Jurnal ELTIKOM :
Jurnal Teknik Elektro, Teknologi Informasi dan Komputer

Vol. 5, No. 2, Desember 2021, hal. 81-91
ISSN 2598-3245 (Print), ISSN 2598-3288 (Online)
DOI: http://doi.org/10,31961/eltikom.v5i2.428
Tersedia online di http://eltikom.poliban.ac.id

PENGENALAN JAMUR YANG DAPAT DIKONSUMSI MENGGUNAKAN METODE TRANSFER LEARNING PADA CONVOLUTIONAL NEURAL NETWORK

Elok Iedfitra Haksoro, Abas Setiawan
Program Studi Sarjana Teknik Informatika, Universitas Dian Nuswantoro, Semarang, Indonesia
e-mail: elokiedfitra@gmail.com, abas.setiawan@dsn.ac.id

Diterima: 14 April 2021 – Direvisi: 17 Mei 2021 – Disetujui: 19 Mei 2021

ABSTRACT

Not all mushrooms are edible because some are poisonous. The edible or poisonous mushrooms can be identified by paying attention to the morphological characteristics of mushrooms, such as shape, color, and texture. There is an issue: some poisonous mushrooms have morphological features that are very similar to edible mushrooms. It can lead to the misidentification of mushrooms. This work aims to recognize edible or poisonous mushrooms using a Deep Learning approach, typically Convolutional Neural Networks. Because the training process will take a long time, Transfer Learning was applied to accelerate the learning process. Transfer learning uses an existing model as a base model in our neural network by transferring information from the related domain. There are Four base models are used, namely MobileNets, MobileNetV2, ResNet50, and VGG19. Each base model will be subjected to several experimental scenarios, such as setting the different learning rate values for pre-training and fine-tuning. The results show that the Convolutional Neural Network with transfer learning method can recognize edible or poisonous mushrooms with more than 86% accuracy. Moreover, the best accuracy result is 92.19% obtained from the base model of MobileNetsV2 with a learning rate of 0,00001 at the pre-training stage and 0,0001 at the fine-tuning stage.

Keywords*: convolutional neural network, edible mushrooms, MobileNets, MobileNetV2, transfer learning*.

ABSTRAK

Tidak semua jamur dapat dikonsumsi karena beberapa diantaranya ada yang beracun. Untuk mengenali jamur yang bisa dikonsumsi dan yang tidak bisa dikenali dengan memperhatikan ciri-ciri morfologi jamur seperti bentuk payung, warna, dan tekstur. Timbul permasalahan yaitu dari beberapa jamur beracun memiliki ciri morfologi yang sangat mirip dengan jamur yang dapat dikonsumsi, sehingga jamur dengan banyak kemiripan tersebut akan sulit dikenali dan dapat menyebabkan terjadinya kesalahan pengidentifikasian jamur. Penelitian ini bertujuan untuk mengenali jamur yang bisa dikonsumsi atau yang beracun dengan pendekatan Deep Learning khususnya Convolutional Neural Network. Tetapi karena proses pelatihan ini membutuhkan waktu yang lama, digunakan metode Transfer Learning untuk mempercepat proses pembelajaran. Transfer Learning merupakan metode pembelajaran yang cara kerjanya adalah menerapkan dan melatih model dari suatu domain dengan cara mentransfer informasi dari domain yang terkait. Digunakan empat Base Model yaitu MobileNets, MobileNetV2, ResNet50, dan VGG19. Setiap Base Model akan diberlakukan beberapa skenario percoban seperti, melihat perbedaan nilai Learning Rate pada saat pre-training dan fine-tuning. Hasil pengujian memperlihatkan bahwa metode Transfer Learning Convolutional Neural Network dapat mengenali jamur yang bisa dikonsumsi dengan akurasi lebih dari 86%. Hasil nilai akurasi terbaik ialah 92.19% yang didapatkan dari penerapan base model MobileNetsV2 dengan nilai learning rate 0,00001 pada tahap pre-training dan 0,0001 pada tahap fine-tuning.

Kata Kunci*: convolutional neural network, jamur dapat dikonsumsi, MobileNets, MobileNetV2, transfer learning*.

I. PENDAHULUAN

INDONESIA merupakan negara beriklim tropis yang kaya akan sumber alam hayati, salah satu keanekaragaman sumber daya alam hayati tersebut ialah jamur [1]. Jamur dapat dikonsumsi sebagai makanan oleh manusia, namun tidak semua jenis jamur, beberapa diantaranya hanya dapat

dikonsumsi sebagai obat dan terdapat pula jamur yang sama sekali tidak bisa dikonsumsi karena beracun. Semua jenis jamur dapat ditemukan diberbagai objek seperti batang tumbuhan, tempat-tempat basah atau tempat yang kaya akan zat organik, dan juga dapat ditemukan pada pohon mati, kotoran ternak, tanah, dan sampah [2].

Dalam mengenali jamur yang bisa dikonsumsi dan yang tidak, bisa dikenali dengan memperhatikan ciri-ciri morfologi jamur seperti bentuk payung, warna, tekstur payung, dan ciri lain yang dapat dilihat [3]. Timbul permasalahan, beberapa jamur beracun memiliki ciri morfologi yang sangat mirip dengan jamur yang dapat dikonsumsi, akan cukup sulit jika hanya dilihat menggunakan mata telanjang untuk mengidentifikasi jamur tersebut termasuk beracun atau dapat dikonsumsi. Mengenali jamur yang bisa dikonsumsi sangat diperlukan, karena jika jamur yang dikonsumsi merupakan jamur beracun, maka akan mengakibatkan keracunan yang dapat mengganggu fungsi organ pencernaan.

Sebelumnya sudah terdapat penelitian tentang mengidentifikasi jenis jamur yang bisa dikonsumsi dengan yang tidak karena beracun menggunakan beberapa metode. Prayoga et al 2019 melakukan penelitian yang mengidentifikasi jenis jamur yang bisa dikonsumsi dengan jamur beracun menggunakan metode Naive Bayes Classifier dengan hasil akhir akurasi 86.06% yang tergolong kedalam *good classification* [4]. Selanjutnya Zubair & Muslikh 2017 melakukan penelitian yang mengidentifikasi jenis jamur dapat dikonsumsi dengan yang tidak menggunakan metode K-Nearest Neighbor dengan hasil akhir akurasi tertinggi hingga 99% yang berada pada nilai *k* adalah 60 [3]. Kedua penelitian tersebut menggunakan data yang sudah diekstraksi dari sampel gambar yang ada dan bukan merupakan data gambar. Sehingga terdapat kemungkinan untuk informasi yang hilang atau tidak sesuai kerena tidak langsung menggunakan gambar sebagai data yang akan dilatih. Beberapa pendekatan algoritma *Machine Learning* terutama *Deep Learning* yang menggunakan media dataset seperti gambar atau suara dapat memiliki hasil yang lebih baik dan akurat.

Convolutional Neural Network (CNN) merupakan salah satu jenis *neural network* yang bisa digunakan untuk mengenali objek pada sebuah image. CNN termasuk dalam jenis *Deep Neural Network* yang merupakan pengembangan dari *Multiplayer Perceptron* (MLP) yang didesain untuk mengolah data dua dimensi [5]. Beberapa penelitian yang dilakukan menggunakan metode CNN mendapatkan hasil akurasi yang bagus. Penelitian yang dilakukan oleh I Wayan Suartika E. P dkk 2016 tentang klasifikasi citra menggunakan *Convolutional Neural Network* pada Caltech 101 didapatkan hasil akurasi sebesar 20%-50% untuk menentukan kebenaran dari klasifikasi citra objek [5]. Penelitian kedua yang dilakukan oleh Halprin Abhirawa, dkk 2017 tentang pengenalan wajah menggunakan *Convolution Neural Network* mendapatkan hasil akurasi 75.79% yang diimplementasikan pada data *testing* The Extended Yale Face Database [6].

Dari beberapa penelitian yang telah disebutkan sebelumnya, diusulkan penggunaan *Convolutional Neural Network* dengan *transfer learning* sebagai metode untuk mengenali jenis jamur yang dapat dikonsumsi atau yang beracun yang ada di Indonesia. Secara umum terdapat dua kontribusi utama pada penelitian ini. Pertama, penggunaan data gambar secara langsung untuk mengenali jamur yang dapat dikonsumsi atau beracun belum pernah diteliti sebelumnya. Kedua, penggunaan CNN dengan *transfer learning* pada beberapa model jaringan syaraf yang diujicobakan untuk mengenali jamur yang dapat dikonsumsi atau beracun dengan akurasi yang dapat diterima.

II. METODE PENELITIAN

A. Prosedur Pengambilan Data

Dataset yang digunakan diambil dari situs Kaggle oleh Thomas Stjernegaard Jeppesen (dari https://*www.kaggle.com/c/fungi-challenge-fgvc-2018/data*). Dataset ini berupa file foto jamur. Dataset yang didapatkan berisi sejumlah 89.196 data gambar dengan berbagai jenis jamur. Pada penelitian ini, digunakan sembilan jenis jamur yang akan dikelompokkan menjadi dua kelas. Kesembilan jenis jamur tersebut diantaranya adalah empat jenis jamur yang dapat dikonsumsi dan lima jamur yang tidak bisa dikonsumsi karena beracun. Empat jamur yang dapat dikonsumsi adalah *Agaricus Bisporus* atau jamur kancing, *Auricularia Auricula Judae* atau jamur kuping, *Pleurotus Ostreatus* atau jamur tiram, dan *Flammulina Velutipes* atau jamur enoki yang total jumlah keseluruhan datanya ialah 642 gambar. Sedangkan lima jamur beracun ialah *Amanita Virosa*, *Cortinarius Rubellus*, *Galerina Marginata*, *Gyromitra Esculenta*, *Amanita Phalloides* yang jumlah total datanya ialah 642 gambar. Jadi total jumlah

Gambar 1. Hasil salah satu sampel gambar dari hasil *Data Augmentation*

data gambar jamur keseluruhan berjumlah 1284 gambar jamur.

Pada sejumlah dataset gambar jamur yang dimiliki, kemudian dibagi berdasarkan kelompok dataset yang akan digunakan sebagai data latih dan data uji. Perbandingan untuk setiap kelas jamur yang dapat dikonsumsi dan yang tidak adalah 70% sebagai data latih dan 30% sebagai data uji untuk masing-masing kelas. Pembagian dataset tersebut mempertimbangkan jumlah data yang tidak terlalu banyak dan untuk menghindari overfitting, sehigga diharapkan bisa memiliki hasil akurasi yang tinggi [7]. Terdapat 450 data gambar sebagai data latih dan 192 sebagai data uji untuk setiap kelas. Jika digabungkan kedua kelasnya akan didapatkan jumlah 900 data latih dan 384 data uji.

B. Data Preprocessing dan Augmentation

Preprocessing yang akan dilakukan terhadap data gambar jamur ialah melakukan normalisasi ukuran gambar. Proses preprocessing yang akan dilakukan ialah dengan melakukan proses *cropping. Cropping* dilakukan untuk mempermudah proses pengolahan data pada sebuah citra supaya penelitian dapat terfokus hanya pada objek yang diteliti. Ada beberapa metode *cropping* yaitu *rectangle, square, circle, ellipse,* dan *pollygon.* Metode *cropping* yang akan digunakan pada penelitian ini ialah metode *cropping square* [8].

Pemilihan metode *cropping square* pada penelitian ini karena untuk dilakukannya proses CNN yang optimal dibutuhkan satu ukuran, bentuk, dan resolusi gambar yang sama antara satu dengan yang lainnya. Jika terdapat perbedaan ukuran, bentuk, dan resolusi pada gambar yang digunakan untuk melakukan uji data dapat terjadi kesalahan atau eror pada proses CNN [8]. Selain itu, digunakan juga teknik *data augmentation* untuk menambah jumlah data dan sekaligus mengurangi *overfitting* [9]. Data augmentation tersebut menambahkan data image dengan cara *random flip* secara horizontal terhadap setiap data yang sudah ada dan melakukan rotasi secara random dengan faktor rotasi diantara $[20\% \times 2\pi, -20\% \times 2\pi]$. Gambar 1 memperlihatkan hasil dari data augmentation. Setiap pixel juga di normalisasi dengan membagi setiap nilai intensitas pixelnya dengan 127,5 sehingga akan didapatkan rentang nilai setiap pixelnya 0 sampai 2.

C. Training CNN dengan Transfer Learning

Proses Training CNN akan dilakukan setelah dataset yang dikumpulkan sudah melalui proses *preprocessing* dan *augmentation* penyamarataan ukuran resolusi. *Input* yang digunakan pada proses training ialah gambar berformat jpeg berwarna dengan tiga *channel* yaitu *red, green*, dan *blue*. Adapun arsitektur CNN yang digunakan adalah adaptasi arsitektur menggunakan transfer learning. Transfer Learning digunakan untuk meningkatkan pembelajaran dari satu domain dengan cara mentransfer informasi dari domain yang terkait. Dapat dimisalkan seperti terdapat dua orang yang keduanya sama-sama ingin belajar bermain piano, salah satunya memiliki pengalaman bermain gitar dan satu orang lainnya belum memiliki pengalaman bermain gitar. Satu orang yang memiliki pengalaman bermain gitar mampu mempelajari piano lebih cepat dan lebih baik karena ia dapat memanfaatkan ilmu memainkan

gitarnya untuk mempelajari permainan piano. Memanfaatkan ilmu bermain gitar untuk mempelajari cara bermain piano, contoh inilah yang bisa menunjukkan konsep dari *Transfer Learning* [10].

Dari ilustrasi permainan piano tadi, memiliki penerapan yang sama pada pengenalan pola terhadap data gambar termasuk data gambar jamur pada penelitian ini. Beberapa peneliti membuat arsitektur algoritma (yang diadaptasi dari CNN) yang diterapkan pada dataset yang besar seperti Image-Net. Kemudian, peneliti yang lain untuk kasus yang menggunakan data dengan distribusi yang serupa bisa menggunakan model algoritmanya dengan tidak melakukan *training* ulang atau melakukan *training* pada sebagian arsitektur modelnya. Selain dapat memperoleh kemungkinan akurasi yang tinggi, melakukan *transfer learning* dapat mempersingkat waktu dan menyederhanakan model arsitektur algoritmanya.

Terdapat dua tahap utama pada transfer learning yaitu *pre-training* dan finetuning. Proses *pre-training* menggunakan base model yang bobotnya dibekukan atau tidak optimasi untuk mempercepat proses training. Sedangkan finetuning untuk melakukan training terhadap model yang dihasilkan dari *pre-training* dengan cara melakukan training pada sebagian lapisan pada base model sebagai upaya untuk memperoleh akurasi yang lebih baik. Untuk penarapan metode *Transfer Learning* pada penelitian ini, digunakan empat *base model* diantaranya ialah VGG19 [11], Resnet50 [12], MobileNets [13], dan MobileNetV2 [14]. Keempat model tersebut akan digunakan dan hasil akurasi terbaik yang didapat tiap modelnya akan dibandingkan dan diambil hasil manakah Metode *Transfer Learning* yang cocok untuk penelitian ini.

Pada penelitian ini digunakan *library* Tensorflow 2. Sebagai contoh model VGG-19 dibuat dengan bobot yang sudah dilatih di ImageNet. Dengan mengklasifikasi argumen *include_top=False*, jaringan klasifikasi atau *output* layer tidak akan disertakan pada saat *pre-training*, dimana hal ini ideal untuk ekstraksi fitur [15]. Selain itu pengklasifikasian dilakukan dengan ditambahkan beberapa lapisan jaringan di atasnya dan dilakukan pelatihan klasifikasi pada tingkat atas. Pembekuan *convolutional base* penting dilakukan sebelum model disusun dan dilatih. Hal ini dilakukan untuk mencegah perbaruan bobot dalam lapisan tertentu selama proses *pre-training*.

Dalam percobaan mengekstrasi fitur, lapisan yang dilatih hanya beberapa lapisan di atas model dasar VGG-19. Selama pretrainign, pembaruan bobot dari jaringan tidak dilakukan. Melatih kembali untuk menyempurnakan (atau "*fine-tune*") bobot pada lapisan atas dari model *pre-training* sebelumnya bersamaan dengan pengklasifikasi yang ditambahkan adalah salah satu cara untuk meningkatkan performa. Semakin tinggi suatu lapisan, semakin terspesialisasi lapisan tersebut, ini berlaku bagi kebanyakan jaringan konvolusional.

Fitur yang sangat sederhana dan umum kemudian digeneralisasikan ke hampir semua jenis gambar bisanya dilakukan oleh beberapa lapisan pertama. Semakin tinggi lapisan yang bisa dicapai, fiturnya akan semakin spesifik untuk kumpulan data tempat model dilatih. Tujuan dilakukannya penyempurnaan atau "*fine-tuning*" ini adalah untuk menyesuaikan fitur khusus agar berfungsi pada kumpulan data baru yang dilatih, bukan menimpanya [15].

Karena penelitian ini menggunakan binary classification maka fungsi aktivasi yang digunakan pada *output* layer ialah fungsi Sigmoid. Setelahnya dilakukan perbandingan antara hasil *output* (yang sudah diaktivasi) dengan hasil prediksi *output* menggunakan sebuah fungsi *Loss Function* yang merupakan dimulainya proses *Backward Pass* atau proses *Back-propagation*. *Loss Function* digunakan untuk mengukur seberapa bagus performa dari pelatihan CNN pada penelitian ini. Jenis *Loss Function* yang digunakan pada penelitian ini ialah *Cross Entropy*. Proses *Back-propagation* merupakan proses yang bertujuan untuk memperbarui nilai bobot dan bias yang didapatkan dari proses *Forward Pass* berdasarkan nilai *error* yang didapat dari tahap *Loss Function*. Proses pembaharuan nilai bobot dan bias menggunakan Adam [16].

1) VGG19

VGGNet atau Visual Geometry Group Network adalah *deep neural network* dengan operasi *multilayer*. *Convolutional Neural Network* merupakan dasar dari VGGNet. VGG19 merupakan model yang sederhana karena pada bagian lapisan konvolusional atas menggunakan 3x3 lapisan konvolusional untuk menambah tingkat kedalaman jaringan. Lapisan *max-pooling* digunakan pada VGG19 untuk mengurangi volume. VGG19 menggunakan dua lapisan *fully connected* dengan jumlah 4096 neuron [11].

Pada saat proses pelatihan, lapisan konvolusiona digunakan sebagai ekstraksi fitur dan lapisan *max-*

TABEL 1
STRUKTUR MOBILENETV2 [11]

Input	Operator	Output
$h \times w \times k$	1x1 conv2d, ReLU	$h \times w \times (tk)$
$h \times w \times tk$	3x3 dwise s=s, ReLU	$\frac{h}{s} \times \frac{w}{s} \times (tk)$
$\frac{h}{s} \times \frac{w}{s} \times tk$	Linear 1x1 conv2d	$\frac{h}{s} \times \frac{w}{s} \times k^i$

pooling digunakan untu mengurangi dimensi fitur. Pada lapisan pertama konvolusi, diterapkan 64 *kernel* (berukuran 3x3) untuk mengesktrasi fitur yang berasal dari *input* gambar. Sedangkan untuk lapisan fully connected digunakan untuk vektor fitur. Vektor fitur yang sudh terpilih selanjutnya dikenakan pada PCA san SVD untuk mengurangi dimensi dan pemilihan fitur dari data gambar untuk hasil klasifikasi yang lebih baik. PCA dan SVD lebih bermanfaat untuk digunakan daripada teknik reduksi lainnya, hal ini karena PCA dan SVD lebih cepat dan secara numerik dapat bekerja lebih stabil [11].

2) ResNet50

Residual Network atau yang biasa disebut dengan ResNets adalah *deep convolutional network* yang ide dasar jairngan ini adalah untuk melewati proses blok dari lapisan konvolusi dengan menggunakan koneksi pintas. ResNets memiliki blok dasar yang bernama Bottleneck Block yang memiliki dua aturan dalam desain sederhananya. Aturan pertama ialah untuk ukuran peta fitur *output* yang sama, setiap lapisannya memiliki jumlah filter yang sama. Aturan kedua ialah jika ukuran peta fitur *output*-nya terbagi menjadi dua, mumlah filter untuk setiap lapisannya menjadi dua kali lipat [12].

Pengambilan sampel dilakukan secara langsung oleh lapisan konvolusi yang memiliki langkah mengubah setiap *byteplot* menjadi gambar RGB dan batch *normalization* yang dilakukan tepat setelah setiap konvolusi dan sebelum proses aktivasi ReLU. Perubahan setiap byteplot menjadi gambar RGB mengubah ukurannya menjadi 224x224 dimensi dan mengurangi rata-rata RGB data dataset dari masing-masing pixel [12].

Saat *input* dan *output* berada pada dimensi yang sama, pintasan identitas digunakan. Ketika dimensi bertambah, pintasan proyeksi digunakan untuk mencocokan dimensi melalui *Pointwise Convolution*. Dalam dua kasus tersebut, ketika pintasan melalu dua ukuran fitur map, pintasan-pintasan tersebut dilakukan dengan mengubah setiap *byteplot* menjadi gambar RGB. Jaringan berakhir dengan memiliki 1000 lapisan *fully connected* dengan aktivasi softmax. Total jumlah lapisan berbobot ialah 50, dengan 25.534.592 jumlah parameter yang dapat dilatih [12].

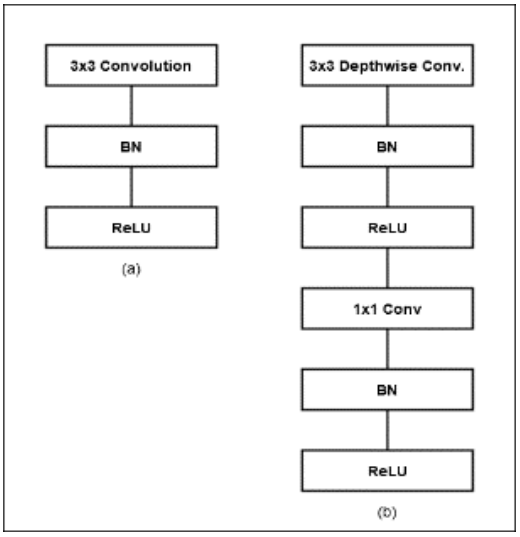
3) MobileNets

MobileNets versi pertama dibangun berdasarkan pada arsitektur ramping yang menggunakan *Depthwise Separable Convolutions* untuk membangun sebuah jaringan saraf dengan bobot yang ringan. *Depthwise Separable Convolutions* merupakan lapisan inti dari base model MobileNets. MobileNets menggunakan 3x3 *Depthwise Separable Convolutions* dengan proses komputasinya delapan hingga sembilan kali lebih sedikit dari standart proses konvolusi dengan hanya sedikit mengurasi akurasi [13].

Depthwise Separable Convolutions merupakan konvolusi faktorisasi yang memfaktorkan konvolusi standar menjadi *depthwise convolution* dan sebuah konvolusi 1x1 yang disebut *pointwise convolution*. *Depthwise Convolution* mengaplikasikan satu filter untuk setiap *input channel* pada model MobileNet. Lalu pointwise convolution mengaplikasikan konvolusi 1x1 untuk mengkombinasikan hasil dari Depthwise Convolution. Pada konvolusi standart, *input* di filter dan dikombinasikan menjadi satu set *output* dalam satu langkah. *Depthwise Separable Convolutions* sendiri terbagi menjadi dua lapisan, yaitu lapisan terpisah untuk proses pemfilteran dan pengkombinasian. Faktorisasi tersebut berefek secara drastis mengurangi ukuran model dan proses komputasi. Untuk kedua lapisan tersebut, MobileNet menggunakan *batchnorm* dan non linier ReLU [13].

Depthwise Separable Convolutions relatif sangat efisien digunakan pada konvolusi standar. Proses yang dilakukan hanya menyaring *input* channels dan tidak mengkombinasikan channels tersebut untuk membuat filter baru. Filter baru dihasilkan dari menambahkan lapisan yang menghitung kombinasi linier dari *output Depthwise Convolution* melalui konvolusi 1x1 [13].

MobileNet memiliki Depthwise Separable Convolution sebagai lapisan intinya, namun tidak untuk lapisan pertama. Lapisan pertama pada model MobileNet merupakan lapisan konvolusi penuh.



Gambar 2. (a) lapisan konvolusi standart dengan batchnorm dan ReLU, (b) lapisan Depthwise dan Pointwise dengan batchnorm dan ReLU

Batchnorm dan ReLU non-linier mengikuti semua lapisan yang ada, kecuali pada lapisan *fully connected* terakhir dimana tidak memiliki non-linier dan tergabung dalam lapisan softmax untuk digunakan sebagai klasifikasi. Rata-rata akhir pada pooling mengurangi resolusi spasial menjadi berjumlah satu sebelum lapisan *fully connected*. Konvolusi Depthwise dan Pointwise dihitung secara terpisah, dan jumlah lapisan yang dimiliki model MobileNets ialah 28 lapisan [13].

4) MobileNetV2

MobileNetV2 merupakan model yang dikembangkan dari model sebelumnya, yaitu MobileNets. Jaringan ini memaksa State of The Art supaya model komputer vision lebih bisa disesuaikan secara *mobile*. MobileNetV2 secara signifikan dapat mengurangi jumlah operasi dan memori yang dibutuhkan dengan tetap menghasilkan akurasi yang sama [14]. Stuktur MobileNetV2 secara umum diperlihatkan pada Tabel 1.

MobileNetV2 juga menggunakan *Depthwise Separable Convolution* sebagai lapisan utamanya. Ide dasar dari hal tersebut ialah mengganti operator dari *full convolutional* dengan versi faktorisasi yang membagi konvolusi menjadi dua lapisan. Lapisan pertama ialah lapisan *Depthwise Convolution* yang melakukan penyaringan ringan dengan menerapkan *single convolutional filter* untuk setiap *input* channel. Lapisan kedua ialah 1x1 convolution atau disebut *Pointwise Convolution*, dimana lapisan ini bertanggung jawab dalam membangun fitur baru dengan mengkomputasikan kombinasi linier dari setiap *input channel* diperlihatkan pada Gambar 2 [14].

Pada setiap lapisan Li pada *Deep Neural Network*, masing-masing memiliki tensor aktivasi dari dimensi *hi x wi x di*. Setiap *input* set dari gambar nyata, set aktivasi (untuk setiap lapisan *Li*) membentuk "*manifold of interest".* diasumsikan bahwa "*manifold of interest"* adalah dimensi rendah yang dapat ditangkap dengan cara memasukkan lapisan Linear Bottleneck ke dalam blok konvolusi. Untuk mencegah terjadinya banyak informasi yang rusak, sangat penting untuk menggunakan lapisan linier [14].

Pada Tabel 1 dijelaskan struktur badan dari MobileNetV2. Model ini memiliki jumlah lapisan *fully convolution* sebanyak 32 filter dan 19 lapisan sisa bottleneck. Digunakan ReLU sebagai non-linearitas karena kemampuannya yang dapat digunakan pada komputasi berpresisi rendah. Untuk ukuran Kernel selalu digunakan ukuran 3x3 sebagai standart dari jaringan modern, dan selama pelatihan berlangsung *dropout* dan *normalization batch* juga dimanfaatkan [14].

III. HASIL DAN PEMBAHASAN

A. Skenario Percobaan

Percobaan dalam penelitian ini ialah menguji arsitektur model manakah yang memiliki hasil akurasi terbaik untuk menjalankan Metode *Transfer Learning* yang diapllikasikan pada dataset jamur dengan dua kelas yaitu jamur yang dapat dikonsumsi dan jamur yang tidak dapat dikonsumsi. Pada Tabel 2 ditunjukkan skenario percobaan yang dilakukan peneliti sehingga didapatkan nilai akurasi terbaik dari

TABEL 2
RINCIAN SKENARIO PERCOBAAN

Kode Skenario	Model yang dipakai untuk TL	Learning Rate *Pre-training*	Learning Rate *Fine-tuning*	Lapisan yang di *Fine-tuning*
1	VGG-19	0,00005	0,0005	7 – 22
2	Resnet50	0,00005	0,0005	58 – 175
3	MobileNets	0,00005	0,0005	29 – 87
4	MobileNetV2	0,00005	0,0005	57 – 155
5	VGG-19	0,00001	0,0001	7 – 22
6	Resnet50	0,00001	0,0001	58 – 175
7	MobileNets	0,00001	0,0001	29 – 87
8	MobileNetV2	0,00001	0,0001	57 – 155

(a) (b)

Gambar 3. (a) proses *pre-training* VGG-19 dengan *learning rate* 0,00001, (b) proses *fine-tuning* VGG-19 dengan *learning rate* 0,0001

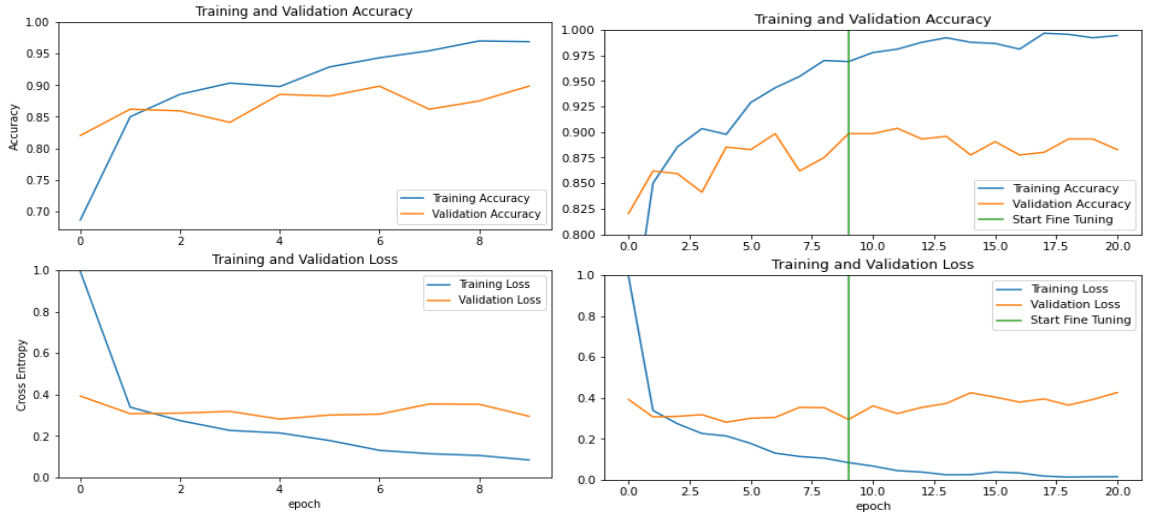
(a) (b)

Gambar 4. (a) proses *pre-training* VGG-19 dengan *learning rate* 0,00005, (b) proses *fine-tuning* VGG-19 dengan *learning rate* 0,0005

berbagai aturan *Learning Rate* yang digunakan pada proses *pre-training* dan *finetuning* untuk setiap model dasar yang digunakan.

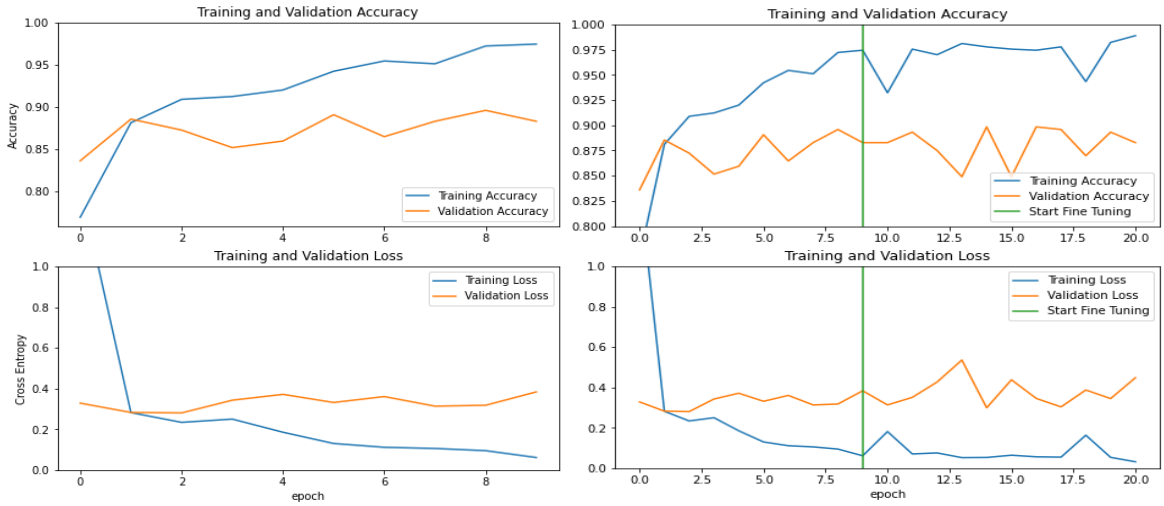
B. Proses Training pada VGG-19

Beri nomer persamaan secara urut di dalam tanda kurung dan letakkan pada tepi kanan, seperti (1). Untuk membuat persamaan lebih singkat, dapat pula digunakan tanda solidus (/), fungsi exp, atau eksponen yang sesuai. Gunakan tanda kurung untuk menghindari ambiguitas di dalam pembagian.



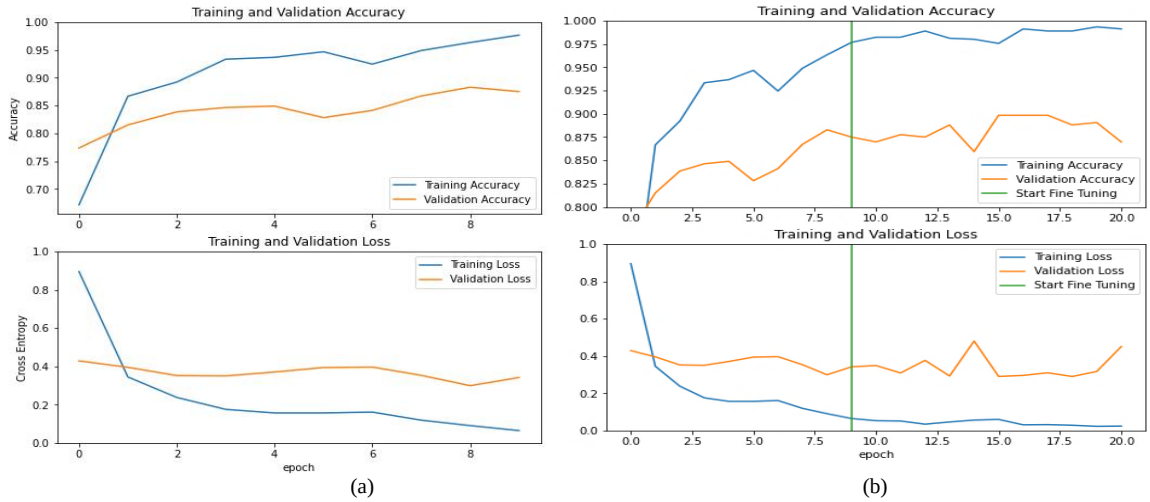
(a) (b)

Gambar 5. (a) proses *pre-training* ResNet50 dengan *learning rate* 0,00001, (b) proses *fine-tuning* ResNet50 dengan *learning rate* 0,0001



(a) (b)

Gambar 6. (a) proses *pre-training* ResNet50 dengan *learning rate* 0,00005, (b) proses *fine-tuning* ResNet50 dengan *learning rate* 0,0005



(a) (b)

Gambar 7. (a) proses *pre-training* MobileNets dengan *learning rate* 0,00001, (b) proses *fine-tuning* MobileNets dengan *learning rate* 0,0001

Pada pelatihan pertama digunakan nilai learning rate 0,00001 untuk tahap *pre-training* dan 0,0001 untuk tahap *fine-tuning*. Pada tahap *pre-training* didapatkan nilai akurasi tertinggi 0,8646 yang terjadi

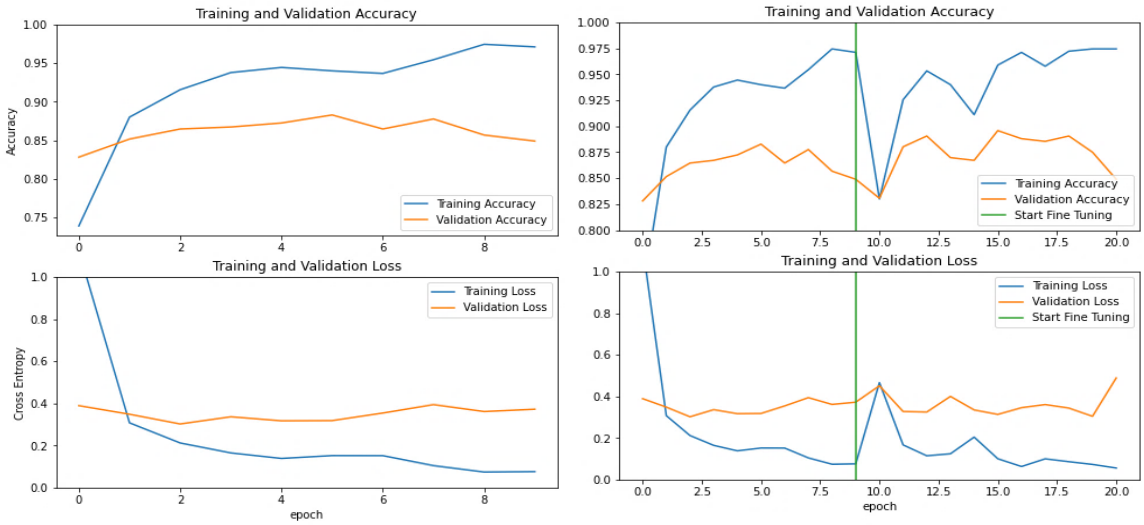
di epoch terakhir. Sedangkan pada tahap *fine-tuning* didapatkan nilai akurasi tertinggi 0,8776 yang terjadi di epoch ke 19 dari 20, Perjalanan pelatihan tersebut diperlihatkan pada Gambar 3. Selanjutnya, pada pelatihan kedua digunakan nilai learning rate 0,00005 untuk tahap *pre-training* dan 0,0005 untuk tahap *fine-tuning*. Pada tahap *pre-training* didapatkan nilai akurasi tertinggi 0,8828 yang terjadi di epoch ke 9 dari 10, Sedangkan pada tahap *fine-tuning* didapatkan nilai akurasi tertinggi 0,8984 yang terjadi di epoch ke 19 dari 20, Perjalanan pelatihan tersebut diperlihatkan pada Gambar 4.

C. Proses Training pada ResNet50

Pada pelatihan pertama digunakan nilai learning rate 0,00001 untuk tahap *pre-training* dan 0,0001 untuk tahap *fine-tuning*. Pada tahap *pre-training* didapatkan nilai akurasi tertinggi 0,8984 yang terjadi di epoch ke 7 dari 10, Sedangkan pada tahap *fine-tuning* didapatkan nilai akurasi tertinggi 0,8984 yang terjadi di separuh perjalanan epoch. Perjalanan pelatihan tersebut diperlihatkan pada Gambar 5. Selanjutnya, pada pelatihan kedua digunakan nilai *learning rate* 0,00005 untuk tahap *pre-training* dan 0,0005 untuk tahap *fine-tuning*. Pada tahap *pre-training* didapatkan nilai akurasi tertinggi 0,8958 yang terjadi di epoch ke 9 dari 10, Sedangkan pada tahap *fine-tuning* didapatkan nilai akurasi tertinggi 0,8958 yang terjadi di epoch ke 17 dari 20, Perjalanan pelatihan tersebut diperlihatkan pada Gambar 6.

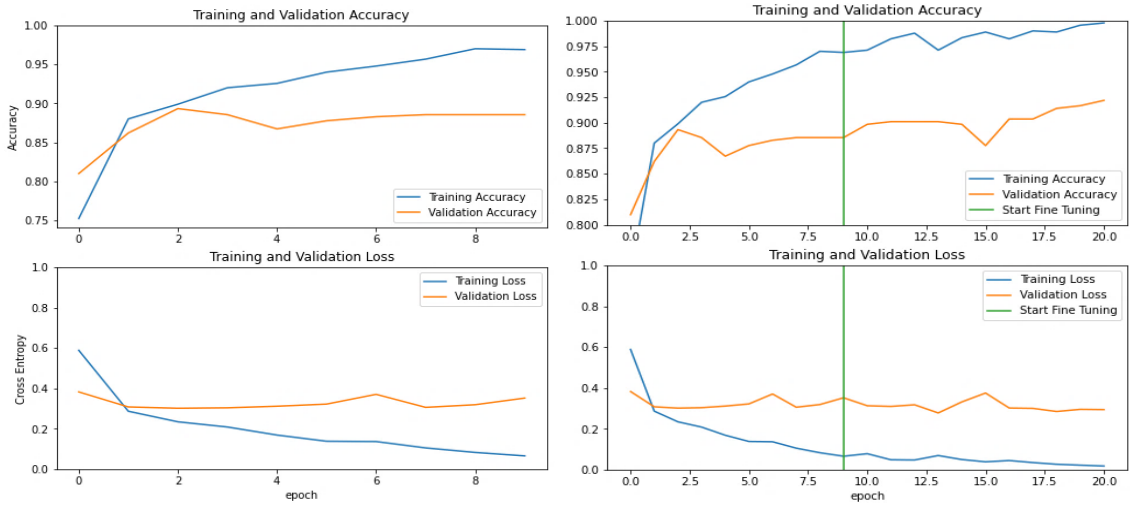
D. Proses Training pada MobileNets

Pada pelatihan pertama digunakan nilai *learning rate* 0,00001 untuk tahap *pre-training* dan 0,0001



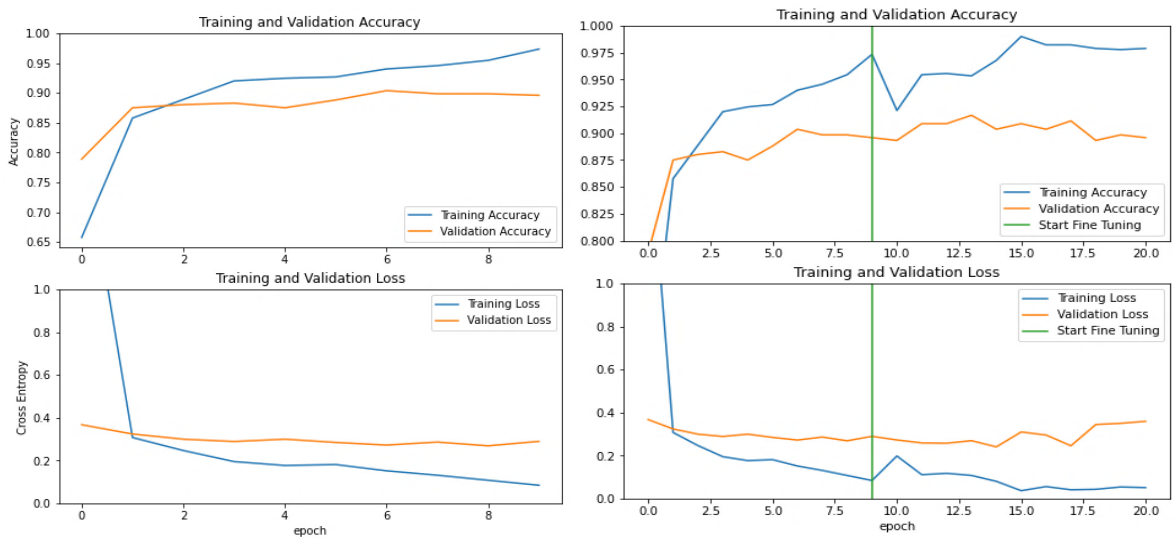
(a) (b)

Gambar 8. (a) proses *pre-training* MobileNets dengan *learning rate* 0,00005, (b) proses *fine-tuning* MobileNets dengan *learning rate* 0,0005



(a) (b)

Gambar 9. (a) proses *pre-training* MobileNetV2 dengan *learning rate* 0,00001, (b) proses *fine-tuning* MobileNetV2 dengan *learning rate* 0,0001



(a) (b)

Gambar. 10. (a) proses *pre-training* MobileNetV2 dengan *learning rate* 0,00005, (b) proses *fine-tuning* MobileNetV2 dengan *learning rate* 0,0005

TABEL 3
HASIL KESELERUHAN SKENARIO PERCOBAAN

Kode Skenario	Model yang dipakai untuk TL	*Learning Rate Pretraining*	*Learning Rate Finetuning*	Akurasi *Pre-training*	Akurasi terbaik (*fine-tuning*)
1	VGG-19	0,00005	0,0005	88,82%	89,84%
2	Resnet50	0,00005	0,0005	89,58%	89,58%
3	MobileNets	0,00005	0,0005	88,28%	89,58%
4	MobileNetV2	0,00005	0,0005	90,36%	91,67%
5	VGG-19	0,00001	0,0001	86,46%	87,76%
6	Resnet50	0,00001	0,0001	89,84%	89,84%
7	MobileNets	0,00001	0,0001	88,28%	89,84%
8	MobileNetV2	0,00001	0,0001	89,32%	92,19%

untuk tahap *fine-tuning.* Pada tahap *pre-training* didapatkan nilai akurasi tertinggi 0,8828 yang terjadi di *epoch* 9 dari 10, Sedangkan pada tahap *fine-tuning* didapatkan nilai akurasi tertinggi 0,8984 yang terjadi di *epoch* 15 dari 20, Perjalanan pelatihan tersebut diperlihatkan pada Gambar 7. Selanjutnya, pada pelatihan kedua digunakan nilai *learning rate* 0,00005 untuk tahap *pre-training* dan 0,0005 untuk tahap *fine-tuning.* Pada tahap *pre-training* didapatkan nilai akurasi tertinggi 0,8828 yang terjadi di *epoch* 6 dari 10, Sedangkan pada tahap *fine-tuning* didapatkan nilai akurasi tertinggi 0,8958 yang terjadi di *epoch* 15 dari 20, Perjalanan pelatihan tersebut diperlihatkan pada Gambar 8.

E. Proses Training pada MobileNetV2

Pada pelatihan pertama digunakan nilai *learning rate* 0,00001 untuk tahap *pre-training* dan 0,0001 untuk tahap *fine-tuning.* Pada tahap *pre-training* didapatkan nilai akurasi tertinggi 0,8932 yang terjadi di *epoch* ke 3 dari 10, Sedangkan pada tahap *fine-tuning* didapatkan nilai akurasi tertinggi 0,9219 yang terjadi di *epoch* terakhir. Perjalanan pelatihan tersebut diperlihatkan pada Gambar 9. Selanjutnya, pada pelatihan kedua digunakan nilai *learning rate* 0,00005 untuk tahap *pre-training* dan 0,0005 untuk tahap *fine-tuning.* Pada tahap *pre-training* didapatkan nilai akurasi tertinggi 0,9036 yang terjadi di *epoch* ke 7 dari 10, Sedangkan pada tahap *fine-tuning* didapatkan nilai akurasi tertinggi 0,9167 yang terjadi di *epoch* ke 13 dari 20, Perjalanan pelatihan tersebut diperlihatkan pada Gambar 10,

F. Hasil Keseluruhan Skenario

Setelah melakukan percobaan sebanyak delapan kali dengan menggunakan input berukuran 160x160 pixel. Dimana ke-delapan percobaan ini menggunakan empat *base model* yang berbeda yaitu MobileNets, MobileNetV2, ResNet50, dan VGG19. Untuk setiap *base model* yang digunakan, masing-masing dilatih dua kali dengan nilai *learning rate* yang berbeda yaitu 0,00001 dan 0,00005 pada tahap *pretraining* dan 0,0001 dan 0,0005 pada tahap *fine-tune*. Tabel 3 merupakan hasil keseluruhan percobaan. Hasil akurasi terbaik ialah 92,19% didapat dari menggunakan *base model* MobileNetV2

ditahap *fine-tuning* dengan *learning rate* 0,0001. Hasil terbaik ini selaras dengan penelitian sebelumnya yang telah dilakukan [14], bahwa didalam arsitektur MobileNet V2 terdapat *inverted residual connections* dan *linier bottleneck* yang dapat meningkatkan akurasi.

Hasil akurasi terendah ditahap *pre-training* ialah 86,46%, dimana hasil tersebut dicapai saat menerapkan *base model* VGG-19 dengan *learning rate* 0,00001. Meskipun *pre-training* VGG-19 memiliki akurasi terendah saat *pre-training*, setelah proses *fine-tuning*, akurasi yang didapat sebelumnya meningkat dan tidak terpaut jauh hasilnya dengan model yang lain. Karena model MobileNetV2 merupakan model *Transfer Learning* terbaik pada penelitian ini, maka model tersebut digunakan sebagai model untuk mengenali jamur yang bisa dikonsumsi atau yang beracun.

IV. KESIMPULAN

Berdasarkan hasil penelitian pada dataset jamur dapat disimpulkan bahwa penggunaan metode *Transfer Learning Convolutional Neural Network* mampu memberikan hasil akurasi sebesar 92,19% dalam pengklasifikasian jamur yang bisa dikonsumsi dengan yang tidak. Dengan hasil tersebut, dapat dikatakan bahwa penelitian ini dapat membantu manusia dalam mengenali jamur yang beracun dengan yang tidak. Dari hasil percobaan yang telah dijabarkan sebelumnya, dapat disimpulkan bahwa arsitektur MobileNetV2 merupakan arsitektur yang sesuai untuk digunakan karena memberikan hasil akurasi tertinggi yaitu sebesar 92,19% dibandingkan dengan ketiga arsitektur lainnya. Ketiga arsitektur yang lainnya mendapatkan akurasi sebesar 87,76% dengan *base model* VGG-19, dan 89,84% dengan *base model* Resnet50 dan MobileNets. Nilai akurasi yang didapatkan ini akan sangat berpengaruh terhadap bagaimana sistem dapat mengenali jamur yang dapat dikonsumsi dengan yang tidak.

DAFTAR PUSTAKA

[1] I. Annissa, Ekamawanti, H. Artuti, and Wahdina, "Keanekaragaman Jenis Jamur Makrokopis Di Arboretum Sylva Universitas Tanjungpura," *J. Hutan Lestari*, vol. 5, no. 4, pp. 969–977, 2017.
[2] W. Darwis, Desnalianif, and R. Supriati, "Inventarisasi Jamur Yang Dapat Dikonsumsi Dan Beracun Yang Terdapat Di Hutan Dan Sekitar Desa Tanjung Kemuning Kaur Bengkulu," *J. Ilm. Konserv. Hayati*, vol. 07, no. 02, pp. 1–8, 2011.
[3] A. Zubair and A. R. Muslikh, "Identifikasi jamur menggunakan metode k-nearest neighbor dengan ekstraksi ciri morfologi," *Semin. Nas. Sist. Inf.*, no. September, pp. 965–972, 2017.
[4] S. A. Prayoga, I. Nawangsih, and T. N. Wiyatno, "Implementasi Metode Naïve Bayes Classifier Untuk Identifikasi Jenis Jamur," vol. 14, no. 1, pp. 55–66, 2019.
[5] I. W. S. E. P, A. Y. Wijaya, and R. Soelaiman, "Klasifikasi Citra Menggunakan Convolutional Neural Network (CNN) pada Caltech 101," *J. Tek. ITS*, vol. 5, no. 1, 2016.
[6] H. Abhirawan, Jondri, and A. Arifianto, "Pengenalan Wajah Menggunakan Convolutional Neural Networks (CNN)," *Univ. Telkom*, vol. 4, no. 3, pp. 4907–4916, 2017.
[7] A. Gholamy, V. Kreinovich, and O. Kosheleva, "Why 70/30 or 80/20 Relation Between Training and Testing Sets : A Pedagogical Explanation," *Dep. Tech. Reports*, pp. 1–6, 2018.
[8] Y. Yuhandri, "Perbandingan Metode Cropping Pada Sebuah Citra Untuk Pengambilan Motif Tertentu Pada Kain Songket Sumatera Barat," *Komtekinfo*, vol. 6, no. 1, pp. 95–105, 2019.
[9] S. C. Wong, A. Gatt, V. Stamatescu, and M. D. McDonnell, "Understanding Data Augmentation for Classification: When to Warp?," *2016 Int. Conf. Digit. Image Comput. Tech. Appl. DICTA 2016*, 2016.
[10] K. Weiss, T. M. Khoshgoftaar, and D. D. Wang, *A survey of transfer learning*, vol. 3, no. 1. Springer International Publishing, 2016.
[11] M. Mateen, J. Wen, Nasrullah, S. Song, and Z. Huang, "Fundus image classification using VGG-19 architecture with PCA and SVD," *Symmetry (Basel).*, vol. 11, no. 1, 2019.
[12] E. Rezende, G. Ruppert, T. Carvalho, F. Ramos, and P. De Geus, "Malicious software classification using transfer learning of ResNet-50 deep neural network," *Proc. - 16th IEEE Int. Conf. Mach. Learn. Appl. ICMLA 2017*, vol. 2017-Decem, no. January 2018, pp. 1011–1014, 2017.
[13] A. G. Howard *et al.*, "MobileNets: Efficient convolutional neural networks for mobile vision applications," *arXiv*, 2017.
[14] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 4510–4520, 2018.
[15] F. Chollet, "Transfer learning and chess," *TensorFlow*, 2017. [Online]. Available: https://www.tensorflow.org/tutorials/images/transfer_learning#configure_the_dataset_for_performance. [Accessed: 03-Nov-2020].
[16] S. Sena, "Pengenalan Deep Learning Part 3 : BackPropagation Algorithm," *Medium*, 2017. [Online]. Available: https://medium.com/@samuelsena/pengenalan-deep-learning-part-3-backpropagation-algorithm-720be9a5fbb8.