IMPLEMENTASI METODE CNN DALAM KLASIFIKASI GAMBAR JAMUR PADA ANALISIS *IMAGE PROCESSING*

(Studi Kasus: Gambar Jamur dengan Genus *Agaricus* dan *Amanita*)

TUGAS AKHIR

Ocktavia Nurima Putri

16611103

PROGRAM STUDI STATISTIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS ISLAM INDONESIA

YOGYAKARTA

2020

HALAMAN PERSETUJUAN PEMBIMBING

TUGAS AKHIR

Judul	:	Implementasi Metode CNN dalam Klasifikasi Gambar Jamur pada Analisis *Image Processing* (Studi Kasus: Gambar Jamur dengan Genus *Agaricus* dan *Amanita*)
Nama Mahasiswa	:	Ocktavia Nurima Putri
NIM	:	16611103

TUGAS AKHIR INI TELAH DIPERIKSA DAN DISETUJUI UNTUK DIUJIKAN

Yogyakarta, (isikan tanggal persetujuan pembimbing)

Pembimbing

(Dr. techn. Rohmatul Fajriyah, S.Si., M.Si.)

HALAMAN PENGESAHAN

TUGAS AKHIR

IMPLEMENTASI METODE CNN DALAM KLASIFIKASI GAMBAR JAMUR PADA ANALISIS *IMAGE PROCESSING*

(Studi Kasus: Gambar Jamur dengan Genus *Agaricus* dan *Amanita*)

Nama Mahasiswa : Ocktavia Nurima Putri

NIM : 16611103

TUGAS AKHIR INI TELAH DIUJIKAN
PADA TANGGAL: 16 Mei 2020

Nama Penguji:	**Tanda Tangan**
1. Ayundyah Kesumawati, S.Si., M.Si.	……..
2. Tuti Purwaningsih, S.Stat., M.Si.	……..
3. Dr. techn. Rohmatul Fajriyah, S.Si., M.Si.	……..

Mengetahui,
Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam
Universitas Islam Indonesia

Prof. Riyanto, S.Pd., M.Si., Ph.D.

KATA PENGANTAR

Assalamu'alaikum Wr.Wb

Dengan menyebut nama Allah SWT, Sang pencipta langit dan bumi serta segala isinya, penulis panjatkan puja dan puji syukur kepada-Nya, yang telah melimpahkan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan tugas akhir yang berjudul "Implementasi Metode CNN dalam Klasifikasi Gambar Jamur pada Analisis *Image Processing* (Studi Kasus: Gambar Jamur dengan Genus *Agaricus* dan *Amanita*)". Shalawat serta salam juga tak lupa penulis ucapkan kepada Nabi Muhammad SAW yang telah diutus ke bumi sebagai lentera bagi hati manusia.

Selesainya tugas akhir ini tidak lepas dari bantuan dan dukungan dari berbagai pihak. Tiada ungkapan yang lebih pantas diberikan selain ucapan terima kasih yang tulus kepada:

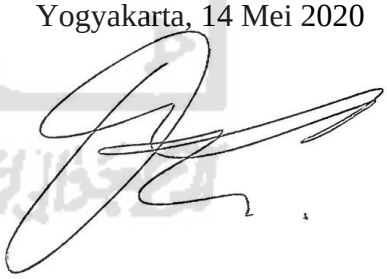
1. Bapak Prof. Riyanto S.Pd., M.Si., Ph.D, selaku Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Islam Indonesia.
2. Bapak Dr. Edy Widodo, S.Si., M.Si, selaku Ketua Program Studi Statistika, Universitas Islam Indonesia.
3. Ibu Dr. techn. Rohmatul Fajriyah, S.Si., M.Si, selaku Dosen Pembimbing tugas akhir yang telah mengorbankan waktu luangnya dan selalu membimbing dalam penyusunan tugas akhir.
4. Bapak dan Ibu Dosen Program Studi Statistika, Universitas Islam Indonesia yang telah mendidik saya selama masa perkuliahan.
5. Kedua orang tua terbaik sedunia, Mama dan Papa yang telah banyak berkorban, senantiasa mendukung saya secara lahir dan batin, dan tak pernah putus doanya disepertiga malam demi kebaikan anak-anaknya.
6. Mbak Nina dan Mas Banar, yang selalu menjadi penyemangat dan tempat melepas penat setiap pulang.
7. Kesayangan Rima. Gifa, Maudi, Lina, Cinmey, Alfa, dan Farhan, sahabat

terbaik semasa kuliah dari jaman cupu sampe pada bisa dandan.

8. Temen-temen "The Hobbit", Febri, Afi, Amalia, yang selalu nanya kapan pulang dan doain dari jauh.
9. Mei, sahabat baru yang selalu bikin aku pulang malem buat sekedar nongkrong di burjo atau muter-muter Jogja.
10. Naya, makasih udah sabar dan selalu jawab pertanyaan gue yang aneh-aneh. *I can't do this without you, I owe you A LOT.*
11. Yolan, Rinna, Bang Tebe, Theddy, semua temen-temen yang selalu support aku supaya nggak suntuk tinggal di Jogja.
12. EDS UII, LABMA UII, LSF 2017, LSF 2018, Saung Mimpi, GEN, KKN 289, GDI dan HIPWEE yang telah menjadi wadah untuk berbagi pengalaman dan berkembang.
13. Temen-temen satu bimbingan Ibu Ema, terutama Iqbal dan Widia yang udah jadi temen ambisku.
14. Lau, makasih udah mau direpotin dan bantu dalam pengerjaan skripsiku.
15. Seluruh teman-teman jurusan statistika angkatan 2016 yang selama 4 tahun telah berjuang bersama untuk menyandang gelar sarjana.

Wassalamualaikum Wr.Wb

Yogyakarta, 14 Mei 2020



(Ocktavia Nurima Putri)

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

PERNYATAAN

Dengan ini saya menyatakan bahwa dalam Tugas Akhir ini tidak terdapat karya yang sebelumnya pernah diajukan untuk memperoleh gelar kesarjanaan di suatu Perguruan Tinggi dan sepanjang pengetahuan saya juga tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan orang lain, kecuali yang diacu dalam naskah ini dan disebutkan dalam daftar pustaka.

Yogyakarta, 14 Mei 2020

(Ocktavia Nurima Putri)

ABSTRAK

IMPLEMENTASI METODE CNN DALAM KLASIFIKASI GAMBAR JAMUR PADA *IMAGE PROCESSING*

(Studi Kasus: Gambar Jamur dengan Genus *Agaricus* dan *Amanita*)

Ocktavia Nurima Putri
Program Studi Statistika, Fakultas MIPA
Universitas Islam Indonesia

Jamur adalah tumbuhan tanpa klorofil dan hidup bergantung dengan makhluk hidup lainnya. Terdapat lebih dari 1.500.000 spesies jamur di dunia dan hanya sekitar 74.000 diantaranya yang teridentifikasi. Di Eropa Utara, terdapat beberapa genus jamur yang banyak tumbuh dan berkembang dengan luas, diantaranya adalah *Agaricus* yang dikenal sebagai jamur layak konsumsi seperti *Agaricus Bisporus* (jamur kancing), dan genus jamur *Amanita*, spesies dari genus jamur tersebut dikenal sebagai jamur paling beracun di dunia, salah satunya disebut sebagai *death caps* yang dapat menyebabkan iritasi dan berpotensi mematikan. Namun morfologi dari beberapa jamur beracun ini bisa menyerupai jamur yang dapat dikonsumsi. Oleh karena itu, penelitian ini bertujuan untuk mengklasifikasikan gambar jamur berdasarkan kategori genusnya. Metode *Deep Learning* yang saat ini memiliki hasil paling signifikan dalam pengenalan citra adalah *Convolutional Neural Network* (CNN). Analisis yang dilakukan menghasilkan akurasi sebesar 62% menggunakan *optimizer* Adam dengan *epoch* 100 dan menggunakan skenario perbandingan data *train validation* 80% : 20%, ukuran kernel 3x3, dan *learning rate* sebesar 0,001.

Kata Kunci: *Genus* Jamur, Klasifikasi Gambar, CNN.

ABSTRACT

CNN IMPLEMENTATION FOR MUSHROOM IMAGE CLASSIFICATION IN IMAGE PROCESSING ANALYSIS

(Case Study: Genus Mushroom Images Agaricus and Amanita)

Ocktavia Nurima Putri

Department of Statistics, Faculty of Mathematics and Natural Sciences

Universitas Islam Indonesia

Mushroom is a group of plant without xhlorophyll and depends upon prepared nutrients. There are over 1,500,000 species of mushrooms on earth and only around 74,000 are identified. There are various types of mushroom genus in Northern Europe, such as Agaricus and Amanita. Agaricus commonly known as an edible mushroom like Agaricus Bisporus, meanwhile Amanita is a genus that is known as the most toxic known mushrooms found worldwide. One of its species is called death caps that contains enough toxin to cause irritation and kill a person. However, these incredibly toxic are very similar in appearance to edible button mushrooms. Therefore, this study aims to classify mushroom images based on its genus using the famous Convolutional Neural Network (CNN) algorithm as a sophisticated technique in drawing tasks that is still part of deep learning. Based on the analysis carried out, the best architectural results is obtained using the proposed scheme for 80% : 20% train validation data, 3x3 kernel size, and learning rate of 0.001 with the accuracy result 62%.

Keywords: *Mushroom Genus, Image classification, CNN.*

BAB 1. PENDAHULUAN

1.1 Latar Belakang Masalah

Salah satu komoditas hortikultura dengan prospek yang baik adalah jamur. Jamur merupakan salah satu kelompok fungi, yaitu tumbuhan tanpa klorofil dan hidup bergantung dengan makhluk hidup lainnya. Secara umum, jamur dapat didefinisikan sebagai organisme *eukariotik* yang mempunyai inti dan *organel*. Jamur tersusun dari *hifa* yang merupakan benang-benang sel tunggal panjang, sedangkan kumpulan *hifa* disebut dengan *miselium*. Jamur mudah dikenal dengan melihat warna *miselium*nya. *Miselium* merupakan *massa* benang yang cukup besar dibentuk dari *hifa* yang saling membelit pada saat jamur tumbuh (Volk and Wheeler 1993). Jamur dapat digunakan sebagai bahan pangan dan digunakan dalam bidang kesehatan. Tumbuhan ini juga berperan sebagai pengurai dari rantai makanan organisme.

Kelompok jamur yang menarik untuk dilihat diversitas dan potensinya adalah jamur makro. Jamur makro sebagian besar merupakan anggota dari filum *Basidiomycota* dan *Ascomycota* (Hibbett, et al. 2007). Jamur makro tumbuh pada batang kayu lapuk, permukaan tanah, atau serasah. Di berbagai belahan dunia, banyak jamur liar yang berperan sangat penting dalam kehidupan manusia di pedesaan (Sarma, Sarma and Patriana 2010, Cai, Pettenella and Vidale 2011).

Terdapat lebih dari 1.500.000 spesies jamur di dunia dan hanya sekitar 74.000 diantaranya yang telah teridentifikasi. Beberapa jamur yang telah teridentifikasi banyak bermanfaat dalam bidang pangan, kesehatan, ekonomi, dan agrikultur. Beberapa lainnya dapat menyebabkan infeksi bagi manusia (Tasyarini, Triswaty and Susantina 2006). Di Eropa Utara, terdapat beberapa genus jamur yang banyak tumbuh dan berkembang dengan luas, diantaranya adalah *Agaricus* dan *Amanita*. Genus jamur *Agaricus* dikenal sebagai jamur yang memiliki banyak spesies layak konsumsi seperti *Agaricus Bisporus* (jamur kancing). Lain halnya dengan genus jamur *Amanita*, spesies dari genus jamur tersebut dikenal sebagai jamur paling beracun di dunia, salah satunya disebut sebagai *death caps* yang dapat menyebabkan iritasi dan berpotensi mematikan. Namun morfologi dari beberapa

jamur beracun ini bisa menyerupai jamur yang dapat dikonsumsi. Hal tersebut dapat membahayakan apabila manusia salah memilih jamur yang ditemui di alam bebas.

Pengklasifikasian makhluk hidup pada umumnya menggunakan konsep spesies dan genus. Jamur dapat diidentifikasi berdasarkan karakteristik ukuran, warna, dan bentuk dari tudung dan batangnya. Namun, identifikasi jamur masih sulit dilakukan karena banyaknya jenis jamur, kurangnya pengetahuan tentang jamur, dan kurangnya ahli dalam bidang jamur. Tidak semua jamur memiliki nama, beberapa lainnya memiliki lebih dari satu. Oleh karena itu, dibutuhkan suatu algoritma untuk mempermudah sistem dalam mengenali, membandingkan, dan mempelajari secara otomatis struktur jamur menggunakan data citra agar dapat ditemukan perbedaan antaraa jamur yang dapat dikonsumsi dan beracun.

Pada saat ini tidak dapat dipungkiri bahwa perkembangan teknologi informasi sangat cepat dan cenderung membantu manusia dalam menyelesaikan pekerjaan dengan lebih cepat dan dalam waktu yang singkat. Dalam era yang serba digital ini sangat memungkinkan untuk terciptanya sebuah komputasi yang dapat mengolah informasi dari suatu citra atau gambar untuk pengenalan objek secara otomatis. Pengolahan citra atau *image processing* adalah suatu proses pengolahan citra dengan menggunakan komputer menjadi sebuah citra yang memiliki kualitas yang lebih baik. Tujuan dari pengolahan citra ini adalah memperbaiki kualitas suatu citra sehingga dapat diinterpretasi dengan mudah oleh manusia atau atau sebuah mesin.

Deep learning telah menjadi salah satu topik hangat dalam dunia *machine learning* karena kapabilitasnya yang signifikan dalam memodelkan berbagai data kompleks seperti citra dan suara. Metode *Deep Learning* yang saat ini memiliki hasil paling signifikan dalam pengenalan citra adalah *Convolutional Neural Network* (CNN) (Fukushima 1980). *Convolutional Neural Network* atau CNN merupakan salah satu metode *deep learning*, dimana metode tersebut telah banyak diaplikasikan pada data gambar. Metode CNN berhasil mengungguli *machine learning* seperti metode SVM pada kasus klasifikasi gambar, yang saat ini memiliki hasil paling signifikan dalam pengenalan citra dikarenakan CNN memiliki cara kerja menyerupai fungsi otak pada manusia, dimana komputer akan diberikan data citra untuk dipelajari, dilatih mengenali setiap elemen visual pada citra serta

memahami setiap pola citranya, hingga nantinya komputer mampu mengidentifikasi citra tersebut (P, Wijaya and Soelaiman 2016).

Kemajuan terbaru pada CNN telah menjadikannya sebagai suatu teknik yang canggih dalam tugas klasifikasi gambar. CNN dapat mempelajari fitur hirarki yang digunakan untuk klasifikasi gambar, yang mana pendekatan hirarki itu mampu mempelajari fitur yang lebih kompleks dengan lapisan yang lebih tinggi, sehingga keakuratan metode CNN untuk mengklasifikasi gambar akan bisa lebih tinggi (Xu, Feng and Mi 2017). Kemampuan CNN diklaim sebagai model terbaik untuk memecahkan permasalahan *object detection* dan *object recognition*. Pada tahun 2012, penelitian tentang CNN dapat melakukan pengenalan citra digital dengan akurasi yang menyaingi manusia pada *dataset* tertentu (Coates, Lee and Ng. 2011). Beberapa penelitian mengenai pengolahan citra dengan menggunan metode CNN mendapatkan hasil akurasi yang bagus, seperti penelitian yang dilakukan oleh Nurhikmat (2018) dalam penelitiannya mengenai klasifikasi gambar wayang golek, pada penelitian tersebut diperoleh hasil akurasi data *training* sebesar 95% dan akurasi data *validation* sebesar 90%. Sedangkan untuk data *testing* yakni sebesar 93%.

Berdasarkan uraian latar belakang diatas, maka peneliti melakukan analisis dengan judul "Implementasi Metode CNN dalam Klasifikasi Gambar Jamur pada Analisis *Image Processing*" dengan studi kasus gambar genus jamur *Agaricus* dan *Amanita*. Dengan penelitian ini diharapkan dapat diketahui tingkat akurasi model yang diperoleh dan seberapa mampu model CNN mengklasifikasi gambar jamur berdasarkan jenis genus dengan akurat.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang ada, rumusan masalah dalam penelitian ini adalah sebagai berikut:

1. Bagaimana hasil metode *deep learning* menggunakan CNN untuk memprediksi gambar jamur berdasarkan macam genusnya?
2. Bagaimana hasil akurasi klasifikasi gambar jamur menggunakan CNN?

1.3 Batasan Masalah

Adapun batasan masalah dalam penelitian ini antara lain:

1. Data yang digunakan merupakan data gambar jamur dengan 2 macam genus yang masing-masing berjumlah sebanyak 208 gambar untuk *training* dan 52 gambar untuk *testing*.
2. Gambar yang digunakan memiliki ukuran pixel.
3. Klasifikasi dilakukan berdasarkan jenis genus, yaitu *Agaricus* dan *Amanita.*
4. *Software* yang digunakan untuk membantu implementasi *deep learning* ialah Python 3.6.2.
5. *Packages* yang digunakan dalam analisis pengolahan gambar dengan metode CNN yakni Keras pada *software* Python.
6. Hasil akurasi diatas 50% dikatakan baik.

1.4 Tujuan Penelitian

Tujuan dari penelitian ini adalah sebagai berikut:

1. Mengetahui bagaimana implementasi metode *deep learning* menggunakan CNN untuk memprediksi gambar jamur berdasarkan macam *genus*.
2. Mengetahui pengukuran akurasi klasifikasi gambar jamur menggunakan CNN.

1.5 Manfaat Penelitian

Manfaat yang diharapkan dalam penelitian ini adalah untuk mengembangkan informasi dan pengetahuan mengenai implementasi dari metode *deep learning* dalam mengklasifikasikan gambar, khususnya untuk memprediksi jenis genus jamur yang dapat dikonsumsi dan beracun berdasarkan gambar yang dapat bermanfaat untuk khalayak umum.

BAB 2. TINJAUAN PUSTAKA

2.1 Penelitian Menggunakan Metode *Convolutional Neural Network*

Referensi penelitian terdahulu yang menggunakan metode CNN untuk memproses klasifikasi objek penelitiannya dirangkum dalam tabel berikut:

Tabel 2.1. Tinjauan Pustaka Penelitian Menggunakan Metode CNN

No.	Nama Peneliti	Judul Penelitian	Tahun	Isi
1.	(Meng, et al. 2014)	*Prediction of plant pre-microRNAs and their microRNAs in genome-scale sequences using structure-sequence features and Convolutional Neutral Network*	2014	Penelitian ini mengembangkan model klasifikasi terpadu, miPlantPreMat berdasarkan *feature* susunan struktur dan CNN untuk mengidentifikasi kedua tanaman pra-miRNA dan miRNA. Untuk melatih model terdapat 152 *feature* baru yang terkait dengan sekuensial. Mereka memperoleh *subset* terbaik dari 47 *feature* dengan menerapkan pemilihan *feature* untuk digunakan dalam metode *Back Convolutional Neural Network-Recursive Feature Elimination* (B-CNN-RFE) untuk klasifikasi pra-miRNA pabrik. Dengan menggunakan metode ini, terdapat 63 *feature* untuk klasifikasi tanaman miRNA. Model ini mencapai

No.	Nama Peneliti	Judul Penelitian	Tahun	Isi
				akurasi sekitar 90% dengan menggunakan *dataset* tanaman dari sembilan jenis tumbuhan, termasuk *Arabidopsis thaliana, Glycine max, Oryza sativa, patogen Physcomitrella, medentago truncatula, sorghum bicolor, Arabidopsis lyrata, Zea mays dan Solanum lycopersicum*. Selain itu dengan menggunakan *miPlantPreMat*, 522 *Solanum lycopersicum* miRNAs diidentifikasi dalam urutan *genom lanopersikum Solanum*.
2.	(P, Wijaya and Soelaiman 2016)	Klasifikasi Gambar Menggunakan *Convolutional Neural Network* (CNN) pada Caltech 10	2016	Penelitian ini melakukan klasifikasi dataset caltech 101 menggunakan CNN, gambar yang dianalisis sebanyak 390, dimana sebanyak 150 masuk dalam kategori unggas, selebihnya yakni *crocodile*, *cougar*, dan *face*. Keberhasilan klasifikasi yang diperoleh untuk kategori unggas yakni 20% sedangakan untuk kategori *crocodile*, *cougar*, dan *face* sebesar 50%. Penelitian ini

No.	Nama Peneliti	Judul Penelitian	Tahun	Isi
				membuktikan bahwa klasifikasi menggunakan CNN relatif handal terhadap perubahan parameter yang dilakukan, dan dengan menggunakan *data training* yang optimal maka akan menghasilkan klasifikasi yang baik.
3.	(Dewa, Fadhilah and Afiahayat 2018)	*Convolutional Neural Networks for Handwritten Javanese Character Recognition*	2018	Penelitian ini mengembangkan perangkat lunak dengan fitur pengolahan gambar menggunakan metode CNN untuk pengenalan karakter tulisan tangan Aksara Jawa yang kemudian dibandingkan dengan hasil dari model *Multilayer Perceptron* (MLP) dari sisi akurasi klasifikasi dan waktu latih. Hasil pengujian menunjukkan akurasi dari model CNN lebih unggul dari model MLP meskipun CNN membutuhkan waktu latih lebih lama dibandingkan dengan MLP.

No.	Nama Peneliti	Judul Penelitian	Tahun	Isi
4.	(Nurhikmat 2018)	Implementasi *Deep Learning* untuk *Image Classification* Menggunakan Algoritma *Convolutional Neural Network* (CNN) Pada Gambar Wayang Golek	2018	Pada penelitian ini dilakukan klasifikasi terhadap gambar wayang golek menggunakan metode CNN, gambar yang diguankan ialah sebanyak 60, dengan menggunakan 64x64 piksel, *learning rate* 0,001, filter 3x3, *epoch* 20, diperoleh hasil akurasi *data training* sebesar 95% dan akurasi *data validation* sebesar 90%. Sedangkan untuk *data testing* yakni sebesar 93%.

No.	Nama Peneliti	Judul Penelitian	Tahun	Isi
5.	(Rohim, Sari and Tibyani 2019)	*Convolutional Neural Network* (CNN) untuk Pengklasifikasian Citra Makanan Tradisional	2019	Pada penelitian ini dilakukan pengklasifikasian terhadap citra makanan tradisional. Hasil penelitian ini menunjukkan dalam membangun arsitektur model *Convolutional Neural Network* untuk pengklasifikasian citra makanan trdisional membutuhkan 4 layer konvolusional, 4 *layer maxpooling*, dan 2 layer *fully connected*. Arsitektur tersebut didapatkan karena mendapatkan nilai *loss value* terkecil dengan nilai 0.000044 pada *epoch* ke 15 saat proses pembelajaran dan mendapatkan nilai 73% presisi, 69% *recall*,, dan 69% Fscore.
6.	(Sumardi 2019)	Implementasi Algoritma CNN Dalam Klasifikasi Gangguan Mata Menggunakan Pendekatan	2019	Penelitian ini bertujuan untuk mengklasifikasi gambar fundus berdasarkan kategori gangguan matanya, dengan menggunakan algoritma CNN. Berdasarkan hasil analisis yang dilakukan, diperoleh hasil arsitektur terbaik dengan

No.	Nama Peneliti	Judul Penelitian	Tahun	Isi
		*Image Processing (*Studi Kasus: Gambar Fundus *Maculopathy, Pathological myopia, Rhegmatogen eousRD,* dan Normal*)*		menggunakan skenario perbandingan *data train testing* 90%, ukuran kernel 3x3, *Optimizer* Adam, 300 *epoch*, *learning rate* sebesar 0,0001, *batch size* 50, dan menggunakan jenis gambar *color* (RGB) mampu memperoleh akurasi sebesar 95% dengan *loss* sebesar 11,4%, dan dapat dengan tepat memprediksi seluruh gambar pada kategori *maculopathy*, *pathological myopia*, dan normal, sedangkan *rhegmatogeneousRD* hanya berhasil diprediksi dengan tepat sebanyak 4 gambar dari total 5 gambar.

2.2 Penelitian Tentang Klasifikasi Gambar Jamur

Kemudian sebagai referensi dari objek penelitian ini, dirangkum beberapa penelitian terdahulu yang menggunakan gambar jamur sebagai objek penelitian. Berikut rangkuman penelitian terdahulu terkait dengan pengklasifikasian gambar jamur:

Tabel 2.2. Tinjauan Pustaka Penelitian Menggunakan Gambar Jamur

No.	Nama Peneliti	Judul Penelitian	Tahun	Isi
1.	(Lidasan and Tagacay 2018)	*Mushroom Recognition using Neural Network*	2018	Penelitian ini dilakukan untuk mengklasifikasikan citra jamur menggunakan *neural network* dengan *image processing* berdasarkan *order* dan *family* jamur. Penelitian ini juga dilakukan untuk mengethaui apakah jamur tersebut layak untuk dikonsumsi. Aplikasi yang digunakan adalah algoritma *GrabCut* untuk segmentasi gambar dan *Probabilistic Neural Network* (PNN) sebagai *classifier* yang melatih dan mengklasifikasikan gambar jamur. Penelitian ini menggunakan 133 gambar jamur sebagai data latih dan menghasilkan akurasi sebesar 92%.

No.	Nama Peneliti	Judul Penelitian	Tahun	Isi
2.	(Rahmat, et al. 2018)	*Fungus Image Identification using K-Nearest Neighbor*	2018	Penelitian ini dilakukan untuk mengidentifikasi jamur beracun menggunakan metode *K-Nearest Neighbor* (KNN). Gambar diolah melalui proses *pre-processing* dengan menggunakan *gray-scaling*, segmentasi gambar menggunakan *canny edge* dan *thresholding*, dan ekstraksi fitur dengan menggunkan metode *zoning*. Penelitian ini menggunakan 40 gambar jamur yang menghasilkan akurasi model sebesar 90%.
3.	(Kavitha, Suganthi and Jose 2018)	*Ensemble Deep Learning for Prediction of Palatable Mushrooms*	2018	Penelitian ini bertujuan untuk mengklasifikasikan jamur yang layak dikonsumsi dan jamur beracun berdasarkan bagian tudung, batang, dan warna jamur. Fokus utama penelitian ini adalah untuk meminimumkan *False Positive Rate* menggunakan metode *Convolutional Neural Network* (CNN) dengan beberapa *optimizer* seperti RMSProp, adagrad, dan adamax. Nilai

No.	Nama Peneliti	Judul Penelitian	Tahun	Isi
				akurasi yang dihasilkan oleh model adalah diantara 80%-85% dari data uji.
4.	(Verma and Dutta 2018)	*Mushroom Classification Using ANN and ANFIS Algorithm*	2018	Pada penelitian ini dilakukan klasifikasi untuk mengidentifikasi berbagai macam jamur yang layak dan tidak layak dikonsumsi. Teknik klasifikasi yang diimplementaskan adalah *Artificial Neural Network* (ANN), *Adaptive Neuro Fuzzy Inference System* (ANFIS), dan *Naïve Bayes*. Hasil analisis dievaluasi menggunakan akurasi, MAE, dan *kappa statistic*. Model ANFIS menunjukkan hasil yang lebih baik dari model lainnya dengan akurasi tertinggi dan MAE terendah.
5.	(Ottom, Alawad and Nahar 2019)	*Classification of Mushroom Fungi Using Machine Learning Techniques*	2019	Pada penelitian ini dilakukan pengklasifikasian terhadap citra jamur untuk mengidentifikasi jamur beracun dan jamur tidak beracun. Klasifikasi dilakukan dengan pendekatan beberapa

No.	Nama Peneliti	Judul Penelitian	Tahun	Isi
				Teknik seperti *Neural Network* (NN). *Support Vector Machine* (SVM), *Decision Tree*, dan *k-Nearest Neighbor* (kNN). Hasil dari analisis menunjukkan bahwa metode kNN memberikan model terbaik dengan akurasi sebesar 94%

2.3 Perbedaan Penelitian Sebelumnya dengan Penelitian Ini

Setelah mempelajari beberapa penelitian terdahulu yang berkaitan dengan metode dan objek yang digunakan pada penelitian ini, maka dapat diketahui perbedaan penelitian ini dengan penelitian sebelumnya yang disajikan dalam tabel berikut.

Tabel 2.3. Perbedaan Penelitian Sebelumnya dengan Penelitian Ini

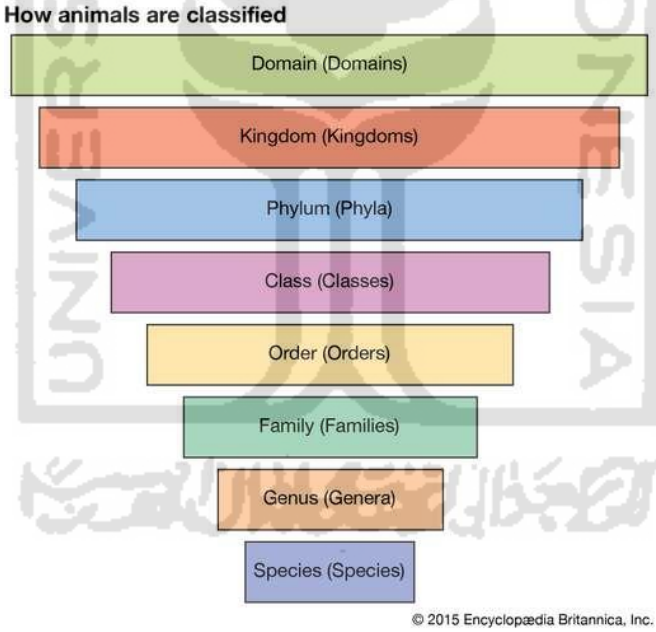
Nama Peneliti	Judul Penelitian	Tahun	Isi
Ocktavia Nurima Putri	Implementasi Metode CNN dalam Klasifikasi Gambar Jamur pada Analisis *Image Processing*	2020	Perbedaan penelitian ini dengan penelitian sebelumnya terletak pada gambar jamur yang diklasifikasikan, dimana peneliti-peneliti lainnya menggunakan gambar jamur secara umum, sedangkan gambar jamur pada penelitian ini diklasifikasikan berdasarkan genus. Arsitektur CNN untuk mendapatkan pengukuran hasil klasifikasi terbaik pada penelitian ini

Nama Peneliti	Judul Penelitian	Tahun	Isi
			menggunakan perbandingan parameter *epoch* sebesar 50, 100, 200, 300, dan 500, serta *optimizer*, yaitu Adam, RMSProp, dan SGD. Analisis yang dilakukan menghasilkan akurasi sebesar 62% menggunakan *optimizer* Adam dengan *epoch* 100 dan menggunakan skenario perbandingan data *train validation* 80% : 20%, ukuran kernel 3x3, *optimizer* Adam, 100 epoch, dan *learning rate* sebesar 0,001.

BAB 3. LANDASAN TEORI

3.1 Genus

Genus adalah tingkatan kategori dalam taksonomi yang digunakan dalam klasifikasi biologi yang berada dibawah famili dan diatas spesies. Genus merupakan kumpulan dari spesies yang memiliki karakteristik serupa. Pada tahun 2016 terdapat sekitar 510.000 nama genus yang telah dipublikasi. (Rees, et al. 2017) Dalam sistem tata nama binomial, nama suatu spesies makhluk hidup terdiri atas dua kata, yaitu nama genusnya (diawali dengan huruf kapital) dan nama penunjuk spesiesnya dengan ditulis atau cetak miring. Misalnya, *Agaricus bisporus*, nama ilmiah untuk spesies jamur, menandakan bahwa jamur tersebut tergolong ke dalam genus *Agaricus*. (Britannica, Genus 2017)



Gambar 3.1. Struktur Taksonomi (Britannica, Genus 2017)

3.2 Jamur

Jamur adalah tumbuhan tanpa klorofil dan hidup bergantung dengan makhluk hidup lainnya. Secara umum, jamur dapat didefinisikan sebagai organisme *eukariotik* yang mempunyai inti dan organel. Jamur tersusun dari *hifa* yang merupakan benang-benang sel tunggal panjang, sedangkan kumpulan *hifa* disebut

dengan *miselium*. Miselium merupakan massa benang yang cukup besar dibentuk dari *hifa* yang saling membelit pada saat jamur tumbuh. Jamur mudah dikenal dengan melihat warna miseliumnya (Volk and Wheeler 1993).

Bagian penting tubuh jamur adalah suatu struktur berbentuk tabung menyerupai seuntai benang panjang, ada yang tidak bersekat dan ada yang bersekat. Hifa dapat tumbuh bercabang-cabang sehingga membentuk jaring-jaring, bentuk ini dinamakan miselium. Pada satu koloni jamur ada *hifa* yang menjalar dan ada hifa yang menegak. Biasanya *hifa* yang menegak ini menghasilkan alat-alat pembiak yang disebut *spora*, sedangkan *hifa* yang menjalar berfungsi untuk menyerap nutrien dari substrat dan menyangga alat-alat reproduksi. *Hifa* yang menjalar disebut *hifa vegetatif* dan *hifa* yang tegak disebut *hifa fertil*. Pertumbuhan *hifa* berlangsung terus-menerus dibagian apikal, sehingga panjangnya tidak dapat ditentukan secara pasti. Diameter *hifa* umumnya berkisar 3-30 µm. Jenis jamur yang berbeda memiliki diameter *hifa* yang berbeda pula dan ukuran diameter itu dapat dipengaruhi oleh keadaan lingkungan (Carlile and Watkinson 1994).

Setiap jamur mencakup di dalam salah satu dari kategori taksonomi, dibedakan atas dasar tipe spora, morfologi *hifa* dan siklus seksualnya. Kelompok-kelompok ini antara lain *Zygomycetes*, *Ascomycetes*, *Basidiomycetes* dan *Deuteromycetes* (Mc-Kane 1996). Manusia telah mengkonsumsi jamur sebagai bahan pangan, obat-obatan, dan kuliner sejak zaman kuno. Beberapa jamur sengaja dibudidayakan, tetapi sebagian besar jamur yang dapat dimakan dikumpulkan dari alam. Orang Romawi dan Yunani memperlakukan jamur sebagai jenis makanan khusus (Miles and Chang 2004).

Untuk menentukan jamur ke dalam kelas dapat dikonsumsi atau beracun sangat sukar dilakukan. Salah satu cara untuk menentukannya adalah dengan mengetahui dengan tepat spesies dari jamur tersebut. Pengalaman sangat menentukan dalam mengenali karakteristik perbedaan jamur yang dapat dikonsumsi dengan spesies beracun.

3.2.1 *Agaricus*

Agaricus adalah genus yang terkenal luas akan jamurnya yang dapat dikonsumsi dan memiliki sebagian kecil spesies yang beracun jika dikonsumsi (Kerrigan, et al. 2006, Bau, Bao and Li 2014, Zhao, et al. 2012a). Ciri-ciri *Agaricus* yaitu memiliki *carpophores* berbentuk payung, permukaan *pileus* yang bersisik, insang yang terpisah dari batang, dan cincin yang melekat pada batang. *Agaricus* merupakan fungi *saprotrof* yang hidup pada kayu, hutan, halaman, tepi jalan, tanah lapang, dan halaman berumput. *Agaricus* dapat ditemui dari dasar laut hingga area pegunungan (Cappelli 1984), dan juga di beberapa area yang gersang.

Gambar 3.2. Jamur *Agaricus* (See 2018)

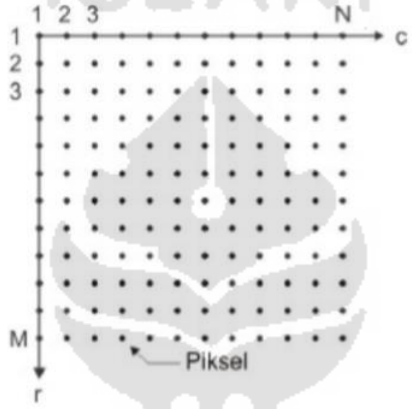
3.2.2 *Amanita*

Amanita adalah genus dari beberapa spesius jamur dengan famili *Amanitaceae.* Beberapa spesies dari genus *Amanita* beracun bagi manusia. Karakteristiknya antara lain adalah memiliki spora putih, cincin pada batang yang yang terletak dibawah tudung, dan kerudung yang membentuk *volva.* Genus jamur ini juga dikenal memiliki jamur yang paling beracun yaitu, jamur *destroying angel* dan *death caps.* (Britannica, Amanita 2017)

Gambar 3.3. Jamur *Amanita* (See 2018)

3.3 Citra Digital

Sebuah gambar didefinisikan sebagai suatu fungsi dua dimensi $f(x,y)$ dimana x dan y adalah koordinat spasial, dan amplitudo fungsi pada setiap pasangan titik (x,y) merupakan intensitas atau kecerahan gambar pada titik tersebut (Gonzales 2004). Gambar digital dikodekan dalam bentuk matriks yang mana indeks baris dan kolom menyatakan suatu titik pada gambar tersebut dan elemen matriksnya (piksel) menyatakan tingkat keabuan pada titik tersebut. Berikut ilustrasi koordinat suatu piksel dari sebuah gambar dengan x adalah baris dan y adalah kolom.



Gambar 3.4. Ilustrasi koordinat suatu piksel dari sebuah gambar (Sumardi 2019)

Dari gambar 3.4 diatas, diketahui bahwa piksel yang ditunjukkan dengan tanda panah berada pada koordinat (M,4), jika direpresentasikan dalam bentuk matriks dapat dilihat seperti pada persamaan berikut (Yusuf 2017):

$$f = \begin{bmatrix} f(1,1) & f(1,2) & \cdots & f(1,N) \\ f(2,1) & f(2,2) & \cdots & f(2,N) \\ \vdots & \vdots & \vdots & \vdots \\ f(M,1) & f(M,2) & \cdots & f(M,N) \end{bmatrix} \quad (3.1)$$

Keterangan :

M = Banyaknya kolom pada *array* citra

N = Banyaknya baris pada *array* citra

3.3.1 Pengolahan Citra Digital

Pengolahan citra digital (*image processing*) merupakan proses mengolah piksel-piksel dalam citra digital untuk suatu tujuan tertentu. Sebuah gambar disebut

dengan citra digital apabila gambar yang dihasilkan berasal dari proses sebuah komputer, kamera, *scanner* atau perangkat elektronik lainnya. Pengolahan citra digital diproses oleh komputer dengan menggunakan algoritma. Citra digital direpesentasikan dengan matriks, sehingga pengolahan pada citra digital pada dasarnya memanipulasi elemen-elemen matriks yang berupa piksel (A'la 2016).

Image processing adalah suatu bidang yang berorientasi pada pengolahan gambar yang telah lama berkembang. Teknik-teknik yang dilakukan dalam pengolahan gambar digunakan untuk mentransformasi suatu gambar ke gambar yang lain, yang mana untuk perbaikan informasi dilakukan oleh manusia melalui penyusunan algoritmanya. Algoritma pengolahan gambar sangat berguna diawal perkembangan sistem visual, yang dapat dilakukan untuk mengolah suatu gambar sebelum diolah atau dianalisis lebih jauh yakni seperti penajaman gambar, menonjolkan fitur tertentu dari suatu gambar, mengompresi gambar dan mengoreksi gambar yang tidak jelas atau blur (U. Ahmad 2005).

3.4 *Artificial Intelligence*

Kecerdasaran Buatan atau *Artificial Intelligence* (AI) adalah teknik yang digunakan untuk meniru kecerdasan yang dimiliki oleh makhluk hidup maupun benda mati untuk menyelesaikan sebuah persoalan. Kecerdasan buatan (*artificial intelligence*) merupakan sebuah studi tentang bagaimana membuat komputer melakukan hal-hal yang pada saat ini dilakukan lebih baik oleh manusia (Rich and Knight 1991). Menurut John McCarthy (1956) *Artificial Intelligence* (AI) bertujuan untuk mengetahui atau memodelkan proses berpikir manusia dan mendesain mesin sehingga bisa menirukan perilaku manusia yang dikutip dari penelitian (Nurhikmat 2018).

Berdasarkan penelitian yang dikutip dari (Pujoseno 2018), menurut H. A. Simon (1987) kecerdasan buatan (*artificial intelligence*) merupakan kawasan penelitian, aplikasi dan instruksi yang terkait dengan pemrograman komputer untuk melakukan sesuatu hal yang dalam pandangan manusia adalah cerdas. Dengan demikian diharapkan komputer bisa menirukan beberapa fungsi otak manusia, seperti berpikir dan belajar. Sistem kecerdasan komputer ini bisa dilatih atau

dilakukan pembelajaran terhadap komputer itu sendiri atau biasa disebut dengan *machine learning.*

3.4.1 *Deep Learning*

Deep learning merupakan salah satu jenis algoritma jaringan saraf tiruan yang menggunakan *metadata* sebagai *input* dan mengolahnya menggunakan sejumlah lapisan tersembunyi (*hidden layer*) transformasi non-linier dari data masukan untuk menghitung nilai *output*. Algortima pada *deep learning* memiliki fitur yang unik, yaitu sebuah fitur yang mampu mengekstraksi secara otomatis. Hal ini berarti algoritma yang dimilikinya secara otomatis dapat menangkap fitur yang relevan sebagai keperluan dalam pemecahan suatu masalah. Algortima semacam ini sangat penting dalam sebuah kecerdasan buatan karena mampu mengurangi beban pemrograman dalam memilih fitur yang eksplisit. Algortima ini dapat digunakan untuk memecahkan permasalahan yang perlu pengawasan (*supervised*), tanpa pengawasan (*unsupervised*), dan semi terawasi (*semi supervised*) dalam berbagai aplikasi seperti pengenalan citra, pengenalan suara, klasifikasi teks, dan sebagainya. Dalam jaringan saraf tiruan tipe *deep learning* setiap lapisan tersembunyi bertanggung jawab untuk melatih serangkaian fitur unik berdasarkan *output* dari jaringan sebelumnya. Algortima ini akan menjadi semakin komplek dan bersifat abstrak ketika jumlah lapisan tersembunyi (*hidden layer*) semakin bertambah banyak. Jaringan saraf yang dimiliki oleh *deep learning* terbentuk dari hirarki sederhana dengan beberapa lapisan hingga tingkat tinggi atau banyak lapisan (*multi layer*). Berdasarkan hal itulah *deep learning* dapat digunakan untuk memecahkan masalah kompleks yang lebih rumit dan terdiri dari sejumlah besar lapisan transformasi non-linier (LeCun, Bengio and Hinton 2015).

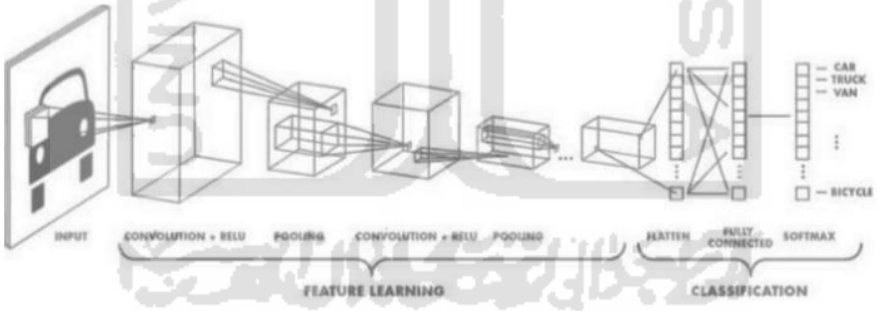
3.4.2 *Machine Learning*

Machine Learning atau pembelajaran mesin adalah pendekatan dalam *artificial intelligence* yang banyak digunakan untuk menggantikan atau menirukan perilaku manusia untuk menyelesaikan masalah atau melakukan otomatisasi. Sesuai namanya, *machine learning* mencoba menirukan bagaimana proses manusia

atau makhluk cerdas belajar dan menggeneralisasi. Setidaknya ada dua aplikasi utama dalam *machine learning* yaitu, klasifikasi dan prediksi (A. Ahmad 2017).

3.5 *Convolutional Neural Network*

Convolutional Neural Network atau CNN merupakan metode *neural network* yang paling terkenal dan banyak diminati. CNN mampu memproses data berdimensi tinggi seperti video dan gambar. Cara kerja CNN hampir serupa dengan *neural network* pada umumnya, hanya saja perbedaan yang paling utama yakni menggunakan kernel 2 dimensi atau dimensi tinggi pada tiap unit dalam lapisan CNN yang akan dilakukan konvolusi. Kernel dalam CNN digunakan untuk menggabungkan fitur spasial dengan bentuk spasial yang menyerupai *media input*. Kemudian CNN menggunakan berbagai parameter untuk mengurangi jumlah variabel agar mudah untuk dipelajari (Khan, et al. 2018). Nama "*Convolutional Neural Network*" menunjukkan bahwa jaringan tersebut menggunakan operasi matematika yang disebut konvolusi (Goodfellow, Bengio and Courville 2016). Kemudian CNN dilatih untuk mempelajari fitur objek untuk memprediksinya. Ilustrasi CNN secara umum dapat dilihat pada gambaran arsitektur berikut,



Gambar 3.5. Gambaran arsitektur CNN (Prabhu 2018)

Berdasarkan gambar 3.5 diketahui bahwa terdapat 2 bagian lapisan arsitektur, yakni antara lain (Utami 2018):

a. *Feature Learning (Feature Extraction Layer)*

   Pada bagian ini terdapat lapisan yang berguna untuk menerima *input* gambar secara langsung diawal dan memprosesnya sampai menghasilkan *output* data *multidimension array*. Lapisan yang ada pada proses ini terdiri dari lapisan konvolusi dan lapisan *pooling*, dimana setiap proses lapisan tersebut akan

menghasilkan *feature maps* berupa angka-angka yang merepresentasikan gambar untuk kemudian diteruskan pada bagian lapisan klasifikasi.

b. Lapisan Klasifikasi (*Classification Layer*)

Lapisan ini terdiri dari beberapa lapisan yang berisi neuron yang terkoneksi penuh (*fully connected*) dengan lapisan lain. Lapisan ini menerima *input* dari *output layer* bagian *feature learning* yang kemudian di proses pada *flatten* dengan tambahan beberapa *hidden layer* pada *fully connected* hingga menghasilkan *output* berupa akurasi klasifikasi dari setiap kelas.

3.5.1 *Convolution Layer*

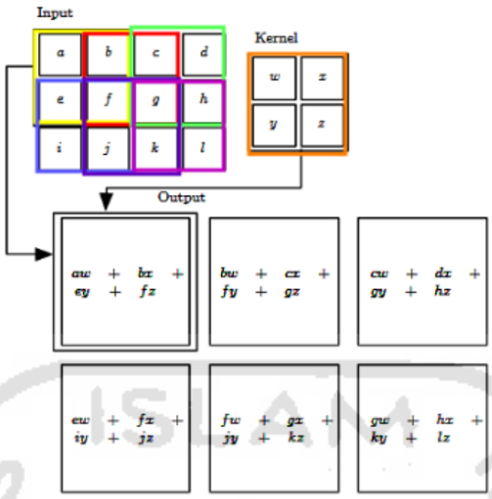
Convolution layer atau lapisan konvolusi adalah jenis operasi linier khusus. Jaringan konvolusional hanyalah jaringan saraf yang menggunakan konvolusi sebagai ganti dari penerapan matriks umum di setidaknya satu dari lapisannya. Konvolusi merupakan salah satu operasi matematis yang digunakan pada pengolahan gambar. Operasi ini menerapkan fungsi keluaran (*output*) sebagai *feature maps* dari masukan (*input*) gambar. *Input* dan *output* ini dapat dilihat sebagai dua argumen yang mempunyai nilai riil. Secara khusus, operasi konvolusi dapat ditulis dengan rumus berikut:

$$s(t) = (x * w)(t) \tag{3.2}$$

Konvolusi didefinisikan untuk setiap fungsi integral yang didefinisikan diatas, dan dapat digunakan untuk tujuan lain selain mengambil rata-rata tertimbang. Fungsi $s(t)$ memberikan *output* tunggal yaitu *feature maps*, argumen pertama yaitu input yang merupakan x dan argumen kedua w sebagai kernel atau filter. Jika dilihat *input* sebagai gambar dua dimensi, maka dapat diasumsikan bahwa t sebagai piksel dan menggantinya dengan i dan j. Maka dari itu, operasi untuk konvolusi ke dalam input lebih dari satu dimensi dapat ditulis sebagai berikut:

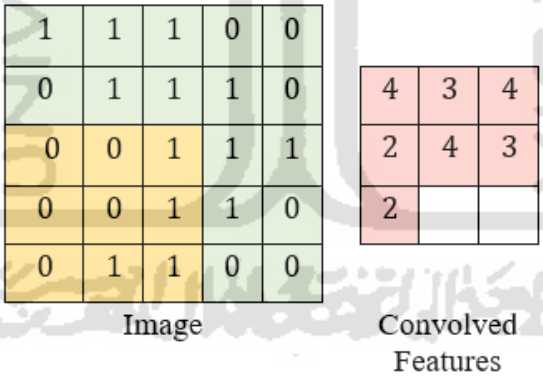
$$S_{(i,j)} = (I * K)_{(i,j)} - \sum_m \sum_n I_{(i-m,j-n)} K_{(m,n)} \tag{3.3}$$

Untuk lebih mudah memahami proses penghitungan konvolusi yang dilakukan menggunakan rumusan 3.3 dapat diilustrasikan dengan gambar berikut, Untuk lebih mudah memahami proses pengrhitungan konvolusi yang dilakukan menggunakan rumusan 3.3 dapat diilustrasikan dengan gambar berikut,



Gambar 3.6. Ilustrasi perhitungan konvolusi (Goodfellow, Bengio and Courville 2016)

Dalam pengolahan gambar, konvolusi berarti mengaplikasikan sebuah kernel (kotak kuning) pada gambar di semua *offset* yang memungkinkan seperti ilustrasi pada gambar berikut,



Gambar 3.7. Gambaran Operasi Konvolusi

Kotak hijau secara keseluruhan adalah gambar yang akan dilakukan konvolusi. Kernel bergerak dari sudut kiri atas ke kanan bawah. Sehingga hasil konvolusi dari gambar 3.7 dapat dilihat dari gambar di sebelah kanan. Tujuannya dilakukan konvolusi pada data gambar yaitu untuk mengekstraksi fitur dari gambar *input*. Konvolusi akan menghasilkan transformasi linear dari data *input* sesuai informasi spasial pada data. Bobot pada lapisan tersebut mengspesifikasi kernel konvolusi yang digunakan, sehingga kernel konvolusi dapat dilatih berdasarkan

input pada CNN (P, Wijaya and Soelaiman 2016). Kemudian dimensi output dari *feature maps* dapat dihitung menggunakan rumus berikut,

$$output\ f.maps\ = \frac{W-N+2P}{S} + 1 \quad (3.4)$$

Keterangan:

W = ukuran input

N = ukuran filter

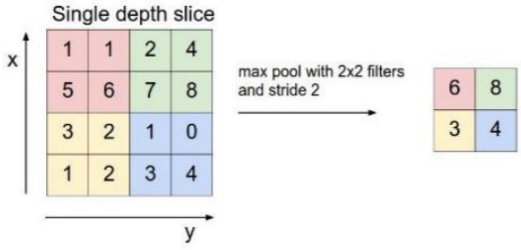
P = *zero padding*

S = *stride*

3.5.2 *Pooling Layer*

Pooling layer adalah lapisan fungsi untuk *feature maps* sebagai masukan dan mengolahnya dengan berbagai operasi statistik berdasarkan nilai piksel terdekat. Pada model CNN, lapisan *pooling* biasanya disisipkan secara teratur setelah beberapa lapisan konvolusi. Lapisan *pooling* yang dimasukkan di antara lapisan konvolusi secara berturut-turut dalam susunan arsitektur model CNN dapat secara progresif mengurangi ukuran volume *output* pada *feature maps*, sehinga dapat mengurangi jumlah parameter dan perhitungan di jaringan, dan untuk mengendalikan *overfitting*. Hal penting dalam pembuatan model CNN adalah dengan memilih banyak jenis lapisan *pooling* yang dalam hal ini dapat menguntungkan kinerja model (Lee, Gallagher and Tu 2015).

Lapisan *pooling* bekerja di setiap tumpukan *feature map* dan melakukan pengurangan pada ukurannya. Bentuk lapisan *pooling* umumnya dengan menggunakan filter dengan ukuran 2x2 yang diaplikasikan dengan langkah sebanyak dua dan beroperasi pada setiap irisan dari inputnya. Berikut ini adalah contoh gambar operasi *max-pooling*:



Gambar 3.8. Operasi *Max Pooling* (Pokharna 2016)

Kotak yang berwarna merah, hijau, kuning dan biru pada sisi kiri merupakan kelompok kotak yang akan dipilih nilai maksimumnya. Sehingga hasil dari proses tersebut dapat dilihat pada kumpulan kotak disebelah kanannya. Proses tersebut memastikan fitur yang didapatkan akan sama meskipun objek citra mengalami translasi (pergeseran). Penggunaan *pooling layer* pada CNN hanya bertujuan untuk mereduksi ukuran citra sehingga dapat dengan mudah digantikan dengan sebuah *convolution layer* dengan *stride* yang sama dengan *pooling layer* yang bersangkutan. *Stride* merupakan parameter yang menentukan berapa jumlah pergeseran filter. Jika nilai *stride* adalah satu, maka filter akan bergeser sebanyak satu piksel secara horizontal lalu vertikal. Semakin kecil *stride* yang digunakan, maka semakin detail informasi yang didapatkan dari sebuah *input*, namun membutuhkan komputasi lebih jika dibandingkan dengan *stride* yang besar.

3.5.3 *Fully Connected Layer*

Lapisan *fully connected layer* adalah suatu lapisan yang terbentuk dari kumpulan hasil proses konvolusi. Lapisan ini memiliki *input* dan proses sebelumnya untuk menentukan fitur manakah yang memiliki hubungan atau korelasi dengan kelas tertentu. Selain itu fungsi yang diperoleh dari *fully connected layer* berfungsi untuk menyatukan semua node menjadi satu dimensi (Nour Arrofiqoh and Harintaka 2018).

Umumnya proses *fully connected* ini terletak pada akhir arsitektur, prosesnya direpresentasikan sebagai perkalian matriks sederhana yang diikuti dengan penambahan vektor bias dan menerapkan fungsi non-linear sebagai berikut:

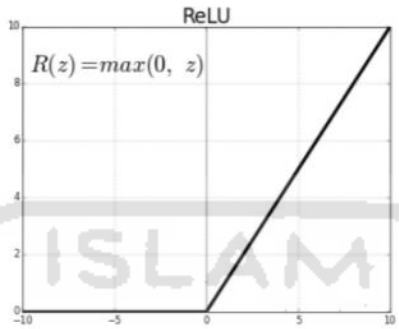
$$y = f(W^T x + b) \tag{3.5}$$

dimana y merupakan vektor *output* yang berisi fungsi dari W yakni matriks sederhana yang memiliki bobot koneksi antar unit, lalu dikalikan dengan x sebagai vektor *input*, dan ditambah dengan b sebagai vektor bias (Khan, et al. 2018).

3.5.4 Fungsi Aktifasi ReLu

Rectified linear unit, yang dikenal sebagai ReLU, adalah cara paling umum dan dasar untuk memperkenalkan non-linearitas ke dalam jaringan saraf. Fungsi ini hanya max (0, x) (Rohim, Sari and Tibyani 2019).

Pada dasarnya fungsi ReLU (*Rectified Linear Unit*) melakukan "*treshold*" dari 0 hingga *infinity*. Fungsi ini menjadi salah satu fungsi yang populer saat ini. Berikut ini grafik fungsi aktivasi ReLU:



Gambar 3.9. Fungsi aktivasi ReLU (Farrukh 2019)

Pada fungsi ini masukan dari neuron-neuron berupa bilangan negatif, maka fungsi ini akan menerjemahkan nilai tersebut kedalam nilai 0, dan jika masukan bernilai positif maka output dari neuron adalah nilai aktivasi itu sendiri. Fungsi aktivasi ini memiliki kelebihan yaitu dapat mempercepat proses konvigurasi yang dilakukan dengan *Stochastic Gradient Descent* (SGD) jika dibandingkan dengan fungsi sigmoid dan tanh. Namun aktivasi ini juga memiliki kelemahan yaitu aktivasi ini bisa menjadi rapuh pada proses training dan bisa membuat unit tersebut mati. (Nurhikmat 2018)

3.5.5 *Backpropagation*

Proses untuk memperbaharui nilai filter dan bobot pada jaringan adalah proes propagasi balik. Perhitungan perubahan nilai bobot dihitung dimulai dari lapisan *fully connected*. Pada lapisan ini perubahan bobot dicari dengan mencari derivatif *loss function* terhadap bobot (Zhang 2016). Pelatihan *backpropagation* meliputi tiga tahap. Tahap pertama adalah tahap maju (*forward*). Tahap ini menghitung maju tahap *layer input* sampai tahap *layer output* dengan menggunakan fungsi aktivasi yang telah ditentukan. Tahap kedua adalah tahap mundur (*backward*), pada tahap ini selisih antara *output* jaringan dengan target yang diinginkan merupakan kesalahan yang terjadi. Kesalahan tersebut dipropagasikan mundur, mulai dari garis yang terhubung langsung dengan setiap unit pada *layer output*. Kemudian tahap yang ketiga adalah tahap yang akan memodifikasi bobot

untuk menurunkan tingkat kesalahan yang terjadi (Jumarwanto, Hartanto and Prastiyanto 2009).

Algoritma pelatihan *back propagation* adalah sebagai berikut (Siang 2005):

Tahap pertama: Propagasi maju (*forward*)

a. Langkah 0: Inisialisasi bobot dengan memberikan nilai acak (nilai acak kecil -0.5 s/d 0.5).
b. Langkah 1: Jika kondisi penghentian belum terpenuhi, lakukan langkah 2-9
c. Langkah 2: Untuk setiap pasangan data pelatihan lakukan langkah 3-8
d. Langkah 3: Tiap unit masukan menerima sinyal dan meneruskannya ke unit tersembunyi di atasnya
e. Langkah 4 : Hitunglah semua output di unit tersembunyi tersebut $z_j (j = 1,2,\dots,p)$:

$$z\_net_j = V_{0j} + \sum_{i=1}^{n} x_i v_{ij} \quad (3.6)$$

$$z_j = f(z\_net_j) = \frac{1}{1+e^{-z\_net_j}} \quad (3.7)$$

Keterangan:

z_j = nilai unit tersembuny ke-j

V_{0j} = bobot *layer input* bias ke unit tersembunyi ke-j

x_i = unit *input* ke-i

v_{ij} = bobot unit *input* ke-i ke layer tersembunyi ke-j

z_j = nilai unit tersembunyi ke-j menggunakan fungsi aktivasi

e = nilai konstanta

f. Langkah 5: Hitunglah semua keluaran jaringan di unit y_k $(k = 1,2,\dots,m)$:

$$y\_net_k = W_{0k} + \sum_{j=1}^{p} z_j w_{jk} \quad (3.8)$$

$$y_k = f(y\_net_k) = \frac{1}{1+e^{-y\_net_k}} \quad (3.9)$$

Keterangan:

y_net_k = nilai unit *output* ke-k

W_{0k} = bobot unit tersembunyi bias ke unit output ke-k

w_{jk} = bobot unit tersembunyi ke-j ke unit output ke-k

y_k = nilai unit *output* ke-k menggunakan fungsi aktivasi sigmoid

Tahap kedua: Propagasi mundur

g. Langah 6: Hitung faktor δ unit keluaran berdasarkan error pada setiap y_k $(k = 1,2,\dots,m)$:

$$\delta_k = (t_k - y_k)\, f'(y\_net_k) = (t_k - y_k)\, y_k(1 - y_k) \quad (3.10)$$

Hitung suku perubahan bobot w_{kj} dengan laju percepatan α.

$$\Delta w_{jk}\, \alpha \delta_k z_j;\ k = 1,2,\dots,m;\ j = 0,1,2,\dots,p \quad (3.11)$$

Keterangan:

δ_k = nilai *error* unit *output*

tt_k = nilai target *output*

α = *learning rate*

Δw_{jk} = perubahan bobot unit tersembunyi ke-j ke unit *output* ke-k

h. Langkah 7: Hitung faktor δ unit tersembunyi berdasarkan error pada setiap unit tersembunyi z_j $(j = 1,2,\dots,p)$.

$$\delta\_net_j = \sum_{k=1}^{n} \delta_j w_{kj} \quad (3.12)$$

faktor δ unit tersembunyi:

$$\delta_j = \delta\_net_j\, f'(z\_net_j) = \delta\_net_j z_j(1 - z_j) \quad (3.13)$$

Hitung suku perubahan bobot v_{ji}:

$$\Delta v_{ij} = \alpha \delta_j x_i;\ j = 1,2,\dots,p;\ i = 0,1,2,\dots,n \quad (3.14)$$

Keterangan:

δ_j = nilai *error* unit tersembunyi

Δv_{ij} = perubahan bobot unit *input* ke-i ke unit tersembunyi ke-j

Tahap ketiga: Perubahan bobot

i. Langkah 8: Hitunglah semua perubahan pada bobot. Rumus perubahan bobot garis yang menuju ke unit keluaran:

$$w_{kj}(baru) = w_{kj}(lama) + \Delta w_{kj}\ (k = 1,2,\dots,m\ ;\ j = 0,1,2,\dots,p) \quad (3.15)$$

Perubahan bobot garis yang menuju ke unit tersebunyi:

$$v_{ji}(baru) = v_{ji}(lama) + \Delta v_{ji} \; (j = 1,2,\ldots,p \; ; \; i = 0,1,2,\ldots,n) \quad (3.16)$$

j. Langkah 9: Menguji apakah kondisi berhenti sudah terpenuhi. Kondisi berhenti ini terpenuhi jika nilai kesalahan yang dihasilkan lebih kecil dari nilai kesalahan referensi.

3.5.6 *Learning Rate*

Penggunaan parameter *learning rate* memiliki pengaruh penting terhadap waktu yang dibutuhkan untuk tercapainya target yang diinginkan. Secara perlahan akan mengoptimalkan nilai perubahan bobot dan menghasilkan *error* yang lebih kecil (Fajri 2011). Variabel *learning rate* menyatakan suatu konstanta yang bernilai antara 0.1-0.9. Nilai tersebut menunjukkan kecepatan belajar dari jaringannya. Jika nilai *learning rate* yang digunakan terlalu kecil maka terlalu banyak *epoch* yang dibutuhkan untuk mencapai nilai target yang diinginkan, sehingga menyebabkan proses *training* membutuhkan waktu yang lama. (Hermawan 2006). Tidak ada aturan yang pasti mengenai *learning rate*, tetapi semakin besar learning rate maka ketelitian jaringan akan semakin berkurang, tetapi berlaku sebaliknya. Apabila *learning rate*-nya semakin kecil, maka ketelitian jaringan akan semakin besar atau bertambah dengan konsekuensi prosesnya akan memakan waktu yang semakin lama.

3.5.7 *Categorical Cross-Entropy Loss*

Fungsi kerugian (atau fungsi tujuan, atau fungsi skor optimisasi) adalah salah satu dari dua parameter yang diperlukan untuk menyusun model. Tujuan yang dioptimalkan sebenarnya adalah rata-rata dari *array output* di semua titik data. (Usage of Loss Function 2017) Salah satu *loss function* yang biasa digunakan dalam proses klasifikasi yakni *Categorical Cross-Entropy Loss*.

3.5.8 *Softmax Classifier*

Softmax classifier adalah generalisasi untuk beberapa kelas. *Softmax classifier* memberikan *output* yang sedikit lebih intuitif (probabilitas kelas yang dinormalisasi) dan juga memiliki interpretasi probabilistik yang akan langsung dijelaskan (Rosebrock 2016). Tujuan dari lapisan ini yaitu untuk memprediksi *output* klasifikasi dalam bentuk nilai probabilitas, dimana nilai probabilitas kelas terbesar merupakan *output* prediksi kelas yang diperoleh. *Softmax* juga memberikan hasil yang lebih intuitif dan memiliki hasil interpretasi probabilistik yang lebih baik dibandingkan dengan algoritma klasifikasi lainnya. *Softmax* memungkinkan peneliti untuk menghitung nilai probabilitas untuk semua label. Hasil dari label yang ada, akan diambil sebuah vektor nilai yang mempunyai nilai riil dan merubahnya menjadi vektor dengan nilai antara nol dan satu. Jika semua hasil dijumlah maka akan bernilai satu

3.6 *Optimizer*

Algoritma optimisasi bertujuan untuk menemukan bobot optimal, meminimalkan kesalahan dan memaksimalkan akurasi. Selama proses pelatihan, parameter (bobot) model diubah untuk mencoba dan meminimalkan fungsi kerugian, agar mampu memprediksi seakurat mungkin. Namun bagaimana cara yang tepat, kapan, dan berapa banyaknya pengubahan itu masih belum bisa dipastikan, maka disinilah pengoptimal masuk. Mereka menyatukan fungsi kerugian dan parameter model dengan memperbarui model dalam menanggapi *output* dari fungsi kerugian. Dalam istilah yang lebih sederhana, pengoptimalisasi membentuk model yang kita punya ke dalam bentuk yang paling akurat dengan memanfaatkan bobotnya.

3.6.1 *Stochastic Gradient Descent (*SGD*)*

Gradient Descent adalah salah satu algoritma paling populer dalam melakukan optimasi pada *artificial neural network.* Algoritma ini digunakan untuk meng*update* sebuah parameter dalam hal ini adalah bobot (*weight*) dan bias. Algoritma ini cukup sederhana untuk dipahami. Pada dasarnya algoritma ini berfungsi untuk mengurangi inisial *weight* dengan “sebagian” dari nilai *gradient*

yang sudah didapatkan. *Gradient Descent* bekerja dengan cara meminimalkan fungsi *J(θ)* yang memiliki parameter θ dengan memperbarui parameter ke suatu arah menurun. Tujuan pengoptimalan dari algoritma ini untuk menemukan parameter yang dapat meminimalkan *loss function* (Ruder 2018).

Meskipun SGD cukup sederhana, namun dapat sangat efektif jika diatur dengan baik (dalam hal parameter *hyper step* dan momentumnya). Adapun rumusnya didefinisikan sebagai berikut (Scarpino,2018).

$$\theta_t + 1 \; = \; \theta_t \; - \; \eta \nabla L(\theta, \mathcal{D}^{(i)}) \; ; \; \mathcal{D}^{(i)} \; = \; \{(x^{(i)}, y^{(i)})\} \tag{3.17}$$

Dengan,

$$\nabla L(\theta, \mathcal{D}^{(i)}) \; = \; \sum_{i=1}^{m} (h(x^{(i)}; \theta) \; - \; y^{(i)})^2 \tag{3.18}$$

Dimana $h(x; \theta) \; = \; \theta^T x$ merupakan model regresi linear, sedangkan $\theta \; \in \; RN$ adalah parameter model, dan $\mathcal{D}^{(i)} \; = \; \{(x^{(i)}, y^{(i)})\}_{i=1}^{m}$ adalah sampel-sampel dalam data pelatihan.

3.6.2 RMSProp

Root Mean Square Propagation atau RMSprop adalah versi khusus Adagrad yang dikembangkan oleh Profesor Geoffrey Hinton di kelas jaringan sarafnya. Alih-alih membiarkan semua gradien terakumulasi untuk momentum, itu hanya mengakumulasi gradien di jendela tetap. RMSprop mirip dengan Adaprop, yang merupakan pengoptimal lain yang berupaya menyelesaikan beberapa masalah yang dibiarkan terbuka oleh Adagrad. RMSprop adalah pengoptimal yang memanfaatkan besarnya gradien terbaru untuk menormalkan *gradient*, yang menjaga rata-rata bergerak di atas gradien *root mean square* oleh karena itu disebut dengan Rms. $f' \; (\theta_t)$ menjadi turunan dari *loss* sehubungan dengan parameter pada langkah waktu t. Dalam bentuk dasarnya, diberikan tingkat langkah α dan istilah peluruhan γ dilakukan pembaruan berikut:

$$r_t \; = \; (1 \; - \; \gamma) \, f' \; (\theta_t)^2 \; + \; \gamma r_t - 1 \tag{3.19}$$

$$v_{\,t+1} = \frac{\alpha}{\sqrt{r_t}} f' \; (\theta_t) \tag{3.20}$$

$$\theta_t + 1 \; = \; \theta_t \; - \; v_{t+1} \tag{3.21}$$

Keterangan:

α: tingkat pembelajaran awal (*learning rate*)

r_t: rata-rata eksponensial dari kotak gradien

θ_t: gradien pada waktu t sepanjang v^j

RMSProp memiliki beberapa keunggulan yakni merupakan pengoptimal yang sangat kuat yang memiliki informasi kelengkungan semu. Selain itu, dapat menangani tujuan stokastik dengan sangat baik, sehingga berlaku untuk pembelajaran *batch mini* (documentation 2013).

3.6.3 Adam

Adam adalah algoritma pengoptimalan yang dapat digunakan sebagai ganti dari prosedur *stochastic gradient descent* klasik untuk memperbarui *weight network* secara iteratif berdasarkan data training. Adam adalah algoritma yang populer di bidang *deep learning* karena ia mencapai hasil yang baik dengan cepat.

$$g_t = \nabla_{\theta_{t-1}} L(\theta_{t-1}) \tag{3.22}$$

$$m_t = (1-\beta) g_t + \beta m_{t-1} \tag{3.23}$$

$$\hat{m}_t = \frac{m_t}{1-\beta_t} \tag{3.24}$$

$$\hat{v}_t = \frac{v_t}{1-\gamma_t} \tag{3.25}$$

$$\theta_{t-1} = \theta_t - \alpha \frac{m_t}{\sqrt{\hat{v}_t + \varepsilon}} \tag{3.26}$$

Hasil empiris menunjukkan bahwa Adam bekerja dengan baik dalam praktiknya dan lebih baik dibandingkan dengan *stochastic optimization method* lainnya (Brownlee 2017).

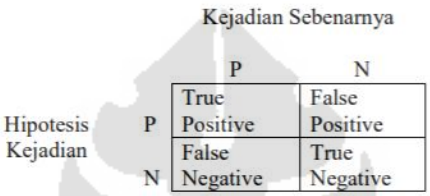
3.7 *Confusion Matrix*

Penentuan baik atau tidaknya performa suatu model klasifikasi dapat dilihat dari parameter pengukuran performanya, yaitu tingkat akurasi, *recall*, dan presisi.

a. *Accuracy*, persentasi keakuratan *classifier* dalam mengklasifikasi data uji dengan benar.
b. *Error rate (misclassification rate)* dari *classifier* adalah 1-*accuracy.*
c. *Precision* merupakan ukuran ketepatan untuk mengetahui persentase *tuple* yang diberi label positif dan benar-benar positif.

d. *Recall* atau *true positive rate* merupakan porsi dari *tuple* positif yang diklasifikasi dengan benar.
e. Alternatif untuk menggabungkan *precision* dan *recall* menjadi satu ukuran adalah dengan pendekatan *f-measure* yang juga dikenal sebagai *f1-score*. *F-measure* menilai ketepatan dan kekokohan sebuah *classifier* dengan rata-rata harmonic dari nilai *precision* dan *recall*-nya.

Untuk menghitung faktor-faktor tersebut diperlukan sebuah matrik yang biasa disebut *confusion matrix*. Salah satu *confusion matrix* yang kerap digunakan dalam pengukuran dapat dilihat pada Gambar 3.10.



Gambar 3.10. *Confusion Matrix* (Fawcett 2006)

Berdasarkan gambar 3.10 terdapat beberapa nilai didalam matriks yaitu "*True Positive*" (TP), "*True Negative*" (TN), "*False Positive*" (FP), dan "*False Negative*" (FN), seluruh kemungkinan kejadian sebenarnya positif (P) dan seluruh kemungkinan kejadian sebenarnya negatif (N). Nilai tersebut dapat digunakan untuk menghitung akurasi.

3.8 Akurasi

Akurasi adalah sebuah matrik untuk mengevaluasi hasil dari model klasifikasi. Akurasi merupakan pembagian dari prediksi model yang dianggap benar dengan jumlah total yang diprediksi, perhitungan tersebut didefinisikan sebagai berikut (Klasifikasi: Akurasi 2018):

$$Akurasi = \frac{Jumlah\ prediksi\ benar}{Jumlah\ total\ prediksi} \quad (3.14)$$

BAB 4. METODOLOGI PENELITIAN

4.1 Populasi dan Sampel Penelitian

Populasi dan sampel dalam penelitian ini adalah seluruh gambar jamur yang telah dikategorikan berdasarkan *genus*-nya. Terdapat 2 kategori genus jamur yang digunakan, yaitu *Agaricus* dan *Amanita*. Gambar jamur tersebut disortir dan diambil sampel sebanyak 260 gambar yang kemudian disortir sebanyak 208 gambar untuk data *training* dan 52 gambar untuk data *validation* dari setiap kategorinya, sehingga total data yang digunakan adalah sebanyak 520 gambar jamur.

4.2 Jenis dan Sumber Data

Data yang digunakan dalam penelitian ini adalah data sekunder. Data diambil dari halaman *Mushroom's Classification – Common Genus's Image* di situs *Kaggle* (www.kaggle.com) yang diakses pada tanggal 1 Januari 2020. Dari situs tersebut diperoleh data berupa gambar dari 2 macam genus jamur yang umum tumbuh di Eropa Selatan. Data berasal dari *Mycologist's Society of Northen Europe* yang dirilis pada tahun 2018 dan dapat digunakan secara bebas.

4.3 Variabel Penelitian

Adapun macam-macam variabel yang digunakan dalam penelitian ini ialah sebagai berikut:

Tabel 4.1. Definisi Variabel Penelitian

Gambar	Variabel	Definisi Operasional
	*Agaricus*	*Agaricus* adalah jenis jamur dengan tudung berdaging, spora berwarna cokelat kehitaman, Anggota *Agaricus* juga memiliki batang atau *stipe*, yang mengangkatnya di atas objek di mana jamur tumbuh, atau substrat,

Gambar	Variabel	Definisi Operasional
		dan cadar parsial, yang melindungi insang berkembang dan kemudian membentuk cincin atau anulus pada tangkai.
	*Amanita*	*Amanita* adalah jamur yang dikenal indah namun beracun. Tudungnya berwarna merah, jingga, hingga kecoklatan dan memiliki *spora* berwarna putih. Tudung *Amanita* cenderung kering. Spesies *Amanita* pada umumnya memiliki insang putih dan cincin pada batangnya.

4.4 Metode Analisis Data

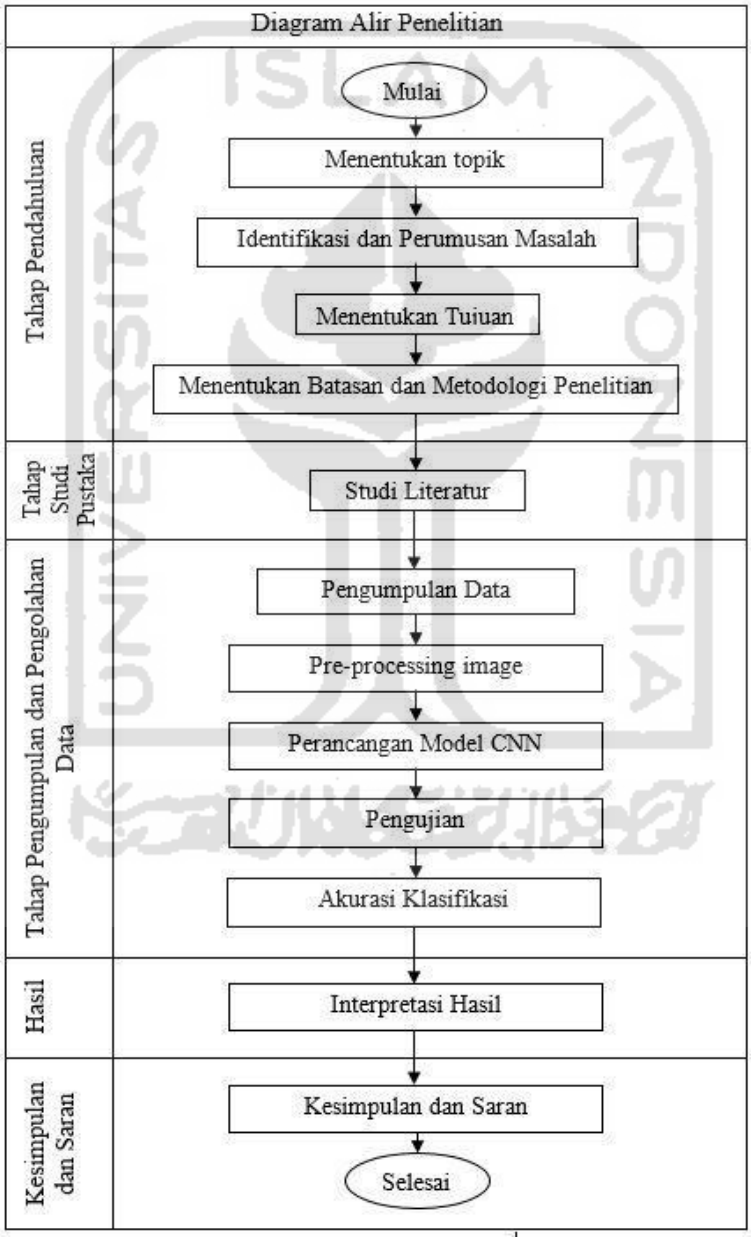
Pada penelitian ini digunakan metode untuk mengklasifikasi gambar jamur dengan algoritma *Convolutional Neutral Network* (CNN) yang merupakan salah satu metode dalam *Deep Learning* yang terkenal mampu mengklasifikasi data gambar dengan baik. Proses untuk mengolah algoritma CNN dibantu oleh *software* Python 3.6.2 dengan *package* Keras.

Gambar jamur akan diklasifikasikan menggunakan algoritma CNN yang akan mengeksekusi data dengan operasi konvolusinya hingga membentuk suatu pola. CNN memiliki banyak lapisan karena merupakan sistem *deep learning*, lapisan-lapisan yang terdapat dalam arsitektur CNN antara lain berisi lapisan konvolusi dan aktivasi, *pooling*, *flatten*, *fully connected*, dan *output*, yang mana dikelompokkan menjadi 2 tahapan, yakni *feature learning* dan *classification*. Pembuatan arsitektur CNN yang baik akan berdampak pada kemampuan hasil klasifikasi gambar untuk semua kategori, oleh karena itu pada penelitian ini menekankan untuk melakukan perbandingan parameter sebagai upaya

mendapatkan arsitektur yang terbaik. Adapun arsitektur yang digunakan dalam penelitian ini ialah seperti pada gambar 4.1 sebagai berikut,

4.5 Tahapan Penelitian

Langkah atau tahapan yang dilakukan pada penelitian ini digambarkan melalui Gambar 4.1 berikut ini :



Gambar 4.1. Diagram Alur Penelitian

Berdasarkan diagram alir penelitian diatas, diketahui bahwa terdapat 5 tahapan yang dilakukan dalam penelitian ini, yakni:

1. Tahap pendahuluan merupakan tahapan awal dalam penelitian ini. Langkah pertama yang dilakukan adalah menentukan topik. Dalam hal ini, penelitian yang dilakukan didasarkan pada kurangnya pengetahuan tentang klasifikasi jamur berdasarkan genus. Selanjutnya, dilakukan identifikasi dan perumusan masalah dimana setelah peneliti mengetahui apa topik yang ingin diangkat, peneliti juga harus mengetahui apa saja pokok-pokok permasalahan yang dapat diselesaikan, dan apa saja yang dapat mendasari adanya penelitian ini. Kemudian barulah ditentukan tujuannya, untuk menentukan tujuan sendiri perlu adanya kesesuaian dari hasil perumusan masalah, tujuan diharapkan mampu menjawab setiap permasalahan yang ada dari topik yang diangkat. Setelah itu, dibuat batasan masalah dan merancang metodologi penelitian agar penelitian ini terarah pada tujuan yang ada, dan merancang metodologi akan memudahkan peneliti untuk memulai langkah-langkah dan mempersiapkan mulai dari hal kecil yang dibutuhkan oleh penelitian ini misal: *software* yang digunakan, data yang digunakan, bagaimana data tersebut harus diolah, dan bagaimana tahapan analisis yang efektif untuk kemudian dapat dijelaskan dengan runtut dan jelas.
2. Kemudian tahapan selanjutnya ialah melakukan studi literatur atau studi pustaka, tahapan ini merupakan tahapan yang perlu dilakukan oleh setiap peneliti, mengingat perlunya untuk memperkaya referensi dibidang yang akan kita teliti. Melihat penelitian lain yang telah dilakukan dan hampir serupa dapat memudahkan peneliti untuk mengembangkan dan menemukan hal-hal baru lainnya yang dapat di eksplor.
3. Pada tahap ketiga dilakukan pengumpulan dan pengolahan data. Pertama dilakukan pengumpulan data sesuai dengan topik yang telah ditentukan, kemudian melakukan perbaikan data atau mempersiapkan data tersebut agar dapat diproses oleh algoritma, perbaikan data ini disebut dengan *pre-processing*, langkah pada *pre-processing* sendiri meliputi penyeragaman

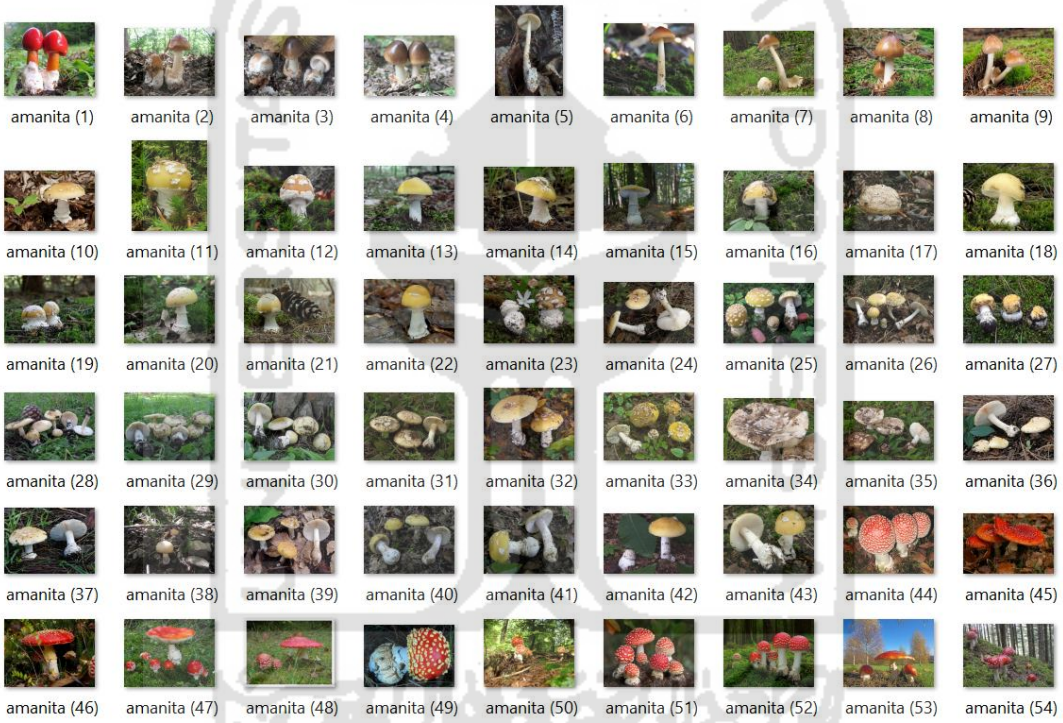
ukuran gambar yang akan di-*input*. Lalu diteruskan dengan merancang arsitektur CNN yang terbaik, umumnya dalam CNN memiliki 2 tahapan, yaitu tahap *feature learning* dan *classification*. *Input* gambar pada model CNN menggunakan citra yang berukuran 64x64x3. Angka tiga yang dimaksud adalah sebuah citra yang memiliki 3 *channel* yaitu *Red*, *Green*, dan *Blue* (RGB). Citra masukan kemudian akan diproses terlebih dahulu melalui proses konvolusi dan proses *pooling* pada tahapan *feature learning*. Jumlah proses konvolusi pada rancangan ini memiliki dua lapisan konvolusi. Setiap konvolusi memiliki jumlah filter dan ukuran kernel yang berbeda. Kemudian dilakukan proses *flatten* atau proses mengubah *feature map* hasil *pooling layer* kedalam bentuk *vector*. Proses ini biasa disebut dengan tahap *fully connected layer*. Setelah mendapatkan arsitektur terbaik, selanjutnya dilakukan pengujian menggunakan gambar yang telah melalui *pre-processing*. Lalu setelah itu akan diperoleh hasil akurasi terbaik dari gambar-gambar tersebut, yang mana hasil yang terbaik dengan kriteria akurasi yang tinggi akan digunakan modelnya untuk memproses data uji. Hingga akhirnya diperoleh hasil final untuk akurasi data uji, yang mana digunakan sebagai penentu atas kemampuan CNN mengklasifikasi terhadap 2 kategori gambar genus jamur.

4. Setelah itu pada tahap interpretasi hasil akan diuraikan kemampuan model CNN dalam mengklasifikasi gambar genus jamur dengan tepat sesuai dengan jumlah data yang ada.
5. Tahap akhir ialah membuat kesimpulan atas keseluruhan rangkaian penelitian yang telah dilakukan, dan kesimpulan hasil yang diperoleh sesuai dengan tujuan dan rumusan masalah yang telah ditentukan diawal. Tak lupa pula memberikan saran-saran sebagai pertimbangan atas kekurangan pada penelitian ini untuk dilakukan penelitian lanjutan atau dikembangkan kembali selanjutnya yang dapat berguna bagi peniliti selanjutnya.

BAB 5. HASIL DAN PEMBAHASAN

5.1 Rancangan Pengujian

Penelitian ini menggunakan 2 kategori gambar genus jamur, yakni *Agaricus* dan *Amanita*. *Dataset* yang telah siap akan dibagi menjadi data *training* dan *testing*. Data *training* dan *testing* dibuat dalam 2 *file* terpisah dimana di dalamnya terdapat *file* yang berisi masing-masing kategori gambar sesuai dengan jumlah proporsinya.



Gambar 5.1. Pengelompokkan Gambar Jamur

Peneliti menetapkan setiap kategori menggunakan data sebanyak 260 dari masing-masing genus jamur dimana perbandingan skenario datanya yakni 80% : 20%. Perbandingan data tersebut didasarkan pada *pareto principle* yang umum digunakan dalam *data mining*, dimana prinsip tersebut menyatakan bahwa 80% kejadian dihasilkan dari 20% sisanya. Maka total data *training* yang digunakan adalah (260 x 80) : 100 = 208, sehingga data *validation* yang digunakan adalah sebesar 260 – 208 = 52.

Tabel 5.1. Skenario pembagian data

Data *Training*	Data validasi
80%	**20%**
208	52

Model terbaik CNN dapat ditentukan dengan membandingkan beberapa paremeter demi tercapainya hasil klasifikasi terbaik. Parameter yang dimaksud antara lain adalah seperti perbandingan jumlah *epoch*¸ jenis *optimizer*, dan jumlah data *train* dan *validation.*

```
"""
Parameters
"""

img_width, img_height = 64,64
batch_size = 30
samples_per_epoch = 416
validation_steps = 104
nb_filters1 = 32
nb_filters2 = 64
conv1_size = 3
conv2_size = 3
pool_size = 2
classes_num = 2
lr = 0.001
```

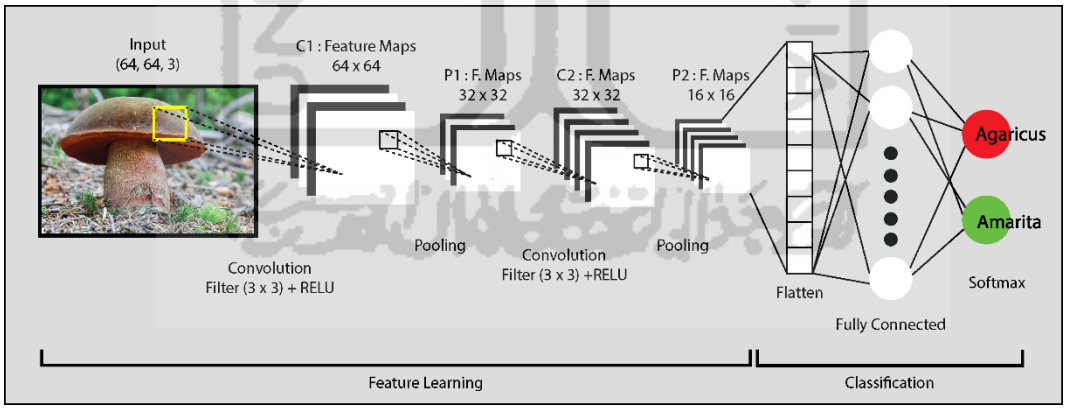
Gambar 5.2. Parameter Penelitian

Keterangan:

Img_width, img_height = lebar dan panjang gambar
Batch_size = ukuran *batch*
Samples_per_epoch = jumlah sampel per *epoch*
Validation_steps = jumlah step validasi
Nb_filters = jumlah filter
Conv_size = ukuran konvolusi
Pool_size = ukuran pool
Classes_num = jumlah kelas
Lr = *learning rate*

Gambar 5.2 menunjukkan penentuan awal dari beberapa parameter yang dibutuhkan dalam model CNN. Input gambar pada model CNN menggunakan citra berukuran 64x64x3, artinya ukuran panjang dan lebar dari citra adalah 64, sedangkan angka 3 menunjukkan citra yang memiliki 3 *channel* yaitu *Red*, *Green*, dan *Blue* (RGB). *Batch size* yang digunakan berukuran 32. *Batch size* merupakan jumlah sampel yang disebarkan ke dalam struktur *neural network*. Sampel per *epoch* adalah jumlah *data testing* yang digunakan dalam tahap pelatihan. Jumlah sampel yang digunakan adalah sebanyak 416 data. Jumlah data validasi yaitu 104 gambar. Kemudian terdapat *number of filter* yang dimasukkan kedalam proses konvolusinya. Pada tahap konvolusi pertama digunakan jumlah filter sebanyak 32 dan pada konvolusi kedua digunakan jumlah filter sebanyak 64. Kemudian untuk ukuran kernel yang digunakan yaitu 3x3. Kernel adalah sebuah matriks untuk menghitung dan mendeteksi suatu pola yang digunakan pada saat proses *convolution*. Ukuran *pooling* diberikan nilai 2. *Pooling* adalah Proses mengurangi dimensi dari *feature map* (*down sampling*). Penelitian ini menggunakan 2 kelas gambar genus jamur dengan nilai *learning rate* 0,001.

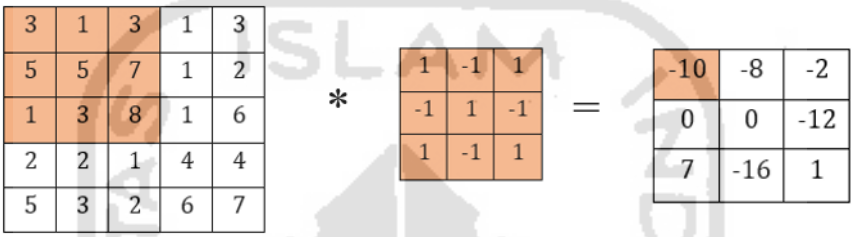
5.2 Merancang Arsitektur CNN



Gambar 5.3. Gambaran Arsitektur CNN

Gambar 5.3 merupakan arsitektur jaringan pada proses *training* untuk menghasilkan model yang optimal. Berdasarkan gambar 5.3 diketahui bahwa arsitektur terbagi dalam 2 tahapan, yakni *feature learning* dan *classification*. Arsitektur diatas dapat dijelaskan melalui penjelasan dibawah ini:

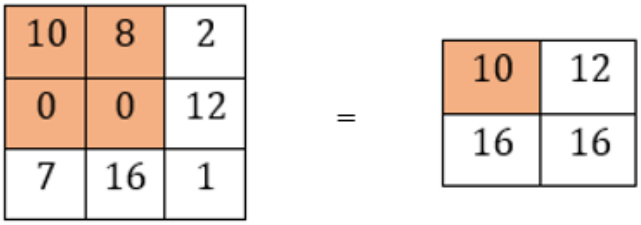
1. Proses konvolusi pertama digunakan kernel berukuran 3x3 dan jumlah filter sebanyak 32 filter, proses konvolusi ini adalah proses kombinasi antara dua buah matriks yang berbeda untuk menghasilkan suatu nilai matriks yang baru. Agar lebih memahami cara kerja dari proses konvolusi, peneliti akan menggunakan sampel matriks pada *input image*. Gambar 5.4 menunjukkan proses konvolusi dengan ukuran kernel 3x3 dan *stride* 1, dimana *stride* merupakan pergeseran kernel terhadap matriks *input.*



Gambar 5.4. Perhitungan Proses Konvolusi

Setelah proses konvolusi, maka ditambahkan sebuah aktivasi fungsi yaitu ReLU (*Retrified Linear Unit*). Fungsi aktivasi ini bertujuan untuk mengubah nilai negatif menjadi nol (menghilangkan nilai negatif dalam sebuah matriks hasil konvolusi). Hasil konvolusi ini memiliki ukuran yang sama yakni 64x64 karena pada saat proses konvolusi digunakan nilai *padding* 0.

2. Proses *pooling*, merupakan pengurangan ukuran matriks dengan menggunakan operasi *pooling*. Pada dasarnya *pooling layer* terdiri dari sebuah filter dengan ukuran tertentu yang akan secara bergantian bergeser pada seluruh area *feature map*. Penelitian ini menggunakan *max-pooling* untuk mendapatkan nilai matriks yang baru berukuran 2x2 dengan cara mengambil nilai paling maksimum dari setiap *window*. Terlihat pada gambar 5.5 *output* dari proses ini memiliki nilai yang paling maksimum yang di ambil dari matriks *feature map* hasil konvolusi.



Gambar 5.5. Proses *Pooling*

3. Proses konvolusi kedua meneruskan hasil dari proses *pooling* pertama yakni dengan input matriks gambar sebesar 32x32 dengan jumlah filter sebanyak 64 dan dengan ukuran kernel 3x3. Proses konvolusi kedua ini sama-sama menggunakan fungsi aktivasi ReLU .
4. Proses selanjutnya adalah proses *pooling* yang kedua, proses ini hampir sama dengan proses *pooling* hang pertama, namun ada perbedaan pada nilai output akhir dari matriksnya . *Output* yang dihasilkan memiliki ukuran gambar 16x16.
5. Hasil dari proses konvolusi yang dilakukan mendapatkan output feature maps yang berbentuk *multidimensional array*, oleh sebab itu proses selanjutnya dilakukan tahap *flatten* mengubah *feature map* menjadi sebuah vektor, agar dapat di input pada tahap *fully connected layer* untuk kemudian diklasifikasi.
6. Selanjutnya *fully connected*, pada tahap ini digunakan hanya satu *hidden layer* pada jaringan MLP (*Multi Layer Perceptron*). *Flatten* disini mengubah output *pooling layer* menjadi sebuah *vector*. Sebelum melakukan proses klasifikasi atau memprediksi gambar, pada proses ini digunakan nilai *dropout* yang merupakan sebuah teknik regulasi jaringan syaraf dengan tujuan memilih beberapa neuron secara acak dan tidak akan dipakai selama proses pelatihan, dengan kata lain neuron-neuron tersebut dibuang secara acak. Tujuan dari proses ini yaitu mengurangi *overfitting* pada saat proses *training*.
7. Setelah itu dilanjutkan pada proses klasifikasi, yang mana dengan bantuan aktivasi *softmax* akan diklasifikasi input sesuai dengan terget kategorinya yakni pada 2 kategori gambar jamur (*Agaricus* dan *Amanita*).

Kemudian model yang dibentuk menggunakan arsitektur tersebut memiliki jumlah total parameter sebesar 4.214.466 dengan rincian sebagai berikut,

Tabel 5.2. Perhitungan Parameter CNN

No.	Nama	Size	Parameter
0	Input	64*64*3	0
1	Conv2d_1	(64+(2*1)-(3-1)) = 64*64*32	((3*3*3)+1)*32 = 896
2	MaxPool_1	32*32*32	0
3	Conv2d_2	(32+(2*1)-(3-1)) = 32*32*64	((3*3*32)+1)*64 = 18496
4	MaxPool_2	16*16*64	0
5	Flatten	16384	0

No.	Nama	Size	Parameter
6	Dense	256	(16384*256)+256 = 4194560
7	Output	2	(256+1)*2 = 514
Total			4.214.466

s

Parameter yang dilatih pada model ini berjumlah 4.214.466 parameter. Tabel 5.2 merupakan model yang terbentuk dari hasil *training*. Untuk menghitung input kedalam konvolusi digunakan rumus "input_size + 2*padding - (filter_size - 1)". Setiap proses konvolusi akan membuat ukuran citra semakin mengecil. Hingga ukuran terakhir yang digunakan sebelum masuk pada *fully connected layer* yakni sebesar 16x16 piksel dan 64 parameter. Selanjutnya dilakukan perubahan data matriks menjadi vektor agar dapat masuk kedalam proses *fully connected*, sehingga terdapat sebanyak 16384 neuron yang akan diteruskan. Kemudian jumlah neuron tersebut ditambahkan dengan proses *dropout*, dan kemudian dilakukan klasifikasi atas 2 kategori genus jamur yang telah diuji. Proses pertama kali yakni *input* gambar, dengan ukuran 64x64x3 dimana 64x64 merupakan ukuran panjang x lebar dari gambar dan 3 merupakan kode untuk *channel* warna atau RGB. Konvolusi pertama menggunakan kernel 3x3 dan filter 32 lapisan yang kemudian diaktivasi menggunakan aktivasi ReLU untuk memperoleh node paling besar untuk kemudian diteruskan pada proses konvolusi kedua, sehingga untuk hasil konvolusi pertama memiliki *output* ukuran gambar sebesar 64x64x32 dan parameter yang dihasilkan berjumlah 896, nilai tersebut diperoleh dari hasil perhitungan ((3x3x3)+1 bias)x 32 filter. Lalu proses konvolusi kedua, meneruskan data yang telah diproses pada proses sebelumnya, yakni ukuran gambar 32x32x64 dengan menggunakan kernel 3x3 dan filter 64, maka diperoleh jumlah parameter sebesar 18.496, yang diperoleh dari perhitungan ((3x3x32)+1 bias)x 64 filter. Setelah melakukan 2 kali konvolusi diawal, kemudian dilanjutkan pada proses *pooling*. Pada proses ini parameter tidak perlu dihitung, oleh karena itu tidak terdapat jumlah parameter atau total parameter bernilai 0. Setelah itu dilanjutkan pada proses *dropout*, dengan menggunakan nilai batas atau *learning rate* sebesar 0,001 artinya untuk nilai piksel dibawah 0,001 akan dibuang atau tidak ikut digunakan pada proses selanjutnya.

5.3 Model Hasil *Training*

Setelah melalu beberapa proses dalam algoritma *Convolutional Neural Network* (CNN) didapatkan hasil *training* dan *validation*. Proses ini menggunakan jumlah 50 epoch, nilai *learning rate* 0,001, dan *optimizer* Adam. Berikut hasil yang diperoleh berdasarkan parameter yang telah ditentukan:

	precision	recall	f1-score	support
agaricus	0.53	0.60	0.56	52
amanita	0.53	0.46	0.49	52
accuracy			0.53	104
macro avg	0.53	0.53	0.53	104
weighted avg	0.53	0.53	0.53	104

Gambar 5.6. Hasil Prediksi

Berdasarkan gambar 5.2 dapat diketahui bahwa nilai akurasi dari *training* model adalah sebesar 0,53 atau 53%, dimana nilai *f1-score* dari masing-masing kelas adalah 56% untuk *Agaricus* dan 49% untuk *Amanita*. Sedangkan berdasarkan gambar 5.3 nilai-nilai yang dihasilkan dari keluaran akhir *epoch* adalah akurasi sebesar 95,86%, validation loss sebesar 126,82%, dan *validation accuracy* sebesar 65,38%. Melalui hasil *epoch* berikut dapat diketahui bahwa terdapat korelasi antara nilai akurasi dan nilai *loss* pada data *train* dengan banyaknya *epoch* atau iterasi. Semakin besar *epoch* yang digunakan, maka nilai akurasi pada data train semakin tinggi. Berbanding terbalik dengan nilai akurasi, semakin besar epoch yang digunakan maka nilai *loss* yang dihasilkan pada pelatihan data semakin rendah. Berdasarkan hal tersebut, maka dapat disimpulkan bahwa untuk memperkecil nilai *loss* yang didapatkan maka dapat dilakukan dengan cara memperbanyak jumlah *epoch* pada proses *training*, sehingga model akan menghasilkan nilai akurasi yang lebih tinggi. Sedangkan, tidak terdapat korelasi atau hubungan dengan banyaknya *epoch* untuk nilai akurasi dan nilai *loss* pada *data test*.

```
Epoch 45/50
13/13 [==============================] - 48s 4s/step - loss: 0.1642 - acc: 0.9245 - val_loss: 1.1560 - val_acc: 0.6635
Epoch 46/50
13/13 [==============================] - 48s 4s/step - loss: 0.1729 - acc: 0.9274 - val_loss: 1.1474 - val_acc: 0.6154
Epoch 47/50
13/13 [==============================] - 53s 4s/step - loss: 0.1875 - acc: 0.9271 - val_loss: 1.0719 - val_acc: 0.5962
Epoch 48/50
13/13 [==============================] - 58s 4s/step - loss: 0.1659 - acc: 0.9304 - val_loss: 1.3442 - val_acc: 0.5577
Epoch 49/50
13/13 [==============================] - 55s 4s/step - loss: 0.1263 - acc: 0.9534 - val_loss: 1.3916 - val_acc: 0.6058
Epoch 50/50
13/13 [==============================] - 61s 5s/step - loss: 0.1258 - acc: 0.9586 - val_loss: 1.2682 - val_acc: 0.6538
```

Gambar 5.7. Hasil Perhitungan *Epoch*

5.4 Perbandingan *Epoch* dengan *Optimizer*

Perbandingan beberapa parameter diperlukan untuk menghasilkan model terbaik CNN. Pada penelitian ini, akan dibandingkan beberapa *epoch*, yang merupakan langkah yang dilakukan pada proses pembelajaran jaringan saraf, dimana besarnya *epoch* yang telah ditetapkan akan mempengaruhi besaran proses pembelajaran dan berhenti tepat pada nilai *epoch* yang telah ditentukan tersebut, *epoch* sendiri prosesnya serupa dengan iterasi, namun *epoch* adalah iterasi dengan rambatan balik. Berdasarkan penelitian-penelitian sebelumnya, jumlah *epoch* yang optimal dipengaruhi berbagai faktor seperti *learning rate, optimizer,* maupun jumlah data. Namun semakin banyaknya *epoch* yang bertambah, semakin sering bobot yang ada pada jaringan di-*update.* Maka dapat diasumsikan besar *epoch* akan linear dengan jumlah dataset. Perbedaan jumlah *epoch* yang terlalu sedikit biasanya akan menghasilkan akurasi yang konstan atau tidak memiliki perbedaan siginifikan. Sehingga, *epoch* yang digunakan adalah sebanyak 50, 100, 200, 300, dan 500 dengan tiga *optimizer* yaitu, Adam, RMSporp, dan SGD.

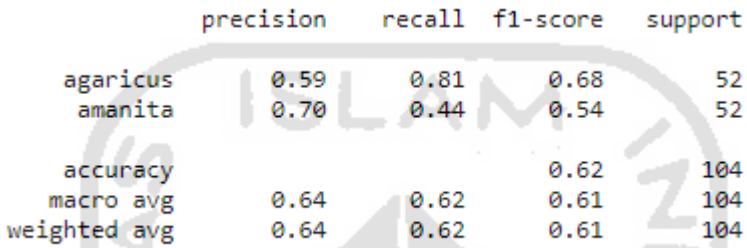
5.4.1 Perbandingan Epoch dengan Optimizer Adam

Penentuan nilai dari *epoch* biasanya tergantung peneliti dengan melihat banyak sampel. *Optimizer* Adam sendiri dikenal sebagai parameter yang biasanya menghasilkan model paling baik dalam CNN. Berikut adalah hasil perbandingan *epoch* dari hasil training menggunakan Adam.

Tabel 5.3. Perbandingan *Epoch* dengan *Optimizer* Adam

Epoch	***Accuracy Validaton***	***Loss Validation***
50	0,6250	0,8411
100	0,6442	1,6569
200	0,6246	2,4371
300	0,6250	2,4068
500	0,6442	2,8902

Berdasarkan iterasi *epoch* yang dihasilkan dapat diketahui bahwa dengan menggunakan *optimizer* Adam, *epoch* 100 dan 500 menghasilkan nilai *accuracy validation* yang sama dan paling tinggi diantara yang lainnya yaitu 0,6442. Namun nilai *loss validation* yang dihasilkan berbeda, dimana *epoch* 100 memiliki nilai *loss* yang lebih rendah sehingga dapat dikatakan bahwa diantara kelima percobaan, jumlah *epoch* 100 adalah yang terbaik.



	precision	recall	f1-score	support
agaricus	0.59	0.81	0.68	52
amanita	0.70	0.44	0.54	52
accuracy			0.62	104
macro avg	0.64	0.62	0.61	104
weighted avg	0.64	0.62	0.61	104

Gambar 5.8. Hasil Akurasi Model dengan *Optimizer* Adam dan *Epoch* 100

Gambar 5.6 merupakan hasil dari akurasi model CNN menggunakan *epoch* 100 dengan *optimizer* Adam. Akurasi yang didapatkan adalah sebesar 62%, dimana akurasi dari masing-masing kategori adalah 68% untuk jamur *Agaricus* dan 54% untuk jamur *Amanita*.

5.4.2 Perbandingan Epoch dengan Optimizer RMSProp

Kemudian, dilakukan pengujian serupa menggunakan optimizer RMSProp dengan epoch sebanyak 50, 100, 200, 300, dan 500. Akurasi terbaik yang dihasilkan terdapat pada *epoch* dengan jumlah 300 yaitu sebesar 0,6442, dimana nilai *loss validationnya* adalah sebesar 2,8250. Namun apabila dimasukkan kedalam pertimbangan, nilai *loss validation* pada *epoch* 50 jauh lebih kecil daripada *epoch* 300 yaitu sebesar 1,2789 dengan nilai akurasinya adalah 0,5769.

Tabel 5.4. Perbandingan *Epoch* dengan *Optimizer* RMSProp

Epoch	***Accuracy Validation***	***Loss Validation***
50	0,5769	1,2789
100	0,5865	2,2299

200	0,6154	2,8735
300	0,6442	2,8250
500	0,6154	3,3107

Nilai akurasi model CNN yang dihasilkan oleh program biasanya lebih kecil daripada hasil akurasi validasi pada iterasi *epoch*. Apabila dibandingkan antara *epoch* 50 dan 300, dapat diketahui melalui tabel berikut bahwa akurasi dari *epoch* 50 lebih besar yaitu 54%.

	precision	recall	f1-score	support
agaricus	0.54	0.54	0.54	52
amanita	0.54	0.54	0.54	52
accuracy			0.54	104
macro avg	0.54	0.54	0.54	104
weighted avg	0.54	0.54	0.54	104

Gambar 5.9. Hasil Akurasi Model dengan *Optimizer* RMSProp dan *Epoch* 50

Gambar 5.9 merupakan *output* dari nilai akurasi dengan menggunakan *epoch* 50. Dapat diketahui bahwa seluruh nilai yang dihasilkan baik pada *Agaricus* maupun *Amanita* adalah 0.54, sehingga akurasi yang dihasilkan adalah sebesar 54%.

Tabel 5.5. Perbandingan *Optimizer* RMSProp dengan *Epoch* 50 dan 300

***Epoch* 50**	***Epoch* 300**
54%	48%

Melalui perbandingan keduanya, dapat diketahui bahwa nilai *loss validation* yang terlalu tinggi cukup berpengaruh terhadap kebaikan model CNN, dimana semakin kecil nilai *loss validation* maka akan lebih baik akurasi model yang didapatkan.

5.4.3 Perbandingan Epoch dengan Optimizer SGD

Optimizer terakhir yang akan diuji adalah SGD atau *Stochastic Gradient Decent*. SGD merupakan parameter optimasi yang lebih sederhana daripada

optimizer sebelumnya. Pada pengujian ini, dapat diketahui bahwa hamper seluruh hasil validasi akurasi terus meningkat seiring dengan bertambahnya *epoch*. Akurasi tertinggi dimiliki oleh *epoch* 500 sebesar 0,7019 dengan nilai *loss validation* sebesar 2,0604. Namun nilai *loss validation* tersebut juga merupakan yang paling tinggi diantara yang lainnya, sedangkan yang terendah dimiliki oleh *epoch* 50 yaitu sebesar 0,7850 dengan nilai *accuracy validation* sebesar 0,5865.

Tabel 5.6. Perbandingan *Epoch* dengan *Optimizer* SGD

Epoch	***Accuracy Validaton***	***Loss Validation***
50	0,5865	0,7850
100	0,6058	1,4890
200	0,6538	1,5250
300	0,5865	1,9170
500	0,7019	2,0604

Berikut adalah nilai akurasi model CNN yang diperoleh dari masing-masing *epoch* 50 dan 500.

Tabel 5.7. Perbandingan *Optimizer* SGD dengan *Epoch* 50 dan 500

***Epoch* 50**	***Epoch* 500**
55%	51%

Berdasarkan perbanding tersebut, dapat disimpulkan bahwa pada percobaan menggunakan *optimizer* SGD, model terbaik dihasilkan oleh *epoch* 50 sebesar 55%, hal tersebut diakibatkan oleh nilai *loss validation* yang paling rendah dibandingkan dengan percobaan dengan *epoch* lainnya.

	precision	recall	f1-score	support
agaricus	0.55	0.54	0.54	52
amanita	0.55	0.56	0.55	52
accuracy			0.55	104
macro avg	0.55	0.55	0.55	104
weighted avg	0.55	0.55	0.55	104

Gambar 5.10. Hasil Akurasi Model dengan *Optimizer* SGD dan *Epoch* 50

Gambar 5.10 menunjukkan hasil akurasi model dengan *epoch* 50. Dari gambar tersebut dapat diketahui bahwa *output* pada kedua kategori memiliki nilai yang serupa, yaitu dengan nilai *f1-score* sebesar 0.54 pada *Agaricus* dan 0.55 pada *Amanita.* Adapun akurasi yang dihasilkan pada akhirnya adalah sebesar 55%.

5.5 Hasil Klasifikasi Model Terbaik

Pada tahap ini akan dilakukan perbandingan untuk menentukan model terbaik dari keseluruhan model yang telah dibuat. Berikut adalah tabel dari masing-masing model terbaik berdasarkan *optimizer* dan *epoch*-nya.

Tabel 5.8. Perbandingan *Optimizer* SGD

Optimizer	***Accuracy Validation***	***Loss Validation***	***Accuracy***
Adam 100	0,6442	1,6569	62%
RMSProp 50	0,5769	1,2789	54%
SGD 50	0,5865	0,7850	55%

Berdasarkan perbandingan dari tabel diatas, dapat diketahui bahwa *optimizer* Adam dengan *epoch* sebanyak 100 dapat menghasilkan nilai akurasi yang lebih tinggi dibandingkan dengan *optimizer* lainnya.

BAB 6. PENUTUP

6.1 Kesimpulan

Berdasarkan analisis yang telah dilakukan, diperoleh kesimpulan antara lain:

1. Implementasi CNN untuk mengklasifikasikan gambar jamur *Agaricus* dan *Amanita* dilakukan dengan dua tahapan utama yaitu *feature learning* yang terdiri dari 2 tahap konvolusi dan 2 tahap *pooling*. Kemudian, dilanjutkan dengan tahan klasifikasi dimana arsitektur terbaik didapatkan dengan membandingkan beberapa parameter yakni *epoch* dan *optimizer*. Diperoleh perbandingan akurasi dari 3 jenis *optimizer* yaitu 62% untuk *optimizer* Adam dengan *epoch* 100, 54% untuk *optimizer* RMSProp dengan *epoch* 50, dan 55% untuk *optimizer* SGD dengan *epoch* 50.
2. Hasil akurasi yang diperoleh untuk mengklasifikasikan gambar jamur menggunakan CNN yakni sebesar 62% dengan menggunakan skenario perbandingan data *train validation* 80% : 20%, ukuran kernel 3x3, *optimizer* Adam, 100 epoch, dan *learning rate* sebesar 0,001.

6.2 Saran

Sebagai upaya perbaikan dan pengembangan, maka penelitian berikutnya dapat:

1. Menggunakan *background correction* pada tahapan *pre-processing.* Data gambar pada penelitian ini memiliki *background* yang terlalu luas dan cenderung tidak fokus kepada objek yang akan diklasifikasi sehingga mempengaruhi nilai akurasi yang dihasilkan.
2. Menggunakan metode *image classification* yang terbaru dengan akurasi yang lebih baik.

DAFTAR PUSTAKA

Ahmad, Abu. 2017. "Mengenal Artificial Intelligence, Machine Learning, Neural Network, dan Deep Learning." *Jurnal Teknologi Indonesia.*

Ahmad, Usman. 2005. *Pengolahan Citra Digital & Teknik Pemrogramannya.* Yogyakarta: Graha Ilmu.

A'la, Fiddin Yusfida. 2016. *Deteksi Retak Permukaan Jalan Raya Berbasis Pengolahan Citra Menggunakan Metode Ekstraksi Ciri Wavelet.* Yogyakarta: Universitas Muhammadiyah Yogyakarta.

Bau, Tolgo, Hai Ying. Bao, and Yu Li. 2014. "A revised checklist of poisonous mushrooms in China." *Mycosystema 33* 517–548.

Britannica. 2017. *Amanita.* Accessed February 23, 2020. https://www.britannica.com/science/Amanita.

—. 2017. *Genus.* Accessed February 24, 2020. https://www.britannica.com/science/genus-taxon.

Brownlee, Jason. 2017. *Gentle Introduction to the Adam Optimization Algorithm for Deep Learning.* July 3. https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/.

Cai, Matia, Davide Pettenella, and Enrico Vidale. 2011. "Income generation from wild mushrooms in marginal rural areas." *Forest Policy Econ* III (XIII): 221-226. doi: https://doi.org/10.1016/j. forpol.2010.10.001.

Cappelli. 1984. *Agaricus, Fungi Europaei.* Italy: Candusso, Alassio.

Carlile, Michael., and Sarah Watkinson. 1994. *The Fungi.* London: Academic Press.

Coates, Adam, Honglak Lee, and Andrew Y. Ng. 2011. "An Analisys of Singe-Layer Network in Unsupervised Feature learning."

Dewa, Chandra Kusuma, Amanda Lailatul Fadhilah, and A. Afiahayat. 2018. "Convolutional Neural Networks for Handwritten Javanese Character Recognition." *IJCCS (Indonesian Journal of Computing and Cybernetics Systems)* (IJCCS (Indonesian Journal of Computing and Cybernetics Systems)) XII: 83-94. doi:10.22146/ijccs.31144.

documentation, climin 0.1. 2013. *rmsprop.* https://climin.readthedocs.io/en/latest/rmsprop.html.

Fajri, Nazar Iskandar. 2011. "Prediksi Suhu dengan Menggunakan Algoritma-Algortima yang Terdapat pada Artificial Neural Network." *Thesis* (Institut Teknologi Bandung).

Farrukh, Muhammad Umar. 2019. "Modeling on Feature Vectors in Compressed Spaces by the use of Neural network techniques." Accessed April 23, 2020. doi:10.13140/RG.2.2.29790.59203.

Fawcett, Tom. 2006. "An introduction to ROC analysis." *Pattern Recognition Letters 27* 861-874.

Fukushima, Kunihiko. 1980. "Neocognitron: A Self-Organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position." *Biological Cybermetics.*

Gonzales, R. P. 2004. *Digital Image Processing (Pemrosesan Citra Digital).* 2. Translated by S. Handayani. Vol. 1. Yogyakarta: Andri Offset.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning.* Cambridge: MIT Press.

Hermawan, Arief. 2006. *Jaringan Saraf Tiruan: Teori dan Aplikasi.* Yogyakarta: ANDI.

Hibbett, D.S., M. Binder, J.F. Bischoff, M. Blackwell, P.F. Cannon, O.E. Eriksson, S. Huhndorf, and T. James. 2007. "A higher-level phylogenetic classification of the fungi." *Mycological Research 111* 509-547.

Jumarwanto, Arif, Rudi Hartanto, and Dhihik Prastiyanto. 2009. "Apllikasi Jaringan Syaraf Tiruan Backpropagation Untuk Memprediksi Penyakit THT di Rumah Sakit Mardi Rahayu Kudus." *Jurnal Teknik Elektro* 1 (1).

Kavitha, S. Regina Lourdhu Suganthi, and Jency Jose. 2018. "Ensemble Deep Learning for Prediction of Palatable." *International Journal of Engineering & Scientific Research* 6 (1): 239-245.

Kerrigan, R.W., P. Callac, J. Guinberteau, M.P. Challen, and L.A. Parra. 2006. "Agaricus section Xanthodermatei: a phylogenetic reconstruction with commentary on taxa." *Mycologia 97* 1292–1315.

Khan, S., H. Rahmani, S. Shah, and D.M Bennamoun. 2018. *A Guide to Convolutional Neural Networks for Computer Vision.* New York: Morgan & Claypool Publishers.

2018. *Klasifikasi: Akurasi.* Accessed February 20, 2020. https://developers.google.com/machine-learning/crashcourse/classification/accuracy?hl=id.

LeCun, Yan, Yoshua Bengio, and Geoffrey Hinton. 2015. "Deep Learning. Nature, 521 (7533), 436-444." *Nature* 436–444.

Lee, C.-Y., P. W. Gallagher, and Z Tu. 2015. "Generalizing Pooling Functions in Convolutional Neural Networks : Mixed, Gated, and Tree., arXiv:1509.08985v2 [stat.ML]. Diakses pada tanggal 20 Januari 2019." *Neural and Evolutionary Computing arXiv:1509.08985 [stat.ML]*.

Lidasan, Johaira U., and Martina P. Tagacay. 2018. "Mushroom Recognition using Neural Network." *IJCSI International Journal of Computer Science Issues* XV (5): 52-57. doi:https://doi.org/10.5281/zenodo.146765.

Mc-Kane, L. 1996. *Microbiology Applied and Practice.* New York: McGrawHill Book Companies.

Meng, Jun, Dong Liu, Chao Sun, and Yushi Luan. 2014. "Prediction of plant pre-microRNAs and their microRNAs in genome-scale sequences using structure-sequence features and support vector machine." *BMC Bioinformatics.* doi: 10.1186/s12859-014-0423-x.

Miles, P.G, and S.T. Chang. 2004. *Mushrooms: Cultivation, Nutritional Value, Medicinal Effect, and Environmental Impact.* 2nd. Boca Raton: CRC Press.

Nour Arrofiqoh, Erlyna, and Harintaka. 2018. "Implementasi Metode Convolutional Neural Network Untuk Klasifikasi Tanaman Pada Citra Resolusi Tinggi." *Geomatika* 24 (2): 61-68.

Nurhikmat, Triano. 2018. "Implementasi Deep Learning Untuk Image Classification Menggunakan Algoritma CNN Pada Citra Wayang Golek." Accessed 2 7, 2020. doi:https://dspace.uii.ac.id/handle/123456789/7843.

Ottom, Mohammad Ashraf, Noor Aldeen Alawad, and Khalid M. O. Nahar. 2019. "Classification of Mushroom Fung Using Machine Learning Techniques."

International Journal of Advanced Trends in Computer Science and Engineering 8: 2378-1385. doi:https://doi.org/10.30534/ijatcse/2019/78852019 .

P, I Wayan Suartika E., Arya Yudhi Wijaya, and dan Rully Soelaiman. 2016. "Klasifikasi Citra Menggunakan Convolutional Neural Network pada Caltech 101, Institut." *JURNAL TEKNIK ITS Vol. 5, No. 1, (2016) ISSN: 2337-3539 (2301-9271 Print)* (Teknologi Sepuluh November) 65-69.

Pokharna, Harsh. 2016. *The best explanation of Convolutional Neural Networks on the Internet!* Accessed 29 July.

Prabhu, Rhagav. 2018. "Understanding of Convolutional Neural Network (CNN) — Deep Learning." March 4. Accessed February 2020, 2020. https://medium.com/@RaghavPrabhu/understanding-of-convolutionalneural-network-cnn-deep-learning-99760835f148.

Pujoseno, Jimmy. 2018. "Implementasi Deep Learning Menggunakan Convolutional Neural Network."

Rahmat, R. F., T. Aruan, S. Purnamawati1, S. Faza, T. Z. Lini, and Onrizal. 2018. "Fungus Image Identification using K-Nearest Neighbor." *IOP Conference Series: Materials Science and Engineering* 420. doi:10.1088/1757-899X/420/1/012097.

Rees, T., L. Vandepitte, W. Decock, and B Vanhoorne. 2017. "IRMNG 2006–2016: 10 YEARS OF A GLOBAL TAXONOMIC DATABASE." *Biodiversity Informatics.* 12: 1-44.

Rich, Elaine, and Kevin Knight. 1991. *Artificial Intelligence.* New York: McGraw-Hill Inc.

Rohim, Akhmad, Yuita Arum Sari, and Tibyani. 2019. "Convulotional Neural Network (CNN) untuk Pengklasifikasian Citra Makanan Tradisional." *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer* III: 7037-7042. http://j-ptiik.ub.ac.id/index.php/j-ptiik/article/view/5851.

Rosebrock, Adrian. 2016. *Softmax Classifiers Explained.* September 12. Accessed February 20, 2020. https://www.pyimagesearch.com/2016/09/12/softmax-classifiersexplained/.

Ruder, Sebastian. 2018. *An overview of gradient descent optimization algorithms.* May 30. http://ruder.io/optimizing-gradient-descent/.

Sarma, TC, I Sarma, and BN Patriana. 2010. "Wild edible mushrooms used by some ethnic tribes of Western Assam." *The Bioscan 3* 613-625.

See, May. 2018. *Mushrooms classification - Common genus's images.* Accessed January 1, 2020. https://www.kaggle.com/maysee/mushrooms-classification-common-genuss-images.

Siang, Jong Jek. 2005. *Jaringan Syaraf Tiruan dan Pemrograman menggunakan MATLAB.* Yogyakarta: ANDI.

Sumardi, Devina Gilar Fitri Ayu. 2019. "Implementasi Algoritma CNN Dalam Klasifikasi Gangguan Mata Menggunakan Pendekatan Image Processing (Studi Kasus: Gambar Fundus Maculopathy, Pathological myopia, RhegmatogeneousRD, dan Normal) ."

Tasyarini, E, Triswaty, and Susantina. 2006. "Relationship between Metabolites Candida spp with Pathogenesis of Candidiasis." *Maranatha Medical Journal 6(1)* 52-66.

2017. *Usage of Loss Function.* Accessed February 20, 2020. https://keras.io/losses/.

Utami, Pertiwi Bekti. 2018. "Klasifikasi Gambar Dengan Deep Learning Menggunakan Metode Convolutional Neural Network." Accessed February 22, 2020. https://dspace.uii.ac.id/handle/123456789/8081.

Verma, S.K., and Maitreyee Dutta. 2018. "Mushroom Classification Using ANN and ANFIS Algorithm." *IOSR Journal of Engineering (IOSRJEN)* 8 (1): 94-100.

Volk, W. A., and M.F. Wheeler. 1993. *Mikrobiologi Dasar.* Jakarta: Erlangga.

Xu, Kele, Dawei Feng, and Haibo Mi. 2017. "Deep Convolutional Neural Network Based Early Automated Detection of Diabetic Retinopathy Using Fundus Image." *Molecules* 1-7. doi:https://doi.org/10.3390/molecules22122054.

Yusuf, M. Taufik. 2017. "Membedakan Objek Menggunakan Metode Thresholding dan Fungsi Morfologi." Accessed February 20, 2020. https://www.scribd.com/doc/47932066/Membedakan-ObjekMenggunakan-Metode-Thresholding-Dan-Fungsi-Morfologi.

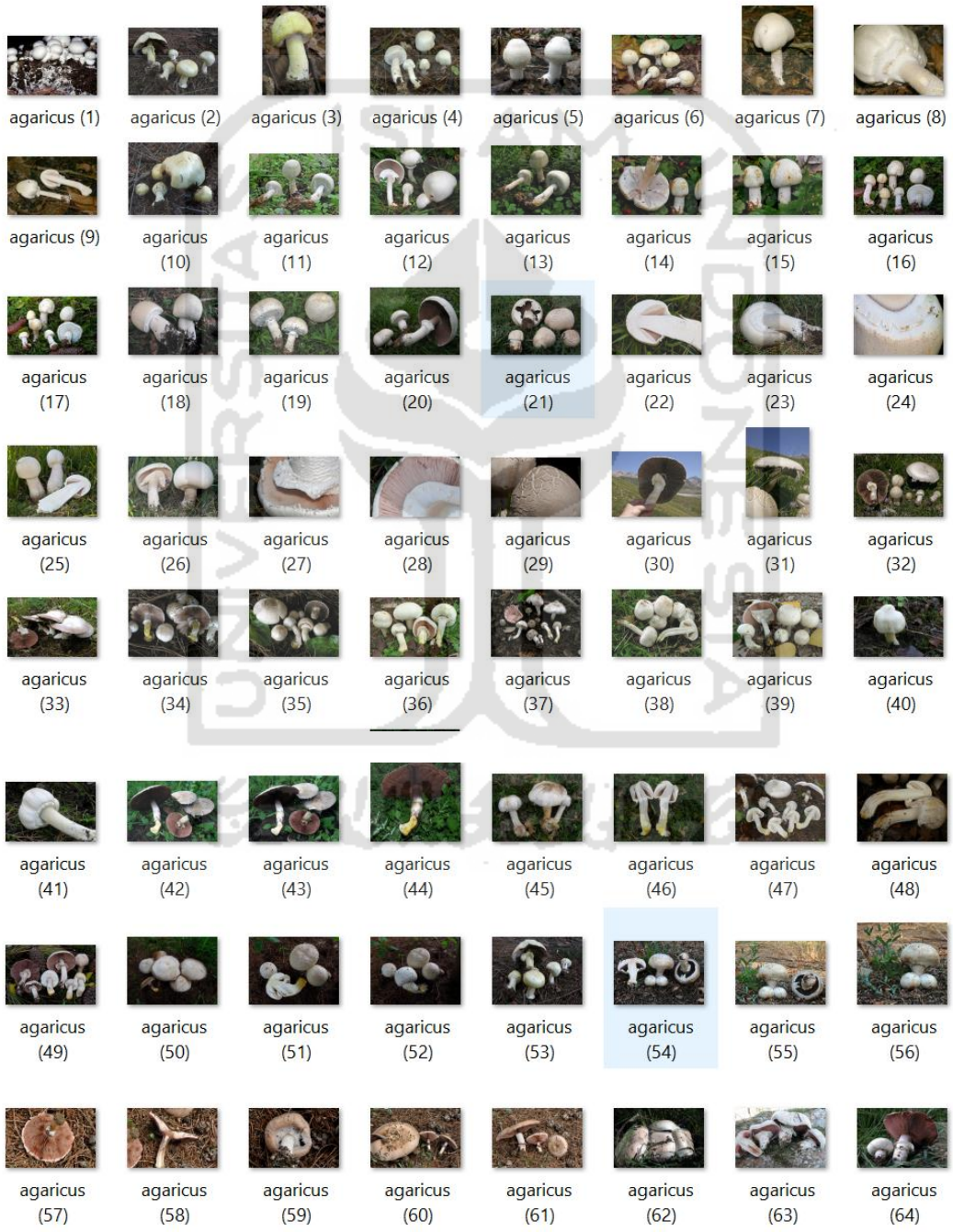
Zhang, Z. 2016. *Derivation of Backpropagation in Convolutional Neural Network (CNN).* Tennessee : University of Tennessee.

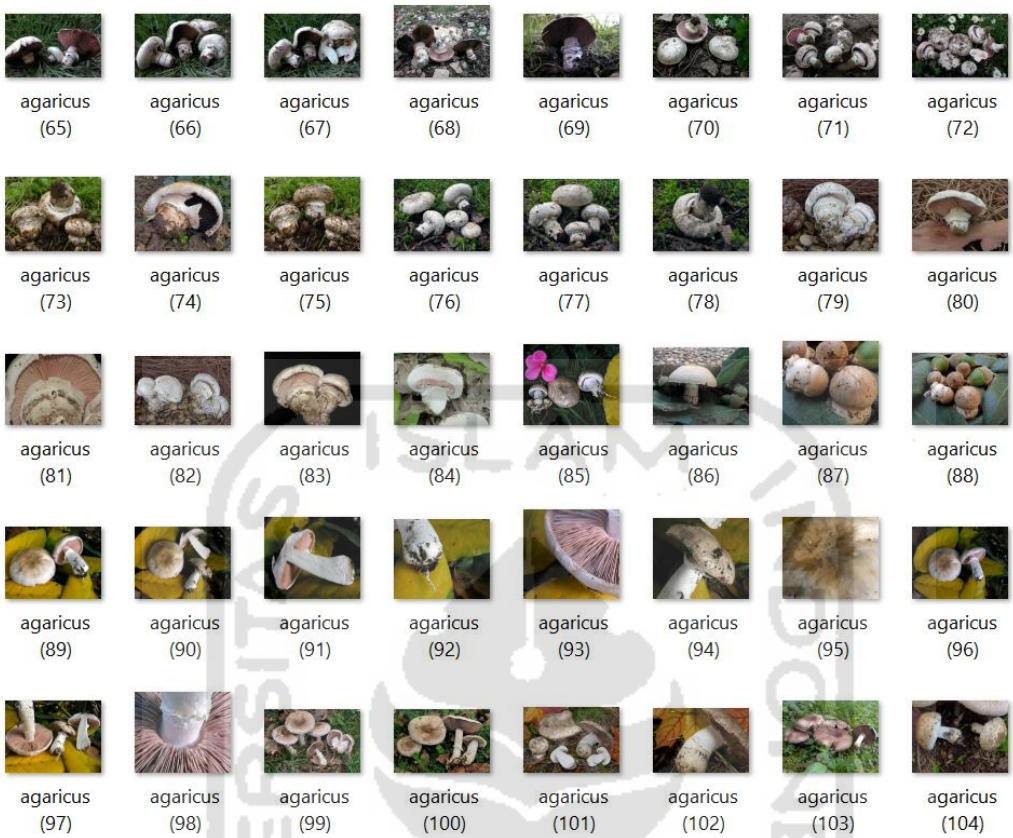
Zhao, R.L., D.E. Desjardin, P. Callac, L.A. Para, J. Guinberteau, K. Soytong, S. Karunarathna, and Y. Zhang. 2012a. "Two species of Agaricus sect. Xanthodermatei from Thailand." *Mycotaxon 122* 187–195.

LAMPIRAN

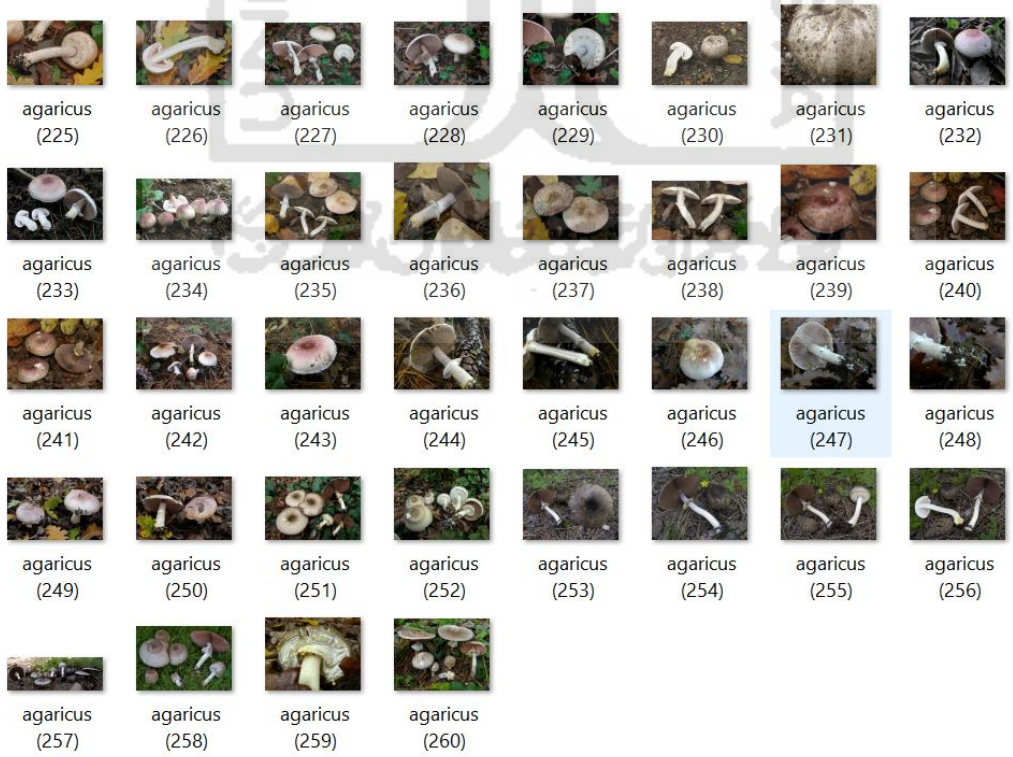
Lampiran 1 Data Gambar Jamur

a. Agaricus

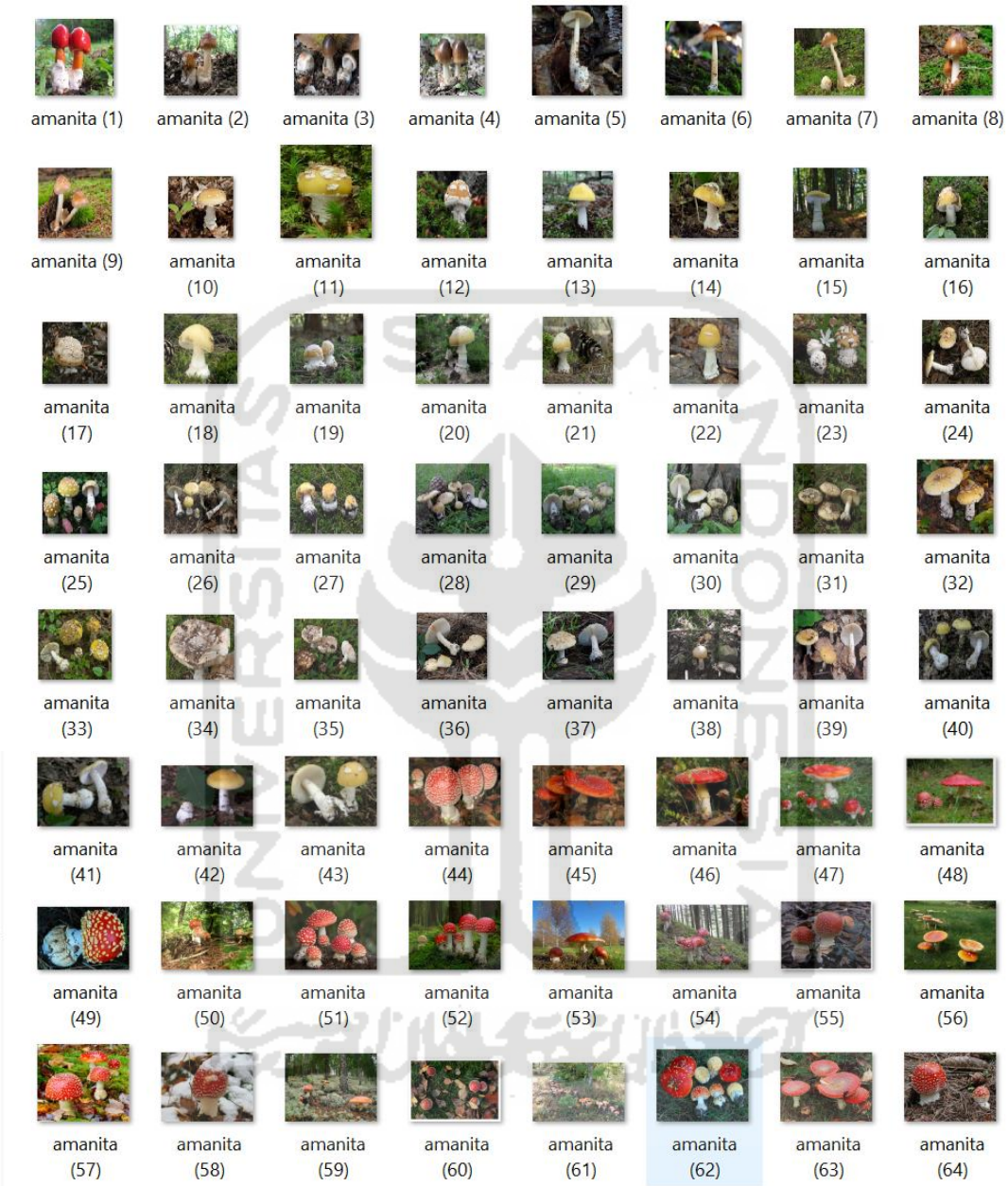


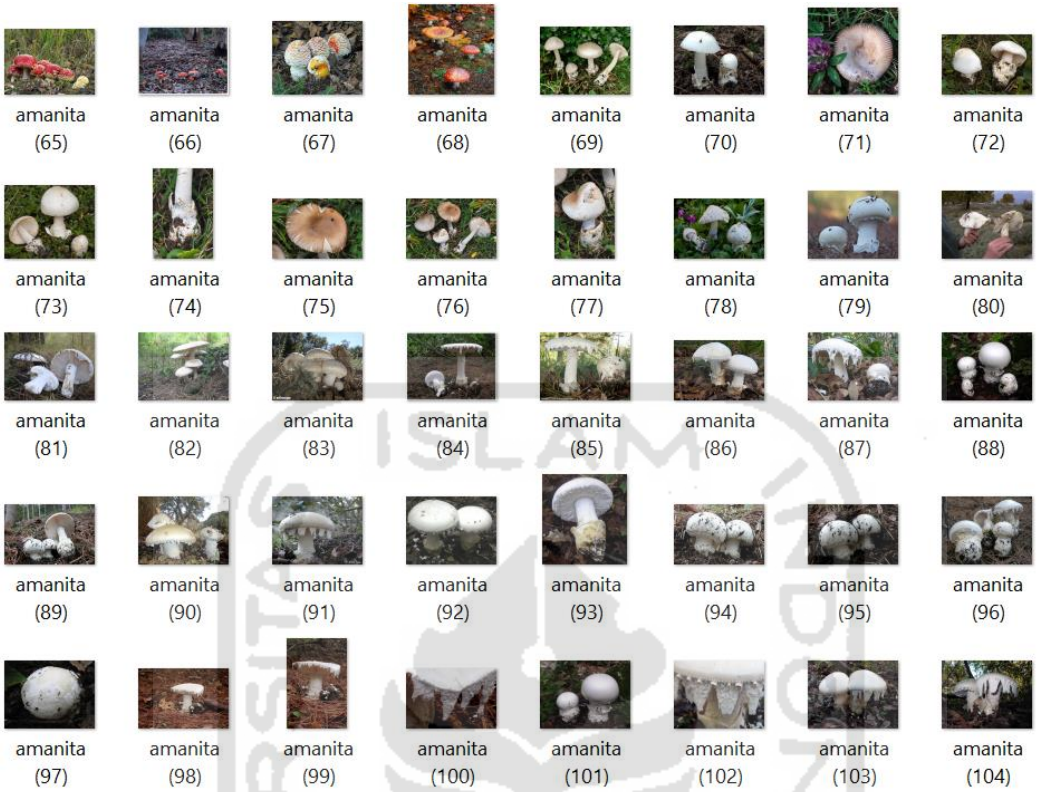

agaricus (65)
agaricus (66)
agaricus (67)
agaricus (68)
agaricus (69)
agaricus (70)
agaricus (71)
agaricus (72)
agaricus (73)
agaricus (74)
agaricus (75)
agaricus (76)
agaricus (77)
agaricus (78)
agaricus (79)
agaricus (80)
agaricus (81)
agaricus (82)
agaricus (83)
agaricus (84)
agaricus (85)
agaricus (86)
agaricus (87)
agaricus (88)
agaricus (89)
agaricus (90)
agaricus (91)
agaricus (92)
agaricus (93)
agaricus (94)
agaricus (95)
agaricus (96)
agaricus (97)
agaricus (98)
agaricus (99)
agaricus (100)
agaricus (101)
agaricus (102)
agaricus (103)
agaricus (104)

...

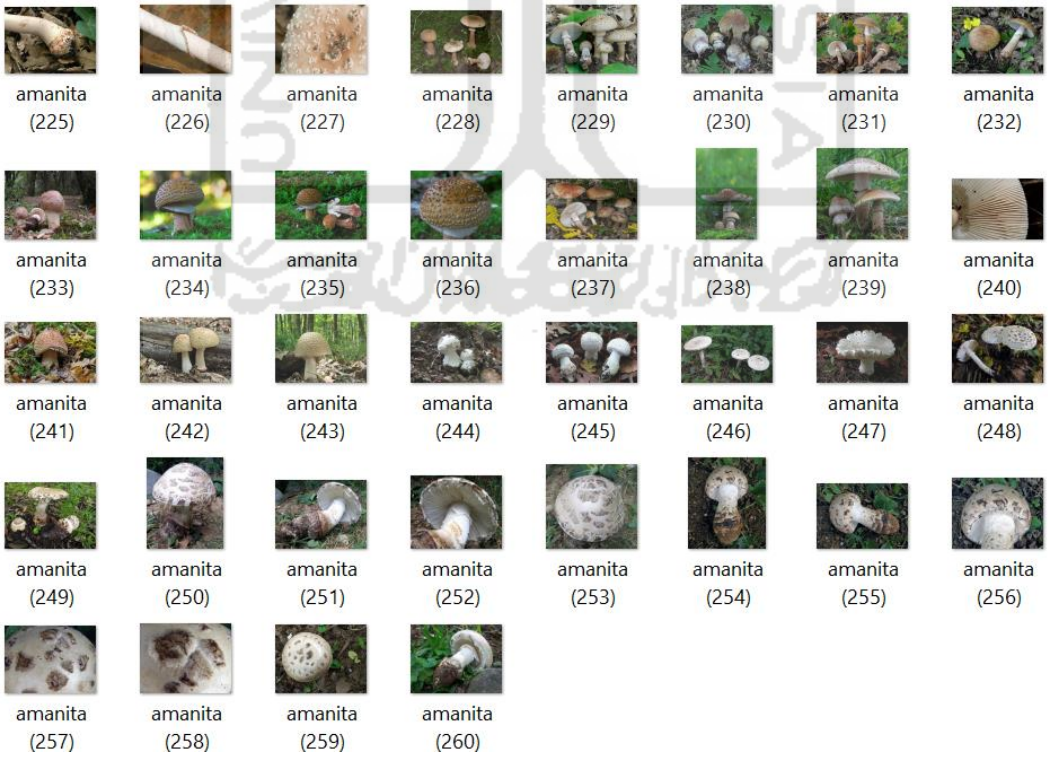

agaricus (225)
agaricus (226)
agaricus (227)
agaricus (228)
agaricus (229)
agaricus (230)
agaricus (231)
agaricus (232)
agaricus (233)
agaricus (234)
agaricus (235)
agaricus (236)
agaricus (237)
agaricus (238)
agaricus (239)
agaricus (240)
agaricus (241)
agaricus (242)
agaricus (243)
agaricus (244)
agaricus (245)
agaricus (246)
agaricus (247)
agaricus (248)
agaricus (249)
agaricus (250)
agaricus (251)
agaricus (252)
agaricus (253)
agaricus (254)
agaricus (255)
agaricus (256)
agaricus (257)
agaricus (258)
agaricus (259)
agaricus (260)

b. Amanita


amanita (1)
amanita (2)
amanita (3)
amanita (4)
amanita (5)
amanita (6)
amanita (7)
amanita (8)
amanita (9)
amanita (10)
amanita (11)
amanita (12)
amanita (13)
amanita (14)
amanita (15)
amanita (16)
amanita (17)
amanita (18)
amanita (19)
amanita (20)
amanita (21)
amanita (22)
amanita (23)
amanita (24)
amanita (25)
amanita (26)
amanita (27)
amanita (28)
amanita (29)
amanita (30)
amanita (31)
amanita (32)
amanita (33)
amanita (34)
amanita (35)
amanita (36)
amanita (37)
amanita (38)
amanita (39)
amanita (40)
amanita (41)
amanita (42)
amanita (43)
amanita (44)
amanita (45)
amanita (46)
amanita (47)
amanita (48)
amanita (49)
amanita (50)
amanita (51)
amanita (52)
amanita (53)
amanita (54)
amanita (55)
amanita (56)
amanita (57)
amanita (58)
amanita (59)
amanita (60)
amanita (61)
amanita (62)
amanita (63)
amanita (64)


amanita (65)
amanita (66)
amanita (67)
amanita (68)
amanita (69)
amanita (70)
amanita (71)
amanita (72)
amanita (73)
amanita (74)
amanita (75)
amanita (76)
amanita (77)
amanita (78)
amanita (79)
amanita (80)
amanita (81)
amanita (82)
amanita (83)
amanita (84)
amanita (85)
amanita (86)
amanita (87)
amanita (88)
amanita (89)
amanita (90)
amanita (91)
amanita (92)
amanita (93)
amanita (94)
amanita (95)
amanita (96)
amanita (97)
amanita (98)
amanita (99)
amanita (100)
amanita (101)
amanita (102)
amanita (103)
amanita (104)

...


amanita (225)
amanita (226)
amanita (227)
amanita (228)
amanita (229)
amanita (230)
amanita (231)
amanita (232)
amanita (233)
amanita (234)
amanita (235)
amanita (236)
amanita (237)
amanita (238)
amanita (239)
amanita (240)
amanita (241)
amanita (242)
amanita (243)
amanita (244)
amanita (245)
amanita (246)
amanita (247)
amanita (248)
amanita (249)
amanita (250)
amanita (251)
amanita (252)
amanita (253)
amanita (254)
amanita (255)
amanita (256)
amanita (257)
amanita (258)
amanita (259)
amanita (260)

Lampiran 2 Program Python

```
from          keras.preprocessing.image          import
ImageDataGenerator
from keras import optimizers
from keras.layers.convolutional import Conv2D
from keras.layers.convolutional import MaxPooling2D
from keras.layers.core import Activation
from keras.layers.core import Flatten
from keras.layers.core import Dense
from keras.layers.core import Dropout
from keras.models import Sequential
from keras import callbacks

DEV = False
argvs = sys.argv
argc = len(argvs)
if argc > 1 and (argvs[1] == "--development" or argvs[1]
== "-d"):
    DEV = True
if DEV:
    epochs = 2
else:
    epochs = 500 ##100

train_data_path  =  'D:/mushrooms-classification-common-
genuss-images/Mushrooms/data/train'
validation_data_path   =   'D:/mushrooms-classification-
common-genuss-images/Mushrooms/data/validation'

"""
Parameters
"""

img_width, img_height = 64,64
batch_size = 30
samples_per_epoch = 416
validation_steps = 104
nb_filters1 = 32
nb_filters2 = 64
conv1_size = 3
conv2_size = 3
pool_size = 2
classes_num = 2
lr = 0.001
```

```
model = Sequential()
model.add(Conv2D(nb_filters1,(conv1_size,  conv1_size),
padding ="same",
input_shape=(img_width, img_height, 3)))
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(pool_size,
pool_size)))
model.add(Conv2D(nb_filters2, (conv2_size, conv2_size),
padding ="same"))
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(pool_size,
pool_size),
                       dim_ordering='th'))
model.add(Flatten())
model.add(Dense(256))
model.add(Activation("relu"))
model.add(Dropout(0.5))
model.add(Dense(classes_num, activation='softmax'))
model.compile(loss='categorical_crossentropy',
              optimizer=optimizers.RMSprop(rho=0.9),
              metrics=['accuracy'])

train_datagen = ImageDataGenerator(
 rescale=1./255,
 shear_range=0.2,
 zoom_range=0.2,
 horizontal_flip=True)
test_datagen = ImageDataGenerator(rescale=1./255)
train_generator = train_datagen.flow_from_directory(
 train_data_path,
 target_size=(img_height, img_width),
 batch_size=batch_size,
 class_mode='categorical')
validation_generator                                    =
test_datagen.flow_from_directory(
 validation_data_path,
 target_size=(img_height, img_width),
 batch_size=batch_size,
 class_mode='categorical')

"""
Tensorboard log
"""
log_dir                  =                  './tf-log/tf-
log(epoch=100,lr=0.001,Op=adam)/'
tb_cb     =     callbacks.TensorBoard(log_dir=log_dir,
histogram_freq=0)
```

```
cbks = [tb_cb]

from time import time
import time
from      tensorflow.python.keras.callbacks      import
TensorBoard
name = "jamur {}".format(int(time.time()))
tensorboard                                              =
TensorBoard(log_dir='logs/{}'.format(name))
tensorboard

model.fit_generator(
    train_generator,
    samples_per_epoch=samples_per_epoch,
    epochs=epochs,
    validation_data=validation_generator,
    validation_steps=validation_steps,
    callbacks=[tensorboard])

target_dir    =    'D:/mushrooms-classification-common-
genuss-
images/Mushrooms/data/models/model(epoch=100,lr=0.001,O
p=adam)/'
if not os.path.exists(target_dir):
 os.mkdir(target_dir)
model.save('D:/mushrooms-classification-common-genuss-
images/Mushrooms/data/models/model(epoch=100,lr=0.001,O
p=adam)/model.h5')
model.save_weights('D:/mushrooms-classification-common-
genuss-
images/Mushrooms/data/models/model(epoch=100,lr=0.001,O
p=adam)/weights.h5')

import numpy
test_steps_per_epoch                                     =
numpy.math.ceil(validation_generator.samples             /
validation_generator.batch_size)

predictions                                              =
model.predict_generator(validation_generator,
steps=test_steps_per_epoch)
# Get most likely class
predicted_classes = numpy.argmax(predictions, axis=1)
predicted_classes

true_classes = validation_generator.classes
```

```
class_labels                                                    =
list(validation_generator.class_indices.keys())

import sklearn.metrics as metrics
report   =   metrics.classification_report(true_classes,
predicted_classes, target_names=class_labels)
print(report)
```

Lampiran 3 Tabel Hasil Penelitian

Optimizer	*Epoch*	*Accuracy Validation*	*Loss Validation*
Adam	50	0,6250	0,8411
	100	0,6442	1,6569
	200	0,6246	2,4371
	300	0,6250	2,4068
	500	0,6442	2,8902
RMSProp	50	0,5769	1,2789
	100	0,5865	2,2299
	200	0,6154	2,8735
	300	0,6442	2,8250
	500	0,6154	3,3107
SGD	50	0,5865	0,7850
	100	0,6058	1,4890
	200	0,6538	1,5250
	300	0,5865	1,9170
	500	0,7019	2,0604