JII: Jurnal Inovasi Informatika Universitas Pradita
e-Issn: 2686-1615, p-Issn: 2527-4007

Plat Nomor Kendaraan dengan *Convolution Neural Network*

Djarot Hindarto[1], **Handri Santoso**[2]

[1,2]Magister Teknologi Informasi; Universitas Pradita; Alamat Scientia Business Park, Jl. Gading Serpong Boulevard No.1, Curug Sangereng, Kelapa Dua, Tangerang, Banten; Telp (021) 55689999; email: djarot.hindarto@student.pradita.ac.id, handri.santoso@pradita.ac.id

Abstrak: Perkembangan teknologi *Deep Learning* sangat bagus dalam melakukan deteksi *Object*. Salah satunya adalah deteksi pada plat nomor kendaraan. Metode ini dapat diterapkan pada *Computer Vision* untuk memproses *image* dengan menggunakan metode *DensetNet*121, *NasNetLarge*, *VGG16* dan *VGG19*. Perbedaan yang paling mendasar antara *Machine Learning* dengan *Deep Learning* adalah memasukan *Hidden Layer* dan yang membedakan proses *Deep Learning* menggunakan *neuron* sebagai proses dari input, proses sampai dengan output. *Feature extraction* dilakukan langsung dengan proses *Deep Learning*. Secara waktu, training model dengan *Deep Learning* sangat lama, jika dibandingkan dengan *Machine Learning*. Dataset berasal dari Kaggle, kemudian dilakukan *training* dengan empat model *Deep Learning*, sehingga menhasilkan model. Terdapat perbedaan dalam melakukan proses *training*. Sebelum melakukan proses *Training*, dilakukan proses *pre-paration* dari Dataset *Image*. Dataset dibagi menjadi dua bagian, Dataset *Training* dan Dataset *Testing*. Setelah model *Training* selesai dilanjutkan dengan proses *Testing* dan mengukur performansi akurasi model. Akurasi dari ke empat model hasil *training Deep Learning* juga disajikan.

Kata kunci: *Deep Learning*, *DensetNet121*, *NasNetLarge*, *VGG16*, *VGG19*

Abstract: *The development of Deep Learning technology is very good at detecting Objects. One of them is detection on the vehicle number plate. This method can be applied to Computer Vision to process images using DensetNet121, NasNetLarge, VGG16 and VGG19 methods. The most basic difference between Machine Learning and Deep Learning is the inclusion of a Hidden Layer and what distinguishes the Deep Learning process using neurons as a process from input, process to output. Feature extraction is done directly with the Deep Learning process. In terms of time, training models with Deep Learning are very long, when compared to Machine Learning. The dataset comes from Kaggle, then training is carried out with four Deep Learning models, resulting in a model. There are differences in conducting the training process. Before carrying out the Training process, a pre-paration process from the Image Dataset is carried out. The dataset is divided into two parts, the Training Dataset and the Testing Dataset. After the training model is completed, it is continued with the Testing process and measuring the performance of the model's accuracy. The accuracy of the four models resulting from Deep Learning training is also presented.*

Keywords: Deep Learning, DensetNet121, NasNetLarge, VGG16, VGG19

1. Pendahuluan

Dalam memonitor kendaraan saat ini sudah tidak perlu dimonitor secara manual, seperti memonitor kendaraan yang lewat di jalan raya atau memonitor lewat CCTV. Tentunya secara manual memerlukan ketelitian petugas. Proses ini yang tidak akurat, jika dibandingkan dengan menggunakan teknologi *Internet of Things* dan metode *Deep Learning* yang menjadi *trend* saat ini dan tahun-tahun ke depan. Metode *Deep Learning* merupakan bagian *Machine Learning* berbasis *Neural Network*. Selain itu menggunakan banyak sekali *hidden layers* yang mampu mempelajari fitur-fitur data secara otomatis. Berbeda dengan *Machine Learning* dan *Feature Extraction* sudah ada di dalam metode *Deep Learning*. *Deep learning*, banyak diaplikasikan di berbagai bidang, seperti *Computer Vision, Natural Language Processing*, Audio, dan lain-lain. Bidang *Computer Vision* saat ini juga mengalami perkembangan yang sangat pesat. Dimana *resolution* gambar yang sangat kecilpun mampu dilakukan *computation* yang cukup cepat.

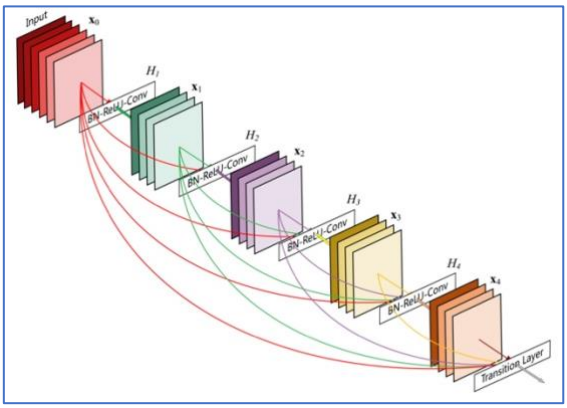
Teknologi *Machine Learning* banyak mendukung berbagai aspek modern yang ada di masyarakat, mulai dari penelusuran web hingga penyaringan konten di media sosial. Rekomendasi di situs web komersial, disamping itu dengan munculnya banyak *product Smartphone* dengan berbagai fitur seperti kamera. Sistem *Machine Learning* digunakan untuk mengidentifikasi objek di dalam gambar, menyalin ucapan ke dalam teks, mencocokkan item berita, mencocokan postingan dari berbagai media. Melakukan

pemilihan hasil penelusuran yang relevan. Semua ini dikarenakan banyak pemanfaatan teknologi *Deep Learning* secara meluas di berbagai aplikasi *image processesing*.

Kajian ini bertujuan untuk membandingkan model *DensetNet*121, *NasNetLarge*, *VGG16* dan *VGG19*. *Dataset* didownload dari Kaggle.com. *Deep Learning* adalah bagian dari *Machine Learning* dengan menggunakan *Deep Neural Network* untuk solusi *Machine Learning* yang lebih dalam. Saat ini perkembangan *Deep Learning* sangat pesat untuk berbagai bidang seperti *Computer Vision*, Audio, *Text* dan lain-lain. Hampir semua bidang menggunakan kemampuan *Deep Learning* untuk menyelesaiakan masalah yang rumit.

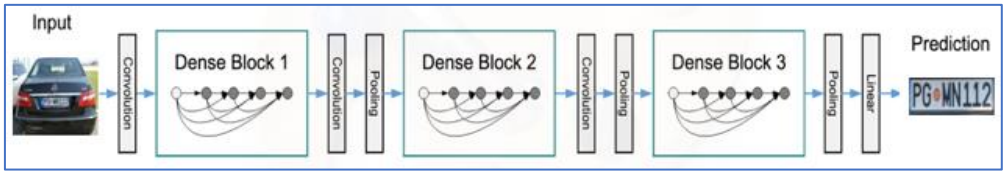
Convolutional Neural Network (CNN) dijadikan pendekatam *Machine Learning* yang dominan untuk deteksi *object vision*. Telah diperkenalkan 20 tahun lalu (Y. LeCun, B. Boser, J. S. Denker, D. Henderson, 1989), peningkatan komputer dan *Structure Network* memungkinkan *Training Deep Learning*.

Teknik *Machine Learning* konvensional masih terbatas dalam kemampuan untuk memproses data asli *image*. Selama beberapa dekade, membangun sistem pengenalan pola atau pembelajaran mesin membutuhkan rekayasa yang cermat dan keahlian domain yang cukup besar untuk merancang ekstrak fitur, yang mengubah data *image* asli (seperti nilai piksel suatu gambar) menjadi vektor fitur yang sesuai dengan subsistem pembelajaran. Dimana *Classifier* dapat mendeteksi pola dalam inputan.



Gambar 1 Lima Blok pertumbuhan k = 4, setiap lapisan sebagai input semua fitur (Huang et al., 2017).

Pada gambar 1, x0 yang dilewatkan melalui *Convolution Network*. Jaringan terdiri dari lapisan L, masing-masing yang mengimplementasikan transformasi non-linier Hl(·), di mana l adalah indeks lapisan. Hl (·) dapat menjadi fungsi komposit operasi seperti *Batch Normalization* (BN) (Ioffe & Szegedy, 2015), unit linier yang diperbaiki (ReLU) (Xavier Glorot, Antoine Bordes, 2011), *Pooling* (Y. LeCun, L. Bottou, Y. Bengio, 1998), atau *Convolution* (Konv). Output dari `lapisan ke-th disebut sebagai x`.



Gambar 2. Sebuah DenseNet dengan 3 blok Dense (Huang et al., 2017).

Dense connectivity

Untuk meningkatkan informasi aliran antar lapisan diusulkan konektivitas yang berbeda pola dengan memperkenalkan koneksi langsung dari lapisan manapun ke semua lapisan berikutnya. Gambar 1 memperlihatkan tata letak *DenseNet* yang dihasilkan secara skematis. Akibatnya, Lapisan l th menerima fitur dari semua lapisan sebelumnya, x0,..., x l − 1, sebagai masukan:

$$x_l = H_l([x_0, x_1, \ldots, x_{l-1}])$$

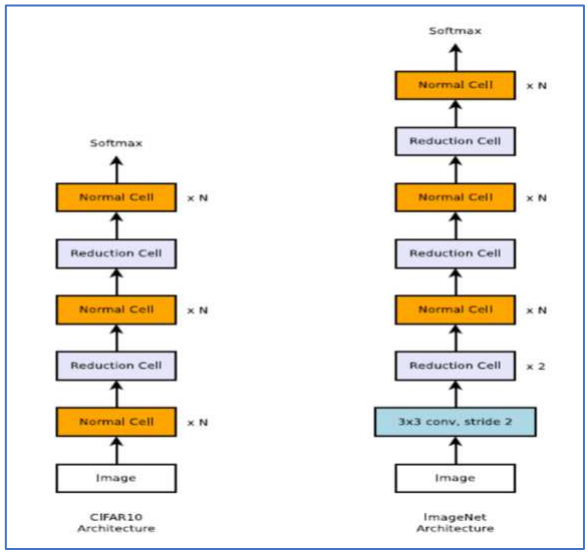
di mana [x0,x1,...,xl − 1] mengacu pada rangkaian fitur diproduksi di lapisan 0,...,l −1. Karena itu konektivitas *Denset* disebut dengan arsitektur *Convolution Network Den*se (DenseNet). Dengan menggabungkan beberapa input dari Hl (·) dalam persamaan diatas menjadi satu sensor.

Pooling layers

Operasi penggabungan yang digunakan dalam persamaan diatas tidak layak ketika ukuran fitur berubah. Namun, bagian penting dari *Convolution Network* adalah lapisan *down-sampling* yang mengubah ukuran fitur. Untuk memfasilitasi *down-sampling* dalam arsitektur pembagian jaringan menjadi beberapa blok *dense* yang terhubung seperti gambar 2 disebut dengan lapisan antar blok sebagai transisi lapisan. Kemudian melakukan konvolusi dan penyatuan. Transisi lapisan yang digunakan terdiri dari *batch* normal lapisan dan lapisan konvolusi 1 × 1 diikuti oleh 2 × 2 lapisan *Pooling Average*.

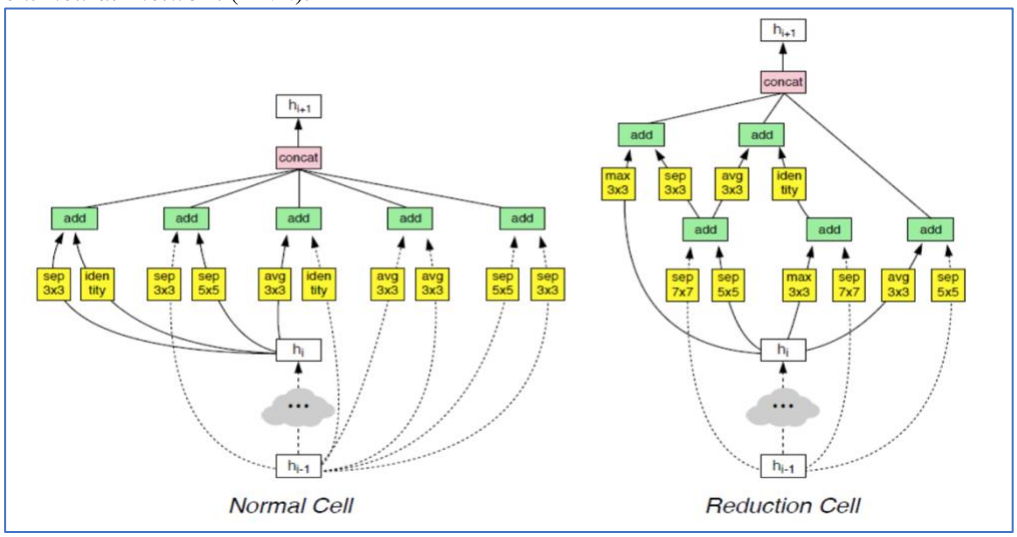
NasNetLarge

Sebelum membahas tentang *NasNetLarge*, terlebih dulu mengenal istilah *Neural Architecture Search* (NAS) untuk *Cells*.



Gambar 3 Arsitektur untuk *CIFAR*-10 dan *ImageNet*.

NASNet merupakan model berdasarkan cara mencari blok arsitektur terbaik pada dataset yang kecil, setelah itu melakukan penyalinan arsitektur terbaik yang ditemukan. Kemudian tahap berikutnya digunakan pada dataset yang besar yakni *ImageNet*. Blok-blok tersebut disusun sedemikian rupa membentuk variasi arsitektur *NASNet*, variasi yang kecil adalah *NASNetMobile* atau *NASNet-A* (4 @ 1056). Arsitektur *NASNet* berisikan dua blok utama yaitu *normal cell* dan *reduction cell*. *Layer* dari kedua *cell* tersebut menggunakan *Recurent Neural Network* (RNN).



Gambar 4 *Normal Cell* dan *Reduction Cell* pada *NASNet-A* Sumber(Sik-Ho Tsang)

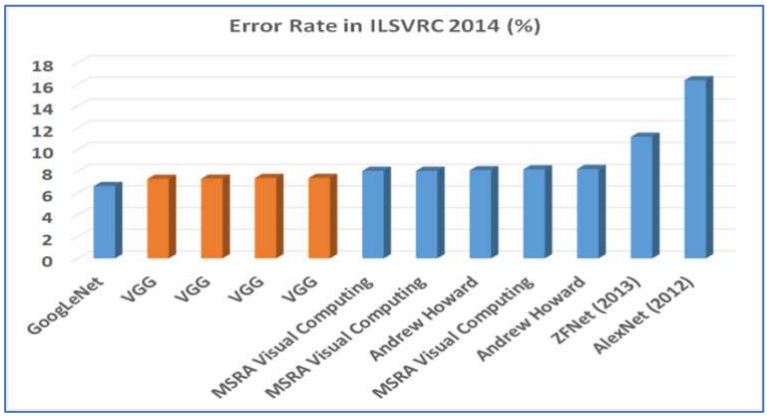
NASNetLarge

Arsitektur *NASNetLarge* yang ada di *Library* Keras.io adalah *NASNet-A* (6 @ 4032) atau *NASNetLarge*. Sama seperti *NASNetMobile*, *NASNetLarge* terdiri dari sekumpulan *normal cell* dan reduction cell, salah satu perbedaannya adalah dalam penggunaan *normal cell*, pada *NASNetLarge* terdapat 6 *normal cell* yang disusun secara sequensial. *NASNetLarge* berhasil mencapai performa *state-of-the-art* untuk klasifikasi gambar *ImageNet* dengan akurasi top-1 82.7% (Zoph B, Vasudevan, Shlens J & Brain, n.d.). Berdasarkan pustaka keras, model ini memiliki jumlah *layer* sebanyak 1039 yang terbagi ke dalam 18 *normal cell* dan 3 *reduction cell*.

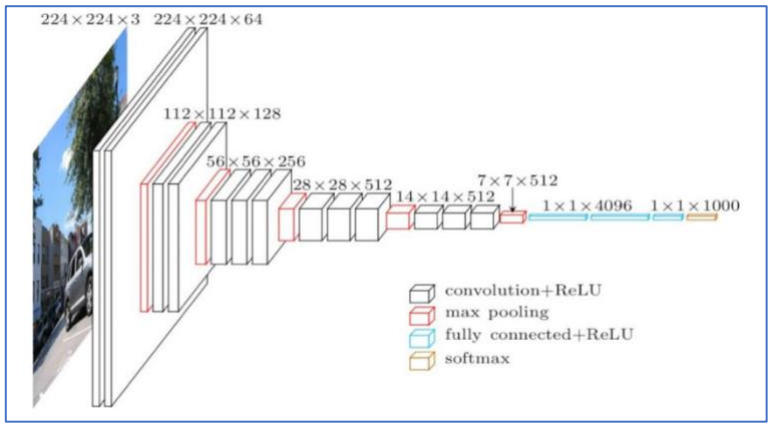
Penggunaan metode pencarian untuk menemukan arsitektur v*olutional* pada *dataset interest*. Metode pencarian yang digunakan dalam pekerjaan ini adalah *Neural Architecture Search* (NAS), kerangka kerja yang diusulkan (Barret Zoph, n.d.). Di dalam NAS, sampel *Recurrent Neural Network* (RNN) terdapat anak jaringan dengan arsitektur yang berbeda. Jaringan anak dilakukan *Training* dengan konvergensi untuk mendapatkan beberapa akurasi pada set validasi yang tertunda. Arsitektur yang dihasilkan mendekati atau melampaui *state-of-the-Art* di *CIFAR*-10 dan *ImageNet* pada *Dataset* dengan permintaan komputasi yang lebih sedikit daripada arsitektur yang dirancang (Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, 2016), (Ioffe & Szegedy, 2015), (Xingcheng Zhang, Zhizhong Li, Chen Change Loy, 2017). Hasilnya sangat penting karena banyak *state-of-the-Art Computer Vision* seperti deteksi objek (Jonathan Huang, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio Guadarrama, n.d.), deteksi wajah (Florian Schroff, Dmitry Kalenichenko, n.d.), lokalisasi gambar (Tobias Weyand, Ilya Kostrikov, n.d.) memperoleh im-fitur usia atau arsitektur dari klasifikasi *ImageNet* model. Misalnya, kami menemukan bahwa fitur gambar obyek diperoleh dari *ImageNet* yang digunakan dalam kombinasi dengan *Faster-Framework* RCNN mencapai deteksi objek bagus. Akhirnya, menunjukkan bahwa penggunaan kembali arsitektur yang dipelajari untuk melakukan klasifikasi *ImageNet*.

***Visual Geometry Group* (*VGG*)**

Dalam sejarah *VGGNet* (Karen Simonyan, n.d.), *VGGNet* ditemukan oleh *VGG* (*Visual Geometry Group*) dari Universitas Oxford, Meski V*GGNet* adalah *runner-up*, dan bukan pemenang *ILSVRC* (*ImageNet Large Scale Visual Recognition Competition*) 2014 dalam klasifikasi, yang telah meningkat secara signifikan dibandingkan *ZFNet* (Pemenang tahun 2013) (Matthew D Zeiler, n.d.) dan *AlexNet* (Pemenang tahun 2012) (Alex Krizhevsky, Ilya Sutskever, n.d.). *GoogLeNet* adalah pemenang *ILSVLC* 2014. Namun, *VGGNet* mengalahkan *GoogLeNet* dan menang pada *Localization* di *ILSVRC* 2014.

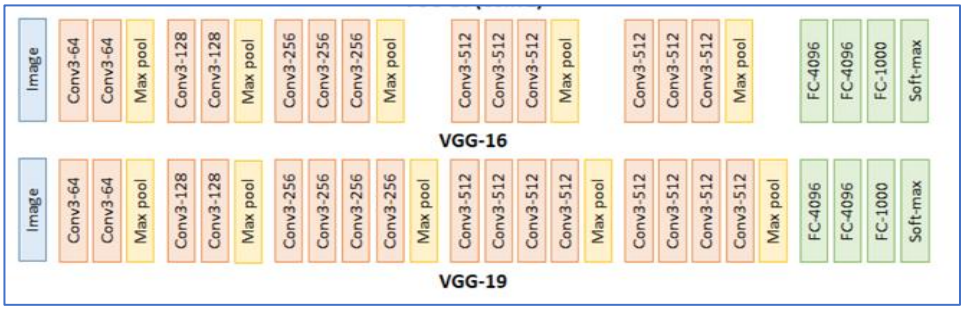


Gambar 5 *ILSVRC* 2014, *Error rate*



Gambar 6 Arsitektur *VGG16*

VGG16 merupakan model *Convolutional Neural Network* (*CNN*) yang diusulkan oleh K. Simonyan dan A. Zisserman dari University of Oxford dalam Jurnal dengan judul "*Very Deep Convolutional Networks for Large-Scale Image Recognition*". Model ini mencapai akurasi pengujian 92,7% top-5 di *ImageNet*, yang menggunakan *Dataset* lebih dari 14 juta gambar yang termasuk dalam 1000 kelas. Itu adalah salah satu model terkenal yang diajukan ke *ILSVRC*-2014. Itu membuat peningkatan dari AlexNet dengan mengganti filter berukuran kernel besar (masing-masing 11 dan 5 di lapisan konvolusi pertama dan kedua) dengan beberapa filter berukuran kernel 3x3 satu demi satu. *VGG16* dilakukan *Training* selama berminggu-minggu dan menggunakan GPU NVIDIA *Titan Black*.

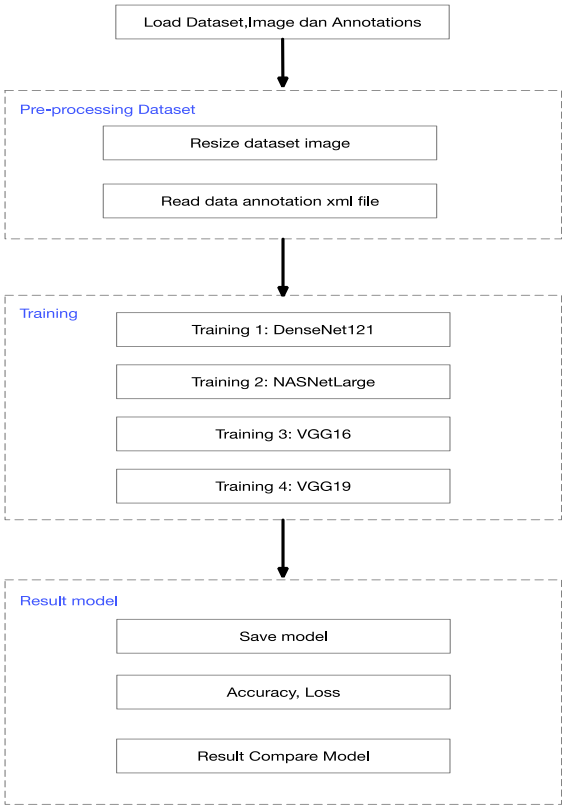


Gambar 7 *VGG16* & *VGG19*

VGG-16 memperoleh tingkat kesalahan 8,8% yang berarti *Deep Learning Network* masih dapat ditingkatkan dengan menambahkan jumlah lapisan. *VGG-19* memperoleh tingkat kesalahan 9,0% yang berarti *Deep Learning* tidak membaik dengan menambahkan jumlah lapisan. Dengan demikian, tidak perlu menambahkan lapisan dan eksperimen dihentikan.

2. Metode Penelitian

Masalah deteksi kendaraan ini dengan metode *Deep Learning* menggunakan model *DenseNet*121, *NasNetLarge*, *VGG16* dan *VGG19*. Dataset yang digunakan adalah Kaggle dan dilakukan download untuk dilakukan deteksi dengan empat model *Deep Learning* tersebut. Hasilnya dari model tersebut dapat diketahui akurasi, *loss* dan *performance* modelnya. Dari model yang dihasilkan oleh empat metode tersebut akan dilakukan perbandingan.



Gambar 8 Metode yang diusulkan

Dataset yang akan diproses untuk empat model adalah sama, yaitu terdiri atas 2 folder, *image* folder dan a*nnotation* folder.

Langkah 1: Membaca semua *dataset*, gambar dan *annotation*. *File Annotation* merupakan *file* yang berisikan informasi mengenai gambar kendaraan. Berisikan Nama *Image*, Jenis *file* png, lebar gambar, dan tinggi gambar.

Langkah 2: *Pre-processing Dataset*, yaitu *Resize dataset image*, yang bertujuan menyamakan semua ukuran *file*, sehingga mudah dilakukan proses *Training*. Membaca informasi dataset dari *file* xml. *File* xml ini merupakan label.

Langkah 3: *Training* untuk ke empat model. *Training* model *DenseNet121*, model *NASNetLarge*, model *VGG16* dan model *VGG19*.

Langkah 4: *Result* model, *save* model, *accuracy* dan *loss*, *Compare* model.

3. Hasil dan Pembahasan

Dataset dari Kaggle adalah *dataset* berbentuk *Image* plat nomor kendaraan dari negara India. Jumlah dataset terdiri atas 433 gambar dan 433 *file Annotation* sebagai keterangan dari informasi gambar plat nomor kendaraan. *Type file Annotation* adalah *file* xml. dan *type file* gambar menggunakan *file* png.

Gambar 9 Plat nomor kendaraan

Gambar 10 *File Annotations* dalam bentuk xml

Gambar 9, adalah sebagai *dataset image* yang akan dilakukan *Training*. Gambar 10 sebagai *Label dataset*, selain itu juga informasi mengenai penjelasan dari *dataset image*. Berikut isi dari *dataset* Cars0.xml. Informasi mengenai letak *file* gambar, nama *file image*. Informasi lainnya lebar gambar, tinggi gambar, kedalaman gambar. Informasi mengenai kotak plat kendaraan juga disebutkan. Xmin, ymin, xmax dan ymax. Informasi ini akan digunakan dalam proses labeling informasi gambar.

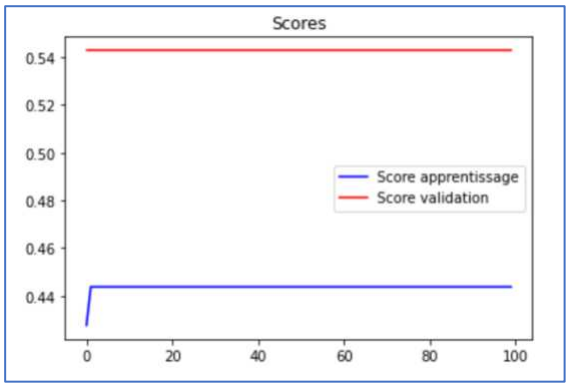
Model *DenseNet121*

```
model = build_densenet()
annealer = ReduceLROnPlateau(monitor='val_accuracy', factor=0.5,
                             patience=5, verbose=1, min_lr=1e-3)
checkpoint = ModelCheckpoint('model.h5', verbose=1, save_best_only=True)
# Generates batches of image data with data augmentation
datagen = ImageDataGenerator(rotation_range=360, width_shift_range=0.2,
                             height_shift_range=0.2, zoom_range=0.2,
                             horizontal_flip=True, vertical_flip=True)
datagen.fit(X_train)
hist = model.fit(datagen.flow(X_train, y_train, batch_size=BATCH_SIZE),
                 steps_per_epoch=X_train.shape[0] // BATCH_SIZE,
                 epochs=EPOCHS, verbose=2,callbacks=[annealer, checkpoint],
                 validation_data=(X_val, y_val))
```

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 224, 224, 3)]     0
_________________________________________________________________
conv2d (Conv2D)              (None, 224, 224, 3)       84
_________________________________________________________________
densenet121 (Functional)     (None, None, None, 1024)  7037504
_________________________________________________________________
global_average_pooling2d (Gl (None, 1024)              0
_________________________________________________________________
batch_normalization (BatchNo (None, 1024)              4096
_________________________________________________________________
dropout (Dropout)            (None, 1024)              0
_________________________________________________________________
dense (Dense)                (None, 256)               262400
_________________________________________________________________
batch_normalization_1 (Batch (None, 256)               1024
_________________________________________________________________
dropout_1 (Dropout)          (None, 256)               0
_________________________________________________________________
root (Dense)                 (None, 4)                 1028
=================================================================
Total params: 7,306,136
Trainable params: 7,219,928
Non-trainable params: 86,208
```

Gambar 11 *Architecture DenseNet121*.

Dataset dilakukan training dengan menggunakan *DensetNet121*, informasi model terdapat pada gambar 12. Densenet121(*functional*) dengan *Output Shape* (None, 7, 7, 1024) dan param 7037504. Arsitektur diatas akan memproses input gambar sebanyak 433 dan *file Annotation*.xml sebanyak 433 sebagai Label.



Gambar 12 Skor *Train DenseNet121*

Hasil *Training* akurasi: 49,42%, Loss: 24,97%. Hasilnya belum bagus dan memungkinkan untuk merubah atau menggunakan model *Deep Learning Convolution Neural Network* yang lainnya.

Model *NASNetLarge*

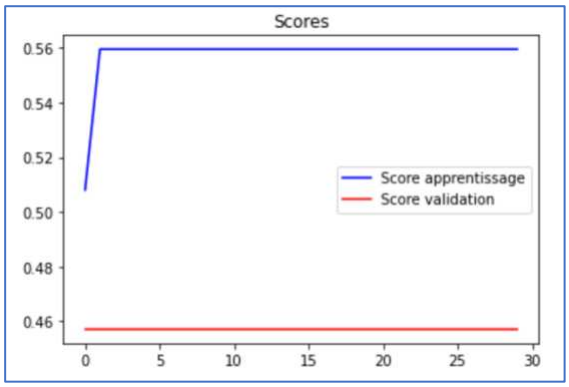
Gambar 13. Model NasNetLarge.

Dataset dilakukan training dengan menggunakan *NASNetLarge*, informasi model terdapat pada gambar 14. NASNetLarge(functional) dengan *Output Shape* (None, 11, 11, 4032) dan param 84916818. Kemudian dilanjutkan dengan *Split Dataset* dengan membagi data *training* 90% dengan data *testing* 10%. Setelah itu melakukan Fit pada proses *Training* dengan komposisi seperti di atas, dengan model

```
#  ------------------- model NasNetLarge -------------------
model = Sequential()
model.add(NASNetLarge(weights="imagenet", include_top = False,
                      input_shape=(IMAGE_SIZE, IMAGE_SIZE, 3)))
model.add(Flatten())
model.add(Dense(128, activation="relu"))
model.add(Dense(128, activation="relu"))
model.add(Dense(64, activation="relu"))
model.add(Dense(4, activation="sigmoid"))

model.layers[-6].trainable = False

model.summary()
```

```
Model: "sequential_5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
NASNet (Functional)          (None, 11, 11, 4032)      84916818
_________________________________________________________________
flatten_4 (Flatten)          (None, 487872)            0
_________________________________________________________________
dense_16 (Dense)             (None, 128)               62447744
_________________________________________________________________
dense_17 (Dense)             (None, 128)               16512
_________________________________________________________________
dense_18 (Dense)             (None, 64)                8256
_________________________________________________________________
dense_19 (Dense)             (None, 4)                 260
=================================================================
Total params: 147,389,590
Trainable params: 62,472,772
Non-trainable params: 84,916,818
_________________________________________________________________
```

NASNetLarge, *Dataset* dilakukan *Split*, kemudian baru dilakukan fit untuk proses *Training*.
Epoch setiap proses berbeda-beda, dikarenakan keterbatasan perangkat keras komputer. *Epoch* digunakan 30 iterasi. Hasil dari proses training mencapai akurasi Loss: 11,02% dan Accuracy 49,42%.



Gambar 14. Score Train NasNetLarge.

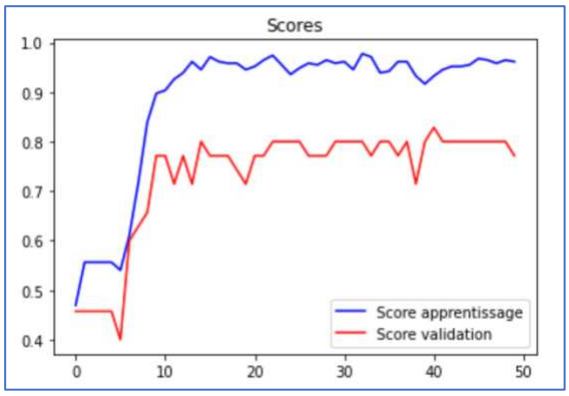
Model *VGG16*

```
# Create the model
model = Sequential()
model.add(VGG16(weights="imagenet", include_top = False,
                input_shape=(IMAGE_SIZE, IMAGE_SIZE, 3)))

model.add(Flatten())
model.add(Dense(128, activation="relu"))
model.add(Dense(128, activation="relu"))
model.add(Dense(64, activation="relu"))
model.add(Dense(4, activation="sigmoid"))

model.layers[-6].trainable = False

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Functional)           (None, 7, 7, 512)         14714688
_________________________________________________________________
flatten (Flatten)            (None, 25088)             0
_________________________________________________________________
dense (Dense)                (None, 128)               3211392
_________________________________________________________________
dense_1 (Dense)              (None, 128)               16512
_________________________________________________________________
dense_2 (Dense)              (None, 64)                8256
_________________________________________________________________
dense_3 (Dense)              (None, 4)                 260
=================================================================
Total params: 17,951,108
Trainable params: 3,236,420
Non-trainable params: 14,714,688
_________________________________________________________________
```

Gambar 15 *Model VGG16*

Dataset dilakukan training dengan menggunakan *VGG16*, informasi model terdapat pada gambar 16 *VGG16*(*functional*) dengan *Output Shape* (None, 7, 7, 512) dan param 14714688. *Dataset* dilakukan split dengan membagi data *training* 90% dan data *testing* 10%. Kemudian dilakukan proses *training* untuk menghasilkan model dari *VGG16*. Untuk *variable Epoch* dibuat 30, supaya tidak terlalu lama. Setelah itu dilakukan pengukuran *training* dengan memberikan data *testing*.



Gambar 16 *Score train VGG16*

Hasil dari proses *training* mencapai akurasi *Loss*: 0,65% dan *Accuracy* 83,90%. Hasil ini lebih baik dari penggunaan model *Deep Learning* seperti metode sebelumnya, untuk kasus pendeteksian sebelumnya. Penelitian dilanjutkan dengan menggunakan model *Deep Learning* lainnya seperti *VGG19*.

Model *VGG19*

```
# -------- Create the model VGG 19 ------
model = Sequential()
model.add(VGG19(weights="imagenet", include_top = False,
                input_shape=(IMAGE_SIZE, IMAGE_SIZE, 3)))
model.add(Flatten())
model.add(Dense(128, activation="relu"))
model.add(Dense(128, activation="relu"))
model.add(Dense(64, activation="relu"))
model.add(Dense(4, activation="sigmoid"))

model.layers[-6].trainable = False

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg19 (Functional)           (None, 7, 7, 512)         20024384
_________________________________________________________________
flatten (Flatten)            (None, 25088)             0
_________________________________________________________________
dense (Dense)                (None, 128)               3211392
_________________________________________________________________
dense_1 (Dense)              (None, 128)               16512
_________________________________________________________________
dense_2 (Dense)              (None, 64)                8256
_________________________________________________________________
dense_3 (Dense)              (None, 4)                 260
=================================================================
Total params: 23,260,804
Trainable params: 3,236,420
Non-trainable params: 20,024,384
_________________________________________________________________
```

Gambar 17 Model *VGG19*

Dataset dilakukan *training* dengan menggunakan *VGG19*, informasi model terdapat pada gambar 18 *VGG19* (*functional*) dengan *Output Shape* (None, 7, 7, 512) dan param 20024384. *Dataset* dilakukan split dengan membagi data *training* 90% dan data testing 10%. Kemudian dilakukan proses *training* untuk menghasilkan model dari *VGG19*. Untuk *variable Epoch* dibuat 30, supaya tidak terlalu lama. Setelah itu dilakukan pengukuran *training* dengan memberikan data *testing*.

Gambar 18 *Score train VGG19*

Hasil dari proses *training* mencapai akurasi *Loss*: 0,65% dan *Accuracy* 85,5%. Dari percobaan dengan membandingan ke empat metode *Convolutional Neural Network* (*CNN*), maka dengan menggunakan *VGG19* menghasilkan *score* akurasi yang meningkat.

Tabel 1: Komparasi dari model yang telah dilakukan *training*

Model	Loss	Accuracy
DenseNet121	24,97%	49,42%
NASNetLarge	11,02%	49,42%
VGG16	0,65%	83,90%
VGG19	0,65%	85,05%

4. Kesimpulan

Deep Learning memungkinkan model komputasi dari berbagai lapisan proses untuk mempelajari representasi data dari berberapa tingkat abstraksi (Lecun et al., 2015). *VGG16* dan *VGG19* telah menampakan hasil yang bagus untuk kasus deteksi plat nomor kendaraan. Sedangkan untuk *DensetNet121* dan *NASNetLarge* menampilkan hasil yang kurang bagus. Saran dari penggunaan model *CNN* terutama untuk model *DenseNet121* dan *NASNetLarge*, perlu dilakukan uji coba dengan menggunakan cara yang berbeda untuk mendapatkan hasil terbaik. Berisi hasil yang diharapkan dari penelitian yang telah dicapai serta seberapa besar hasil penelitian ini berdampak kepada masyarakat. Dalam kesimpulan juga dapat berisi saran perbaikan yang diharapkan dapat dilakukan oleh peneliti selanjutnya.

Daftar Referensi

Alex Krizhevsky, Ilya Sutskever, G. E. H. (n.d.). ImageNet Classification with Deep Convolutional Neural Networks. *NIPS'12: Proceedings of the 25th International Conference on Neural Information Processing Systems*, *1*, 1097–1105.

Barret Zoph, Q. V. Le. (n.d.). Neural Architecture Search with Reinforcement Learning. *ICLR 2017 Conference Submission*.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, Z. W. (2016). Rethinking the Inception Architecture for Computer Vision. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. https://doi.org/10.1109/CVPR.2016.308

Florian Schroff, Dmitry Kalenichenko, J. P. (n.d.). FaceNet: A Unified Embedding for Face Recognition

and Clustering. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, *3*, 815–823. https://doi.org/10.1109/CVPR.2015.7298682

Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, *2017-Janua*, 2261–2269. https://doi.org/10.1109/CVPR.2017.243

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *ICML*, *37*, 448–456.

Jonathan Huang, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio Guadarrama, K. M. (n.d.). *Speed/accuracy trade-offs for modern convolutional object detectors*. *3*. https://doi.org/10.1109/CVPR.2017.351

Karen Simonyan, A. Z. (n.d.). VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION. *International Conference on Learning Representations*.

Lecun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444. https://doi.org/10.1038/nature14539

Matthew D Zeiler, R. F. (n.d.). Visualizing and Understanding Convolutional Networks. *European Conference on Computer Vision*, *3*.

Tobias Weyand, Ilya Kostrikov, J. P. (n.d.). PlaNet - Photo Geolocation with Convolutional Neural Networks. *In European Conference on Computer Vision*. https://doi.org/10.1007/978-3-319-46484-8_3

Xavier Glorot, Antoine Bordes, Y. B. (2011). Deep Sparse Rectifier Neural Networks. *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, *15*, 315–323.

Xingcheng Zhang, Zhizhong Li, Chen Change Loy, D. L. (2017). PolyNet: A Pursuit of Structural Diversity in Very Deep Networks. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, *2*. https://doi.org/10.1109/CVPR.2017.415

Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Computation*, *Neural com*, 541–551.

Y. LeCun, L. Bottou, Y. Bengio, P. H. (1998). Gradient based learning applied to document recognition. *Proceedings of the IEEE, 86(11)*.

Zoph B, Vasudevan, Shlens J, Q. V. Le, & Brain, G. (n.d.). Learning Transferable Architectures for Scalable Image Recognition. *Developing Neural Network Image Classification Models Often Requires Significant Architecture Engineering. In This Paper, We Study a Method to Learn the Model Architectures Directly on the Dataset of Interest. As This Approach Is Expensive When the Datase*, *4*.