Indonesian Journal of Computer Science
ISSN 2302-4364 (*print*) dan 2549-7286 (*online*)
Jln. Khatib Sulaiman Dalam, No. 1, Padang, Indonesia, Telp. (0751) 7056199, 7058325
Website: ijcs.stmikindonesia.ac.id | E-mail: ijcs@stmikindonesia.ac.id

Transfer Learning dengan Metode *Fine Tuning* pada Model *Network* VGG16 dan ResNet50

Fauzan Muhammad, Aniati Murni Arimurthy, Dina Chahyati
fauzanmuhammad01@ui.ac.id, aniati@cs.ui.ac.id, dina@cs.ui.ac.id
Universitas Indonesia

Informasi Artikel

Diterima : 23 Des 2022
Direview : 20 Jan 2023
Disetujui : 26 Feb 2023

Kata Kunci

Transfer learning, *fine tuning, cross validation*

Abstrak

Transfer learning adalah prinsip yang digunakan pada *neural network* dengan tujuan membantu pelatihan pada data yang sedikit, mempercepat waktu dan meningkatkan performa pelatihan. Salah satu metode *transfer learning* adalah *fine tuning*. Pada metode tersebut dilakukan pembekuan pada sejumlah lapisan pada model *network* yang dilatih sebelumnya dan melakukan pelatihan pada lapisan yang lainnya dan pada lapisan *feature extractor*. Pada penelitian ini akan dilihat salah satu metode *transfer learning* pada model *network* VGG6 dan ResNet50, yaitu *fine tuning*, dengan melakukan pembekuan dengan jumlah yang lapisan berbeda pada masing-masing model. Kemudian pada kedua model tersebut dilihat kinerja dan akurasinya pada saat pelatihan dan pengujian. Berdasarkan pengujian yang dilakukan, performa terbaik didapat pada *trainable layer* yang paling kecil. pada model *network* VGG16 dengan akurasi 82.0988% dan ResNet50 dengan akurasi 99.3827%.

Keywords

Transfer learning, fine tuning, cross validation

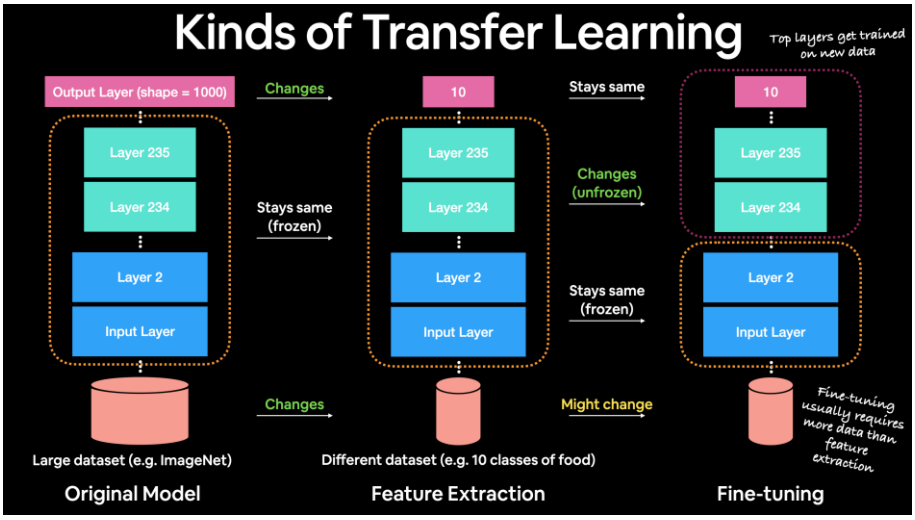
Abstrak

Transfer Learning is a principle used in neural networks with the aim of assisting training on small data sets, speeding up time and improving pelatihan *performance. One method of transfer learning is fine tuning. In this method, freezing a number of layers in the previously trained network model and training the remaining layers and feature extractor layer. In this research, we will look at one of the transfer learning methods on the VGG6 and ResNet50 network models, namely fine tuning, by doing a various number of freeze layers in each model. Then, during training and testing, the performance and accuracy of both models were observed. Based on the tests conducted, the best performance is obtained on the smallest number of trainable layer on the VGG16 network model with 82.0988% accuracy and ResNet50 with 99.3827% accuracy.*

A. Pendahuluan

Transfer learning adalah transfer pengetahuan yang diperoleh jaringan dari suatu tugas, yang memiliki sejumlah besar data, ke tugas baru di mana *dataset* yang ada tidak cukup. Inspirasi pada *transfer learning* adalah jika model dilatih pada kumpulan data yang cukup besar dan umum, model ini akan secara efektif berfungsi sebagai representasi yang umum dari segi visual [1].

Ada tiga pendekatan utama pada *transfer learning*, yaitu sebagai *classifier*, sebagai *feature extractor*, dan *fine tuning*. Perbedaan antara ketiga pendekatan dapat dilihat pada gambar 1. Menggunakan metode *transfer learning* sebagai *classifier* berarti mengambil jaringan yang dilatih untuk masalah serupa dan digunakan langsung pada model. Metode *transfer learning* sebagai *feature extraction* dilakukan ketika tugas baru mirip dengan dataset asli yang dilatih jaringan sebelumnya. Metode *fine tuning* mengekstrak *feature map* yang benar dari model sebelumnya dan menyempurnakannya agar sesuai dengan domain target [2].



Gambar 1. Tiga Pendekatan *Transfer Learning* [3]

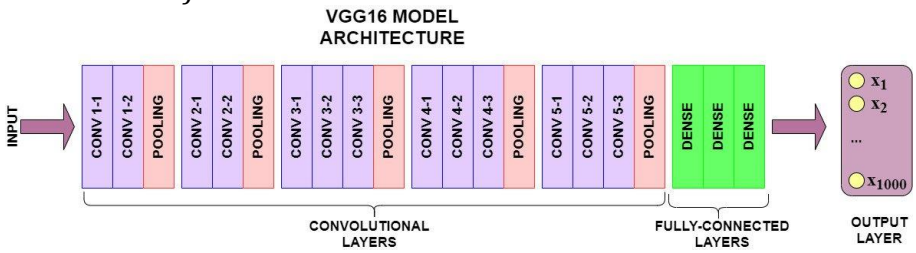
Metode *transfer learning* berkaitan erat pada penelitian yang berhubungan dengan *neural network*. Pada [4], *transfer learning* digunakan untuk melakukan klasifikasi pada karya seni. Sedangkan pada [5], *transfer learning* digunakan untuk klasifikasi angka dengan penulisan tangan.

Penelitian ini berfokus hanya pada metode *tranfer learning* yang ketiga, yaitu *fine tuning*. Pada metode *fine tuning*, *weight* dari *pre-trained network* sudah dioptimalkan untuk belajar dari *dataset* sebelumnya. Ketika *network* ini digunakan dalam penelitian sendiri, proses pelatihan akan dimulai dari nilai bobot (*weight*) yang terakhir dari *dataset* sebelumnya, sehingga proses pelatihan akan menjadi lebih cepat dibandingkan harus menggunakan *weight* awal yang diacak [2]. Pada penelitian ini, metofe *fine tuning* diterapkan pada dua model *network*, yaitu VGG16 dan ResNet50.

VGG16 merupakan model pertama yang akan dilakukan *fine tuning* pada penelitian ini. VGG16 merupakan salah satu model *network* dari VGGNet dikembangkan pada tahun 2014 oleh *Visual Geometry Group* di Universitas Oxford, yang menjadi asal dari penamaan VGGNet. Tiga komponen bangunannya persis sama dengan yang ada di LeNet dan AlexNet, yamg membedakan VGGNet adalah jaringan yang lebih dalam dengan lebih banyak lapisan *convolutional*, *pooling*, and

dense. Selain itu, tidak ada komponen baru yang diperkenalkan di sini. VGG16 terdiri dari 16 lapisan *weight*, yaitu 13 lapisan *convolution* dan 3 lapisan *fully connected*. Arsitekturnya yang seragam membuatnya terkenal di komunitas *deep learning* karena sangat mudah dipahami.

Arsitektur VGG16 terdiri dari serangkaian *convolutional building blocks* seragam dan diikuti oleh *unified pooling layer*, di mana semua lapisan konvolusi adalah filter berukuran kernel 3 × 3 dengan *stride* 1 dan nilai *padding* yang sama dengan *stride* dan semua lapisan *pooling* memiliki ukuran *pool* 2 × 2 dan nilai *stride* 2. Arsitektur VGGNet dikembangkan dengan menumpuk lapisan *convolutional* 3 × 3 dengan lapisan *pooling* 2 × 2 yang disisipkan setelah beberapa lapisan *convolutional*. Lapisan selanjutnya diikuti oleh *classifier* yang terdiri dari lapisan *fully connected* dan *softmax*.



Gambar 2. Arsitektur VGG16 [6]

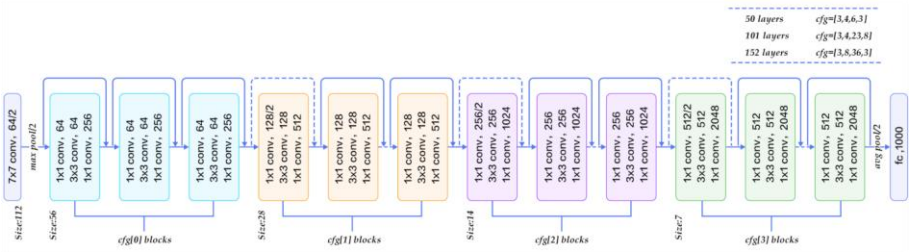
Tumpukan lapisan *convolutional* diikuti oleh tiga lapisan *Fully-Connected* (FC): dua lapisan pertama masing-masing memiliki 4096 buah *channel*, lapiasan ketiga untuk melakukan klasifikasi ILSVRC yang terdiri dari 1000 kelas sehingga berisi 1000 buah *channel* (satu *channel* untuk setiap kelas). Lapisan terakhir adalah lapisan fungsi aktivasi *softmax*. Konfigurasi lapisan *fully connected* seragam padai semua jaringan. Semua lapisan *hidden* dilengkapi dengan fungsi aktivasi *Rectified Non-Linear Unit* (ReLU) [7].

VGG16 mencapai top-5 *error rate* sebesar 8,1% pada *dataset* ImageNet dibandingkan dengan 15,3% yang dicapai oleh AlexNet. VGG19 bahkan lebih baik, yaitu mampu mencapai top-5 *error rate* ~7,4%. Walaupun memerlukan jumlah parameter yang lebih besar dan kedalaman VGGNet yang lebih besar dibandingkan dengan AlexNet, VGGNet membutuhkan jumlah *epoch* yang lebih sedikit untuk mencapai konvergensi karena regularisasi oleh *depth layer* yang lebih besar dan ukuran filter konvolusi yang lebih kecil [8].

Model *network* VGG16 termasuk model *network* yang relatif lama, namun model *network* ini masih digunakan dalam berbagai penelitian saat ini. Pada [9], VGG16 digunakan untuk melakukan klasifikasi pada pakaian. Pada [10], VGG16 digunakan untuk memecahkan masalah identifikasi dan klasifikasi sampah domestik. Pada [11], VGG16 digunakan pada pembangunan model klasifikasi fitur untuk klasifikasi air, lahan pertanian, bangunan, jalan, dan pohon untuk gambar SAR (*Synthetic Aperture Radar*).

Model kedua yang akan dilakukan *fine-tuning* adalah ResNet50. Model *network* ResNet50 dirancang untuk memecahkan masalah *vanishing gradient*, yaitu ketika gradien yang digunakan untuk melakukan *backpropagation* pada model menurun secara eksponensial sehingga mendekati nol. He et al. membuat pintasan yang memungkinkan gradien langsung dilakukan *backpropagation* ke lapisan sebelumnya[12]. Pintasan ini disebut *skip connections*, digunakan untuk mengalirkan informasi dari lapisan sebelumnya di jaringan ke lapisan berikutnya,

membuat jalur pintasan alternatif untuk gradien terus bergulir. kegunaan lain dari *skip connections* adalah memungkinkan model untuk mempelajari fungsi identitas, yang memastikan bahwa lapisan akan bekerja setidaknya sebaik lapisan sebelumnya.



Gambar 3. Arsitektur ResNet50 [13]

Untuk membangun bagian *feature extractor* dari ResNet, dimulai dengan lapisan *convolutional* dan *pooling* dan kemudian menumpuk sejumlah *residual block* untuk membangun jaringan. Pada *residual block*, digunakan representasi shallow yaitu VLAD [14] dan *Fisher Vector* [15]. Untuk kuantisasi vektor, pengkodean vektor residu [16] digunakan karena lebih efektif daripada pengkodean vektor asli. *Classifier* terdiri dari lapisan *fully connected* diikuti oleh *softmax*. Modul residual terdiri dari dua cabang, yaitu *shortcut path* dan *main path*. *Batch normalization* ditambahkan ke setiap lapisan konvolusional untuk mengurangi *overfitting* dan mempercepat pelatihan.

Sama seperti model *network* VGG16, ResNet50 masih digunakan dalam berbagai penelitian saat ini. Pada [17], ResNet50 digunakan untuk deteksi logo dan pengenalan merek dagang. Pada [18], ResNet50 digunakan untuk mengidentifikasi spesies jamur liar dan mengurangi terjadinya keracunan. Pada [19], ResNet50 digunakan untuk pengenalan ekspresi wajah dengan menambahkan tiga lapisan konvolusi sehingga fitur tingkat rendah dan tinggi dapat diekstraksi.

Kontribusi dari penelitian ini adalah untuk melihat bagaimana metode *fine tuning* bekerja pada kedua model *network*, yaitu VGG16 dan ResNet50. Penelitian dilakukan dengan melakukan pembekuan (*freezing*) pada sejumlah layer pada model *network*. *Freezing* pada lapisan model *network* berarti membekukan layer yang sudah dilatih sebelumnya untuk mencegahnya lapisan tersebut dilatih ulang saat pelatihan dilakukan. Model *network* VGG16 dan ResNet50 dipilih karena kedua model *network* mempunyai arsitektur yang berbeda satu sama lain. Model *network* VGG16 menggunakan model konvensional seperti LeNet dan AlexNet. Sedangkan model ResNet50 memperkenalkan *skip connections* pada modelnya.

Penelitian ini disusun secara berurutan dengan bagian B dijabarkan pengujian yang akan dilakukan. Pada bagian C, berisi hasil eksperimen pelatihan pada *dataset*. Pada bagian D berisi kesimpulan dari penelitian.

B. Metode Penelitian

Ada dua pengujian yang dilakukan. Pengujian pertama adalah performa pelatihan pada kedua *network* yaitu VGG16 dan ResNet50. Pada kedua *network* ini dilakukan metode *fine tuning* sebelum nantinya dilakukan pelatihan. Pelatihan pada kedua model dilakukan dengan Google Colab. Pada percobaan, model *network* VGG16 dan ResNet50 diambil dari dari modul keras dengan menggunakan *weight* hasil dari pelatihan dengan ImageNet. Pada kedua model *network*,

dilakukan empat percobaan, yang membedakannya adalah berapa lapisan yang dilakukan *freezing* sehingga menyisakan sejumlah lapisan yang bisa dilatih. Pelatihan dilakukan sebanyak 100 dan 200 *epoch*, dengan ukuran gambar 224*224 dan *batch size* 32. Yang membedakan pelatihan pada *network* VGG16 dan ResNet50 adalah *optimizer* yang digunakan. Pada *network* VGG16 digunakan *optimizer* Adadelta, sedangkan pada *network* ResNet50 digunakan *optimizer* Adam dengan *learning rate* 0,001. *Optimizer* adadelta adalah *optimizer* yang dikembangkan dari *Adagrad* yang mengadaptasi *learning rate* berdasarkan perubahan pembaruan gradien.

Dataset yang digunakan pada penelitian ini adalah *cats, dogs, horses, and humans dataset* oleh Osama Adel Elsayed, yang diambil dari Kaggle. *Dataset* terdiri dari empat kelas. Masing masing kelas mempunyai sebanyak 202 gambar. Jumlah *dataset* gambar yang digunakan adalah 808 gambar. Pelatihan menggunakan 80% dari *dataset*, sedangkan untuk validasi dan pengujian menggunakan masing-masing 10% dari dataset. Untuk *dataset* yang digunakan akan diacak, namun dengan *random seed* yang sama untuk setiap pelatihan untuk menjaga konsistensi.

Gambar 4. Contoh Gambar Dataset *cats, dogs, horses, and humans*

Untuk mengevaluasi kinerja pada pelatihan, metrik yang digunakan adalah *loss* dan akurasi. Metrik tersebut mencakup pada saat pelatihan dan pengujian. *Loss* adalah nilai yang mewakili penjumlahan kesalahan dalam model. Nilai ini mengukur seberapa baik kinerja model. Untuk menghitung *loss*, digunakan *loss function*. Pada kedua model pada penelitian ini, *loss function* yang digunakan adalah *Categorical Cross-Entropy*, yang merupakan gabungan dari fungsi aktivasi *Softmax* dan *Cross-Entropy*. Sedangkan akurasi mengukur seberapa baik prediksi model dengan membandingkan prediksi model dengan *ground truth* dengan hasil akhir berupa nilai persentase.

Pada penelitian ini, metrik *loss* dan akurasi juga akan ditampilkan dalam plot *learning curve*. Secara umum, *learning curve* adalah plot yang menunjukkan waktu (*epoch*) pada sumbu x dan pembelajaran atau peningkatan pada sumbu y. Ada tiga kondisi yang dapat dilihat dari *learning curve*, yaitu *underfit*, *overfit*, dan *good fit*. Plot *learning curve* menunjukkan *underfitting* ketika *training loss* tetap datar terlepas dari pelatihan, dan *validation loss* terus berkurang hingga akhir pelatihan. Plot *learning curve* menunjukkan *overfitting* ketika *training loss* terus berkurang seiring dengan jumlah *epoch*, dan ketika *validation loss* menurun ke satu titik dan mulai meningkat lagi. Plot *learning curve* menunjukkan kondisi *good fit* ketika

training loss menurun ke titik stabilitas, dan *validation loss* menurun ke titik stabilitas dan memiliki celah yang kecil dengan *training loss*.

Pengujian kedua yang dilakukan adalah implementasi *5-Fold cross validation*. Pengujian dilakukan dengan membagi data menjadi lima bagian yang sama. Masing-masing bagian digunakan dalam satu iterasi sebagai data untuk pengujian, dan pada iterasi lainnya sebagai bagian data untuk pelatihan. Karena *cross validation* melibatkan lebih dari satu pelatihan, digunakan *loss* rata-rata dan akurasi rata-rata untuk mengevaluasi hasilnya. Metrik tersebut mencakup pada saat pelatihan dan pengujian.

Gambar 5. Contoh *Cross Validation* [19]

Selanjutnya, hasil penelitian ini dibandingkan dengan hasil dari [4]. Penelitian tersebut dipilih karena menggunakan kedua model *network* pada penelitian ini, yaitu VGG16 dan ResNet50. Pada penelitian [4], kedua model *network* dilatih dengan menggunakan *dataset Best Artworks of All Time* dari Kaggle yang terdiri dari 8355 gambar. Hasil terbaik didapat dari pelatihan dengan parameter *batch size* 32, 400 *epoch*, dan *optimizer* model Adam dengan *learning rate* 0,1. Setelah dilakukan pelatihan pada penelitian tersebut, model VGG16 mendapat nilai akurasi 83.36% pada pelatihan dan 80,25% pada validasi. Sedangkan model ResNet50 mendapatkan nilai akurasi 89,32% pada pelatihan dan 87,15% pada validasi.

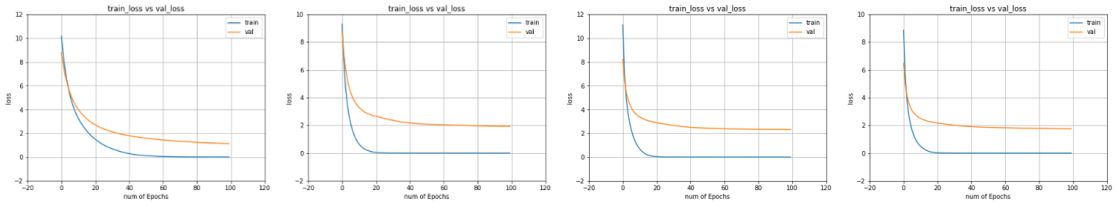
C. Hasil dan Pembahasan

Pada *network* VGG16, dilakukan empat percobaan, yang membedakannya adalah berapa layer yang dilakukan pembekuan sehingga menyisakan sejumlah lapisan yang bisa dilatih. Masing-masing pelatihan menggunakan parameter yang sama, yaitu *batch size* 32, 100 epoch, dan *optimizer* model Adadelta. Proses pelatihan menggunakan 80% dataset, validasi dan pengujian menggunakan masing-masing 10% dari *dataset*. Tabel 1 menunjukkan hasil pelatihan, dimana akurasi tertinggi didapat dari pelatihan dimana terdapat tiga *trainable layer* dengan akurasi 82.0988%.

Tabel 1. Hasil Pelatihan dengan *Fine tuning* pada VGG16 (100 *epoch*)

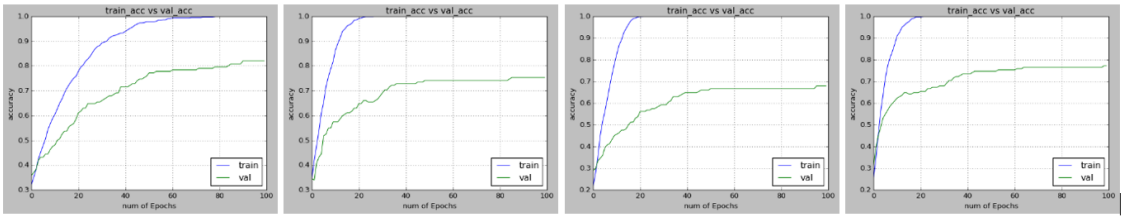
Trainable layer	**Waktu Pelatihan**	***Loss* Pelatihan**	**Akurasi Pelatihan**	***Loss* Pengujian**	**Akurasi Pengujian**
3	420 detik	0.02003	100%	1.1368	82.0988%
8	452 detik	0.00531	100%	1.9315	75.3086%
12	624 detik	0.00775	100%	2.3037	67.9012%
16	744 detik	0.00528	100%	1.7546	77.1605%

Selanjutnya, pada gambar 6 menunjukkan hasil pelatihan dengan memperhatikan progres *loss* pada setiap *epoch* dari 0 sampai 100. Terlihat bahwa proses pelatihan dengan *trainable layer* 3 mempunyai jarak pelatihan dan *validation loss* paling kecil, Dibandingkan dengan *trainable layer* lainnya, nilai *loss* untuk *dataset* pelatihan dan *validation* dengan *trainable layer* 3 lebih lambat menuju nilai *loss* minimum. Hal ini menandakan *learning curve* pada pelatihan dengan *trainable layer* 3 memberikan model yang *good fit*.



Gambar 6. Hasil Pelatihan *loss* pada Metode *Fine Tuning* pada VGG16 dengan 100 epoch dari kiri ke kanan: 3, 8, 12, dan 16 *trainable layer*

Gambar 7 menunjukkan hasil pelatihan dengan memperhatikan progres akurasi pada setiap epoch dari 0 sampai 100. Terlihat bahwa proses pelatihan dengan 3 buah *trainable layer* mempunyai *validation accuracy* paling tinggi. Dibandingkan dengan jumlah *trainable layer* lainnya, nilai *loss* untuk pelatihan dan *validation* dengan 3 buah *trainable layer* lebih lambat menuju nilai akurasi maksimum.



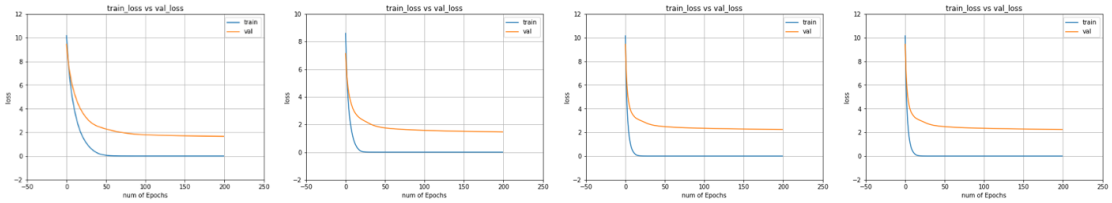
Gambar 7. Hasil *Hasil* Pelatihan Akurasi *pada Metode Fine Tuning* pada VGG16 dengan 100 *epoch* dari kiri ke kanan: 3, 8, 12, dan 16 *trainable layer*

Tabel 2 menunjukkan hasil pelatihan dengan parameter yang digunakan adalah parameter yang sama dengan tabel 1, yaitu *batch size* 32, dan *optimizer* model adadelta. Pelatihan menggunakan 80% dari *dataset*, validasi dan pengujian masing-masing menggunakan 10% dari *dataset* namun dengan jumlah *epoch* 200. Akurasi tertinggi didapat dari pelatihan dimana terdapat enam belas *trainable layer* dengan akurasi 82.0988%.

Tabel 2. Hasil Pelatihan dengan *Fine Tuning* pada VGG16 (200 epoch)

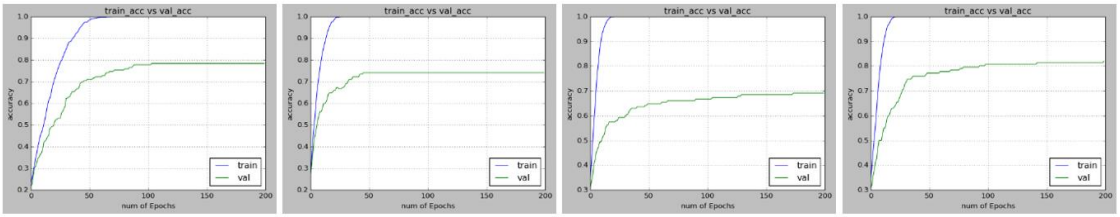
Trainable layer	**Waktu Pelatihan**	***Loss* Pelatihan**	**Akurasi Pelatihan**	***Loss* Pengujian**	**Akurasi Pengujian**
3	843 detik	0.00389	100%	1.6685	78.3951%
8	920 detik	0.00186	100%	1.4601	74.0741%
12	1101 detik	0.00186	100%	2.2447	69.1358%
16	1417 detik	0.00212	100%	1.2675	82.0988%

Selanjutnya, pada gambar 8 menunjukkan hasil pelatihan dengan memperhatikan progres *loss* pada setiap *epoch* dari 0 sampai 100. Terlihat bahwa proses pelatihan dengan *trainable layer* 8 mempunyai celah pelatihan dan *validation loss* paling kecil. Dibandingkan dengan *trainable layer* lainnya, nilai *loss* untuk pelatihan dan validasi dengan tiga buah *trainable layer* lebih lambat menuju nilai *loss* minimum. Ini menandakan *learning curve* pada pelatihan dengan *trainable layer* 3 memberikan model yang *good fit*.



Gambar 8. Hasil Pelatihan *loss* pada Metode *Fine tuning* pada VGG16 dengan 200 *epoch* dari kiri ke kanan: 3, 8, 12, dan 16 *trainable layer*

Gambar 9 menunjukkan hasil pelatihan dengan memperhatikan progres akurasi pada setiap *epoch* dari 0 sampai 100. Terlihat bahwa proses pelatihan dengan *trainable layer* 16 mempunyai *validation accuracy* paling tinggi. Dibandingkan dengan *trainable layer* lainnnya, nilai *loss* untuk pelatihan dan validasi dengan jumlah *trainable layer* 3 lebih lambat menuju nilai akurasi maksimum.



Gambar 9. Hasil Pelatihan Akurasi pada *Metode Fine tuning* pada VGG16 dengan 200 *epoch* dari kiri ke kanan: 3. 8, 12, dan 16 *trainable layer*

Pada *network* VGG16, dilakukan *5-foid cross validation* pelatihan sesuai dengan jumlah *trainable layer* yang berbeda. Masing-masing pelatihan menggunakan parameter yang sama, yaitu *batch size* 32, jumlah *epoch* 100, dan *optimizer* model adadelta. Pelatihan menggunakan 80% dari dataset, *validation* dan pengujian menggunakan masing-masing 10% dari *dataset*. Karena *cross validation* dilakukan, untuk setiap pelatihan model *network* dikonfigurasi ulang untuk setiap pelatihan. Tabel 3 menunjukkan hasil pelatihan, performa tertinggi

didapat dari pelatihan dengan tiga buah *trainable layer* dengan *loss* rata-rata 1.3818, akurasi rata-rata 78.5859%, dan standar deviasi ± 3.6595.

Tabel 3. Hasil Pelatihan dengan *5-Fold Cross Validation* pada VGG16 (100 *epoch*)

Trainable layer	Waktu Total	*Loss* Rata-rata	Akurasi Rata-rata	*Standar Deviasi*
3	1678 detik	1.3818	78.5859%	± 3.6595
8	2119 detik	1.7107	75.1115%	± 4.3701
12	2444 detik	1.8701	70.4240%	± 4.3526
16	3247 detik	1.9546	70.5482%	± 4.8162

Tabel 4 menunjukkan hasil *cross validation* pelatihan dengan nilai parameter sama dengan tabel 3, namun dengan jumlah *epoch* 200. Performa tertinggi didapat dari pelatihan dimana terdapat tiga buah *trainable layer* dengan *loss* rata-rata 1.3531, akurasi rata-rata 78.7125%, dan standar deviasi ± 3.2122.

Tabel 4. Hasil Pelatihan dengan *5-Fold Cross Validation* pada VGG16 (200 *epoch*)

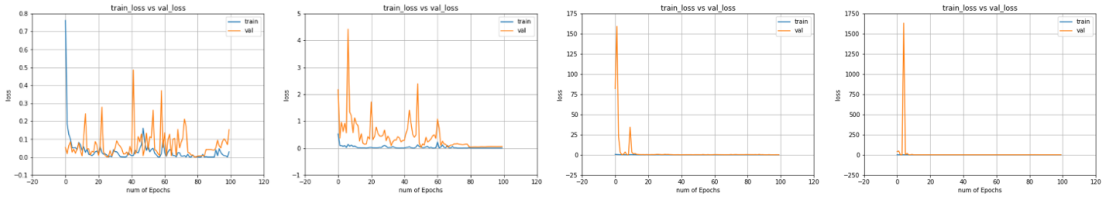
Trainable layer	Waktu Total	*Loss* Rata-rata	Akurasi Rata-rata	*Standar Deviasi*
3	3296 detik	1.3531	78.7125%	± 3.2122
8	3900 detik	2.0014	72.1463%	± 3.6592
12	4887 detik	1.9858	71.5259%	± 2.9631
16	6355 detik	1.7883	71.0443%	± 3.9023

Pada *network* ResNet50, dilakukan empat percobaan, yang membedakannya adalah berapa lapisan yang dilakukan *freeze* sehingga menyisakan sejumlah lapisan yang bisa dilatih. Masing-masing pelatihan menggunakan parameter yang sama, yaitu *batch size* 32, jumlah *epoch* 100, *optimizer* model Adam dengan *learning rate* 0,001. Pelatihan menggunakan 80% dari *dataset*, validasi dan pengujian menggunakan masing-masing 10% dari *dataset*. Tabel 5 menunjukkan hasil pelatihan, dimana akurasi tertinggi didapat dari pelatihan dimana terdapat enam buah *trainable layer* dengan akurasi 99.3827%.

Tabel 5. Hasil Pelatihan pada ResNet50 (100 *epoch*)

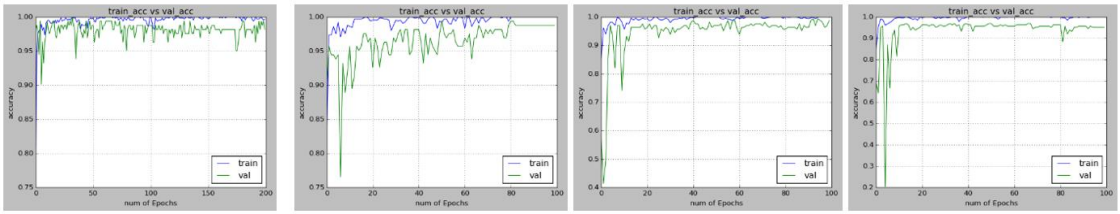
Trainable layer	Waktu Pelatihan	*Loss* Pelatihan	Akurasi Pelatihan	*Loss* Pengujian	Akurasi Pengujian
6	327 detik	0.0290	99.54%	0.1527	99.3827%
16	326 detik	0.0251	100%	0.0561	98.7654%
26	343 detik	0.0014	99.85%	0.0290	98.7654%
38	386 detik	0.0036	100%	0.4602	95.0617%

Selanjutnya, pada gambar 10 menunjukkan hasil pelatihan dengan memperhatikan progres *loss* pada setiap *epoch* dari 0 sampai 100. Terlihat bahwa proses pelatihan, khususnya *trainable layer* 6, *validation loss* mengalami kenaikan dan penurunan pada grafiknya. Hal ini menandakan *learning curve* pada pelatihan dengan *trainable layer* 6 memberikan model yang *overfit*. Hal ini terjadi karena model memiliki kapasitas lebih dari yang dibutuhkan untuk masalah, dan akibatnya terlalu banyak fleksibilitas. Selain itu, *overfit* terjadi karena model dilatih terlalu lama.



Gambar 10. Hasil Pelatihan *loss pada Metode Fine tuning* pada ResNet50 dengan 100 *epoch* dari kiri ke kanan: 6. 12, 26, dan 38 *trainable layer*

Pada gambar 11 menunjukkan hasil pelatihan dengan memperhatikan progres akurasi pada setiap *epoch* dari 0 sampai 100. Terlihat bahwa proses pelatihan dengan jumlah *trainable layer* mempunyai *validation accuracy* paling tinggi. Dibandingkan dengan *trainable layer* lainnnya, nilai *loss* untuk *dataset* pelatihan dan validasi dengan tiga buah *trainable layer* lebih lambat menuju nilai akurasi maksimum.



Gambar 11. Hasil Pelatihan Akurasi *pada Metode Fine tuning* pada ResNet50 dengan 100 *epoch* dari kiri ke kanan: 6. 12, 26, dan 38 *trainable layer*

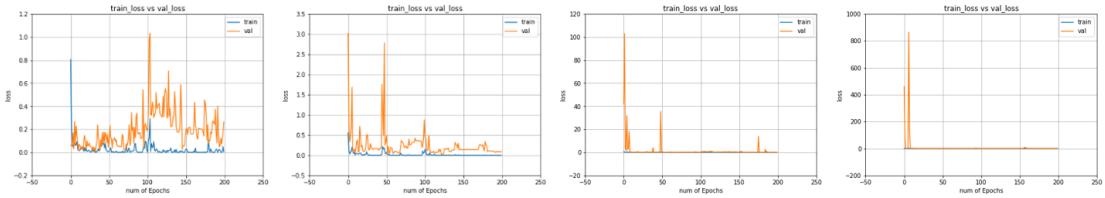
Tabel 6 menunjukkan hasil pelatihan dengan parameter yang digunakan adalah parameter yang sama dengan tabel 5, yaitu *batch size* 32, *optimizer* model Adam dengan *learning rate* 0,001. Pelatihan menggunakan 80% dari *dataset*, *validation* dan pengujian menggunakan masing-masing 10% dari *dataset* namun dengan jumlah *epoch* 200. Akurasi tertinggi didapat dari pelatihan dimana terdapat enam buah *trainable layer* dengan akurasi 98.1481%.

Tabel 6. Hasil Pelatihan dengan *Fine tuning* pada ResNet50 (200 *epoch*)

Trainable layer	**Waktu Pelatihan**	***Loss* Pelatihan**	**Akurasi Pelatihan**	***Loss* Pengujian**	**Akurasi Pengujian**
6	597 detik	0.0067	99.69%	0.2630	98.1481%
16	684 detik	0.0079	100%	0.0864	98.7654%
26	745 detik	0.0063	99.69%	0.1349	97.5309%
38	745 detik	0.0036	100%	0.1063	98.1481%

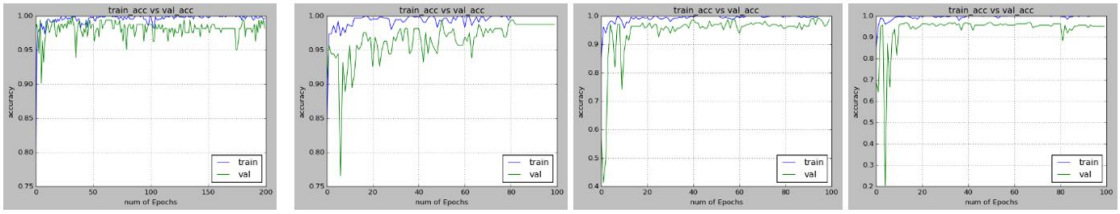
Gambar 12 menunjukkan hasil pelatihan dengan memperhatikan progres *loss* pada setiap *epoch* dari 0 sampai 100. Terlihat bahwa proses pelatihan dengan enam belas buah *trainable layer* mempunyai jarak pelatihan dan *validation loss* paling kecil. Terlihat bahwa proses pelatihan, khususnya *trainable layer* 6 dan 16, *validation loss* mengalami kenaikan dan penurunan pada grafiknya. Hal ini menandakan *learning curve* pada pelatihan dengan *trainable layer* 6 memberikan model yang *overfit*. Hal ini terjadi karena model memiliki kapasitas lebih dari yang

dibutuhkan untuk masalah, dan akibatnya terlalu banyak fleksibilitas. Selain itu, *overfit* terjadi karena model dilatih terlalu lama.



Gambar 12. Hasil Pelatihan *loss pada Metode Fine tuning* pada ResNet50 dengan 200 *epoch* dari kiri ke kanan: 6. 16, 26, dan 38 *trainable layer*

Gambar 13 menunjukkan hasil pelatihan dengan memperhatikan progres akurasi pada setiap *epoch* dari 0 sampai 100. Terlihat bahwa proses pelatihan dengan jumlah *trainable layer* enam belas mempunyai *validation accuracy* paling tinggi.



Gambar 13. Hasil Pelatihan Akurasi *pada Metode Fine tuning* pada ResNet50 dengan 200 *epoch* dari kiri ke kanan: 6. 16, 26, dan 38 *trainable layer*

Sama seperti percobaan pada model *network* VGG16, pada *network* ResNet50, pelatihan dengan 5-fold *cross validation* dilakukan sesuai dengan jumlah *trainable layer* yang berbeda. Masing-masing pelatihan menggunakan parameter yang sama, yaitu *batch size* 32, jumlah *epoch* 200, *optimizer* model Adam dengan *learning rate* 0,001. Pelatihan menggunakan 80% *dataset*, validasi dan pengujian menggunakan masing-masing 10% dataset. Karena *cross validation* dilakukan, untuk setiap pelatihan model *network* dikonfigurasi ulang untuk setiap pelatihan. Tabel 7 menunjukkan hasilnya, dimana performa tertinggi didapat dari pelatihan dengan enam *trainable layer* dengan *loss* rata-rata 0.1282, akurasi rata-rata 98.8858%, dan standar deviasi ± 0.4648.

Tabel 7. Hasil Pelatihan dengan *5-Fold Cross Validation* pada ResNet50 (100 *epoch*)

Trainable layer	**Waktu Total**	***Loss Rata-rata***	***Akurasi Rata-rata***	***Standar Deviasi***
6	1230 detik	0.1282	98.8858%	± 0.4648
16	1270 detik	0.2550	97.8920%	± 1.5022
26	1321 detik	0.2058	97.7724%	± 0.3021
38	1482 detik	0.2010	96.6589%	± 1.2080

Tabel 8 menunjukkan hasil *5-fold cross validation* pelatihan dengan nilai parameter sama dengan tabel 7, namun dengan jumlah *epoch* 200. *Loss* rata-rata

paling rendah adalah pada jumlah *trainable layer* 26 dengan 0.1698. Akurasi rata-rata paling tinggi adalah pada *trainable layer* 16 dengan nilai 98.8873%.

Tabel 8. Hasil Pelatihan dengan *5-Fold Cross Validation* pada ResNet50 (200 *epoch*)

Trainable layer	**Waktu Total**	***Loss* Rata-rata**	**Akurasi Rata-rata**	***Standar Deviasi***
6	2162 detik	0.3841	98.3896%	± 1.3363
16	2548 detik	0.2102	98.8873%	± 0.6051
26	2523 detik	0.1698	97.8966%	± 0.6292
38	3045 detik	0.3790	96.0378%	± 1.011

Berdasarkan pengujian yang dilakukan, dapat disimpulkan bahwa dari pelatihan dengan metode *fine tuning*, performa terbaik didapat pada *trainable layer* yang paling kecil, baik pada network VGG16 maupun ResNet50. Karena fitur tingkat yang lebih tinggi saling terkait (karena kumpulan datanya serupa), *fine tuning* melalui seluruh jaringan tidak benar-benar diperlukan. *Dataset* Imagenet dilatih dengan 1000 kelas, dengan tiga kelas yang sama dengan dataset untuk pelatihan yaitu kucing (*cats*), anjing (*dogs*), dan kuda (*horses*). Jadi, *fine tuning* yang memberikan hasil yang paling baik adalah membekukan jaringan yang telah dilatih sebelumnya.

Selanjutnya, hasil terbaik dari penelitian ini dilakukan perbandingan dengan penelitian dari [3]. Pada penelitian [3], model *network* VGG16 dan ResNet50 dilatih dengan menggunakan *dataset Best Artworks of All Time* dari Kaggle yang terdiri dari 8355 gambar. Hasil terbaik didapat dari pelatihan dengan parameter *batch size* 32, 400 epoch, dan *optimizer* model Adam dengan *learning rate* 0,1.

Tabel 9. Hasil Pelatihan pada ResNet50 (100 *epoch*)

Parameter	**Penelitan Ini**	**Penelitian Pembanding**
Jumlah gambar *dataset*	808 gambar	8355 gambar
Batch size	32	32
Jumlah *epoch*	100	400
Optimizer (VGG16)	Adadelta	Adam (*learning rate* = 0,1)
Optimizer (ResNet50)	Adam (*learning rate* = 0,001)	Adam (*learning rate* = 0,1)
Akurasi Saat Pelatihan (VGG16)	82.10%	83.36%
Akurasi saat pelatihan (ResNet50)	99.38%	89.32%

Perbandingan penelitian dapat dilihat pada tabel 9. Dari tabel ini, dapat dilihat bahwa penelitian pembanding memberikan nilai akurasi terbaik pada pelatihan model *network* VGG16 dengan nilai 83.36%. Sedangkan penelitian ini memberikan nilai akurasi terbaik pada pelatihan model *network* ResNet50 dengan nilai 99.3827%. Dari kedua penelitian ini, dapat juga dilihat bahwa model *network* ResNet50 mempunyai nilai akurasi terbaik dibandingkan dengan VGG16.

D. Simpulan

Penelitian ini melakukan implementasi pada salah satu metode *transfer learning*, yaitu *fine tuning* dengan melakukan pembekuan sehingga menyisakan sejumlah *trainable layer*. Penelitian ini melakukan implamentasi salah satu metode *transfer learning*, yaitu *fine tuning* dengan cara melakukan pelatihan pada model *network* VGG16 dan ResNet50 dengan beberapa lapisan pada model dibekukan. Hasil uji percobaan menunjukkan bahwa melakukan pelatihan model dengan *trainable layer* yang paling sedikit. menghasilkan performa yang lebih baik. Dari hasil uji pada model network VGG16, performa terbaik didapat dari pelatihan pada tiga buah *trainable layer* pada jumlah *epoch* 100 dengan nilai *loss* 1.1368 dan akurasi 82.0988%, dan pelatihan pada enam belas buah *trainable layer* pada jumlah *epoch* 200 dengan nilai *loss* 1.2675 dan akurasi 82.0988%. Sedangkan pada model *network* ResNet50, performa terbaik didapat pada pelatihan dengan enam *trainable layer* pada jumlah *epoch* 100 dengan nilai *loss* 0.1527 dan akurasi 99.3827% dan 200 dengan nilai *loss* 0.0864 dan akurasi 98.7654%. Secara keseluruhan, yaitu pada training dan *cross validation*, performa terbaik didapat pada jumlah trainable layer yang sedikit, yaitu tiga *trainable layer* pada VGG16 dan enam *trainable layer* pada ResNet50.

E. Ucapan Terima Kasih

Penulis mengucapkan terima kasih kepada Ibu Prof. Dr. Ir. Aniati Murni Arymurthy, M.Sc. dan Ibu Dr. Dina Chahyati, S.Kom., M.Kom. untuk bimbingan dan dukungannya demi terselesaikannya penelitian ini.

F. Referensi

[1] Q. Yang, Y. Zhang, W. Dai, & S. J. Pan, *Transfer learning*, Cambridge University Press, 2020.

[2] M. Elgendy, *Deep Learning for Vision Systems*, Manning Publications Co, 2020.

[3] D. Bourke, 04. Transfer Learning with TensorFlow Part 1: Feature Extraction, https://dev.mrdbourke.com, 2021, diakses tanggal 20 November, 2022

[4] G. K. Masilamani and R. Valli, "Art Classification with Pytorch Using Transfer Learning," *2021 International Conference on System, Computation, Automation and Networking (ICSCAN),* pp. 1-5, 2021

[5] K. Kaur, R. Dhir and K. Kumar, "Transfer Learning approach for analysis of epochs on Handwritten Digit Classification" *2021 2nd International Conference on Secure Cyber Computing and Communications (ICSCCC)*, pp. 456-458, 2021

[6] J. McDermott, Hands-on Transfer Learning with Keras and the VGG16 Model, https://www.learndatasci.com/tutorials/hands-on-transfer-learning-keras, 2022, diakses tanggal 20 November 2022

[7] Karen Simonyan and Andrew Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition” *In ICLR*, 2015

[8] Krizhevsky, A., Sutskever, I., and Hinton, G. E. “ImageNet classification with deep convolutional neural *networks*” *In NIPS*, pp. 1106–1114, 2012.

[9] W. Yi, S. Ma, H. Zhang and B. Ma, "Classification and improvement of multi label image based on vgg16 *network*" *2022 3rd International Conference on Information Science, Parallel and Distributed Systems (ISPDS)*, pp. 243-246, 2022.

[10] H. Wang, "Garbage Recognition and Classification System Based on Convolutional Neural *Network* VGG16" *2020 3rd International Conference on Advanced Electronic Materials, Computers and Software Engineering (AEMCSE)*, pp. 252-255, 2020.

[11] J. Tao, Y. Gu, J. Sun, Y. Bie and H. Wang, "Research on vgg16 convolutional neural *network* feature classification algorithm based on Transfer Learning" *2021 2nd China International SAR Symposium (CISS)*, pp. 1-3, 2021.

[12] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770-778, 2016.

[13] A. Rastogi, ResNet50, https://blog.devgenius.io/ResNet50, 2022, diakses tanggal 20 November 2022

[14] H. Jegou, F. Perronnin, M. Douze, J. Sanchez, P. Perez, and C. Schmid, Aggregating local image descriptors into compact codes. *TPAMI*, 2012.

[15] F. Perronnin and C. Dance. "Fisher kernels on visual vocabularies for image categorization" *In CVPR*, 2007.

[16] H. Jegou, M. Douze, and C. Schmid, Product quantization for nearest neighbor search. *TPAMI*, 2011.

[17] Sarwo, Y. Heryadi, E. Abdulrachman and W. Budiharto, "Logo detection and brand recognition with one-stage logo detection framework and simplified ResNet50 backbone" *2019 International Congress on Applied Information Technology (AIT)*, pp. 1-6, 2019.

[18] Q. Jiaxin, Y. Pengfei, L. Haiyan and L. Hongsong, "Research on Classification of Wild Fungi Based on Improved ResNet50 *Network*," *2021 6th International Conference on Image, Vision and Computing (ICIVC)*, pp. 168-173, 2021.

[19] S. Petluru and P. Singh, "Transfer Learning-based Facial Expression Recognition with modified ResNet50," *2022 IEEE World Conference on Applied Intelligence and Computing (AIC)*, pp. 237-241, 2022.

[20] R. Patro, Cross-Validation: K Fold vs Monte Carlo, https://towardsdatascience.com, 2021, diakses tanggal 28 November 2022