolutional Neural Network atau dikenal juga dengan sebutan *ConvNets* adalah sebuah metode untuk memproses data dalam bentuk beberapa array, contohnya yaitu gambar berwarna yang terdiri dari tiga array 2D yang mengandung intensitas piksel dalam tiga jenis warna. *Convolutional Neural Networks* (ConvNets) merupakan penerapan dari *Artificial Neural Networks* (ANN) yang lebih istimewa dan saat ini diklaim sebagai model terbaik untuk memecahkan masalah pengenalan objek.

Secara teknis, *convolutional network* memiliki arsitektur yang dapat dilatih dan terdiri dari beberapa tahap. Masukan dan keluaran dari masing-masing tahap adalah beberapa array yang disebut *feature map* atau peta fitur. Contohnya untuk citra *greyscale*, *input* atau masukan adalah berupa matriks dua dimensi. *Output* dari masing-masing tahap adalah *feature map* hasil pengolahan dari semua lokasi pada citra masukan. Masing-masing tahap terdiri dari tiga lapisan yaitu konvolusi, aktivasi dan *pooling*.

Secara umum, arsitektur dari sebuah *convolution network* ditunjukkan pada **Gambar 3.8** (LeCun, dkk., 2010) sebagaimana digunakan oleh LeCun. Pada gambar tersebut, *Input* dari *CNN* adalah berupa citra dengan ukuran tertentu. Tahap pertama dalam *CNN* adalah tahap konvolusi. Konvolusi dilakukan dengan menggunakan kernel dengan ukuran tertentu. Jumlah kernel yang dipakai tergantung dari jumlah fitur yang dihasilkan. *Output* dari tahap ini kemudian dikenakan fungsi aktivasi, yang bisa berupa fungsi tanh atau *Rectifier Linear Unit* (ReLU). *Output* dari fungsi aktivasi kemudian melalui proses *sampling* atau *pooling*. *Output* dari proses *pooling* adalah citra yang telah berkurang ukurannya, tergantung dari *pooling mask* yang dipakai.



Gambar 3.8 Struktur Convolutional Neural Network

3.7.1. *Convolutions Layer*

Neuron ke (i,j) pada hidden layer , memiliki nilai keaktifan y yang dihitung sesuai dengan Persamaan (3.7), dimana nilai (m,n) pada persamaan tersebut menunjukkan ukuran *local receptive fields*/kernel.

$$y_{i,j} = \sigma(b + (w * x)) \tag{3.7}$$

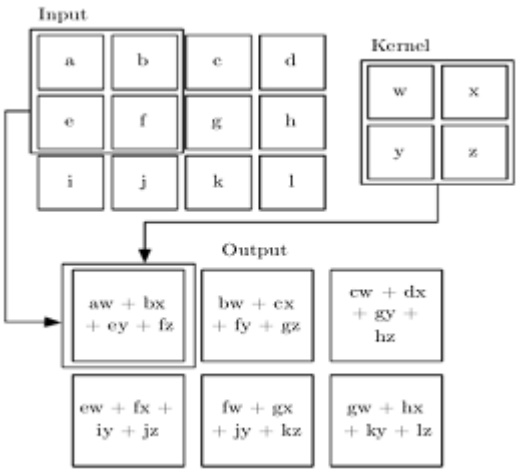
$$(W * x)_{i,j} = \sum_{k=0}^{m} \sum_{l=0}^{n} w_{k,l} x_{i+j,k+l} \tag{3.8}$$

Perkalian antara input dengan kernel di atas (Persamaan (3.8)) yang biasa disebut konvolusi. Namun, menurut Goodfellow, dkk. (2016) konvolusi dilakukan pada kernel yang terbalik, seperti pada Persamaan (3.9). Sedangakan jika kernel tidak dibalik maka fungsi itu disebut cross-correlation. Walaupun begitu, banyak kode pustaka *machine learning* yang menggunakan rumus cross-correlation dan menyebutnya sebagai rumus konvolusi.

$$(W * x)_{i,j} = \sum_{k=0}^{m} \sum_{l=0}^{n} w_{k,l} x_{i-j,k-l} \tag{3.9}$$

Ukuran citra hasil konvolusi berkurang dibandingkan dengan citra awal dan dapat dinyatakan dengan Persamaan (3.10). Dalam hal ini jika citra dengan ukuran 28x28 dikenai konvolusi dengan ukuran kernel 3x3 maka ukuran akhir menjadi 28-3+1 x 28-3+1 =26x26.

$$ukuran_{hasilkonvolusi} = ukuran_{awal} - filtersize + 1 \tag{3.10}$$

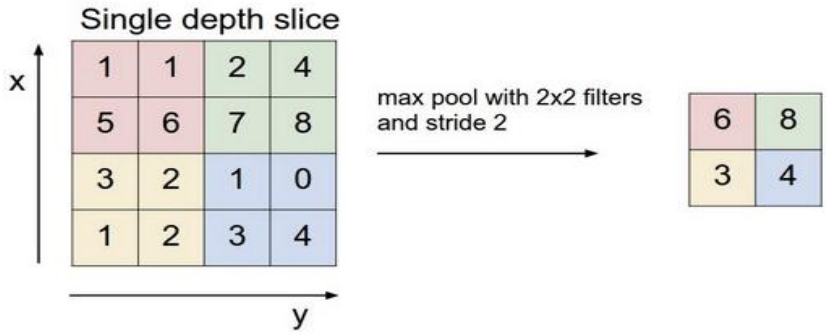


Gambar 3.9 Proses konvolusi pada input array 2D dengan bobot 2D (Rismiyati, 2016)

Pada **Gambar 3.9** diatas menunjukkan ilustrasi proses konvolusi pada citra, yang merupakan array dua dimensi I, dengan bobot K (dua dimensi). Pada gambar tersebut, citra berukuran 4x3 dikonvolusi dengan menggunakan kernel berukuran 2x2. Citra yang dihasilkan adalah berukuran 3x2. Elemen pertama pada citra hasil konvolusi adalah merupakan jumlah dari perkalian bobot kernel dengan nilai citra yang bersangkutan.

3.7.2. *Pooling Layer*

Pooling atau *subsampling* adalah pengurangan ukuran matriks dengan menggunakan operasi *pooling*. Terdapat dua macam *pooling* yang sering dipakai yaitu *average pooling* dan *max-pooling*. Dalam *average pooling*, nilai yang diambil adalah nilai rata-rata, sementara pada *max pooling*, nilai yang diambil adalah nilai maksimal. **Gambar 3.10** menunjukkan operasi *max-pooling.*

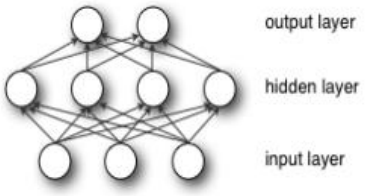


Gambar 3.10 Proses Max-Pooling (Li dan Karpathy, 2015)

Output dari proses *pooling* adalah matriks dengan dimensi yang lebih kecil dibanding dengan matrik awal. Proses konvolusi dan *pooling* dilakukan beberapa kali sehingga didapatkan peta fitur dengan ukuran yang dikehendaki. Peta fitur tersebut akan menjadi input bagi *fully connected neural network.*

3.7.3. *Multi Layer Neural Network*

Hasil dari *layer* konvolusi (konvolusi, aktivasi, pooling) menjadi *input* bagi *multi-layer perceptron neural network*. **Gambar 3.17** menunjukkan *neural network* dengan satu lapisan tersembunyi (LISA Lab, 2015). Pada gambar tersebut, *neural network* dengan satu lapisan tersembunyi pada gambar tersebut bisa dinyatakan sebagai fungsi $f : R^D \rightarrow R^L$, dimana D adalah ukuran dari vektor input x, dan L adalah ukuran dari output vektor $f(x)$. Fungsi tersebut dapat ditulis dalam notasi matriks sebagai $f(x) = G(b^{(2)} + W^{(2)} * s(b^{(1)} + W^{(1)} * x))$,dimana $b(1)$ dan $b(2)$ adalah nilai bias, $W(1)$ dan $W(2)$ adalah matriks bobot, serta G dan s adalah fungsi aktivasi.



Gambar 3.11 Neural Network Satu Lapisan Tersembunyi

Vektor h(x) yang merepresentasikan fungsi pada lapisan tersembunyi bisa dinyatakan sebagai $h(x) = s(b^{(1)} + W^{(1)} x)$, dimana $W^{(1)} \in R^{DxD_h}$ adalah matriks bobot yang menghubungkan vektor input ke lapisan tersembunyi. Sementara fungsi aktivasi s bisa berupa fungsi tanh atau fungsi sigmoid.

Vektor output dapat dinyatakan sebagai $o(x) = G\left(b^{(2)} + W^{(2)} h(x)\right)$. Dalam *CNN, multi layer perceptron* ini diterapkan pada *fully connected layer*. Klasifikasi output dilakukan dengan menggunakan fungsi *softmax.*

3.7.4. *Softmax Layer*

Fungsi *softmax* digunakan untuk metode klasifikasi dengan jumlah kelas yang banyak, seperti regresi logistik multinomial, analisis diskriminan linear *multiclass*, *Naive Bayes Classifier*, dan *Artificial Neural Network* (ANN). *Softmax* adalah sebuah fungsi yang mengubah K-dimensi vektor 'x' yang berupa nilai sebenarnya menjadi vektor dengan bentuk yang sama namun dengan nilai dalam rentang 0-1, yang jumlahnya 1. Fungsi *softmax* digunakan dalam *layer* yang terdapat pada *neural network* dan biasanya terdapat pada *layer* terakhir untuk mendapatkan *output.* Tidak jauh berbeda dengan neuron pada umumnya *softmax* neuron menerima input lalu melakukan pembobotan dan penambahan bias. Tetapi setelah itu neuron pada *softmax layer* tidak menerapkan fungsi aktivasi melainkan menggunakan fungsi *softmax.*

Jika diketahui p adalah input berbobot yang diterima oleh neuron pada *softmax layer* maka aktivasi yi untuk neuron ke-i adalah :

$$y_i = \frac{e^{p_i}}{\sum_{j=1}^{i} e^{p_j}} \tag{3.11}$$

Di mana bagian penyebut pada Persamaan (3.11) merupakan total nilai masing-masing neuron pada output layer. Sehingga bisa dikatakan pada *softmax layer*, *output* merupakan distribusi probabilitas untuk setiap kelas. Penyebutnya memastikan bahwa output ke-i berjumlah mendekati 1. Dengan menggunakan *softmax* kita bisa menafsirkan *output* jaringan y_i^n sebagai perkiraan $p\ (i|x^n)$.

3.8. *Mxnet*

MXNet adalah sebuah kerangka kerja *deep learning open source* yang memungkinkan untuk menentukan, melatih, dan menyebarkan *deep neural network* pada beragam *platform*, mulai dari *cloud infrastructure* hingga *mobile device*s. *Mxnet* merupakan sebuah *library deep learning* yang fleksibel dan efisien. *MXNet* mendukung pemrograman imperatif dan simbolis, yang memudahkan melakukan pemrograman penting dalam memulai *deep learning*. *MXNet* dirancang untuk didistribusikan pada *cloud infrastructrure* yang dinamis, dengan menggunakan server parameter terdistribusi (Li Mu, dkk.) dan dapat mencapai skala yang hampir linier dengan beberapa perangkat *GPU* ataupun *CPU*. *MXNet* mendukung

pemrograman dalam berbagai bahasa termasuk *Python, R, Scala, Julia,* dan *Perl*. Adapun kelebihan yang dimiliki *Mxnet* seperti hanya membutuhkan sedikit *code* dalam proses *training CNN* namun dapat menghasilkan *output* yang representative, membutuhkan *space memory* yang tidak banyak, dan lain-lain.

3.9. *Fatkun Batch* Unduh Gambar

Salah satu teknik dalam pengambilan data adalah dengan melakukan *scraping data.* Cara paling sederhana dalam melakukan *scraping* untuk data gambar adalah dengan mengunduhnya di *internet.* Dari sekian banyak *extension* yang disediakan dalam *web browser,* terdapat satu *extension* yang dapat digunakan sebagai media untuk mengunduh gambar di halaman *internet* dari berbagai sumber yaitu *Fatkun Batch* Unduh Gambar. *Fatkun Batch* Unduh Gambar merupakan salah satu ekstensi yang ditawarkan oleh *Google Chrome* guna memudahkan masyarakat dalam mengunduh gambar yang dibutuhkan. Kemudahan yang diberikan oleh *Fatkun Batch* Unduh Gambar adalah dapat digunakan secara *offline,* tidak berbayar, dapat menyaring resolusi atau *link* yang diinginkan, dan lain-lain.

BAB IV
METODOLOGI PENELITIAN

4.1. Populasi dan Sampel

Populasi dalam penelitian ini yaitu seluruh citra wayang Punakawan yang tersebar dari berbagai sumber dalam *internet.* Sedangkan untuk sampel dalam penelitian ini yaitu 1200 citra atau gambar wayang Punakawan dengan jumlah masing-masing untuk keempat jenis wayang Punakawan sebanyak 300 citra.

4.2. Metode Pengambilan Data

Data dalam penelitian ini diambil dengam menggunakan metode *scraping* dari berbagai sumber di *internet* dan foto secara manual menggunakan *smartphone.*

4.3. Variabel Penelitian

Berikut ini merupakan variabel penelitian beserta penjelasan dari masing-masing variabel :

Tabel 4.1 Variabel Penelitian

Citra Wayang	Variabel	Definisi Operasional Variabel
	Wayang Gareng	Wayang Gareng merupakan wayang Punakawan yang memiliki ciri tubuh dengan hidung besar dengan tangan yang kurang sempurna serta tubuh ukuran sedang.
	Wayang Semar	Wayang semar merupakan wayang Punakawan dengan ciri tubuh paling gemuk diantara jenis Punakawan lainnya. Telunjuknya juga selalu terlihat menunjuk.

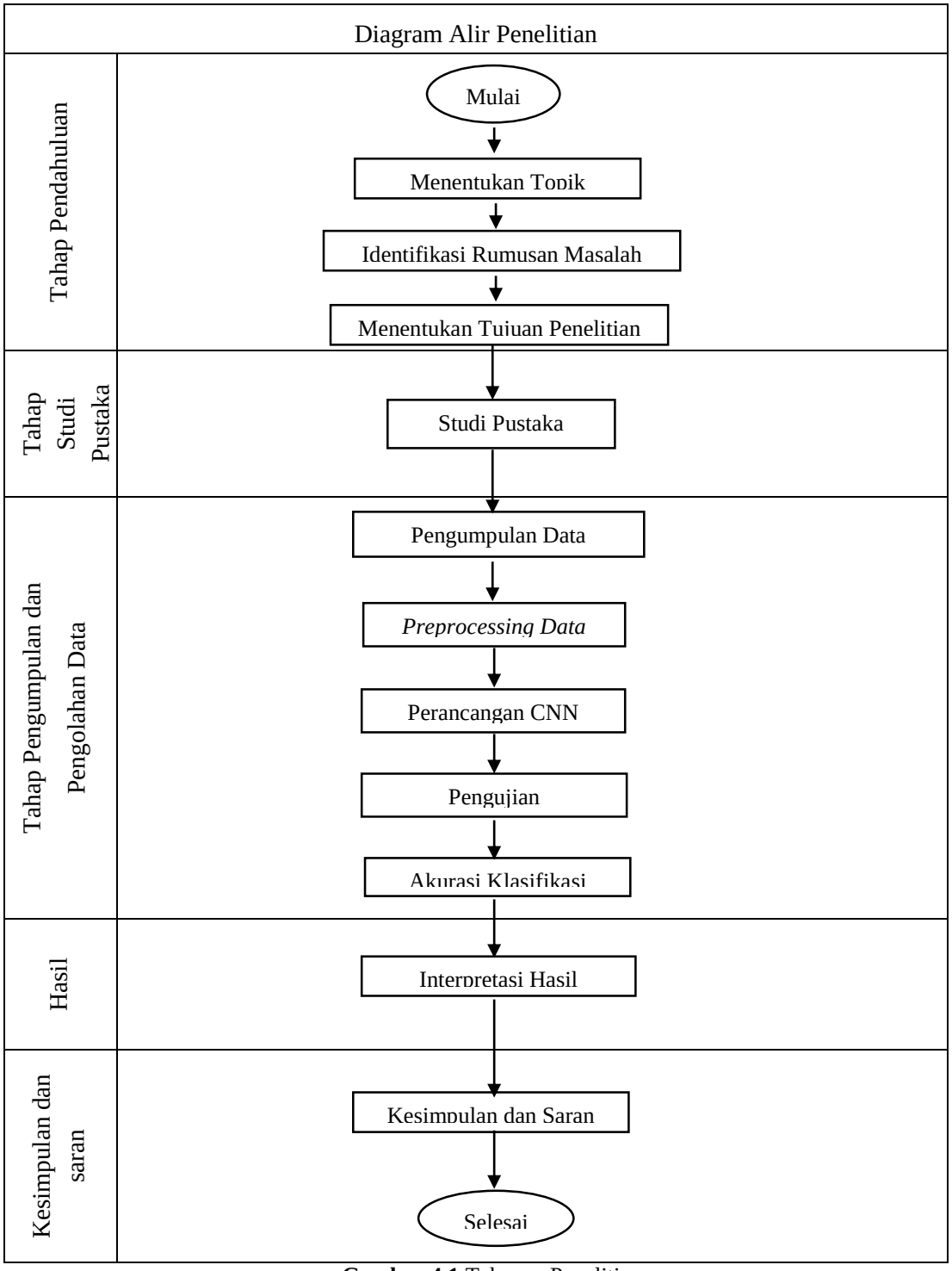
	Wayang Petruk	Wayang Petruk merupakan wayang Punakawan dengan ciri tubuh kurus dan memiliki hidung panjang.
	Wayang Bagong	Wayang Bagong merupakan wayang Punakawan dengan ciri tubuh hampir mirip dengan semar, namun lebih kecil.

4.4. Metode Analisis Data

Metode analisis yang digunakan dalam penelitian ini adalah metode *deep learning* yaitu *Convolutional Neural Network* (CNN) dengan dukungan *Mxnet*. Adapun cara kerja yang dilakukan adalah dengan mengenali objek atau citra sebagai *input* dan *output* yang diharapkan adalah tingkat akurasi pengenalan objek tersebut.

4.5. Tahapan Penelitian

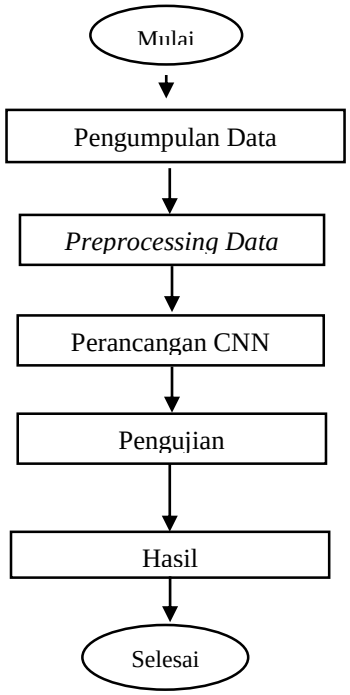
Berikut ini merupakan tahapan penelitian yang dilakukan dalam penelitian ini :



Gambar 4.1 Tahapan Penelitian

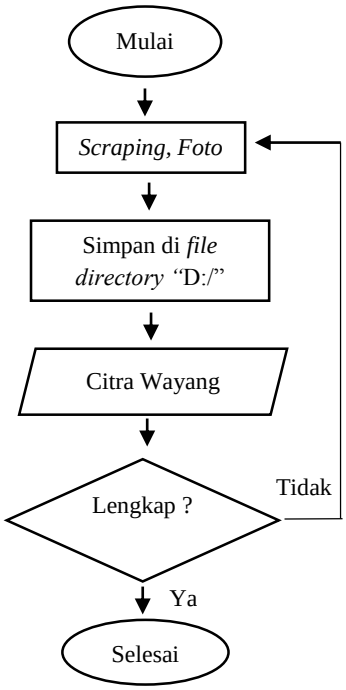
BAB V
HASIL DAN PEMBAHASAN

Dalam bab ini akan dijelaskan rangkaian proses klasifikasi wayang Punakawan menggunakan *CNN* yang didukung oleh kerangka kera *deep learning* yaitu *Mxnet.* Berikut ini merupakan tahapan mulai dari pemrosesan *dataset* hingga dapat dihasilkan hasil klasifikasi dan akurasi model yang akan menjadi bahan bahasan dalam bab ini.



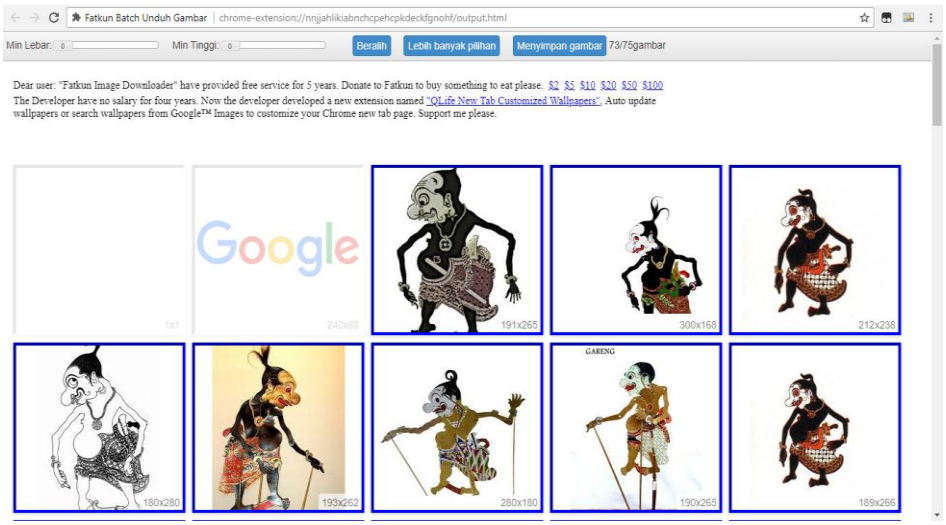
5.1. Pengumpulan Data

Proses pengumpulan data pada penelitian ini adalah menggunakan metode *scraping* dan foto manual oleh penulis. Adapun metode *scraping* yang dilakukan adalah dengan cara *download* gambar terkait dengan objek dalam penelitian ini yaitu gambar keempat jenis wayang Punakawan. Perhatikan **Gambar 5.1** yang merupakan langkah dalam pengumpulan data untuk penelitian.



Gambar 5.1 Proses Pengumpulan Data

Metode *scraping* yang dilakukan adalah dengan cara *download* gambar keempat jenis wayang Punakawan. Untuk mendapatkan gambar tersebut tanpa harus mengunduhnya satu persatu, maka diperlukan sebuah ekstensi dari *browser* yang penulis gunakan yaitu *Chrome*. Salah satu ekstensi yang dapat digunakan yaitu "Fatkun Batch Unduh Gambar". Berikut ini adalah contoh tampilan halaman "Fatkun Batch Unduh Gambar" ketika ingin mengunduh gambar 'Gareng' yang ada di hasil pencarian Google.



Gambar 5.2 Unduh Gambar

Seluruh gambar yang ditampilkan oleh hasil pencarian ‘Gareng’ oleh Google akan tampil seperti pada **Gambar 5.2** dan kemudian dapat dipilih *item* mana yang tidak perlu untuk diunduh. Secara otomatis gambar-gambar tersebut akan tersimpan ke dalam *file directory* komputer. Gambar-gambar tersebut masih harus dilakukan pembersihan manual, seperti menghapus gambar yang tidak berkaitan dengan objek itu sendiri atau mengubah format gambar menjadi ‘jpg’ seluruhnya. Keempat gambar wayang Punakawan tersebut kemudian ditempatkan pada *folder* yang berbeda namun masih pada *file directory* yang sama. Dalam hal ini penulis meletakkan *folder* gambar Punakawan pada *file directory* “D:/” dengan letak masing-masing folder wayang diberi nama “D:/Gareng”, “D:/Semar”,” D:/Petruk”, “D:/Bagong”. Seluruh hasil pengumpulan gambar baik dari hasil *download* maupun foto manual digabungkan masing-masing sesuai dengan keempat objek yang ada. Peneliti menentukan untuk menggunakan 1200 data citra wayang Punakawan dengan jumlah masing-masing jenisnya yaitu 300 wayang. Hasil pengumpulan data dapat dilihat pada Lampiran 2.

5.2. *Preprocessing Data*

Sebagian atau lebih dari data yang tersedia pastinya memiliki ukuran yang berbeda-beda. Oleh karena itu, *preprocessing* citra atau gambar dilakukan untuk menyiapkan citra kemudian diproses lebih lanjut, baik untuk kebutuhan ekstraksi fitur maupun kebutuhan klasifikasi. Dalam hal ini terdapat dua langkah *preprocessing* yang dapat dilakukan dalam proses klasifikasi menggunakan metode *CNN*, yaitu dengan mengubah ukuran citra menjadi 46x46. Kemudian dilakukan konversi warna citra menjadi *greyscale* agar sistem mudah mengenali.

5.2.1. *Resize*

Ukuran citra biasanya ditunjukkan oleh simbol h = *height* dan w = *widtht* dalam dua dimensi (h x w). Pada penelitian-peneltian sebelumnya telah banyak yang menggunakan ukuran 28x28 atau 32x32, namun pada penelitian ini penulis ingin membuat program dapat bekerja dengan citra ukuran 46x46. Adapun pemilihan ukuran ini berdasarkan penelitian yang pernah dilakukan oleh Zufar dan Setiyono (2016) tentang *CNN* untuk pengenalan wajah secara *real time.*

Sebelumnya peneliti juga telah mencoba untuk menggunakan citra ukuran 32x32, akan tetapi akurasi pada *test data* yang diperoleh hanya sebesar 52.5 %. Hal lain yang mendasari juga ialah kemampuan perangkat komputer dalam menjalankan program yang dibuat. Semakin besar ukuran yang digunakan semakin mudah juga program mengenali citra yang ada.

5.2.2. Konversi Data Menjadi *Greyscale*

Konversi citra warna ke citra *greyscale* dilakukan berdasarkan Persamaan (3.1). Persamaan ini dipilih karena lebih banyak dipakai pada kebutuhan *computer vision* (Kanan dan Cottrell, 2012). Pada proses pengubahan citra diketahui bahwa proses pertama yang dilakukan adalah dengan mengambil citra warna sebagai masukan. Dari citra warna yang diperoleh, tinggi dan lebar dari citra dicari untuk kebutuhan pembuatan citra baru. Kemudian dilakukan pemisahan terhadap nilai dari masing-masing komponen *red, green*, dan *blue* (rgb) dari citra warna tersebut. Selanjutnya, satu citra baru dibuat untuk menampung citra hasil perubahan model warna. Proses perulangan dilakukan selama height<h, dan weight<w untuk mendapatkan nilai *greyscale* pada masing-masing *pixel* pada citra baru. Proses ini menghasilkan citra *greyscale.*

5.2.3. Penerapan *Preprocessing*

Preprocessing dilakukan dilakukan dengan melakukan segmentasi dan perubahan ukuran pada citra. Program untuk melakukan *preprocessing* ditunjukkan pada **Gambar 5.3**. Namun sebelum melakukan proses *preprocessing*, harus dilakukan pendefinisian data. Salah satu *package* dalam *R* yang mendukung proses *preprocessing* adalah *package* 'EBImage'. *Package* ini memiliki tujuan fungsional untuk *image processing* dan *image analysis*. Pada **Gambar 5.3** dapat dilihat bahwa *package* ini harus dijalankan pada awal program. Pada baris ke-4 merupakan tahap mengatur *working directory* letak *file* dari citra pertama yang diproses yaitu Gareng. Keempat jenis wayang masing-masing diberi label dengan urutan 0-3 (empat kelas).

```
1
2  library(EBImage)
3
4  setwd("D://gareng/")
5
6  labelgareng <- 0
7
8  df <- data.frame()
9
10 gareng <- list.files()
11
12 w<-46
13
14 h<-46
15
16 gambar_size<-46*46
```

Gambar 5.3 Pendefinisian data

Pada baris ke-8 dibuat *data frame* dari *dataset* Gareng dengan nama 'df'. Seluruh data kemudian diseragamkan ukurannya menjadi 46x46 dengan rincian *width* (w) = 46, *height* (h) = 46.

```
18
19 for (i in 1:length(gareng))
20
21 {
22   semuagareng <- readImage(gareng[i])
23
24   semuagareng_resized <- resize(semuagareng, w=w, h=h)
25
26   semuagareng_gray <- channel(semuagareng_resized, "gray")
27
28   semuagareng_matrix <- semuagareng_gray@.Data
29
30   semuagareng_vector <- as.vector(t(semuagareng_matrix))
31
32   gabunggareng = c(labelgareng,semuagareng_vector)
33
34   df<-rbind(df,gabunggareng)
35
36   write.csv(df, "gabunggareng.csv", row.names = FALSE)
37
38   names(df) <- c('label', paste('pixel', c(1:gambar_size)))
39
40 }
41
42 display(semuagareng_gray)
```

Gambar 5.4 Proses preprocessing

Setelah pendefinisian data, kemudian dimulai dari membuat tahap fungsi perulangan untuk mengubah ukuran citra dan konversi citra warna menjadi *greyscale.* Perintah mengubah ukuran atau *resize* ada di baris ke-24 dengan nama hasil *resize*nya adalah `semuagareng_resized`. Sedangkan perintah untuk konversi citra menjadi *greyscale* terdapat dalam baris selanjutnya yang dinamai dengan `semuagareng_gray`. Kemudian, citra yang telah berukuran 46x46 dan berubah citra warnanya menjadi *greyscale* diubah dalam bentuk matriks lalu diubah lagi

menjadi bentuk vektor. Tahap ini ada dalam baris ke-28 dan 30. Untuk membuat sebuah *file dataset* yang dapat diolah menggunakan *Mxnet,* hasil *preprocessing* sebelumnya disimpan dalam bentuk *file csv*. Format ini salah satu format yang sesuai dengan data angka dan dapat dibaca oleh *Mxnet.* Nilai-nilai piksel yang ada dalam *file* tersebut berada rentang nilai 0-1. Nilai piksel akan semakin mendekati angka satu apabila piksel yang dibaca berada tepat di pola citra. Dalam *file* tersebut, nilai label masing-masing kelas citra digabungkan dengan nilai piksel citra yang telah diubah dalam bentuk vektor. Jadi, dalam satu *file csv* misalnya untuk citra Gareng, terdapat satu kolom label '0' sebanyak jumlah datanya yaitu sebanyak 300 baris. Dan ada pula kolom piksel untuk setiap citra sebanyak 2116 kolom. Proses ini menjelaskan perintah pada baris ke-32 hingga ke-38.

Proses *preprocessing* diatas diulangi hingga sampai pada label ke-3 untuk ketiga citra lainnya yaitu Semar, Petruk, dan Bagong. Adapun dalam program ini Semar, Petruk, dan Bagong diberi label masing-masing secara berurutan adalah 1,2,3. Untuk melihat contoh hasil citra yang telah di*resize* dan menjadi *greyscale*, dapat menuliskan perintah `display(semuagareng_gray)`.

5.2.4. Persiapan *Dataset*

Citra yang telah melewati proses *resize* dan konversi *greyscale* kemudian dibuat *dataset* dalam bentuk matriks beserta labelnya. *Dataset* ini disiapkan dalam bentuk format yang dapat diterima oleh *library* ***Mxnet***. Adapun format *dataset* yang diterima tersebut adalah *CSV* (Comma Separated Value), label citra ada di kolom pertama sedangkan kolom berikutnya adalah nilai piksel ukuran 46 x 46 sehingga terdapat 2116 kolom piksel. Dalam penelitian ini, digunakan dua data set terpisah yang masing-masing berisi data latih (train) atau data uji (test).

Dalam hal ini peneliti menggunakan tiga macam jumlah data *train* dan *test* yang akan diolah, yaitu sebesar 70%, 80 %, dan 90% dari seluruh data untuk *training.* Sedangkan sisa dari masing-masing jumlah tersebut merupakan data uji. Ketiganya akan dibandingkan nilai akurasinya dengan inisialisasi parameter yang sama.

Peneliti menentukan jumlah untuk data *train* atau latih dengan mempartisi sebesar 70% dari masing-masing objek kemudian dijadikan data *train*. Sehingga total data *train* dari keempat objek adalah sebesar 840 dari total 1200 data. Sedangkan jumlah data *test* adalah dari 30 % sisanya yaitu 360 data *test*. Kemudian untuk 80 % partisi data *train*, terdapat 960 data *train* dan 240 data *test.* Dan partisi 90% data *train*, terdapat 1080 data *train* dan 120 data *test.*

Masukan bagi proses klasifikasi menggunakan metode *CNN* adalah berupa gabungan label dan semua nilai piksel dalam bentuk vektor dari citra. Seperti yang telah dijelaskan pada bahasan sebelumnya, masukan bagi pelatihan *CNN* memiliki format *csv* (*comma delimited value)* dan disimpan dalam *directory*. Perhatikan program pembacaan *dataset* pada **Gambar 5.5** dibawah ini.

```
236 garengcsv<- read.csv("D://gareng/gabunggareng.csv")
237
238 semarcsv<- read.csv("D://semar/gabungsemar.csv")
239
240 petrukcsv<- read.csv("D://petruk/gabungpetruk.csv")
241
242 bagongcsv<- read.csv("D://bagong/gabungbagong.csv")
243
244 dim(garengcsv)
245
```

Gambar 5.5 Pembacaan dataset

Misalnya untuk citra petruk, hasil ekstraksi dari gambar menjadi vektor nilai piksel disimpan dalam *file* "D://petruk". Untuk melihat dimensi dari masukan bagi *CNN* menggunakan perintah pada baris ke-244 yaitu dengan menuliskan *syntax* `dim(garengcsv)` sebagai contoh jika ingin melihat dimensi *dataset* wayang Gareng. Kemudian adalah dengan menentukan data *train* dan *test* bagi masukan untuk pelatihan *CNN*. Perhatikan **Gambar 5.6** berikut ini.

```
261 library(caret)
262
263 data_set <- rbind(garengcsv,semarcsv,petrukcsv,bagongcsv)
264
265 train_index <- createDataPartition(data_set$label, p = .90,times = 1)
266
267 train_index <- unlist(train_index)
268
269 train_set <- data_set[train_index,]
270
271 dim(train_set)
272
273 test_set <- data_set[-train_index,] #20%
274
275 dim(test_set)
276
277
278 train_data <- data.matrix(train_set)
279
280 train_x <- t(train_data[, -1])
281
282 dim(train_x)
283
284 train_y <- train_data[,1] #oke
285
286 dim(train_y)
287
288 train_array <- train_x
289
290 dim(train_array)
291
292 dim(train_array) <- c(46,46,1,ncol(train_x))
293
294
295 test_data <- data.matrix(test_set)
296
297 test_x <- t(test_set[,-1])
298
299 dim(test_x)
300
301 test_y <- test_set[,1]
302
303 test_array <- test_x
304
305 dim(test_array)
306
307 dim(test_array) <- c(46,46,1, ncol(test_x))
```

Gambar 5.6 Membuat data latih dan data uji

Dengan menggunakan *package* 'caret' yang ada dalam *R*, memudahkan peneliti dalam memperoleh jumlah data latih (*train*) dan uji (*test*). Dalam penelitian ini digunakan tiga macam jumlah data *train* masukan bagi *CNN*, yaitu 70 %, 80% dan 90% dari keseluruhan data dengan bobot yang sama untuk wayang Gareng, Semar, Petruk, dan Bagong. Pada **Gambar 5.6** diatas menampilkan program dengan jumlah data *train* sebanyak 90% dari keseluruhan data.

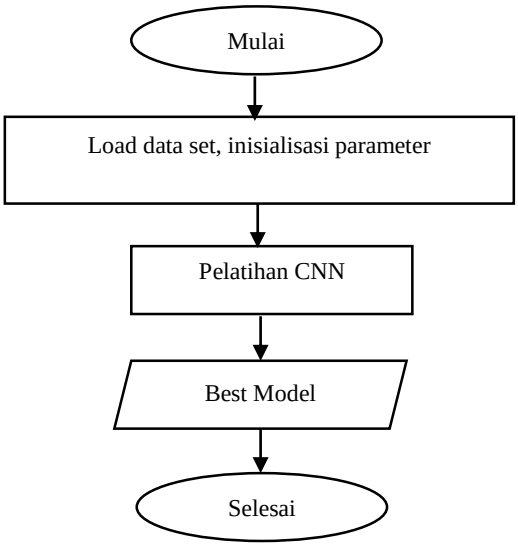
Seluruh *dataset* digabungkan hingga menjadi kesatuan menggunakan perintah `rbind` dan diberi nama `data_set`. Setelah menentukan jumlah data *train* dan *test*, selanjutnya adalah membuat *train set* dan *test set* dari gabungan *dataset* sebelumnya. Pada baris ke-267 diketahui bahwa perintah *unlist* digunakan untuk mengeluarkan data vektor dari daftar 'train_index'. Jadi, *train set* yang diinginkan berupa baris vektor dari *dataset train_index*. Dan untuk *test set* merupakan sisa dari

keseluruhan data dengan *train set* yang telah ditentukan. Hal ini dapat dilihat dari perintah `[-train_index,]` pada baris ke-273.

Dalam *train set* mapun *test set*, terdapat variabel label dan piksel. Dari kedua variabel tersebut akan dipisahkan dan kemudian akan menjadi masukan bagi proses pelatihan *CNN* dan pembuatan model. Dalam **Gambar 5.6**, terdapat dua ekstraksi baru dari *train set* dan *test set*, yaitu data label wayang pada *train set* dan *test set* dan matriks *train set* dan *test set* yang telah di*transpose*. Adapun dari data *train* yang terdiri dari variabel label ditunjukkan dengan nama `train_y`, sedangkan untuk nilai pikselnya ditunjukkan dengan nama `train_array`. Hal ini dapat dilihat dari gambar pada baris ke-284 yang menginginkan hanya kolom pertama dari *train data*. Sedangkan pada baris ke-280 dilihat bahwa hanya menginginkan kolom selain kolom pertama yaitu kolom label. Sama halnya dengan data *test* terdiri dari variabel label yang ditunjukkan dengan nama `test_y`, dan variabel nilai piksel dinamakan dengan `test_array`.

5.3. Rancangan Klasifikasi CNN

Perancangan dari model klasifikasi ditunjukkan pada **Gambar 5.2** dibawah ini.



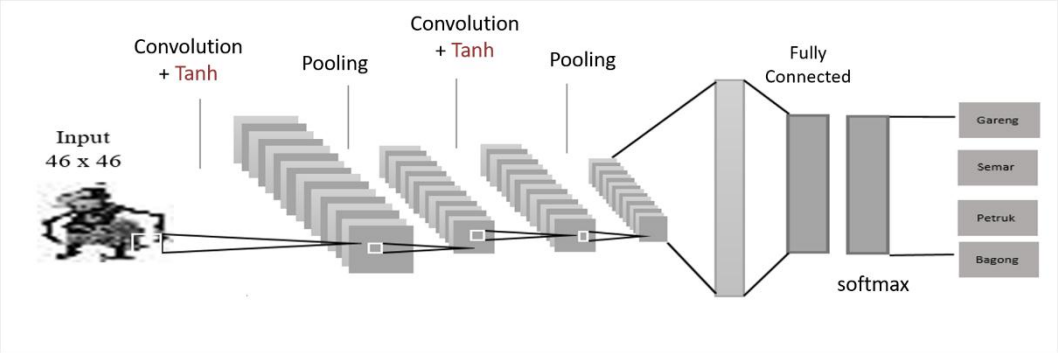
Gambar 5.7 Rancangan model klasifikasi

Berdasarkan **Gambar 5.7** menunjukkan bagaimana proses rancangan model klasifikasi menggunakan *CNN*. Proses pertama yang dilakukan adalah pembacaan

data set. Dari *dataset* tersebut diambil data *train* dan data *test*. Data *train* kemudian digunakan untuk melakukan pelatihan CNN. Proses pelatihan *CNN* dilakukan dengan diawali inisialisasi parameter dan dilakukan berulang-ulang sesuai maksimum perulangan yang ditentukan. Keluaran dari proses ini adalah model terbaik untuk melakukan klasifikasi. Akurasi model tersebut dapat ditentukan dengan melakukan validasi dengan menggunakan data uji.

5.3.1. Rancangan *CNN*

Pelatihan model *CNN* ditunjukkan pada **Gambar 5.9** dibawah ini. Pada arsitektur *CNN* tersebut, citra masukan yang digunakan berukuran 46x46x3. Citra masukan kemudian dilakukan proses konvolusi dan proses *pooling* pada lapisan konvolusi. Dari arsitektur *CNN* tersebut jumlah filter yang digunakan adalah 20 buah. Ukuran filter ini merupakan parameter yang akan dicari nilainya untuk dapat mencapai akurasi yang terbaik.



Gambar 5.8 Arsitektur CNN

Pada proses *pooling* menggunakan ukuran 2x2 dan dengan jumlah pergeseran *mask* sebanyak dua langkah. Jumlah pergeseran ini sering pula disebut dengan *stride.* Penggunaan ukuran ini apabila menggunakan ukuran yang lebih kecil atau ukuran 1x1 berarti sama saja *pooling* tidak dilakukan. Sedangkan apabila menggunakan ukuran yang semakin besar akan lebih banyak menghilangkan informasi dari citra tersebut.

Jumlah lapisan konvolusi adalah salah satu parameter yang menentukan akurasi dari model klasifikasi. Dalam **Gambar 5.8** diatas terlihat bahwa digunakan dua lapisan konvolusi. Tidak ada penentuan standar untuk jumlah lapisan konvolusi

ini, salah satu cara untuk memilih jumlah lapisan yang tepat adalah dengan menemukan kombinasi yang tepat guna menciptakan *dataset* dapat bekerja dalam porsi yang sesuai dalam proses *training model*. Terdapat banyak penelitian terdahulu yang menggunakan dua lapisan konvolusi dalam proses menemukan model terbaik. Contohnya seperti penelitian dari Zufar dan Setiyono (2016) tentang *face recognition* atau dari Yuzhen (2016) tentang *food recognition.* Sebenarnya bisa saja menggunakan lebih banyak lapisan konvolusi, akan tetapi waktu dalam menjalankan *training* akan lebih lama karena parameter yang akan diperbaharui semakin banyak. Penentuan jumlah lapisan konvolusi ini berdasarkan informasi pada konvolusi pertama diteruskan untuk dilanjutkan pada konvolusi kedua. Dalam lapisan konvolusi terdapat *pooling* untuk mengumpulkan informasi baru yang akan dibawa untuk lapisan selanjutnya.

Dalam sebuah pelatihan *CNN*, parameter yang digunakan adalah jumlah *epoch* atau iterasi, jumlah lapisan konvolusi, jumlah filter masing-masing lapisan, ukuran filter, jumlah neuron pada lapisan tersembunyi yang digunakan, serta optimasi. Prinsip kerja proses pelatihan *CNN* sama dengan *Neural Network* (NN) pada umumnya, yaitu dengan dilakukannnya proses *forward propagation* dan *backward propagation* sebanyak *epoch* yang ditentukan atau jika kondisi berhenti awal sudah dipenuhi. Pembaharuan nilai bobot dilakukan setiap kali satu data dibaca, yaitu menggunakan *stochastic gradient descent.* Pada penelitian ini digunakan sebuah algoritma *default* dalam prosedur *stochastic gradient descent* yang bernama *optimizer* 'Adam'. Kata 'Adam' sendiri berasal dari kalimat *adaptive moment estimation.* Jadi, *optimizer* 'Adam' digunakan pada pelatihan *CNN* untuk memperbaharui nilai bobot secara berulang yang berbasis data *training*.

Pada pelatihan *CNN input*nya adalah *dataset* yang telah dibaca pada proses sebelumnya. Inisialisasi parameter dilakukan dengan menentukan jumlah *epoch*, *optimizer adam*, dan ukuran kernel pada masing-masing lapisan konvolusi. Setelah itu dilakukan iterasi sebanyak jumlah *epoch* maksimal dengan melakukan propagasi ke depan (forward propagation) dan pembaharuan nilai bobot.

Pada lapisan konvolusi, matriks masukan dilakukan konvolusi menggunakan filter yang telah ditetapkan ukurannya. Pada gambar tersebut

ditunjukkan masukan dari proses konvolusi yaitu matriks berukuran 3 x tinggi x lebar. Hasil dari proses konvolusi tersebut adalah matriks sejumlah jumlah filter yang ditentukan yaitu sebanyak 20 buah. Hasil konvolusi tersebut kemudian mengalami *pooling* untuk mengurangi ukuran matriks, dan selanjutnya diberikan fungsi aktivasi. Tujuan dilakukannya *pooling* adalah menyederhanakan informasi dari hasil konvolusi dengan cara mengurangi ukuran matriks hasil konvolusi atau dengan kata lain hanya mengambil informasi yang penting saja. Teknik *pooling* yang digunakan adalah *max pooling,* yaitu dengan memilih nilai maksimum di setiap pergerakan kernel pada matriks hasil konvolusi. Adapun fungsi aktivasi yang digunakan setelah proses *pooling* adalah fungsi Tanh (Tangen Hiperbolik). Proses tersebut dilakukan sebanyak lapisan konvolusi yang telah ditetapkan, yakni dua konvolusi.

Hasil dari lapisan konvolusi berfungsi sebagai masukan bagi *MLP* (*Multi Layer Perceptron*). *MLP* terdiri dari lapisan-lapisan yang terdapat dalam lapisan konvolusi, seperti *input layer,* 2 *hidden layer,* dan *output layer.* Masing-masing komponen matriks dari matriks keluaran lapisan konvolusi dianggap sebagai satu nilai masukan bagi *MLP*. Untuk masing-masing neuron pada *hidden layer*, fungsi aktivasi yang digunakan adalah fungsi Tanh.

5.3.2. Penerapan Pelatihan *CNN*

Dalam penelitian ini, *package* ‘mxnet’ dalam *R* digunakan sebagai *package* yang mendukung proses pelatihan *CNN*. *Mxnet* sendiri memuat *command-command* yang berkaitan dengan proses dalam arsitektur *CNN* seperti yang ditunjukkan pada **Gambar 5.8.** Perbedaan *Mxnet* dengan *library* khusus *CNN* lainnya adalah *Mxnet* merupakan *library* sederhana dari *deep learning* yang membutuhkan lebih sedikit *code* dalam pemrogramannya namun keluaran yang dihasilkan sangat representatif dengan tujuan *image classification,* yaitu hasil klasifikasi dan tingkat akurasi model klasifikasi. Terdapat dua lapisan konvolusi dan dua lapisan *hidden layer.* Hal ini ditunjukkan pada **Gambar 5.9** dibawah ini.

```
321 library(mxnet)
322
323 mx_data <- mx.symbol.Variable('data')
324
325 conv_1 <- mx.symbol.Convolution(data = mx_data, kernel = c(3, 3), num_filter = 10)
326
327 tanh_1 <- mx.symbol.Activation(data = conv_1, act_type = "tanh")
328
329 pool_1 <- mx.symbol.Pooling(data = tanh_1, pool_type = "max", kernel = c(2, 2), stride = c(2,2))
330
331
332 conv_2 <- mx.symbol.Convolution(data = pool_1, kernel = c(3,3), num_filter = 20)
333
334 tanh_2 <- mx.symbol.Activation(data = conv_2, act_type = "tanh")
335
336 pool_2 <- mx.symbol.Pooling(data = tanh_2, pool_type = "max", kernel = c(2, 2),stride = c(2, 2))
337
338
339 flat <- mx.symbol.Flatten(data = pool_2)
340
341 drop1 = mx.symbol.Dropout(data=flat, p=0.1)
342
343 fcl_1 <- mx.symbol.FullyConnected(data = drop1, num_hidden = 10)
344
345 tanh_3 <- mx.symbol.Activation(data = fcl_1, act_type = "tanh")
346
347
348 drop1 = mx.symbol.Dropout(tanh_3, p=0.1)
349
350 fcl_2 <- mx.symbol.FullyConnected(data = drop1, num_hidden = 4)
351
352
353 NN_model <- mx.symbol.SoftmaxOutput(data = fcl_2)
354
```

Gambar 5.9 Pelatihan CNN

Proses pelatihan *CNN* pada penelitian ini menggunakan dua lapisan konvolusi yang terdapat di dalam kotak biru pada **Gambar 5.9**. Konvolusi pertama yang diberi nama `conv_1` menggunakan ukuran kernel sebesar 3x3 dan jumlah filter sebanyak 10. Kemudian diaktifasi menggunakan fungsi 'tanh' atau tangen hiperbolik seperti yang ditunjukkan **Gambar 5.9** pada baris ke-327. Lalu dilakukan proses *pooling* untuk memecah hasil konvolusi dalam matriks yang lebih sederhana atau mengurangi matriks hasil konvolusi. *Pooling* pertama menggunakan ukuran kernel 2x2 dan jumlah *stride* yang digunakan sebesar 2x2. Proses *pooling* dilakukan dengan cara mengambil nilai maksimal dari hasil konvolusi dengan *output* berupa matriks berdimensi yang lebih kecil dibanding dengan matrik awal. Hal ini dapat dilihat pada **Gambar 5.9** baris ke-329 bahwa *pool type* yang digunakan adalah 'max'. Pada konvolusi kedua, data yang digunakan adalah hasil *pooling* pada konvolusi pertama, hal ini dapat dilihat dalam *syntax* pada baris ke-332. Pada lapisan kedua, digunakan jumlah filter yang berbeda yaitu 20 filter. Kemudian diaktifasi kembali menggunakan fungsi 'tanh' dan dilakukan *pooling* lagi menggunakan ukuran yang sama.

Untuk memperoleh model yang dapat menghasilkan akurasi terbaik, dalam penelitian ini peneliti melakukan beberapa percobaan menggunakan ukuran kernel

dan jumlah filter yang berbeda. Dan program yang ditampilkan dalam penelitian ini adalah program yang menghasilkan akurasi paling maksimal. Berikut ini merupakan tabel yang memuat inisialisasi perbandingan ukuran kernel dan jumlah filter pada lapisan konvolusi dimana hasilnya akan dibahas pada subbab hasil.

Tabel 5.1 Percobaan parameter pelatihan CNN

Kemungkinan	Konvolusi	Ukuran Kernel	Jumlah Filter
1	Conv_1	5x5	10
	Conv_2	3x3	10
2	Conv_1	5x5	10
	Conv_2	3x3	20
3	Conv_1	5x5	20
	Conv_2	3x3	20
4	Conv_1	3x3	10
	Conv_2	3x3	10
5	Conv_1	3x3	10
	Conv_2	3x3	20
6	Conv_1	3x3	20
	Conv_2	3x3	20

Selanjutnya adalah *hidden layer* yang merupakan *fully connected kayer*. Lapisan inilah yang membedakan *CNN* dengan metode *deep learning* lainnya. Dalam program ini digunakan dua *fully connected layer* seperti yang ditunjukkan dalam kotak hijau pada **Gambar 5.9** tersebut. Langkah berikutnya adalah membuat *flatten* dari hasil *pooling* kedua pada lapisan konvolusi. Data *flat* kemudian diregulasi untuk proses *dropout* pertama dengan menggunakan probabilitas sebesar 10%. Data *dropout* pertama yang bernama `drop_1` dijadikan masukan bagi proses dalam *fully connected layer* pertama. Dalam lapisan ini digunakan jumlah neuron sebanyak 10 (num_hidden=10). Kemudian proses ini diaktifasi menggunakan fungsi ‘tanh’. Dilanjutkan pada *fully connected layer* berikutnya yang diawali dengan proses *dropout* hasil aktivasi lapisan sebelumnya dengan menggunakan probabilitas sebesar 10%. Dan hasil *dropout* digunakan bagi proses *fully connected*

layer dengan jumlah neuron sebanyak 4 sesuai dengan jumlah wayang Punakawan yang akan diklasifikasikan pada penelitian ini. Proses terakhir adalah mengatur aktivasi bagi fungsi *softmax* untuk melakukan prediksi nilai probabilitas. *Softmax function* digunakan pada lapisan *output* dan memiliki tugas untuk mengklasifikasikan.

```
369 mx.set.seed(100)
370
371 device <- mx.cpu()
372
373 model <- mx.model.FeedForward.create(NN_model, X = train_array, y = train_y,
374
375                                      ctx = device,
376
377                                      num.round = 100,
378
379                                      array.batch.size = 250,
380
381                                      optimizer = "adam",
382
383                                      eval.metric = mx.metric.accuracy,
384
385                                      initializer = mx.init.Xavier(factor_type = "in",
386
387                                                   rnd_type = "gaussian",magnitude = 2),
388
389                                      array.layout = "auto")
390
```

Gambar 5.10 Pembuatan model

Berdasarkan pada **Gambar 5.10**, didefinisikan bahwa perangkat yang digunakan adalah *CPU*. Kemudian dibuat model dengan menginisialisasikan parameter jumlah iterasi, *batch size,* dan menggunakan *optimzer Adam.* Sebelumnya, model yang akan dibentuk menggunakan masukan dari data *train* dan data *test* yang telah dijelaskan sebelumnya. Keduanya di inisialkan dengan `x` sebagai `train_array`, dan `y` sebagai `train_y`. Jumlah iterasi yang digunakan adalah sebanyak 100 dengan *batch size* sebesar 250. Jumlah iterasi yang ditampilkan dalam **Gambar 5.11** merupakan nilai iterasi optimal yang telah dicoba oleh peneliti. Adapun dalam penelitian ini, peneliti akan menampilkan perbandingan jumlah iterasi terhadap hasil akurasi model yang akan ditampilkan pada subbab hasil.

Jika dalam penelitian lain menggunakan *learning rate* sebagai parameter yang digunakan, dalam program ini peneliti menggunakan *optimizer Adam* sebagai algoritma *default* dalam hal optimalisasi model. Setelah mendefinisikan model, maka model tersebut akan diuji tingkat akurasi dalam mengklasifikasikan citra

wayang Punakawan. Berikut ini merupakan tahapan yang dilakukan dalam memprediksi model untuk data *train* data *test.*

```
395 predict_probs <- predict(model, train_array)
396
397 predict_probs
398
399 predicted_labels <- max.col(t(predict_probs))-1
400
401 table(train_data[,1],predicted_labels)
402
403 sum(diag(table(train_y, predicted_labels)))/1080
404
405
406
407 predict_probs <- predict(model, test_array)
408
409 predict_probs
410
411 predicted_labels <- max.col(t(predict_probs))-1
412
413 table(test_data[, 1], predicted_labels)
414
415 sum(diag(table(test_y, predicted_labels)))/120
```

Gambar 5.11 Prediksi model

Dalam **Gambar 5.11**, pada baris ke-395 merupakan *syntax* untuk memprediksi peluang antara model dan `train_array` yang merupakan data nilai piksel dari data *train*. Kemudian ingin diketahui dari data *train* yang ada, label termasuk ke dalam kolom data mana saja. Hal ini dapat dijalankan dengan menuliskan *syntax* pada baris ke-399 untuk data *train* dan baris ke-411 untuk data *test.* Label yang telah diklasifikasikan sebelumnya, dibentuk menjadi sebuah tabel dengan menggabungkan dengan matriks *train_data.* Tingkat akurasi model dalam mengklasifikasikan wayang dapat dicari menggunakan *syntax* pada baris ke-403 untuk data *train* dan baris ke-415 untuk data *test* dimana masing-masing dibagi dengan jumlah data *train* dan *test* yang dijalankan. Model hasil pelatihan *CNN* yang telah dijalankan pada program tersebut dapat ditampilkan dengan menggunakan *syntax* dibawah ini.

```
416 graph.viz(NN_model)
```

Gambar 5.12 Visualisasi model

5.4. Pengujian

Pengujian dilakukan untuk melakukan evaluasi terhadap model yang dihasilkan oleh *CNN*. Pengujian model dilakukan untuk model empat kelas dalam

membedakan wayang Punakawan menjadi wayang Gareng, Semar, Petruk, dan Bagong.

5.4.1. Rancangan Pengujian Parameter Optimal

Pengujian model *CNN* adalah dengan mengubah nilai parameter-parameter dalam *CNN* untuk mendapatkan nilai akurasi yang terbaik. Dalam penelitian ini, parameter yang diubah adalah jumlah iterasi atau *epoch* dari 25 hingga 250 *epoch*, jumlah filter pada masing-masing lapisan dinaikkan dari 10 menjadi ke nilai filter 20, dan ukuran kernel pada tiap lapisan konvolusi.

5.4.2. Rancangan Pengaruh Jumlah Data Terhadap Akurasi

Dalam penelitian ini, akan diuji apakah terdapat pengaruh apabila jumlah data *training* dan *testing* berbeda. Pengujian dilakukan untuk tiga jenis jumlah data *train* yang berbeda. Pertama, diuji menggunakan 70 % data *training* dari jumlah keseluruhan data. Kedua, menggunakan 80 % data *training* dari keseluruhan data. Ketiga, menggunakan 90 % data *training* dari keseluruhan data. Selanjutnya akurasi dari penelitian diteliti untuk melihat pengaruh jumlah data latih terhadap akurasi.

5.5. Hasil

Berikut ini merupakan hasil dari penerapan tahapan *preprocessing* dan pelatihan *CNN* menggunakan *Mxnet.*

5.5.1. Hasil *Preprocessing*

Berikut ini adalah beberapa contoh citra yang telah diubah ukuran dan citra warnanya menjadi *greyscale*.

(a)

(b)

(c) (d)

Gambar 5.13 Citra hasil *preprocessing*

Berdasarkan pada **Gambar 5.13** diatas, citra wayang Semar (a), wayang Bagong (b), wayang Petruk (c), dan wayang Gareng (d) menunjukkan hasil pemotongan ukuran dan konversi citra warna menjadi *greyscale.*

5.5.2. Hasil Klasifikasi *CNN*

Dalam memperoleh model terbaik tentunya dibutuhkan nilai-nilai parameter yang terbaik pula. Parameter tersebut adalah jumlah dan ukuran filter masing-masing lapisan, dan jumlah iterasi. Dalam penelitian ini peneliti juga membandingkan hasil akurasi dari jumlah data.

5.5.2.1. Perbandingan Ukuran Kernel dan Jumlah Filter

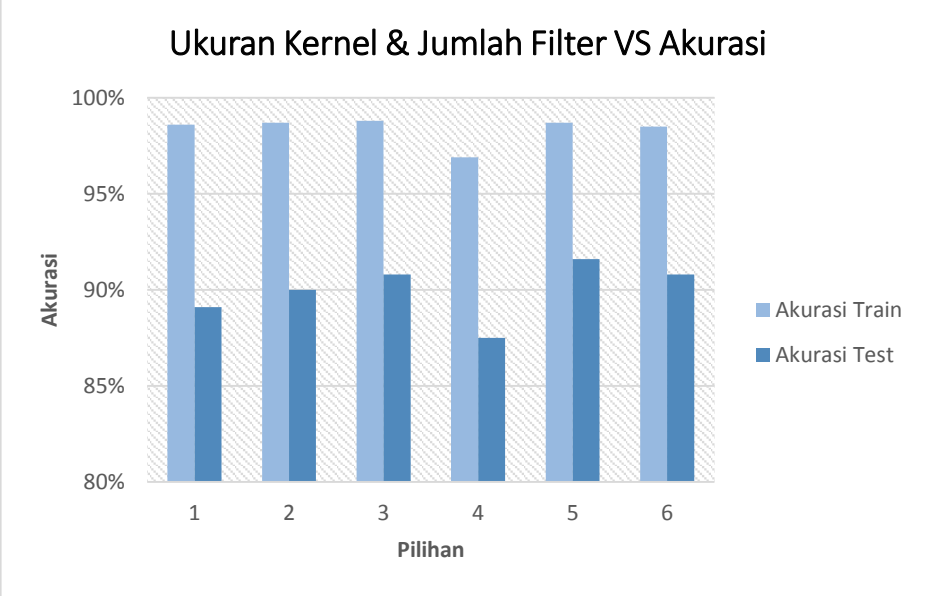
Dalam memperoleh akurasi yang terbaik, beberapa percobaan menggunakan parameter ukuran kernel dan jumlah filter yang berbeda menjadi salah satu langkah yang dapat dilakukan. Berikut ini merupakan hasil akurasi yang diperoleh dengan jumlah filter yang berbeda menggunakan jumlah *training* data sebanyak 90% dan *testing* data sebanyak 10%.

Tabel 5.2 Akurasi berdasarkan ukuran kernel dan jumlah filter

	1	2	3	4	5	6
Akurasi *Train*	98,6 %	98,7 %	98,8 %	96,9 %	**98,7 %**	98,5 %
Akurasi *Test*	89,1 %	90 %	90,8 %	87,5 %	**91,6 %**	90,8 %

Urutan 1-6 pada **Tabel 5.2** merupakan pilihan kemungkinan-kemungkinan yang terdapat pada **Tabel 5.1**. Dapat dilihat bahwa tingkat akurasi terbaik dihasilkan oleh pilihan ke-5. Nilai parameter ini terdapat pada lapisan konvolusi,

baik lapisan konvolusi pertama maupun kedua. Tingkat akurasi tertinggi berada pada pilihan kemungkinan ke-5. Nilai parameter yang terdapat dalam pilihan ke-5 terdiri dari ukuran kernel sebesar 3x3 di lapisan konvolusi pertama dan 3x3 di lapisan konvolusi kedua. Sedangkan jumlah filter 10 pada lapisan konvolusi pertama dan jumlah filter 20 pada lapisan konvolusi kedua. Dari 1080 jumlah data *train*, diketahui model dapat mengenal wayang Punakawan dengan probabilitas sebesar 98,7 %. Sedangkan dari 120 data *test*, diketahui bahwa model dapat mengenal wayang Punakawan dengan probabilitas sebesar 91,6 %.

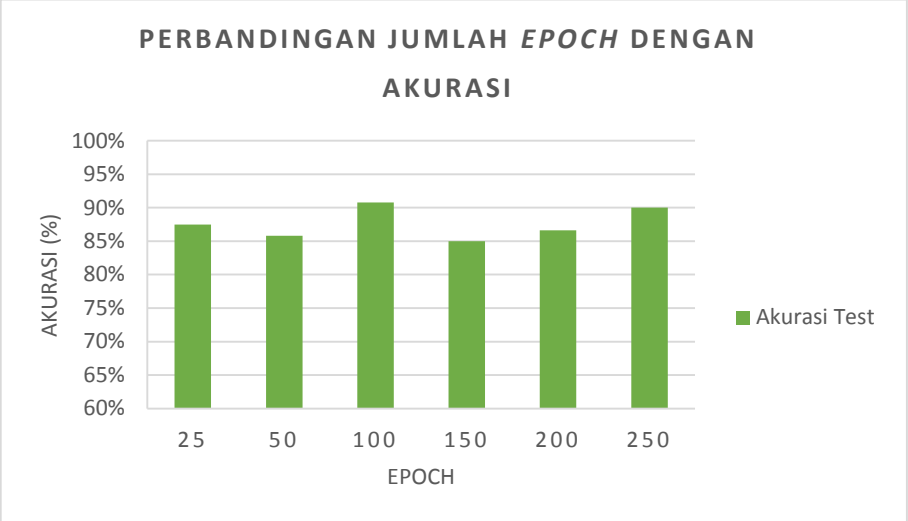


Gambar 5.14 Akurasi berdasarkan ukuran kernel dan jumlah filter

Hasil akurasi tersebut didukung oleh parameter lainnya yaitu jumlah *epoch* sebanyak 100 kali, *optimizer Adam*, dan dua lapisan konvolusi.

5.5.2.2. Perbandingan Jumlah Iterasi (Epoch)

Jumlah iterasi atau biasa disebut juga dengan *epoch* dibandingkan menggunakan beberapa nilai. Adapun jumlah *epoch* yang diuji adalah 25, 50, 100, 150, 200, dan 250. Berikut ini adalah gambaran hasil akurasi dengan jumlah *epoch* yang telah dilakukan.



Gambar 5.15 Perbandingan akurasi berdasarkan jumlah epoch

Berdasarkan **Gambar 5.15** dapat dilihat bahwa akurasi tertinggi dihasilkan oleh nilai *epoch* sebanyak 100. Parameter ini didukung beberapa parameter lainnya seperti jumlah filter sebanyak 10 pada lapisan konvolusi pertama dan 20 pada lapisan kedua. Untuk ukuran kernel pada lapisan pertama dan kedua sebesar 3x3.

5.5.2.3. Perbandingan Jumlah Data

Dalam penelitian ini, digunakan percobaan dalam menentukan jumlah data *train* dan *test* untuk mendapatkan hasil akurasi yang terbaik. Adapun percobaan tersebut menggunakan tiga macam jumlah data *train* yaitu 70 %, 80 % dan 90% dari total 1200 data wayang Punakawan, dan sisanya masing-masing merupakan data *test*. Berikut ini merupakan hasil akurasi menggunakan jumlah data *train* dan *test* yang berbeda.

Tabel 5.3 Perbandingan jumlah data

Data / Akurasi	*Train*			*Test*		
	70%	80 %	90 %	30 %	20 %	10 %
Akurasi	98.5 %	98.75 %	98.7 %	82.2 %	82.5 %	**91.6 %**

Berdasarkan pada **Tabel 5.3** tersebut, peneliti menentukan untuk menggunakan hasil akurasi pada data *test* terbesar yaitu pada jumlah data *test* 10 % dapat menghasilkan akurasi sebesar 91.6 %. Semakin banyak data *train* yang

dilatih, semakin besar pula akurasi pengenalan pada data *test* yang dihasilkan. Jumlah data inilah yang digunakan sebagai langkah awal proses pelatihan *CNN* seperti hasil program yang telah ditampilkan pada bahasan-bahasan sebelumnya.

5.5.2.4. Hasil Klasifikasi Akhir

Berdasarkan beberapa percobaan pencarian parameter terbaik dalam memperoleh akurasi yang terbaik, berikut ini merupakan hasil klasifikasi *CNN* menggunakan jumlah filter sebesar 10 pada lapisan konvolusi pertama dan 20 pada lapisan konvolusi kedua. Adapun ukuran kernel yang digunakan pada lapisan pertama maupun kedua adalah sebesar 3x3. Berikut ini merupakan hasil klasifikasi wayang Punakawan dari jumlah data *test* sebanyak 120 data.

Tabel 5.4 Hasil klasifikasi citra uji

Klasifikasi	Jumlah	Hasil				Berhasil	Gagal	% Berhasil
		Gareng	Semar	Petruk	Bagong			
Gareng	30	28	0	1	1	28	2	93.3 %
Semar	30	1	26	1	2	26	4	86.7 %
Petruk	30	0	2	28	0	28	2	93.3 %
Bagong	30	0	1	1	28	28	2	93.3 %
Rata-rata % berhasil								91.65 %

Berdasarkan **Tabel 5.4** tersebut, ketika sistem diberi masukan 30 citra Gareng, 30 citra Semar, 30 citra Petruk, dan 30 citra Bagong ternyata citra yang dapat dikenali dengan tepat adalah sebanyak 28 untuk citra Gareng, 26 untuk citra Semar, 28 untuk citra Petruk, dan 28 untuk citra Bagong. Pada hasil pengenalan 30 citra Gareng, dua citra lainnya dikenali sebagai citra Petruk dan Bagong. Pada pengenalan 30 citra Semar, diketahui bahwa empat citra lainnya dikenali sebagai 1 citra Gareng, 1 citra Petruk, dan 2 citra Bagong. Pada pengenalan 30 citra Petruk, diketahui bahwa dua citra lainnya dikenali sebagai citra Semar. Dan pada pengenalan 30 citra Bagong, dua citra lainnya dikenali sebagai citra Semar dan Petruk. Rata-rata model mampu mengklasifikasikan keempat jenis wayang

Punakawan dengan akurasi sebesar 91.65 % dengan perhitungan simpangan sebagai berikut.

$$\sigma = \left(\sqrt{\frac{(93.33-91.65)^2+(86.67-91.65)^2+(93.33-91.65)^2+(93.33-91.65)^2}{(4-1)}}\right)\% = 3.33\ \%$$

Sehingga dapat disimpulkan bahwa keakurasian kerja sistem yang menggunakan algoritma *CNN* sebagai algoritma klasifikasi wayang Punakawan adalah sebesar (91.65 ± 3.33) %. Akurasi yang belum sempurna ini dapat disebabkan oleh beberapa faktor, misalnya seperti adanya citra negatif, *noise* dalam *dataset,* kurangnya sampel, dan lain-lain.

BAB VI
PENUTUP

6.1. Kesimpulan

Berdasarkan penelitian dan hasil penerapan metode *Convolutional Neural Network* (CNN) dalam mengklasifikasikan citra wayang Punakawan, dapat diambil kesimpulan sebagai berikut :

1. Citra dapat melewati proses *preprocessing* dengan baik yaitu dengan diubah ukuran dimensinya menjadi 46x46 dan dikonversi citra warnanya menjadi *greyscale.*
2. Akurasi dapat dicapai semakin baik apabila digunakan data *train* yang semakin besar. Hal ini dibuktikan dengan pengujian pada tiga jumlah data *training* berbeda, yaitu 70 %, 80 % dan 90 %. Dari ketiganya, hasil akurasi terbaik pada data uji dihasilkan oleh data *train* sebesar 90 % dan data *test* sebanyak 10 %.
3. Dengan menggunakan data *train* 90 % dan data *test* 10 %, dua lapisan konvolusi, jumlah filter 10 pada lapisan konvolusi pertama dan jumlah filter 20 pada lapisan konvolusi kedua, ukuran kernel pada lapisan konvolusi pertama dan kedua sebesar 3x3 diperoleh akurasi model dalam mengenali empat jenis wayang Punakawan berdasarkan data uji telah sangat baik yaitu sebesar 91.6 %.

6.2. Saran

Berdasarkan hasil penelitian ini, terdapat beberapa saran bagi penelitian selanjutnya khususnya bagi penelitian yang menggunakan metode *Convolutional Neural Network* :

1. Guna dapat mempercepat pemrosesan pelatihan *CNN,* dapat digunakan perangkat *Graphical Processing Unit* (GPU).
2. Dapat mengembangkan program menjadi lebih *attractive,* contohnya dengan menggabungkan dengan ilmu *computer vision.*

3. Mampu membandingkan lebih banyak parameter dalam proses pelatihan *CNN* guna dapat memperoleh hasil akurasi yang terbaik.
4. Mampu mengembangkan perolehan model *CNN* menggunakan *Mxnet* menjadi semakin dapat menghasilkan *output* yang bervariatif, handal dan representatif dalam mengenali objek.

DAFTAR PUSTAKA

Adi N., Kristian, Santoso., Albertus J., dan Suselo, Thomas. 2013. *Algoritma Backpropagation pada Jaringan Saraf Tiruan untuk Pengenalan Pola Wayang Kulit. SemnasIF2013*, ISSN:1979-2328.

Bengio, Y. 2015. *Deep Learning. Literature review*, *Nature* vol 521, 436–444.

Bramer, M., 2007, Principles of Data Mining, Springer-Verlag, London.

Danukusumo, Kefin Pudi. 2017. *Implementasi Deep Learning Menggunakan Convolutional Neural Network Untuk Klasifikasi Citra Candi Berbasis Gpu*. Skripsi, Program Informatika, Universitas Atma Jaya, Yogyakarta.

Deng, L. & Yu, D., 2014, Deep Learning: Methods and Applications, Foundations and Trends in Signal Processing, 7, 3-4, 197-387.

Goodfellow, I., Bengio, Y., & Courville, A., 2016, Deep Learning,http://goodfeli.github.io/dlbook/, diakses 17 Februari 2018.

Ford, A., dan Roberts, A., Agustus 1998, Colour Space Conversions,http:// www.inforamp.net/ ~poynton/ PDFs/ coloureq.pdf.

J. Han and M. Kamber, Data mining: Concepts and Techniques, Second ed. San Fransisco: Elsevier, 2006.

Kanan, C. dan Cottrell, G.W., 2012, Color-to-grayscale: Does the method matter in image recognition?, PLoS ONE, 7(1).

Kalchbrenner, N., Grefenstette, E., Blunsom, P., 2014. A Convolutional Neural Network for Modellng Sentence. Department of Computer Science University of Oxford.

Kusumadewi, S., 2004, Membangun Jaringan Syaraf Tiruan, Yogyakarta: Graha Ilmu.

LeCun, Y., Kavukcuoglu, K. dan Farabet, C., 2010, Convolutional networks and applications in vision, ISCAS 2010 - 2010 IEEE International Symposium on Circuits and Systems: Nano-Bio Circuit Fabrics and Systems, pp.253–256.

Li Mu., Andersen, dkk. *Scaling Distributed Machine Learning with the Parameter Server*. 2014. Carnegie Mellon University, Baidu, Google.

Lu, Yuzhen. 2016. *Food Image Recognition by Using Convolutional Neural Networks (CNNs)1*. *Final Project*, Michigan State University.

Madhukar, M., Again, S., and Chronopoulos, A.T., 2012, New Decision Support Tool for Acute Lymphoblastic Leukemia Classification, dalam Journal of SPIE-IS&T/ Vol. 8295 829518-1.

McAndrew Alasdair, (2004), An Introduction to Digital Image Processing with Matlab. Notes for SCM2511 Image Processing 1, School of Computer Science and Mathematics Victoria University of Technology.

Mulyani, Rachmi. 2016. Pengenalan Suara Pada Sistem Notulen Rapat Menggunakan Convolutional Neural Network (CNN), Skripsi, Program Ilmu Komputer, Universitas Pendidikan Indonesia, Bandung.

Nurgiyantoro, Burhan. 2011. Wayang dan Pengembangan Karakter Bangsa. *Jurnal Pendidikan Karakter 1:1*, Yogyakarta.

Priambodoeko. 2011. *Wayang Punakawan.* *https://www.flickr.com/photos/35840991@N06/5903650462/*. Diakses pada 10 Februari 2018.

Pramudiono, Iko. 2003. *Pengantar Data Mining : Menambang Permata Pengetahuan di Gunung Data.*

Rajagede, Rian Adam. 2016. Deep Learning untuk Pengenalan Pelafalan Huruf Hijaiyah Berharakat, Skripsi, Program Ilmu Komputer, Universitas Gajah Mada, Yogyakarta.

Razi, AR. 2017. Klasifikasi Artikel Berita Berbahasa Indonesia Menggunakan Convolutional Neural Network, Tesis, Program Ilmu Komputer, Universitas Gajah Mada, Yogyakarta.

Rismiyati. 2016. Implementasi Convolutional Neural Network Untuk Sortasi Mutu Salak Ekspor Berbasis Citra Digital, Tesis, Program Ilmu Komputer, Universitas Gajah Mada, Yogyakarta.

Zufar, Muhammad dan Setiyono, Budi. 2016. *Convolutional Neural Networks untuk Pengenalan Wajah Secara Real-Time*. *Jurnal Sains dan Seni ITS*, Vol.5 No.2, 2337-3520.

LAMPIRAN

1. *Syntax* Program Klasifikasi Wayang Punakawan

```
library(EBImage)
setwd("D://gareng/")
labelgareng <- 0
df <- data.frame()
gareng <- list.files()
w<-46
h<-46
gambar_size<-46*46
for (i in 1:length(gareng))
{
  semuagareng <- readImage(gareng[i])
  semuagareng_resized <- resize(semuagareng, w=w, h=h)
  semuagareng_gray <- channel(semuagareng_resized, "gray")
  semuagareng_matrix <- semuagareng_gray@.Data
  semuagareng_vector <- as.vector(t(semuagareng_matrix))
  gabunggareng=c(labelgareng,semuagareng_vector)
  df<-rbind(df,gabunggareng)
  write.csv(df, "gabunggareng.csv", row.names = FALSE)
  names(df) <- c('label', paste('pixel', c(1:gambar_size)))
}
display(semuagareng_gray)
```

```
setwd("D://semar/")

labelsemar <- 1

df <- data.frame()

semar <- list.files()

w<-46

h<-46

gambar_size<-46*46

for (i in 1:length(semar))

{

  semuasemar<- readImage(semar[i])

  semuasemar_resized <- resize(semuasemar, w=w, h=h)

  semuasemar_gray <- channel(semuasemar_resized, "gray")

  semuasemar_matrix <- semuasemar_gray@.Data

  semuasemar_vector <- as.vector(t(semuasemar_matrix))

  gabungsemar=c(labelsemar,semuasemar_vector)

  df<-rbind(df,gabungsemar)

  write.csv(df, "gabungsemar.csv", row.names = FALSE)

  names(df) <- c('label', paste('pixel', c(1:gambar_size)))

}

display(semuasemar_gray)
```

```
setwd("D://petruk/")


labelpetruk <- 2

df <- data.frame()


petruk <- list.files()

w<-46

h<-46

gambar_size<-46*46

for (i in 1:length(petruk))

{
  semuapetruk<- readImage(petruk[i])

  semuapetruk_resized <- resize(semuapetruk, w=w, h=h)

  semuapetruk_gray <- channel(semuapetruk_resized, "gray")

  semuapetruk_matrix <- semuapetruk_gray@.Data

  semuapetruk_vector <- as.vector(t(semuapetruk_matrix))

  gabungpetruk=c(labelpetruk,semuapetruk_vector)

  df<-rbind(df,gabungpetruk)

  write.csv(df, "gabungpetruk.csv", row.names = FALSE)

  names(df) <- c('label', paste('pixel', c(1:gambar_size)))

}
display (semuapetruk_gray)
```

```
setwd("D://bagong/")

labelbagong <- 3

df <- data.frame()

bagong <- list.files()

w<-46

h<-46

gambar_size<-46*46


for (i in 1:length(bagong))

{
semuabagong<- readImage(bagong[i])

  semuabagong_resized <- resize(semuabagong, w=w, h=h)

  semuabagong_gray <- channel(semuabagong_resized, "gray")

  semuabagong_matrix <- semuabagong_gray@.Data

  semuabagong_vector <- as.vector(t(semuabagong_matrix))

  gabungbagong=c(labelbagong,semuabagong_vector)

  df<-rbind(df,gabungbagong)

  write.csv(df, "gabungbagong.csv", row.names = FALSE)

  names(df) <- c('label', paste('pixel', c(1:gambar_size)))

}

display(semuabagong_gray)
```

```
garengcsv<- read.csv("D://gareng/gabunggareng.csv")

semarcsv<- read.csv("D://semar/gabungsemar.csv")

petrukcsv<- read.csv("D://petruk/gabungpetruk.csv")

bagongcsv<- read.csv("D://bagong/gabungbagong.csv")

dim(garengcsv)

library(caret)

complete_set <- rbind(garengcsv,semarcsv,petrukcsv,bagongcsv)

training_index <- createDataPartition(complete_set$label, p =
    .90,times = 1)

training_index <- unlist(training_index)

train_set <- complete_set[training_index,]

dim(train_set)

test_set <- complete_set[-training_index,]

dim(test_set)

train_data <- data.matrix(train_set)

train_x <- t(train_data[, -1])

dim(train_x)

train_y <- train_data[,1]

dim(train_y)

train_array <- train_x

dim(train_array)

dim(train_array) <- c(46,46,1,ncol(train_x))

test_data <- data.matrix(test_set)

test_x <- t(test_set[,-1])

dim(test_x)

test_y <- test_set[,1]

test_array <- test_x

dim(test_array)

dim(test_array) <- c(46,46,1, ncol(test_x))
```

```
library(mxnet)

mx_data <- mx.symbol.Variable('data')

conv_1 <- mx.symbol.Convolution(data = mx_data, kernel = c(3,
     3), num_filter = 10)

tanh_1 <- mx.symbol.Activation(data = conv_1, act_type = "tanh")

pool_1 <- mx.symbol.Pooling(data = tanh_1, pool_type = "max",
     kernel = c(2, 2), stride = c(2,2))

conv_2 <- mx.symbol.Convolution(data = pool_1, kernel = c(3,3),
     num_filter = 20)

tanh_2 <- mx.symbol.Activation(data = conv_2, act_type = "tanh")

pool_2 <- mx.symbol.Pooling(data = tanh_2, pool_type = "max",
         kernel = c(2, 2),stride = c(2, 2))

flat <- mx.symbol.Flatten(data = pool_2)

drop1 = mx.symbol.Dropout(data=flat, p=0.1)

fcl_1 <- mx.symbol.FullyConnected(data = drop1, num_hidden = 10)

tanh_3 <- mx.symbol.Activation(data = fcl_1, act_type = "tanh")

drop1 = mx.symbol.Dropout(tanh_3, p=0.1)

fcl_2 <- mx.symbol.FullyConnected(data = drop1, num_hidden = 4)

NN_model <- mx.symbol.SoftmaxOutput(data = fcl_2)

device <- mx.cpu()

NN_model <- mx.symbol.SoftmaxOutput(data = fcl_2)


model <- mx.model.FeedForward.create(NN_model, X = train_array,
     y = train_y,
ctx = device,

num.round = 100,

array.batch.size = 250,

optimizer = "adam",

eval.metric=mx.metric.accuracy,

initializer = mx.init.Xavier(factor_type = "in",

rnd_type = "gaussian",magnitude = 2),

array.layout = "auto")
```

```
mx.set.seed(100)


predict_probs <- predict(model, train_array)

predict_probs

predicted_labels <- max.col(t(predict_probs))-1

table(train_data[,1],predicted_labels)


sum(diag(table(train_y, predicted_labels)))/1080

predict_probs <- predict(model, test_array)

predict_probs

predicted_labels <- max.col(t(predict_probs))-1

table(test_data[, 1], predicted_labels)

sum(diag(table(test_y, predicted_labels)))/120

graph.viz(NN_model)
```

2. Citra Wayang Punakawan