p-ISSN : 2443-2210
e-ISSN : 2443-2229

Jurnal Teknik Informatika dan Sistem Informasi
Volume 6 Nomor 2 Agustus 2020

Augmentasi Data Pengenalan Citra Mobil Menggunakan Pendekatan *Random Crop*, *Rotate*, dan *Mixup*

http://dx.doi.org/10.28932/jutisi.v6i2.2688

Joseph Sanjaya[#1], Mewati Ayub[✉#2]

[#] *Program Studi Magister Ilmu Komputer, Universitas Kristen Maranatha*
Jl. Surya Sumantri No.65, Bandung
[1]mi1879011@student.it.maranatha.edu
[2]mewati.ayub@it.maranatha.edu

Abstract — **Deep convolutional neural networks (CNNs) have achieved remarkable results in two-dimensional (2D) image detection tasks. However, their high expression ability risks overfitting. Consequently, data augmentation techniques have been proposed to prevent overfitting while enriching datasets. In this paper, a Deep Learning system for accurate car model detection is proposed using the ResNet-152 network with a fully convolutional architecture. It is demonstrated that significant generalization gains in the learning process are attained by randomly generating augmented training data using several geometric transformations and pixel-wise changes, such as image cropping and image rotation. We evaluated data augmentation techniques by comparison with competitive data augmentation techniques such as mixup. Data augmented ResNet models achieve better results for accuracy metrics than baseline ResNet models with accuracy 82.6714% on Stanford Cars Dataset.**

Keywords— **Convolutional Neural Network; Data Augmentation; ResNet**

I. PENDAHULUAN

Dewasa ini perkembangan teknologi semakin pesat, sehingga kehidupan manusia sudah banyak dibantu oleh berbagai mesin *vision*. *Convolutional Neural Networks (CNNs)* termasuk salah satu algoritma yang telah mencapai kemajuan yang signifikan terhadap banyak tugas-tugas mesin *vision* dalam beberapa tahun ini, termasuk klasifikasi citra, deteksi objek, segmentasi semantik, dan seterusnya. Hal ini membuktikan bahwa algoritma *CNN* memiliki kemampuan yang kuat dalam melakukan ekstraksi fitur pada gambar. Oleh karena itu, algoritma ini sering digunakan dalam tugas-tugas pengenalan objek, termasuk dalam pengenalan model mobil [1].

Jaringan *CNN* telah memberikan pencapaian yang signifikan dalam bidang klasifikasi gambar dan pemrosesan gambar karena banyaknya parameter dan kemampuan ekspresi yang kaya. Studi baru-baru ini menunjukkan bahwa kinerja *CNN* secara proporsional sebanding dengan jumlah sampel pelatihan. Sebaliknya, tanpa sampel pelatihan yang cukup, *CNN* dengan berbagai parameter memiliki risiko *overfitting* karena model *CNN* menghafal fitur detail gambar pelatihan yang tidak dapat digeneralisasi. Karena mengumpulkan banyak sampel data akan menyebabkan biaya yang tinggi, maka metode augmentasi data telah umum digunakan. Augmentasi data meningkatkan variasi gambar dengan memanipulasi transformasi dimensi gambar.

Dalam penelitian ini akan diekplorasi arsitektur ResNet pada jaringan *CNN*, dua hal yang menjadi fokus permasalahan utama dalam penelitian ini adalah:

a. Mendapatkan cara untuk menerapkan augmentasi data terhadap arsitektur ResNet untuk tugas mengenali model mobil pada *Stanford Cars Dataset*.
b. Melakukan studi komparatif terhadap arsitektur ResNet dengan dan tanpa teknik augmentasi data yang diusulkan.

Berdasarkan fokus tersebut, maka makalah ini akan membahas tentang bagaimana penggunaan teknik augmentasi data *random crop, random rotate*, dan *mixup* terhadap model *deep learning Convolutional Neural Network* (*CNN*) dengan menggunakan arsitektur ResNet dan membandingkan performa antara model *baseline* dan model yang dilatih menggunakan augmentasi data.

Data pelatihan yang digunakan merupakan *dataset* yang disediakan oleh *Stanford University*. *Dataset* berisi 16.185 gambar dari 196 kelas mobil. Data ini dibagi menjadi 8.144 gambar pelatihan dan 8.041 gambar pengujian, di mana setiap kelas telah dibagi secara kasar menjadi 50-50 [2]. Atribut kelas adalah tahun pembuatan dan model, mis. 2012 Tesla Model S atau 2012 BMW M3 coupe. Hasil dari penelitian ini berupa model *deep learning CNN* berdasarkan arsitektur ResNet yang bisa digunakan untuk mengklasifikasi model mobil dan analisis performa untuk

setiap model.

II. KAJIAN LITERATUR

A. Tinjauan Pustaka

Deteksi kendaraan merupakan topik penting dalam bidang komputer visi, yang telah banyak diteliti Berbagai algoritma deteksi kendaraan telah diusulkan dari berbagai peneliti berdasarkan fitur yang berbeda dan / atau pengklasifikasi yang berbeda. Saat ini, sejumlah besar studi dilakukan untuk deteksi objek tunggal dan telah mencapai hasil yang diinginkan. Mengingat kompleksitas lingkungan perkotaan untuk deteksi kendaraan, perlu untuk mendapatkan keseimbangan yang baik antara akurasi dan efisiensi deteksi. Yun Wei mengusulkan algoritma deteksi kendaraan dua langkah dengan menggunakan fitur gabungan Harr dan HOG [2]. Algoritma dapat memastikan efisiensi operasional yang lebih baik dengan alasan adaptif dengan lingkungan yang kompleks. Proses pengambilan target dua langkah didasarkan pada strategi pendeteksian kasar-ke-halus, yang mengandung terutama dua langkah, yaitu, segmentasi awal target tampilan depan untuk mempersempit wilayah *region of interest* (*ROI*), dan ekstraksi target kendaraan yang tepat di *ROI* [3]. M. Liu mengusulkan *Hierarchical Joint CNN-Based* untuk menangani tugas pengenalan mobil. Penelitiannya signifikan terhadap masalah keamanan konten karena model yang terlatih dapat digunakan untuk mengidentifikasi model mobil dalam sistem pemantauan video [4].

Lee Hu mengusulkan strategi *pooling* berbobot spasial, yang menunjukkan daerah diskriminatif dari gambar *input* yang digunakan oleh *DCNNs* untuk mengklasifikasikan subkategori objek. Metode yang diusulkan ini telah mencapai hasil yang dilaporkan terbaik pada set data publik *Stanford Cars*-196 dan CompCars. Penelitian ini menunjukkan bahwa fitur lokal dan saluran *pooling* spasial dapat diekstraksi dari satu *DCNN* dan *channel pooling* ini dapat dipelajari dalam proses pelatihan *end-to-end* dari *DCNN* [5]. Singkatnya, metode yang disebutkan di atas mengusulkan model deteksi target yang berbeda dari perspektif aplikasi yang berbeda. Dibandingkan dengan keberhasilan yang dicapai untuk deteksi objek tunggal, studi lebih lanjut harus dilakukan untuk deteksi beberapa objek dalam lingkungan yang kompleks dengan akurasi deteksi dan efisiensi yang memadai. Linan Zhang dan Hayden Schaeffer menganalisis hubungan antara kedalaman, regularisasi, dan stabilitas ruang fitur terhadap model *CNN* dengan arsitektur ResNet dan variansinya. Eksperimen komputasi pada varian yang diajukan menunjukkan bahwa keakuratan ResNet dipertahankan dan akurasi tampaknya monoton sehubungan dengan kedalaman [6].

Mark Sandler dan timnya melakukan penelitian terhadap MobileNetV2 yang selanjutnya akan dipergunakan pada implementasi EfficientNet. Mark Sandler mengukur kinerja model pada klasifikasi Imagenet, deteksi objek COCO, segmentasi gambar VOC. Mark Sandler mengevaluasi *trade-off* antara akurasi, dan jumlah operasi yang diukur dengan *multiply-add (MAdd)* [7]. Hyo Lee dan timnya mengusulkan pendekatan pembelajaran mendalam baru untuk *Make and model recognition* (*MMR*) menggunakan arsitektur SqueezeNet. Menurut Hyo Lee arsitektur SqueezeNet dengan koneksi *bypass* antara modul *Fire*, varian dari vanilla SqueezeNet, digunakan untuk penelitian ini, yang membuat sistem *MMR* yang diajukan lebih efisien. Hasil percobaan pada kumpulan data kendaraan skala besar menunjukkan bahwa model yang diusulkan mencapai tingkat akurasi 96,3% pada tingkat peringkat-1 dengan irisan waktu efisien 108,8 ms. Untuk tugas inferensi, model dalam yang digunakan membutuhkan ruang kurang dari 5 MB dan karenanya memiliki kelayakan yang besar dalam aplikasi waktu nyata [8].

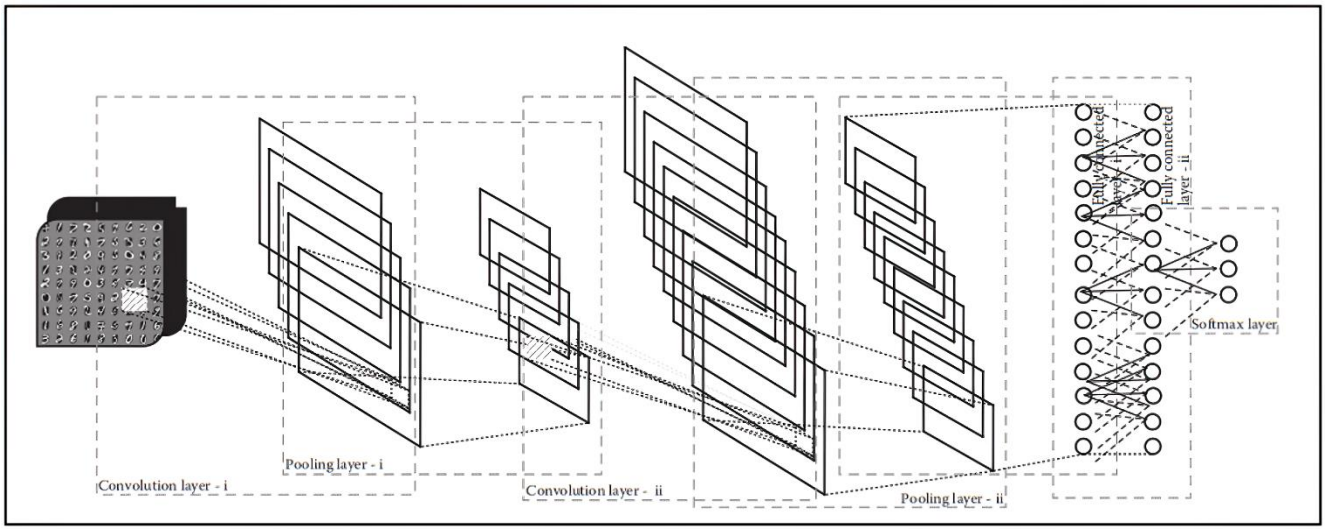
Ryo Takahashi dan timnya mengusulkan penggunaan augmentasi data RICAP (*Random Image Cropping and Patching*) yang secara acak memotong empat gambar dan menggabungkannya untuk membuat gambar pelatihan baru. Selain itu, RICAP mencampur label kelas dari empat gambar, menghasilkan keuntungan yang mirip dengan label *smoothing* [9].

B. Rekayasa Fitur (Features Engineering)

Rekayasa fitur adalah proses pembuatan/pengembangan fitur *input* baru dari data yang sudah ada. Rekayasa fitur merupakan proses mengubah data *raw* menjadi fitur yang lebih baik mewakili masalah yang mendasari model prediktif, sehingga meningkatkan akurasi model pada data yang tidak terlihat [10]. Secara umum, ilmuwan data dapat menganggap pembersihan data sebagai proses *substraction* dan rekayasa fitur sebagai proses *enhancement* pada model.

Proses dari rekayasa fitur dapat terlihat seperti[11]:

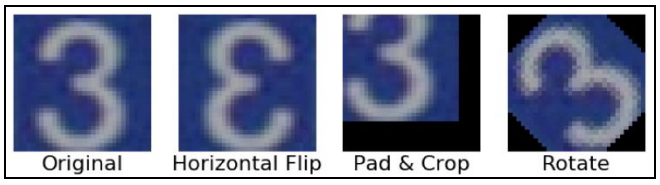
a. *Brainstorm feature*s: mencari *insight* tersembunyi pada data, melihat banyak data, mempelajari rekayasa fitur pada masalah yang dihadapi dan melihat apa yang dapat dilakukan rekayasa.
b. *Devise feature*s: langkah ini berbeda-beda sesuai dengan masalah yang dihadapi, tetapi ilmuwan data dapat menggunakan ekstraksi fitur otomatis, konstruksi fitur manual, atau campuran keduanya.
c. *Select feature*s: dengan memanfaatkan *feature importances* yang berbeda dan metode pemilihan fitur untuk mempersiapkan satu atau lebih "tampilan" untuk model dapat beroperasi.
d. *Evaluate models*: Memperkirakan akurasi model pada data yang tidak terlihat menggunakan fitur yang dipilih.



Gambar 1. *Convolutional Neural Network* [1]

C. Augmentasi Data

Augmentasi data adalah strategi yang memungkinkan praktisi untuk secara signifikan meningkatkan keragaman data yang tersedia untuk model pelatihan, tanpa benar-benar mengumpulkan data baru. Teknik augmentasi data seperti *cropping*, *padding*, dan *flipping horizontal* umumnya digunakan untuk melatih jaringan neural besar. Namun, sebagian besar pendekatan yang digunakan dalam pelatihan jaringan neural hanya menggunakan tipe augmentasi dasar. Sementara arsitektur jaringan neural telah diselidiki secara mendalam. Contoh augmentasi data terhadap gambar dapat dilihat pada gambar 2 [12].



Gambar 2. Contoh Augmentasi Data [12]

D. Convolutional Neural Network (CNN)

*Convolutional Neural Network*s (*CNN*s) dirancang untuk tugas pengenalan gambar dan awalnya diterapkan pada tantangan pengenalan angka tulisan tangan (Fukushima 1980; LeCun 1989). Tujuan desain dasar *CNN* adalah untuk membuat jaringan di mana *neuron* di lapisan awal jaringan akan mengekstraksi fitur visual lokal, dan *neuron*s di lapisan selanjutnya akan menggabungkan fitur-fitur ini untuk membentuk fitur tingkat tinggi [1].

CNN adalah jaringan saraf di mana sinyal diumpankan ke dalam set pasangan lapisan *pooling convolutional* yang ditumpuk (lapisan *convpool*), dan *output* dari *layer* terakhir diumpankan ke dalam setumpuk lapisan yang terhubung sepenuhnya yang memberi makan ke dalam lapisan *softmax*. Gambar 1 menunjukkan arsitektur CNN.

CNN memanfaatkan proses konvolusi. Dengan menggerakkan sebuah kernel konvolusi (filter) berukuran tertentu ke sebuah gambar, komputer mendapatkan informasi representatif baru dari hasil perkalian bagian gambar tersebut dengan filter yang digunakan [13].

E. Convolutional Layer

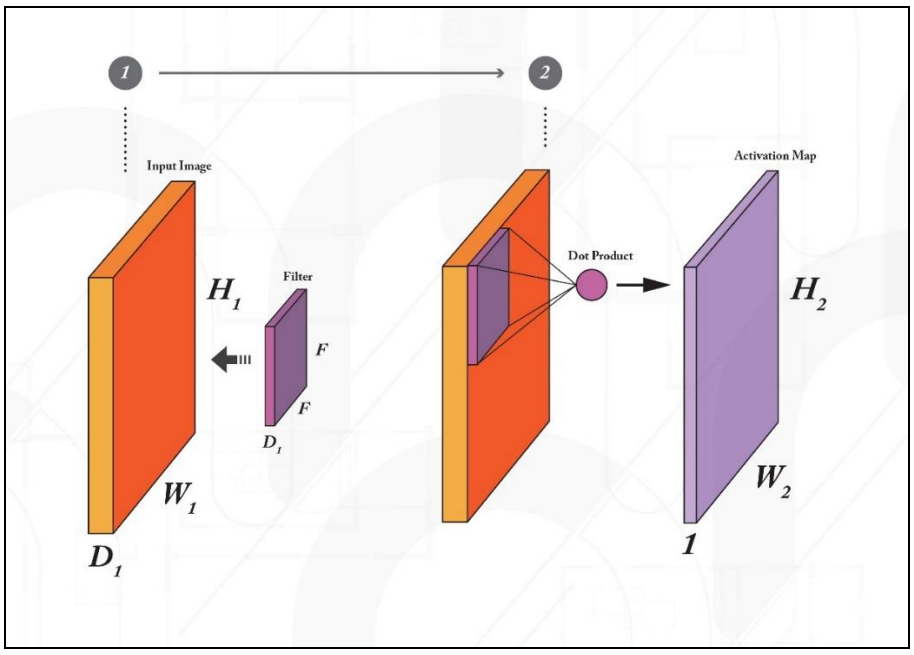
Convolutional layer digunakan untuk mengekstraksi fitur gambar dan menggunakan operasi konvolusi untuk menggantikan operasi perkalian matriks dalam jaringan saraf tradisional untuk mempelajari pemetaan antara lapisan *input* dan *output*. Pembagian parameter dalam operasi konvolusi memungkinkan jaringan untuk mempelajari hanya satu set parameter, yang sangat mengurangi jumlah parameter dan secara signifikan meningkatkan efisiensi komputasi. Operasi konvolusi didefinisikan sebagai:

$$a_{i,j} = \sum_{m=0}^{s} \sum_{n=0}^{s} w_{m,n} x_{i+mj+n} \qquad (1)$$

di mana $w_{m,n}$ adalah bobot kernel konvolusional pada *m* dan *n*; x_{i+mj+n} adalah nilai piksel gambar pada *i* dan *j*; *s* adalah tinggi dan lebar kernel konvolusional.

Proses konvolusi memanfaatkan apa yang disebut sebagai filter. Seperti layaknya gambar, filter memiliki ukuran tinggi, lebar, dan tebal tertentu. Filter ini diinisialisasi dengan nilai tertentu, dan nilai dari filter inilah yang menjadi parameter yang akan di-*update* dalam proses belajar.

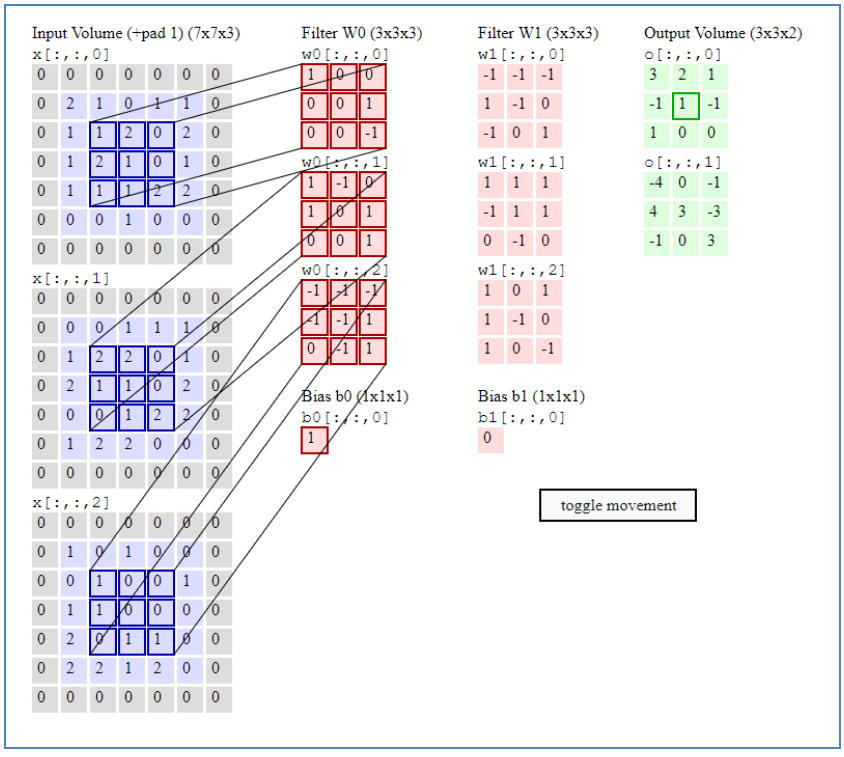
p-ISSN : 2443-2210
e-ISSN : 2443-2229



Gambar 3. Proses pada *Convolutional Layer* [14]

Gambar *input CNN* selalu berbentuk kotak. Proses untuk gambar *non-rectangular* masih belum diketahui [14]. Filter pun mengikuti karakteristik kotak tersebut. Proses ini dapat dilihat pada gambar 3.

Pada setiap posisi gambar, dihasilkan sebuah angka yang merupakan *dot product* antara bagian gambar tersebut dengan filter yang digunakan. Dengan menggeser (*convolve*) filter di setiap kemungkinan posisi filter pada gambar, dihasilkan sebuah peta aktivasi seperti yang dapat dilihat pada gambar 4.



Gambar 4. Peta Aktivasi *Convolution Layer* [14]

JuTISI
Jurnal Teknik Informatika dan Sistem Informasi

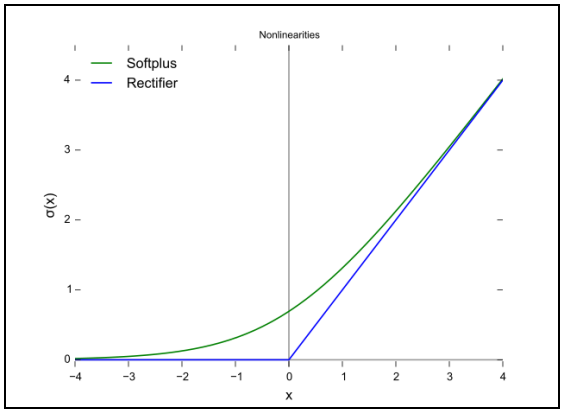
F. Fungsi Aktivasi (Activation Functions)

Rectified Linear Unit (*ReLU*) adalah fungsi aktivasi non-linear yang digunakan dalam *multilayer neural network* atau *deep neural network*. *Rectified Linear Unit* (*ReLU*) biasanya digunakan sebagai fungsi aktivasi di *CNN* [15] untuk menghindari hilangnya gradien dan meningkatkan kecepatan pelatihan. Fungsi *ReLU* didefinisikan sebagai:

$$f(x) = \max(0, x) \quad (2)$$

Menurut persamaan 2, *output ReLU* adalah nilai maksimum antara nol dan nilai *input*. *Output* sama dengan nol ketika nilai *input* negatif dan nilai *input* ketika *input* positif. Dengan demikian, persamaan dapat dirubah menjadi persamaan sebagai berikut:

$$f(x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases} \quad (3)$$



Gambar 5. Representasi *ReLU* [15]

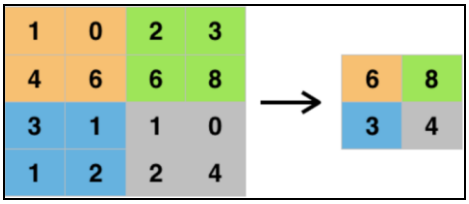
Gambar 5 menampilkan representasi grafis dari fungsi *ReLU*. Dapat diperhatikan bahwa nilai untuk setiap *input* X negatif menghasilkan *output* 0, dan hanya setelah nilai positif dimasukkan fungsi aktif.

Untuk memahami cara kerja *ReLU*, penting untuk memahami efek yang dimilikinya terhadap variabel efek interaksi. Efek interaksi adalah ketika variabel mempengaruhi prediksi tergantung pada nilai variabel terkait. Sebagai contoh, membandingkan skor IQ dari dua sekolah yang berbeda dapat memiliki efek interaksi IQ dan usia. IQ siswa di sekolah menengah lebih baik daripada IQ siswa sekolah dasar, karena usia dan IQ berinteraksi satu sama lain tanpa memandang sekolah. Hal ini dikenal sebagai efek interaksi dan *ReLU* dapat diterapkan untuk meminimalkan efek interaksi. Misalnya, jika A = 1 dan B = 2, dan keduanya memiliki bobot terkait masing-masing 2 dan 3, fungsinya adalah, *f (2A + 3B)*. Jika A meningkat, *output* akan meningkat juga. Namun, jika B adalah nilai negatif yang besar, *output* akan menjadi 0.

Manfaat menggunakan fungsi *ReLU* adalah kesederhanaannya sehingga dapat membuatnya menjadi fungsi yang relatif murah untuk dikomputasi. Karena tidak ada matematika yang rumit, model dapat dilatih dan dijalankan dalam waktu yang relatif singkat. Masalah *vanishing gradient* dapat dihindari pada *ReLU*, tidak seperti fungsi-fungsi alternatif seperti *sigmoid* atau *tanh*. Terakhir, *ReLU* jarang diaktifkan karena untuk semua *input* negatif, *output*-nya nol. *Sparsity* adalah prinsip bahwa fungsi spesifik hanya diaktifkan dalam situasi singkat. Hal ini adalah fitur yang diinginkan untuk jaringan saraf modern, seperti pada *sparse network*, kemungkinan *neuron* memproses bagian berharga dari masalah dengan tepat. Misalnya, model yang memproses gambar ikan mungkin mengandung *neuron* yang khusus untuk identitas mata ikan. *Neuron* spesifik itu tidak akan diaktifkan jika model sedang memproses gambar pesawat terbang. Penggunaan fungsi *neuron* yang ditentukan ini khusus untuk *network sparsity* [16].

G. Pooling Layer

Bagian berikutnya dari *CNN* adalah *pooling layer*. *Pooling* adalah fitur yang biasanya melekat pada arsitektur *CNN*. Gagasan utama di balik lapisan penyatuan adalah untuk "mengumpulkan" fitur dari peta yang dihasilkan dengan menggabungkan filter di atas gambar. Secara formal, fungsinya adalah untuk secara progresif mengurangi ukuran spasial dari representasi untuk mengurangi jumlah parameter dan perhitungan dalam jaringan. Bentuk *pooling* yang paling umum adalah *max pooling* atau mengambil nilai terbesar dari bagian tersebut yang dapat dilihat pada gambar 6. Namun terdapat metode *pooling* lain yang dapat digunakan seperti *average pooling* atau *L2-norm pooling*.



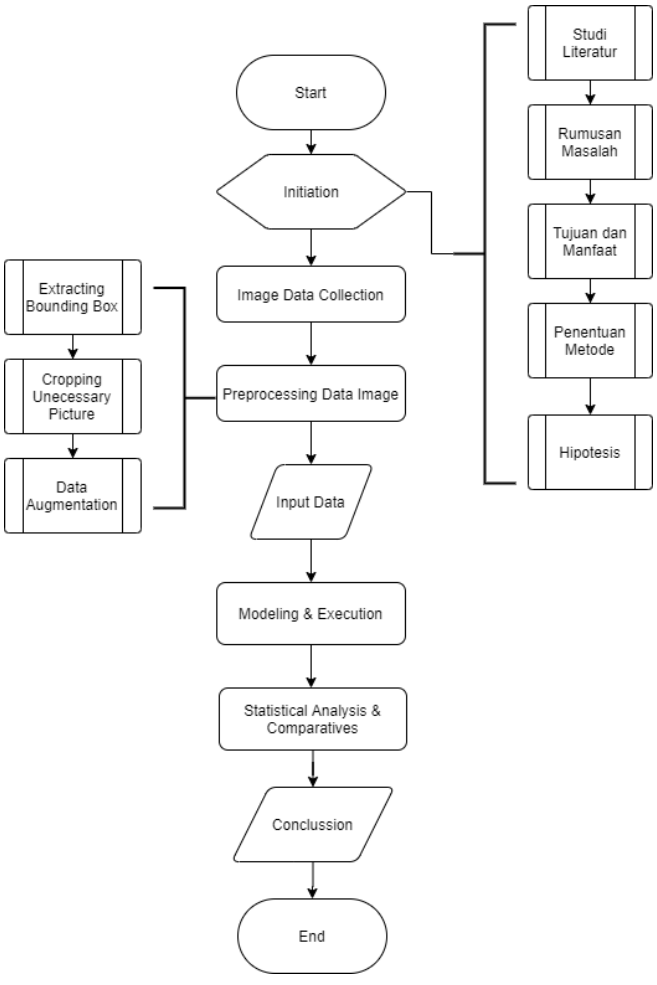
Gambar 6. Cara Kerja *Pooling Layer* [13]

Max pooling dilakukan sebagian untuk membantu *overfitting* dengan menyediakan bentuk abstrak dari representasi. Selain itu, *layer* ini dapat mengurangi biaya komputasi dengan mengurangi jumlah parameter untuk dipelajari dan menyediakan invarian terjemahan dasar ke representasi internal. *Max pooling* dilakukan dengan menerapkan filter *max* ke (biasanya) subregional yang tidak tumpang tindih dari representasi awal.

Gambar 7. *Stanford Cars Dataset [2]*

III. DESAIN PENELITIAN

Penelitian ini merupakan penelitian kuantitatif karena hasil yang diharapkan dengan performa model berdasarkan ResNet arsitektur yang ditentukan berdasarkan performa yang dihasilkan ketika menggunakan augmentasi data dan tidak menggunakan augmentasi data.



Gambar 8. Desain Penelitian

Tahapan penelitian dapat dilihat pada gambar 8 yang melingkupi:

a. *Initiation*
   Pada bagian ini akan dijelaskan tahapan awal pada penelitian ini. Tahap pertama dalam penelitian ini adalah inisiasi dengan melakukan beberapa langkah yaitu. Studi Literatur, menentukan rumusan masalah, menentukan tujuan penelitian, menentukan manfaat penelitian, pemilihan metode, dan pembuatan hipotesis.
b. Pengumpulan data dan *Pre-processing*
   Pada bagian ini akan dijelaskan tahap untuk mengambil data dan mengolah data gambar mobil yang didapatkan melalui *Stanford Cars Dataset* [2]. Diperlukan proses-proses *data pre-processing* untuk membuat data gambar mentah menjadi data yang dapat digunakan dalam metode-metode penelitian yang akan dilakukan.
c. *Model Execution*
   Pada bagian ini akan dilakukan pembuatan model *CNN* berdasarkan arsitektur ResNet sebagai ekstraksi fitur. Setelah itu dilakukan eksekusi model terhadap test data.

IV. PENGUJIAN METODE

A. Pengumpulan Data dan Pre-processing

Data yang digunakan pada penelitian ini adalah data model mobil yang terdapat pada *Stanford Cars Dataset*. Data diambil dari *website Stanford University* yang terintegrasi pada penelitiannya tentang *fine-grained categorization* [2]. Bentuk data yang diambil berupa data gambar mentah yang sulit untuk diproses dikarenakan format label dan *bounding box* yang digunakan bukanlah format umum dengan ukuran data sebanyak 16,185 baris dengan 196 kelas. Contoh dari data gambar mentah dapat dilihat pada gambar 7.

Tabel I
Data Hasil Ektraksi *Bounding Box* dan Label

File Gambar	*Bbox_1 (x,y)*	*Bbox_2 (x,y)*	*Class_id*	*Class_name*	*Bbox_h*	*Bbox_w*
	(39, 116)	(569, 375)	14	Audi TTS Coupe 2012	260	531
	(36, 116)	(868, 587)	3	Acura TL Sedan 2012	472	833
	(85, 109)	(601, 381)	91	Dodge Dakota Club Cab 2007	273	517
	(691, 393)	(1484, 1096)	134	Hyundai Sonata Hybrid Sedan 2012	704	864
	(14, 36)	(133, 99)	106	Ford F-450 Super Duty Crew Cab 2012	64	120

Dapat dilihat pada gambar 7, format data gambar dan label sulit disinkronkan karena memiliki format yang berbeda dan untuk setiap kelas dari label mobil memiliki variansi gambar yang sedikit sehingga berkemungkinan akan menghasilkan model klasifikasi yang cukup buruk. Sehingga dibutuhkan data *pre-processing* untuk mengubah bentuk data *bounding box* dan label tersebut menjadi data yang dapat digunakan dalam analisis model *CNN* lalu memperbanyak data dengan metode augmentasi data.

B. Ekstraksi Bounding Box

Langkah *pre-processing* data awal dimulai dengan melakukan ekstraksi data *bounding box* lalu melakukan integrasi antara data gambar dengan label. Hal ini dilakukan untuk mempersiapkan label untuk klasifikasi mobil yang akan dipelajari oleh model *CNN*. Data awal dari label dan *bounding box* berbentuk *file* Matlab, sehingga langkah pertama adalah melakukan ekstraksi data tabular kedalam *dataframe*. Hasil dari langkah ini dapat dilihat pada tabel I.

Pada tabel I, dapat diperhatikan hasil dari ekstraksi data *bounding box* berbentuk *vertices*/titik koordinat (*x, y*) yang diwakili oleh kolom *Bbox_1* dan *Bbox_2*. Koordinat tersebut mewakili lokasi titik piksel pada gambar yang terdapat pada kolom File Gambar, dengan menggunakan koordinat tersebut dapat dihitung lebar dan tinggi untuk setiap *bounding box*. Hal ini dilakukan untuk dapat membentuk kotak letak mobil pada setiap gambar. Hasil akhir setelah melakukan perhitungan lebar dan tinggi berdasarkan koordinat titik dapat dilihat pada tabel II.

Jika diperhatikan pada tabel II, lebar dan tinggi untuk *bounding box* setiap gambar mobil diwakili oleh kolom *Bbox_h* untuk tinggi dan *Bbox_w* untuk lebar. Proses selanjutnya adalah mencoba mensimulasikan kotak tersebut pada gambar untuk menunjukkan bahwa titik dan ukuran *bounding box* benar. Hasil dari proses ini dapat dilihat pada gambar 9.

Gambar 9. Hasil Integrasi Data *Bounding Box* dengan Gambar

Tabel II
Data Hasil Pengukuran Lebar dan Tinggi *Bounding Box*

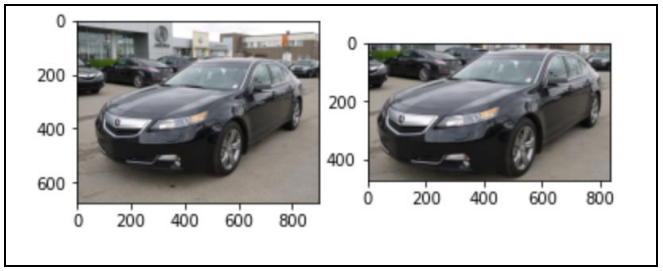
File Gambar	*Bbox_1 (x,y)*	*Bbox_2 (x,y)*	*Class_id*	*Class_name*	*Bbox_h*	*Bbox_w*
	(39, 116)	(569, 375)	14	Audi TTS Coupe 2012	260	531
	(36, 116)	(868, 587)	3	Acura TL Sedan 2012	472	833
	(85, 109)	(601, 381)	91	Dodge Dakota Club Cab 2007	273	517
	(691, 393)	(1484, 1096)	134	Hyundai Sonata Hybrid Sedan 2012	704	864
	(14, 36)	(133, 99)	106	Ford F-450 Super Duty Crew Cab 2012	64	120

Jika diperhatikan pada gambar 9, data titik dan ukuran kotak terlihat sangat cocok untuk gambar mobil. Dengan ini data tersebut dapat dipakai untuk proses selanjutnya yaitu *cropping* gambar.

C. Cropping Gambar yang Tidak Perlu

Proses selanjutnya pada *pre-processing* data gambar. Proses pelatihan pada model *CNN* gambar merupakan proses yang sangat berat sehingga untuk melakukan optimisasi waktu pelatihan perlu pemangkasan data gambar yang tidak diperlukan. Hal ini dilakukan untuk mengurangi perhitungan model terhadap fitur gambar yang tidak berguna sehingga diperlukan *bounding box* untuk setiap model klasifikasi objek pada gambar. *Bounding box* ini berguna untuk membatasi wilayah yang akan dimasukkan pada model klasifikasi.

Pada proses ini akan dilakukan *cropping* terhadap data gambar sesuai dengan *bounding box* yang telah dibuat sebelumnya. Hasil dari proses ini dapat dilihat pada gambar 10.

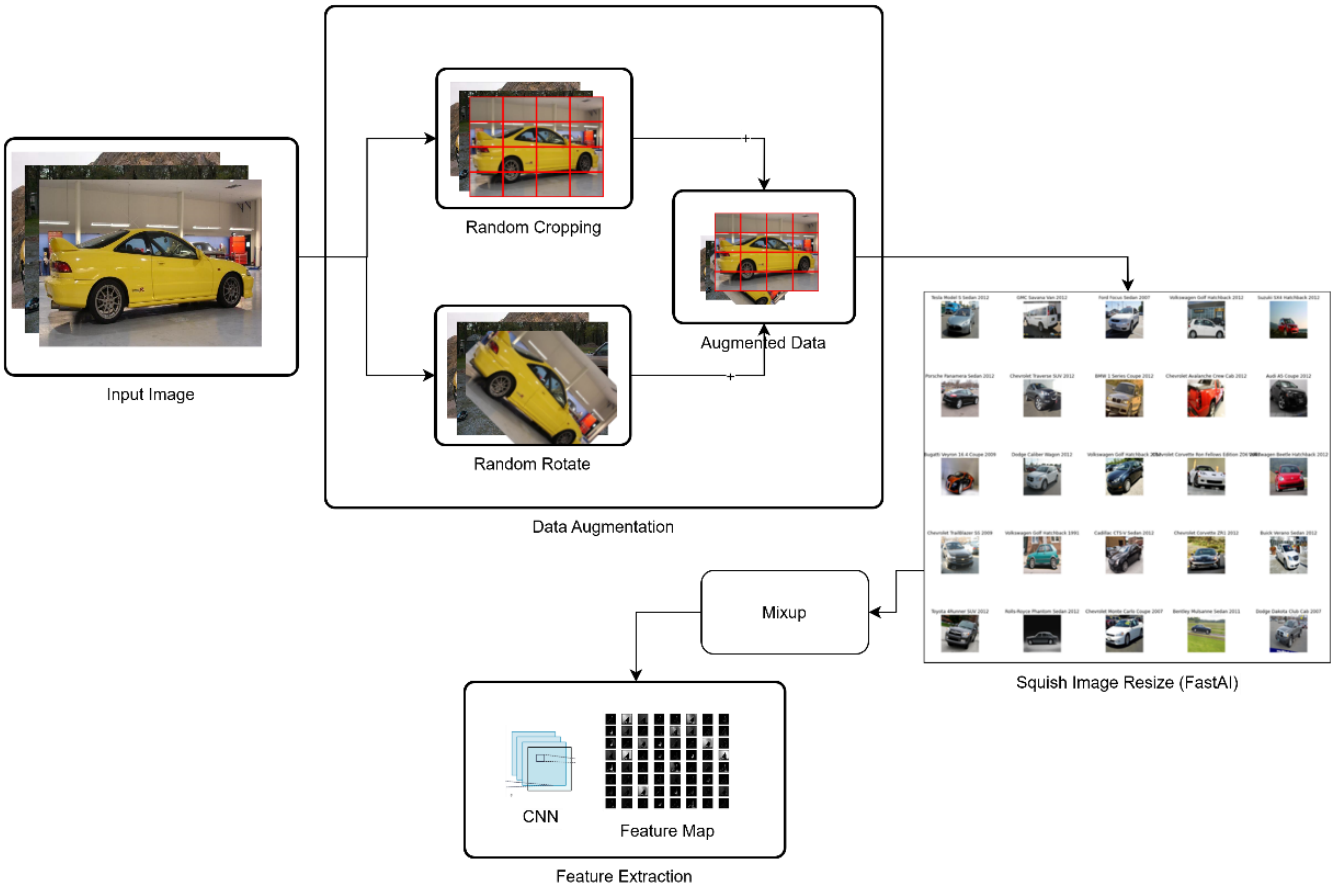


Gambar 10. Hasil *Cropping* Data Gambar

Jika diperhatikan pada gambar 10, hasil *cropping* gambar menghasilkan gambar yang hanya menunjukkan mobil saja. Terlihat bahwa luas wilayah gambar terpangkas sekitar 25% dari luas gambar sebelumnya.

D. Augmentasi Data

Proses selanjutnya dari *pre-processing* gambar yaitu augmentasi data. Salah satu kekurangan dari *Stanford Cars Dataset* adalah jumlah gambar yang mewakili setiap kelas label mobil berjumlah tidak terlalu banyak untuk setiap kelasnya. Hal ini dapat menyebabkan kurangnya fitur yang didapatkan ketika melakukan pelatihan berdasarkan data tersebut, sehingga diperlukan metode untuk memperbanyak gambar tanpa harus melakukan sampel tambahan yaitu augmentasi data.

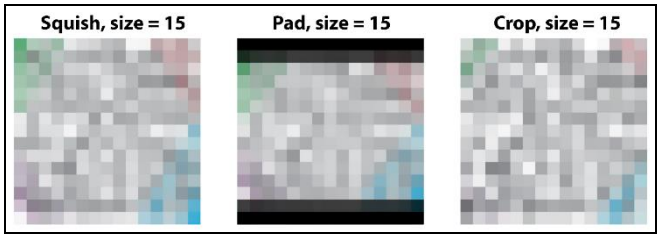


Gambar 11. Proses Augmentasi Data

Pada penelitian ini dilakukan dua metode untuk augmentasi data yaitu: *random crop* dan *random rotation*. Proses ini dapat dilihat pada gambar 11. Dengan memanfaatkan augmentasi data *random crop* dan *random rotation* didapatkan dataset baru yang dapat digunakan pada saat pelatihan.

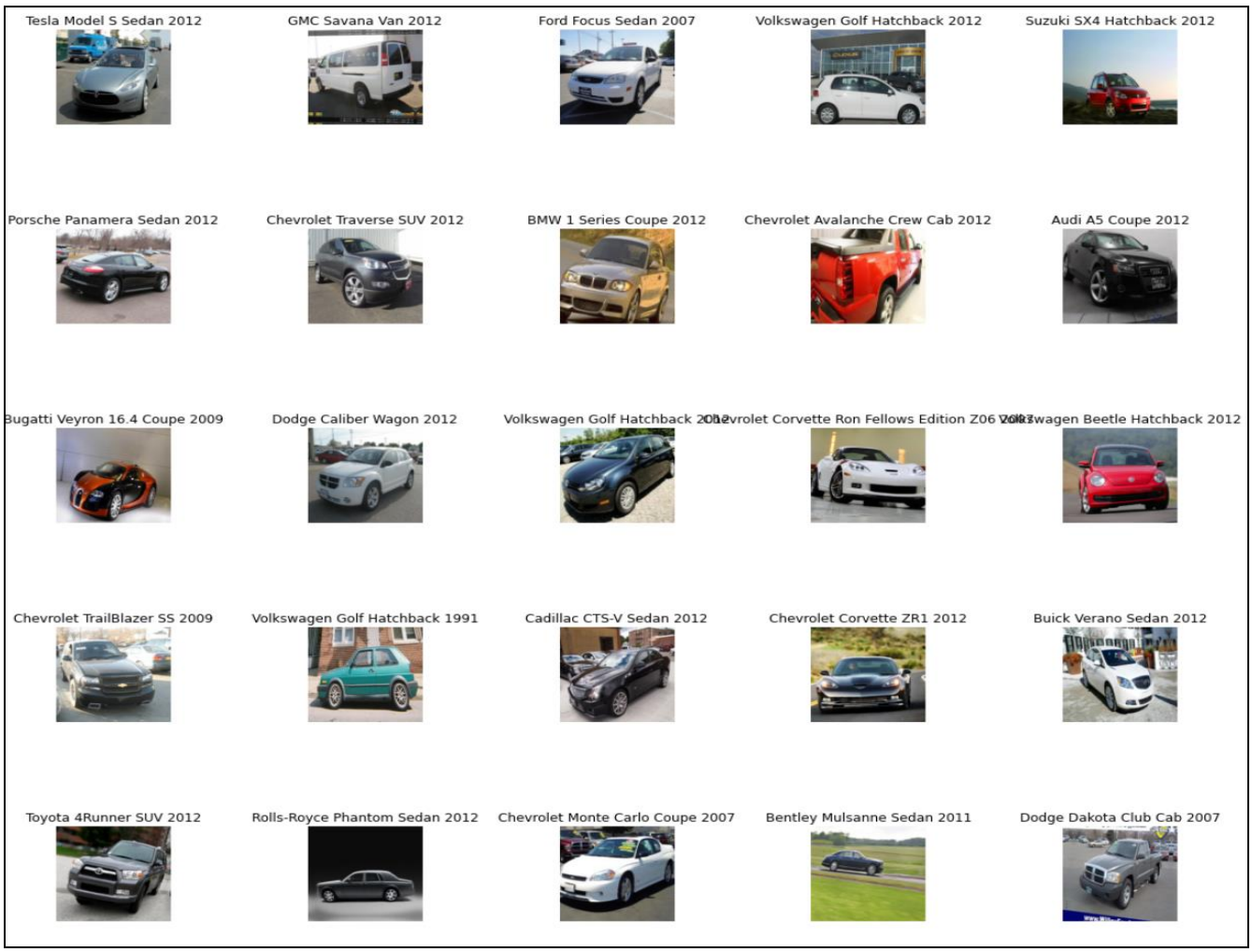
Random crop menghasilkan gambar yang dipotong pada lokasi piksel yang dipilih secara *random* sehingga memungkinkan menghasilkan gambar yang tidak utuh, tetapi pada model *CNN* hal ini bukan masalah karena fitur gambar diambil bukan secara keseluruhan melainkan melalui *sliding windows* berdasarkan fitur yang dipilih.

Random rotation menghasilkan gambar yang dirotasi berdasarkan derajat yang dipilih secara *random*. Dengan ini pada saat pelatihan akan dihasilkan *feature map* yang lebih banyak sehingga fitur yang didapatkan dari setiap kelas bertambah. Setelah data augmentasi selesai digenerasi, dilakukan *resize* dengan menggunakan salah satu fasilitas fast.ai. Metode *resize* yang digunakan adalah "*squish*", dengan menggunakan metode ini ukuran dimensi piksel gambar dibuat sesuai dengan bingkai yang telah ditentukan tanpa melihat *aspect ratio* gambar sebelum dilakukan *resize*. Contoh perbandingan hasil *resize* antara metode yang ada pada *fast.ai* yaitu *padding*, *squish*, dan *crop* dapat dilihat pada gambar 12.



Gambar 12. Perbanding *Resize Methods Fast.ai*

Pada gambar 12, terlihat bahwa setiap gambar di-*resize* menjadi ukuran 15x15 piksel. Gambar pertama di-*resize* menggunakan metode *squish*, gambar kedua dengan metode *padding*, dan gambar ketiga dengan metode *crop*. Pada model *CNN*, input berbentuk *4D array* yaitu *batch_size*, lebar, tinggi, dan *depth* sehingga proses pembangunan arsitektur *CNN* memiliki ketergantungan terhadap *input shape*. Tanpa memiliki input shape yang tetap, arsitektur *CNN* tidak dapat dibentuk dengan baik sehingga dimensi gambar perlu disamakan [1]. Hasil akhir dari augmentasi data ini dapat dilihat pada gambar 13.



Gambar 13. Hasil dari Augmentasi Data

Setelah data baru selesai dibuat, dilakukan metode *mixup* terhadap dataset gambar sebelum dimasukkan pada fase pelatihan model. Metode ini dilakukan menggunakan salah satu modul pada fast.ai yang berisi implementasi teknik augmentasi data yang disebut *mixup*. Metode ini sangat efisien dalam regularisasi terhadap model dalam komputer visi [17].

Pada jurnal artikel *mixup*, Daojun Liang [17] mengusulkan metode pelatihan model tentang campuran gambar set pelatihan. Sebagai contoh, misalkan pada saat fase pelatihan model menggunakan dataset CIFAR-10. Daripada memasukkan model gambar mentah, diambil dua gambar yang berbeda lalu dilakukan kombinasi linear dari kedua gambar tersebut, dengan ini didapatkan *tensor* gambar baru dengan rumus:

$$new_{image} = \mathrm{t} * image1 + (1 - \mathrm{t}) * image2 \qquad (4)$$

Variabel *t* adalah nilai antara 0 dan 1 yang mewakili skala perbandingan antara ukuran gambar 1 dengan gambar 2. Target label yang ditetapkan untuk gambar baru tersebut adalah kombinasi yang sama dari target asli, dengan rumus:

$$new_{target} = \mathrm{t} * target1 + (1 - \mathrm{t}) * target2 \qquad (5)$$

Variabel *t* adalah nilai antara 0 dan 1 yang mewakili bobot pada label gambar, misal terdapat dua mobil pada gambar dan gambar mobil kedua memiliki bobot *t = 0.3* maka label klasifikasi adalah 70% label gambar 1 dan 30% label gambar 2. Hasil akhir dari proses ini berupa gabungan linear antara 2 gambar sehingga model dapat belajar lebih efisien. Contoh gabungan input gambar dengan *mixup* dapat dilihat pada gambar 14.

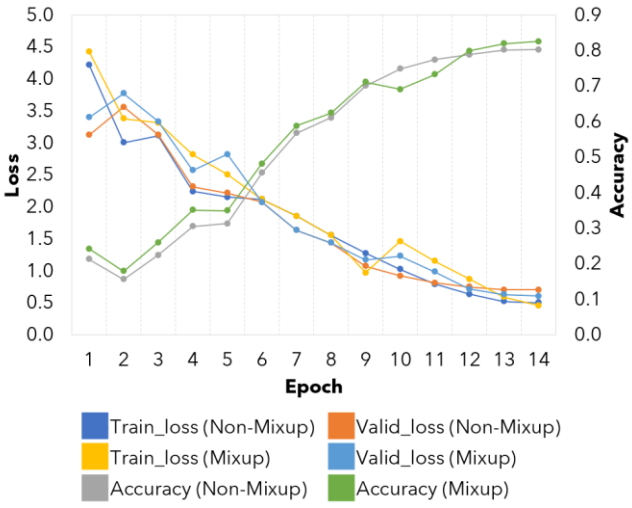
Seperti yang ditunjukkan gambar 14, sedikit sulit bagi mata manusia untuk memahami gambar yang diperoleh dengan cara ini (meskipun memang terlihat bentuk kedua mobil yang berbeda). Satu catatan penting adalah bahwa ketika pelatihan dengan *mixup*, *final loss* (pelatihan atau validasi) akan lebih tinggi daripada saat pelatihan tanpa *mixup*, bahkan ketika akurasi jauh lebih baik.

Gambar 14. Contoh Gambar dengan *Mixup*

Model yang dilatih seperti ini akan membuat prediksi memiliki tingkat kepercayaan yang lebih rendah [17].

E. Eksekusi Model

Percobaan dilakukan terhadap pelatihan awal menggunakan parameter dasar terhadap model *CNN* dengan arsitektur ResNet 152 dengan metode pelatihan *mixup* dan tanpa *mixup*. Hal ini dilakukan untuk melihat perbandingan performa antara kedua metode sehingga didapatkan metode pelatihan yang memiliki performa terbaik dalam metrik akurasi. Percobaan dilakukan sebanyak 14 *epoch* untuk arsitektur yang dilatih dengan *mixup* dan tanpa *mixup*. Hasil dari percobaan ini dapat dilihat pada gambar 15 dan tabel III.



Gambar 15. Hasil Percobaan Pelatihan dengan dan tanpa Mixup

Tabel III
Hasil Akhir Perbandingan Pelatihan

Methods	*Jumlah Epoch*	*Train loss*	*Valid loss*	Akurasi
Non-Mixup	14	0.513145	0.703171	0.791234
Mixup	14	0.556042	0.712334	0.824154

Jika diperhatikan pada tabel III, terdapat *train loss* dan *valid loss* yang didapatkan dari *loss function*. *Loss function* digunakan untuk mengoptimalkan algoritma pembelajaran mesin. *Loss* dihitung berdasarkan *train* dan *validation*, interpretasinya didasarkan pada seberapa baik kinerja model dalam dua set ini. *Train loss* dan *valid loss* adalah jumlah kesalahan yang dibuat untuk setiap sampel dalam *test set* atau *validation set*. Pada tabel terlihat bahwa metode pelatihan dengan menggunakan *mixup* memiliki *train loss* dan *valid loss* yang lebih tinggi dibandingkan dengan tanpa *mixup*. Sesuai dengan yang disampaikan oleh Daojun Liang [17], hal ini dikarenakan model yang dilatih dengan *mixup* memiliki tingkat kepercayaan yang lebih rendah. Pada penelitian ini difokuskan pada performa model dengan metrik akurasi tertinggi sehingga pelatihan akan dilakukan dengan *mixup*.

F. Evaluasi Model

Evaluasi model akan dilakukan menggunakan *Test Time Augmentation* (*TTA*). *TTA* mengacu pada melakukan augmentasi data seperti *neural transfer style*, *flipping* gambars, *cropping* terhadap *test set* yang akan diprediksi. Setelah model memprediksi label kelas dari *test* data yang sudah melalui proses augmentasi, skor dikumpulkan untuk membentuk prediksi akhir dari gambar asli [18]. Hasil akhir dibandingkan berdasarkan metrik akurasi terhadap model ResNet, Hasil perbandingan dapat dilihat pada tabel IV.

Tabel IV
Hasil Perbandingan Performa Model ResNet

Methods	*Top losses*	Akurasi
Baseline ResNet 152	14.45	0.764815
Baseline Resnet 152 + Random Crop & Rotate	10.81	0.801424
Baseline ResNet 152 + Random Crop & Rotate + *Mixup*	11.20	0.826714

Jika diperhatikan pada tabel IV, kolom *top losses* menunjukkan gambar yang diprediksi salah dengan kemungkinan terkecil. Terlihat bahwa *top losses* pada model ini termasuk tinggi dikarenakan pada dataset terdapat gambar yang memiliki fitur sangat mirip, misal terdapat mobil dengan model yang sama tetapi tahun yang berbeda.

p-ISSN : 2443-2210
e-ISSN : 2443-2229

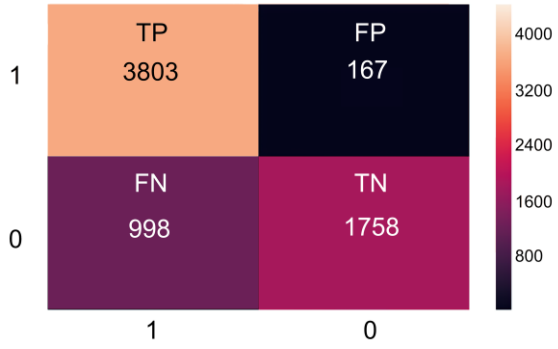
Jika diperhatikan dengan memanfaatkan augmentasi performa model meningkat, terutama ketika menggunakan *mixup*. Tetapi seperti yang disebutkan pada subbab D *loss* akan bertambah pada model yang dilatih menggunakan *mixup*.

Salah satu kasus *overfitting CNN* yang paling serius muncul ketika mengklasifikasikan gambar berdasarkan serangkaian fitur terbatas dan mengabaikan yang lain. Misalnya, jika *CNN* mengklasifikasikan gambar mobil berdasarkan fitur bagian depan mobil, model gagal untuk mengklasifikasikan gambar yang menggambarkan mobil dengan posisi menyamping. Karena augmentasi data mengumpulkan dan memotong gambar secara acak, masing-masing gambar menyediakan wilayah pemotongan yang berbeda di setiap langkah pelatihan. Hal ini diharapkan untuk mendukung *CNN* dalam menggunakan beragam fitur yang lebih luas dari gambar yang sama dan untuk mencegah *CNN* dari *overfitting* ke fitur-fitur dari wilayah tertentu.

Untuk memverifikasi hipotesis ini, divisualisasi daerah di mana *CNN* memusatkan banyak perhatian menggunakan *Class Activation Mapping (CAM)* [19]. *CAM* mengharapkan *CNN* memiliki *global average pooling layer* untuk mendapatkan rata-rata spasial dari *feature map* sebelum lapisan keluaran terakhir. *CAM* menghitung *regional importance* dengan memproyeksikan kembali output ke *feature map*. Hasil dari proses ini dapat dilihat pada gambar 16.

Gambar 16. *Class Activation Mapping (CAM)*

Jika diperhatikan pada gambar 15, gambar pada baris pertama adalah input gambar, gambar pada baris kedua adalah hasil *CAM* pada *baseline* model, dan gambar pada baris ketiga adalah *CAM* pada model dengan augmentasi data. Model dengan augmentasi mendapatkan fitur lebih dibandingkan *baseline* model. Untuk melihat dampak augmentasi terhadap performa klasifikasi model, dilakukan analisis menggunakan matriks *confusion*. Hasil dari analisis ini dapat dilihat pada gambar 17.



Gambar 17. Matriks Confusion Klasifikasi

Gambar 17 menunjukan bahwa dengan menggunakan augmentasi data pada klasifikasi mobil, akan menghasilkan model klasifikasi yang lebih condong ke *True Positive (TP)*.

V. KESIMPULAN

Dari hasil percobaan yang dilakukan pada tabel IV didapatkan kesimpulan bahwa dengan memanfaatkan augmentasi data *random crop*, *rotation*, dan *mixup*, model *CNN* dengan arsitektur ResNet dapat bekerja dengan performa akurasi yang lebih baik. Augmentasi dengan *Mixup* meningkatkan akurasi tetapi meningkatkan *loss*. Dengan memanfaatkan data augmentasi ini fitur yang didapatkan oleh model dapat meningkat sehingga meningkatkan juga informasi untuk klasifikasi.

Sebagai kelanjutan dari penelitian ini, akan dilakukan studi banding arsitektur ResNet dengan algoritma CNN yang lain untuk mendapatkan metrik akurasi, ukuran model, dan kecepatan terbaik dalam pengenalan model mobil, menggunakan dataset yang sama dengan penelitian yang sudah dilakukan.

DAFTAR PUSTAKA

[1] R. Venkatesan and B. Li, *Convolutional neural networks in visual computing: a concise guide* (Data-enabled engineering). Boca Raton: CRC Press, Taylor & Francis Group, CRC Press is an imprint of the Taylor & Francis Group, an informa business, 2018, p. 168.

[2] J. Krause, M. Stark, J. Deng, and L. Fei-Fei, "3D Object Representations for Fine-Grained Categorization," in *2013 IEEE International Conference on Computer Vision Workshops (ICCVW)*, 2013, Sydney, Australia: IEEE, pp. 554-561, doi: 10.1109/ICCVW.2013.77.

[3] Y. Wei, Q. Tian, J. Guo, W. Huang, and J. Cao, "Multi-vehicle detection algorithm through combining Harr and HOG features," (in en), *Mathematics and Computers in Simulation*, vol. 155, pp. 130-145, 2019, doi: 10.1016/j.matcom.2017.12.011.

[4] M. Liu, C. Yu, H. Ling, and J. Lei, "Hierarchical Joint CNN-Based Models for Fine-Grained Cars Recognition," in *Cloud Computing and Security*, vol. 10040, X. Sun, A. Liu, H.-C. Chao, and E. Bertino Eds. Cham: Springer International Publishing, 2016, pp. 337-347.

[5] Q. Hu, H. Wang, T. Li, and C. Shen, "Deep CNNs With Spatially Weighted Pooling for Fine-Grained Car Recognition," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 11, pp. 3147-3156, 2017, doi: 10.1109/TITS.2017.2679114.

[6] L. Zhang and H. Schaeffer, "Forward Stability of ResNet and Its Variants," (in en), *Journal of Mathematical Imaging and Vision*, vol. 62, no. 3, pp. 328-351, 2020, doi: 10.1007/s10851-019-00922-y.

[7] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, Salt Lake City, UT: IEEE, pp. 4510-4520, doi: 10.1109/CVPR.2018.00474.

[8] H. Lee, I. Ullah, W. Wan, Y. Gao, and Z. Fang, "Real-Time Vehicle Make and Model Recognition with the Residual SqueezeNet Architecture," (in en), *Sensors*, vol. 19, no. 5, p. 982, 2019, doi: 10.3390/s19050982.

[9] R. Takahashi, T. Matsubara, and K. Uehara, "Data Augmentation using Random Image Cropping and Patching for Deep CNNs," *IEEE Transactions on Circuits and Systems for Video Technology*, pp. 1-1, 2019 2019, doi: 10.1109/TCSVT.2019.2935128.

[10] A. Zheng and A. Casari, *Feature engineering for machine learning: principles and techniques for data scientists*, First edition ed. Beijing : Boston: O'Reilly, 2018, p. 200.

[11] J. Brownlee. "Discover Feature Engineering, How to Engineer Features and How to Get Good at It." https://machinelearningmastery.com/discover-feature-engineering-how-to-engineer-features-and-how-to-get-good-at-it/ (accessed Jun. 9, 2020).

[12] D. Seita, "1000x Faster Data Augmentation," in *The Berkeley Artificial Intelligence Research Blog*, ed, 2019.

[13] R. Dharmadi, "Mengenal Convolutional Neural Network," in *Medium*, ed, 2018.

[14] A. Karpathy, "Cs231n convolutional neural networks for visual recognition," *Neural networks*, vol. 1, p. 1, 2016.

[15] S. Qiu, X. Xu, and B. Cai, "FReLU: Flexible Rectified Linear Units for Improving Convolutional Neural Networks," in *2018 24th International Conference on Pattern Recognition (ICPR)*, 2018, Beijing: IEEE, pp. 1223-1228, doi: 10.1109/ICPR.2018.8546022.

[16] D. AI. "ReLu." https://deepai.org/machine-learning-glossary-and-terms/relu (accessed Jun. 06, 2020).

[17] D. Liang, F. Yang, T. Zhang, and P. Yang, "Understanding Mixup Training Methods," *IEEE Access*, vol. 6, pp. 58774-58783, 2018, doi: 10.1109/ACCESS.2018.2872698.

[18] J. Nalepa, M. Myller, and M. Kawulok, "Training- and Test-Time Data Augmentation for Hyperspectral Image Segmentation," *IEEE Geoscience and Remote Sensing Letters*, vol. 17, no. 2, pp. 292-296, 2020, doi: 10.1109/LGRS.2019.2921011.

[19] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning Deep Features for Discriminative Localization," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, Las Vegas, NV, USA: IEEE, pp. 2921-2929, doi: 10.1109/CVPR.2016.319.