TUGAS AKHIR – KS184822

PREDIKSI PENGELUARAN PERKAPITA YANG DISESUAIKAN BERDASARKAN CITRA DIGITAL *GOOGLE EARTH* MENGGUNAKAN KOMBINASI *CONVOLUTIONAL NEURAL NETWORK* (CNN) DAN *SUPPORT VECTOR REGRESSION* (SVR)

ASVA ABADILA ROUHAN
NRP 062116 4000 0105

Dosen Pembimbing
Santi Puteri Rahayu, M.Si., Ph.D
Dr. rer. pol. Dedy Dwi Prastyo, S.Si., M.Si.

PROGRAM STUDI SARJANA
DEPARTEMEN STATISTIKA
FAKULTAS SAINS DAN ANALITIKA DATA
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA 2020

TUGAS AKHIR – KS184822

PREDIKSI PENGELUARAN PERKAPITA YANG DISESUAIKAN BERDASARKAN CITRA DIGITAL *GOOGLE EARTH* MENGGUNAKAN KOMBINASI *CONVOLUTIONAL NEURAL NETWORK* (CNN) DAN *SUPPORT VECTOR REGRESSION* (SVR)

ASVA ABADILA ROUHAN
NRP 062116 4000 0105

Dosen Pembimbing
Santi Puteri Rahayu, M.Si., Ph.D
Dr. rer. pol. Dedy Dwi Prastyo, S.Si., M.Si.

PROGRAM STUDI SARJANA
DEPARTEMEN STATISTIKA
FAKULTAS SAINS DAN ANALITIKA DATA
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA 2020

FINAL PROJECT – KS184822

PREDICT HOUSEHOLD EXPENDITURE BASED ON GOOGLE EARTH IMAGES USING THE COMBINATION OF CONVOLUTIONAL NEURAL NETWORK (CNN) AND SUPPORT VECTOR REGRESSION (SVR)

ASVA ABADILA ROUHAN
SN 062116 4000 0105

Supervisors
Santi Puteri Rahayu, M.Si., Ph.D
Dr. rer. pol. Dedy Dwi Prastyo, S.Si., M.Si.

UNDERGRADUATE PROGRAMME
DEPARTMENT OF STATISTICS
FACULTY OF SCIENCE AND DATA ANALYTICS
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA 2020

(Halaman ini sengaja dikosongkan)

LEMBAR PENGESAHAN

PREDIKSI PENGELUARAN PERKAPITA YANG DISESUAIKAN BERDASARKAN CITRA DIGITAL *GOOGLE EARTH* MENGGUNAKAN KOMBINASI *CONVOLUTIONAL NEURAL NETWORK* (CNN) DAN *SUPPORT VECTOR REGRESSION* (SVR)

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Statistika
pada
Program Studi Sarjana Departemen Statistika
Fakultas Sains dan Analitika Data
Institut Teknologi Sepuluh Nopember

Oleh :

Asva Abadila Rouhan
NRP. 062116 4000 0105

Disetujui oleh Pembimbing:

Santi Puteri Rahayu, M.Si., Ph.D. ( )
NIP. 19750115 199903 2 003

Dr. rer. pol. Dedy Dwi Prastyo, S.Si., M.Si. ( )
NIP. 19831204 200812 1 002

Mengetahui,
Kepala Departemen

Dr. Dra. Kartika Fithriasari, M.Si.
NIP. 19691212 199303 2 002

SURABAYA, 22 AGUSTUS 2020

(Halaman ini sengaja dikosongkan)

PREDIKSI PENGELUARAN PERKAPITA YANG DISESUAIKAN BERDASARKAN CITRA DIGITAL *GOOGLE EARTH* MENGGUNAKAN KOMBINASI *CONVOLUTIONAL NEURAL NETWORK* (CNN) DAN *SUPPORT VECTOR REGRESSION* (SVR)

Nama Mahasiswa : Asva Abadila Rouhan
NRP : 062116 4000 0105
Departemen : Statistika
Dosen Pembimbing : Santi Puteri Rahayu, M.Si., Ph.D.
Dr. rer. pol. Dedy Dwi Prastyo, S.Si., M.Si.

Abstrak

Kemiskinan dapat didefinisikan sebagai ketidakmampuan seseorang atau rumah tangga dalam memenuhi kebutuhan pokoknya. Kebanyakan negara menggunakan pendapatan atau konsumsi rumah tangga sebagai standar pengukuran kesejahteraan penduduk. Namun, pengumpulan data secara mendetail dari pintu ke pintu merupakan hal banyak memakan waktu dan biaya. Belakangan ini muncul sumber data alternatif pengganti survei, yaitu citra digital satelit. Citra digital umumnya diolah menggunakan Convolutional Neural Network (CNN). Salah satu arsitektur CNN yang dianggap paling baik adalah VGG16. VGG16 dalam tugas akhir ini digunakan sebagai fixed feature extraction sedangkan pemodelan estimasi kemiskinan di Pulau Jawa menggunakan Support Vector Regression (SVR). Kombinasi kedua metode menghasilkan model dengan performa 0,703 pada tahap testing. Terdapat hampir 80% kesesuaian pada 25% golongan pengeluaran perkapita terendah antara hasil estimasi dan data aktual.

Kata kunci: Citra Satelit, Convolutional Neural Network, Pengeluaran Perkapita, Support Vector Regression, VGG16.

(Halaman ini sengaja dikosongkan)

PREDICT HOUSEHOLD EXPENDITURE BASED ON GOOGLE EARTH IMAGES USING THE COMBINATION OF CONVOLUTIONAL NEURAL NETWORK (CNN) AND SUPPORT VECTOR REGRESSION (SVR)

Name : **Asva Abadila Rouhan**
Student Number : **062116 4000 0105**
Department : **Statistics**
Supervisors : **Santi Puteri Rahayu, M.Si., Ph.D.**
Dr. rer. pol. Dedy Dwi Prastyo, S.Si., M.Si.

Abstract

Poverty can be defined as the inability of a person or a household to satisfy their primary needs. Most countries use household income or expenditure as a standard measurement of population wealth. However, a comprehensive door-to-door data collection is time-consuming and costly. Recently, satellite imagery as an alternative source of data has emerged to replace survey. Digital images are generally processed using Convolutional Neural Network (CNN). One of the CNN architectures that is considered as the most excellent is VGG16. In this final project, VGG16 is used as a fixed feature extractor, while Support Vector Regression (SVR) is used to estimates poverty in Java. The combination of these two methods generates a model with R^2 0,703 in the testing stage. There is almost 80% matching ratio in the 25% lower household expenditure group between the estimated and real data.

Keywords: Convolutional Neural Network, Household Expenditure, Satellite Imagery, Support Vector Regression, VGG16.

(Halaman ini sengaja dikosongkan)

KATA PENGANTAR

Puji syukur kepada kehadirat Allah SWT yang telah memberikan rahmat, hidayah, karunia serta pertolongan-Nya yang tak pernah henti diberikan, sehingga penulis dapat menyelesaikan laporan Tugas Akhir dengan judul "**Prediksi Pengeluaran Perkapita yang Disesuaikan Berdasarkan Citra Digital Google Earth Menggunakan Kombinasi Convolutional Neural Network dan Support Vector Regression**" dengan baik, lancar dan tepat waktu.

Penulis menyadari bahwa Tugas Akhir ini dapat terselesaikan tidak terlepas dari bantuan dan dukungan berbagai pihak. Oleh karena itu, penulis menyampaikan terima kasih kepada:

1. Kedua orang tua, atas segala do'a, nasehat, kasih sayang, dan dukungan yang diberikan kepada penulis demi kesuksesan dan kebahagiaan penulis.
2. Dr. Dra. Kartika Fithriasari, M.Si. selaku Kepala Departemen Statistika dan Dr. Santi Wulan Purnami, S.Si., M.Si. selaku Sekretaris Departemen 1 Bidang Akademik dan Ke-mahasiswaan yang telah memberikan fasilitas, sarana, dan prasarana.
3. Dr. rer. pol. Dedy Dwi Prastyo, S.Si., M.Si. selaku dosen wali selama masa studi dan selaku dosen pembimbing yang telah banyak memberikan saran dan arahan dalam proses belajar di Departemen Statistika.
4. Santi Puteri Rahayu, M.Si., Ph.D. selaku dosen pembimbing yang telah meluangkan waktu dan dengan sangat sabar memberikan bimbingan, saran, dukungan serta motivasi selama penyusunan Tugas Akhir.
5. Dr. rer. pol. Heri Kuswanto, S.Si., M.Si. dan Dr. M. Atok selaku dosen penguji yang selalu sabar dalam mengomentari serta memberikan masukan dan saran dalam penyelesaian Tugas Akhir.

6. Seluruh dosen Statistika ITS yang telah memberikan ilmu dan pengetahuan yang tak ternilai harganya, serta segenap karyawaan Departemen Statistika ITS.
7. Adam dan Bapak Kos yang selalu berusaha memberikan fasilitas internet sehingga Tugas Akhir dapat dikerjakan dengan nyaman.
8. Teman lomba seperjuangan, Arif, Nafis, Naufal, dan Fitria yang selalu bersemangat dalam mengikuti kompetisi.
9. Mbak Ulfa dan Mas Azmi yang bersedia membantu penulis memahami konsep CNN.
10. Anggota *Statistics Computer Course* (SCC) 17/18 dan 18/19 yang memberikan pengalaman berorganisasi dan membantu penulis meningkatkan kemampuan di bidang komputasi.
11. Teman-teman Statistika ITS Σ27, TR16GER yang selalu memberikan dukungan kepada penulis selama ini.
12. Semua pihak yang turut membantu dalam pelaksanaan Tugas Akhir yang tidak bisa penulis sebutkan namanya satu persatu.

Besar harapan penulis untuk mendapatkan kritik dan saran yang membangun sehingga Tugas Akhir ini dapat memberikan manfaat bagi semua pihak yang terkait.

Surabaya, 18 Juli 2020

Penulis

DAFTAR ISI

Halaman

LEMBAR PENGESAHAN v
ABSTRAK vii
ABTRACT ix
KATA PENGANTAR xi
DAFTAR ISI xiii
DAFTAR GAMBAR xv
DAFTAR TABEL xvii
DAFTAR LAMPIRAN xix

BAB I PENDAHULUAN 1
1.1 Latar Belakang 1
1.2 Rumusan Masalah 8
1.3 Tujuan Penelitian 9
1.4 Manfaat Penelitian 9
1.5 Batasan Masalah 10

BAB II TINJAUAN PUSTAKA 11
2.1 Statistika Deskriptif 11
2.2 Citra Digital 12
2.3 *Convolutional Neural Network* (CNN) 13
2.3.1 *Convolutional Layer* 14
2.3.2 *Pooling Layer* 18
2.3.3 *Fully-connected Layer* 19
2.4 *Transfer Learning* 20
2.5 VGG16 21
2.6 *Support Vector Regression* (SVR) 22
2.7 *K-fold Cross Validation* 28
2.8 Ukuran Kebaikan Model 29
2.9 Pengeluaran Perkapita 30
2.10 Google Earth 30

BAB III METODOLOGI PENELITIAN 32
3.1 Sumber Data 33
3.2 Variabel Penelitian dan Struktur Data 33

3.3 Langkah Analisis 35
BAB IV ANALISIS DAN PEMBAHASAN 39
4.1 Karakteristik Pengeluaran Perkapita Kabupaten/Kota di Pulau Jawa 39
4.2 Prediksi Pengeluaran Perkapita Kabupaten/Kota di Pulau Jawa 41
4.2.1 *Pre-Processing* Citra Digital Satelit 42
4.2.2 Pemodelan SVR Berdasarkan *Output* VGG16 47
4.3 Pemetaan Hasil Estimasi Pengeluaran Perkapita Kabupaten/Kota di Pulau Jawa 50
BAB V KESIMPULAN DAN SARAN 55
5.1 Kesimpulan 55
5.2 Saran 55
DAFTAR PUSTAKA 57
LAMPIRAN 61

DAFTAR GAMBAR

Halaman

Gambar 2.1 Gambar Bercitra RGB sebagai Vektor..... 13
Gambar 2.2 Ilustrasi *Convolutional Layer*..... 15
Gambar 2.3 Perbedaan Antara *Stride* 1 dan *Stride* 2 16
Gambar 2.4 Ilustrasi *Padding* Berukuran 1 × 1 17
Gambar 2.5 Operasi Konvolusi dengan 2 *Input Channel* 17
Gambar 2.6 Ilustrasi *Pooling Layer*..... 18
Gambar 2.7 llustrasi *Fully-connected Layer*..... 19
Gambar 2.8 Fungsi Aktivasi ReLU 20
Gambar 2.9 Arsitektur VGG16..... 21
Gambar 2.10 SVR Linear 1 Dimensi..... 23
Gambar 2.11 Tipe *Loss Function*: (a) Linear, (b) kuadratik, (c) Huber..... 24
Gambar 2.12 Ilustrasi Kernel Polinomial 27
Gambar 2.13 Ilustrasi Kernel *Radial Basis Function* 27
Gambar 2.14 Ilustrasi Regresi Satu Dimensi Menggunakan Tiga Kernel SVR 28
Gambar 2.15 Ilustrasi 10-Fold Cross Validation 29
Gambar 3.1 Diagram Alir Penelitian 37
Gambar 4.1 Persebaran Pengeluaran Perkapita 2018 di Pulau Jawa 40
Gambar 4.2 *Boxplot* dan *Histogram* Pengeluaran Perkapita 2018 (ribu/tahun/orang)..... 41
Gambar 4.3 Visualisasi 6 *Filter* Konvolusi Pertama Pada *Layer* Pertama..... 43
Gambar 4.4 Visualisasi 64 *Feature Maps* Pertama Pada *Layer* block1_conv1 44
Gambar 4.5 Visualisasi 64 *Feature Maps* Pertama Pada *Layer* block5_conv3 45
Gambar 4.6 *Output Feature Maps* Pertama pada *Layer* block5_conv3..... 46

Gambar 4.7 Ilustrasi *Fully-connected Layer* VGG16 47
Gambar 4.8 *Density Scatter Plot Hyperparameter* Hasil Bangkitan 48
Gambar 4.9 3D *Scatter Plot* Hubungan Antara H*yperparameter* dan R^2.....50
Gambar 4.10 Perbandingan Nilai Aktual dan Estimasi Pengeluaran Perkapita 51
Gambar 4.11 Hasil Prediksi dan Persentase *Error* Model 52

DAFTAR TABEL

Halaman

Tabel 2.1 Kode Warna RGB 13
Tabel 3.1 Variabel Penelitian 33
Tabel 3.2 Struktur Data Penelitian Sebelum *Feature Extraction* 34
Tabel 3.3 Struktur Data Penelitian Setelah *Feature Extraction* 35
Tabel 4.1 Arsitektur VGG16 Sebagai Feature Extractor 42
Tabel 4.2 Kombinasi 10 Parameter Terbaik Berdasarkan R^2 *Testing* 49
Tabel 4.3 29 Kabupaten/Kota dengan Pengeluaran Terendah 53

(Halaman ini sengaja dikosongkan)

DAFTAR LAMPIRAN

Halaman

Lampiran 1. Sampel *Input* Gambar 61
Lampiran 2. Perubahan Gambar Menjadi 4096 Variabel 62
Lampiran 3. Data Penelitian 63
Lampiran 4. Kombinasi *Hyperparameter* SVR Beserta Performa Model 64
Lampiran 5. Hasil Prediksi (dalam ribuan) 65
Lampiran 6. Sintaks Mengambil Gambar Menggunakan Maps Static API Dengan Python 66
Lampiran 7. Histogram dan Boxplot Dengan R 77
Lampiran 8 Sintaks Pemodelan SVR Dengan R 79
Lampiran 9. Output Pemodelan SVR 81
Lampiran 10. Sintaks 3D Plot Interaktif Dengan Python 83
Lampiran 11. Tahapan Perhitungan VGG16 83
Lampiran 12. Contoh Estimasi Regresi SVR Kernel RBF 91
Lampiran 13. Surat Pernyataan Data 93

(Halaman ini sengaja dikosongkan)

BAB I
PENDAHULUAN

1.1 Latar Belakang

Kemiskinan adalah salah satu masalah mendasar berskala global yang membutuhkan perhatian serius dari pemerintah, khususnya yang terjadi di negara berkembang, seperti Indonesia. Negara-negara di dunia berkomitmen untuk mengentaskan kemiskinan pada tahun 2030 sebagaimana dinyatakan dalam tujuan pertama dari 17 tujuan yang dituangkan dalam *Sustainable Development Goals* (SDGs), yaitu menghapus kemiskinan. Tindakan-tindakan prioritas dalam mengentaskan kemiskinan meliputi meningkatkan akses mata pencaharian berkelanjutan, peluang wirusaha dan sumber daya produktif, menyediakan akses layanan social dasar bagi semua kalangan, secara progresif mengembangkan sistema perlindungan social untuk mendukung mereka yang tidak dapat hidup sendiri, memberdayakan orang yang hidup dalam kemiskinan, mengatasi dampak kemiskinan yang tidak proporsional terhadap perempuan, bekerja dengan pendonor dan penerima yang berminat mengalokasikan Bantuan Pembangunan Resmi (*Official Development Assistance*) untuk mengentaskan kemiskinan, dan mengintensifkan kerja sama internacional untuk pengentasan kemiskinan (Perserikatan Bangsa-Bangsa, n.d.).

Kemiskinan dapat didefinisikan berdasarkan perspektifnya, seperti perspektif social, ekonomi, dan politik. Namun kemiskinan seringkali didefinisikan dalam konteks sosioekonomi, yaitu ketidakmampuan seseorang atau suatu rumah tangga dalam memenuhi kebutuhan pokoknya yaitu sandang, papan, dan pangan. Pada dasarnya, kemiskinan diukur untuk menentukan apakah seseorang dianggap miskin

atau tidak. Para pengambil kebijakan bergantung pada ukuran tersebut, entah ukuran tersebut diukur pada tingkat rumah tangga atau kabupaten/kota, untuk mengerahkan sumber daya mereka ke tempat-tempat yang paling membutuhkan bantuan. Suatu standar pengukuran yang akurat diperlukan untuk membandingkan kemiskinan antar waktu dan antar daerah. Kebanyakan negara menggunakan pendapatan rumah tangga atau konsumsi rumah tangga sebagai dasar penentuan kesejahteraan penduduk miskin (World Bank, 2014). Sayangnya, pengumpulan data yang tepat waktu dan dapat diandalkan merupakan hal yang sulit dan mahal karena membutuhkan pengumpulan data yang mendetail secara langsung dari rumah tangga (Jerven, 2013).

Badan Pusat Statistik (BPS) merupakan Lembaga Pemerintah Non Kementrian yang menyediakan kebutuhan data dan membantu kegiatan statistic di lembaga pemerintahan (Badan Pusat Statistik, n.d.-b). Salah satu data yang dikumpulan oleh BPS adalah Survei Sosial Ekonomi Nasiolan (Susenas). Susenas menjadi sandaran utama untuk memenuhi kebutuhan pemerintah dalam mengimplementasikan pembangunan nasional agar sejalan dengan tujuan pembangunan internasional (SDGs). Unit analisis terkecil yang digunakan pada Susenas adalah rumah tangga. Salah satu produk hasil Susenas yang dilaksanakan pada Maret 2019 yang dirilis oleh BPS pada 1 November 2019 adalah pengeluaran untuk konsumsi penduduk Indonesia. Namun, data yang dirilis merupakan data per provinsi. Sedangkan data konsumsi penduduk (pengeluaran perkapita) per kabupaten masih merupakan data tahun 2018.

Mengingat lamanya pembaruan data dan mahalnya pengumpulan data melalui survei, dalam beberapa tahun terakhir muncul sumber-sumber data baru yang berpotensi

dapat dimanfaatkan untuk memperkirakan kemiskinan seperti data ponsel atau satelit. Pengembangan terbaru dari metode berbasis *machine learning* telah memungkinkan suatu pendekatan intensif untuk mengukur kemiskinan. Noor, Alegana, Gething, Tatem, & Snow, pada tahun 2008 menunjukkan bahwa tingkat cahaya berkorelasi dengan pengukuran kemakmuran berbasis aset di 37 negara Afrika. Data tingkat cahaya ini diperoleh dari pengukuran tingkat intensitas cahaya yang ditangkap secara pasif oleh satelit. Namun, pendekatan ini memiliki performa buruk untuk mengukur daerah yang memiliki pendapatan rendah, tingkat cahaya yang ditangkap satelit tidak dapat dibedakan dengan *noise* yang muncul saat satelit menangkap cahaya (Chen & Nordhaus, 2011). Pendekatan lainnya yang menggunakan data ponsel untuk mengestimasi kemiskinan sebagaimana dilakukan oleh Blumenstock, Cadamuro, & On pada 2015 menunjukkan hasil yang meyakinkan, namun sulit untuk membandingkan hasilnya dengan negara lain karena perbedaan dataset.

Sementara itu, citra digital siang hari muncul sebagai sumber informasi terbaru dalam aktivitas ekonomi. Citra digital siang hari memiliki resolusi gambar yang lebih tinggi daripada citra digital malam hari. Gambar ini mengandung objek yang nampak seperti bangunan, jalan, mobil, ladang pertanian, dan atap yang memungkinkan untuk mengidentifikasi kesejahteraan suatu daerah. Daerah perairan nampak berwarna biru, hijau, atau cokelat tergantung kandungan airnya, sedangkan air beku berwarna putih, abu-abu, atau sedikit biru. Tanaman memiliki nuansa hijau yang berbeda-beda, padang rumput cenderung berwarna hijau pucat, sedangkan hutan berwarna hijau gelap. Tanah pertanian memiliki warna yang lebih terang daripada vegetasi

alami. Tanah kosong biasanya berwarna cokelat, tapi itu bergantung pada kandungan mineralnya. Daerah perkotaan biasanya berwarna perak atau abu-abu dari konsentrasi beton atau bahan bangunan lainnya. Beberapa kota memiliki warna cokelat atau merah tergantung pada bahan yang digunakan untuk atap rumah. Selain warna, objek yang dapat teridentifikasi adalah pola, bentuk, dan tekstur. Perairan merupakan objek yang mudah diidentifikasi karena memiliki bentuk yang unik dan sering nampak dalam peta. Pola lainnya berasal dari tanah. Peternakan biasanya memiliki bentuk geometris yang menonjol daripada pola-pola yang muncul secara alami. Penggundulan hutan seringkali berbentuk persegi. Garis lurus yang nampak hampir pasti merupakan buatan manusia, mulai dari jalan, kanal, hingga batas wilayah (Riebeek, 2013). Objek-objek visual yang terlihat memiliki hubungan linier yang beragam dengan pengeluaran, sehingga dapat menangkap variasi pada daerah miskin (Jean et al., 2016). Jalan raya berkorelasi dengan perkembangan ekonomi, warna tanah, bahan atap, perairan, lahan pertanian, dll. mungkin menyediakan informasi bermanfaat tentang tingkat perkembangan suatu daerah (Pandey, Agarwal, & Krishnan, 2018). Namun, hanya sedikit informasi mengenai apakah objek-objek visual tersebut cukup memberikan informasi dalam mengestimasi kemiskinan. Sejauh ini, hanya sedikit diskusi tentang penerapan langsung model CNN terhadap citra siang hari (Ngestrini, 2019).

Jean et al., 2016 mengenalkan pendekatan baru berbasis *deep learning* untuk mengekstrak fitur-fitur dari citra satelit siang hari yang mengindikasikan kemiskinan yang disebut *transfer learning*. Terdapat dua *output* penelitian Jean et al., yaitu pengeluaran rumah tangga yang didapat dari data survey *Living Standard Measurement Study*

(LSMS) milik World Bank dan tingkat kesejahteraan yang diperoleh dari *Demographics and Health Surveys* (DHS). Kedua survey tersebut diambil pada level *cluster* yang setara dengan desa. Terdapat tiga tahap pada pemodelan mereka, yang pertama dimulai dari *Convolutional Neural Network* (CNN) model yang telah dilatih pada dataset ImageNet. Kemudian model tersebut dilatih ulang (*fine-tune*) berdasarkan informasi yang didapat dari ImageNet untuk memprediksi intensitas cahaya malam yang sesuai dengan input citra satelit siang hari. Pada tahap kedua ini, model dilatih untuk merangkum *input* berdimensi tinggi dari data citra satelit siang hari sebagai *feature* berdimensi rendah yang dapat memprediksi variasi intensitas cahaya malam. Pada tahap akhir, *feature* citra yang diekstrak dari CNN digunakan untuk mengestimasi pengeluaran atau aset pada tingkat *cluster* menggunakan regresi *ridge*. Namun *feature* citra tersebut direduksi terlebih dahulu menjadi 100 dimensi menggunakan *Principal Component Analysis* (PCA). Citra malam hari dapat menampilkan variasi kecil pada daerah dengan pengeluaran rendah. Obyek yang nampak pada citra siang hari, seperti material atap dan jarak ke daerah perkotaan, bervariasi linear dengan pengeluaran, sehingga dapat menangkap variasi diantara *cluster* yang miskin. Prediksi berbasis *cross-validation* yang dilatih pada 4 negara Afrika berbeda, yaitu Nigeria, Tanzania, Uganda, dan Malawi dapat menjelaskan 37% hingga 55% variasi konsumsi rumah tangga dan sebesar 55% hingga 75% dalam menjelaskan variabilitas aset rumah tangga.

Head, Manguin, Tran, & Blumenstock, pada tahun 2017 mencoba mengukur kesejahteraan rumah tangga yang terdiri dari indeks kesejahteraan, pendidikan, akses terhadap air bersih, indeks kesehatan, indeks *anthropometric*, serta

listrik dan ponsel. Pada dasarnya mereka mereplikasi eksperimen yang dilakukan oleh Jean et al., yaitu menggunakan pendekatan *transfer learning*. Arsitektur CNN yang digunakan, VGG16 untuk melakukan *fine-tuning* yang digunakan untuk memprediksi intensitas cahaya malam menggunakan citra satelit siang hari untuk wilayah Rwanda, Nigeria, Haiti, dan Nepal. Mereka menghapus *layer* terakhir CNN, yaitu *fully-connected layer* dan menggantinya dengan *fully-connected layer* baru yang diinisiasi secara acak. *Layer* tersebut dilatih kembali untuk memprediksi intensitas cahaya malam. T*uning* pada CNN dapat dianggap sebagai fungsi yang memetakan citra satelit mentah menjadi *visual features*, dengan memberikan gambar sebagai *input* dan mengekstrak vektor berdimensi 16384. Dengan kata lain, setiap citra satelit akan menghasilkan vektor berdimensi 16384. 16384 variabel tersebut direduksi menjadi 100 variabel menggunakan PCA. Keseratus variabel tersebut dimodelkan menggunakan regresi *ridge* dan menghasilkan R^2 dengan *range* 6% hingga 74%.

Ngestrini pada tahun 2019 melakukan prediksi kemiskinan di Indonesia dengan total gambar sebanyak 18.750 buah. Terdapat tiga metode yang digunakan, yaitu pendekatan *multistep learning*, pendekatan *naïve*, dan pendekatan segmentasi semantic. Pendekatan *multistep learning* yang dimaksud adalah pendekatan yang dikenalkan oleh Jean et al. Ngestrini menggunakan arsitektur VGG-F untuk mengklasifikasikan kelompok intensitas cahaya malam. Pada akhirnya, model tersebut akan dilatih menggunakan regresi *ridge* untuk mengestimasi tingkat kemiskinan berdasarkan ekstraksi citra. Pendekatan *naïve* adalah pendekatan yang secara langsung melatih CNN untuk kebutuhan regresi berdasarkan citra satelit. Sehingga ditambahkan *layer* regresi pada akhir jaringan. Metode ketiga

yaitu pendekatan segmentasi semantik mengacu kepada klasifikasi per-*pixel*. Klasifikasi ini memberikan label/kelas untuk tiap *pixel*. Ngestrini mengkategorikan *feature* dalam sebuah citra menjadi enam kategori, yaitu, vegetasi, tanah, jalan, bangunan, perairan, dan obyek lainnya. Arsitektur CNN yang digunakan merupakan *pre-trained model*. Pendekatan *multistep learning* menghasilkan R^2 sebesar 45% sedangkan pendekatan *naïve* menggunakan arsitektur ResNet10 menghasilkan R^2 maksimum 20%. Pendekatan segmentasi semantic menggunakan arsitektur SegNet dapat menjelaskan variabilitas garis kemiskinan hingga sebesar 48% menggunakan regresi *lasso*. Kombinasi intensitas cahaya malam dan variabel hasil segmentasi menghasilkan R^2 mencapai 56%.

Google API diperlukan untuk mengunduh data peta yang bersifat statis. Namun untuk mengunduh peta menggunakan Maps Static API, Google membebankan biaya sebesar $2 per 1000 akses (Google Maps Platform, 2019). Satu kali pengunduhan peta sama dengan satu kali permintaan akses Maps Static API. Untungnya, Google memberikan kuota gratis sebesar $200 per bulan. Kuota sebesar itu apabila digunakan seluruhnya dapat mengunduh citra sebanyak 200.000 buah. Berdasarkan *file shapefile* (.shp), Pulau Sumatera mencakup 2,5 juta citra, Pulau Jawa 600 ribu citra, Kalimantan 1,4 juta citra, Sulawesi dan Papua 1,8 juta citra. Google selama tahun 2019, telah memperbarui citra Kota Surabaya sebanyak 3 kali, yaitu pada 15 Juni, 3 September, dan terakhir pada 3 November. Citra digital lima ibukota provinsi lainnya yang terletak di Pulau Jawa diperbarui pada Juli 2019, masing-masing dengan tanggal yang berbeda.

Berdasarkan uraian diatas, peneliti ingin memprediksi pengeluaran per kapita menurut kabupaten/kota sebagai *proxy* dalam mengukur pengeluaran rumah tangga berdasarkan citra digital Google Earth menggunakan *Convolutional Neural Network* (CNN) sebagai *feature extractor* dan *Support Vector Regression* (SVR) sebagai pemodelan regresi. Citra yang digunakan adalah citra Pulau Jawa karena memiliki citra yang paling sedikit daripada pulau besar lainnya. Selain itu, data Google Earth lebih mutakhir daripada data BPS. Dengan hanya mengandalkan data yang bersifat public dan gratis, hasil penelitian ini diharapkan dapat memberikan gambaran kasar namun memberikan informasi terkini dalam memetakan tingkat kesejahteraan daerah.

1.2 Rumusan Masalah

BPS membutuhkan waktu kira-kira setengah hingga satu tahun dalam menerbitkan hasil survei. Walaupun hasil tersebut akurat dan terpercaya, terdapat kesenjangan data ketika BPS belum selesai merilis hasil survei. Citra satelit siang hari, seperti Google Earth hadir sebagai alternative data yang dapat dimanfaatkan untuk mengisi kesenjangan data tersebut karena citra tersebut lebih cepat dan lebih sering diperbarui. Persebaran kesejahteraan daerah yang diperbarui secara regular diperlukan sehingga konsekuensi dari suatu kebijakan dapat dimonitor dan dievaluasi dengan cepat dan tanggap. *Image Processing* digunakan untuk memproses suatu informasi suatu gambar. Metode yang umum digunakan adalah *Convolutional Neural Network* (CNN). Model CNN yang sudah ada, yaitu *pre-trained model* dapat digunakan sebagai *fixed feature extraction*. Hasil *feature extraction* ini yang digunakan sebagai variabel independent. Metode *Support Vector Machine* (SVR) digunakan untuk

memodelkan pengeluaran per kapita berdasarkan hasil *feature extraction*. Berdasarkan permasalahan yang telah diuraikan, rumusan masalah dalam penelitian ini adalah bagaimana hasil estimasi pengeluaran per kapita berdasarkan citra satelit menggunakan CNN dan SVR. Hasil estimasi tersebut akan dipetakan menggunakan *choropleth* sehingga didapatkan perkiraan persebaran tingkat kesejahteraan.

1.3 Tujuan Penelitian

Berdasarkan rumusan masalah yang telah dibahas sebelumnya, tujuan yang ingin dicapai dalam penelitian ini sebagai berikut.

1. Mendapatkan informasi terkait karakteristik pengeluaran perkapita kabupaten/kota di Pulau Jawa.
2. Memperoleh estimasi pengeluaran perkapita suatu daerah berdasarkan citra digital Google Earth menggunakan *Convolutional Neural Network* (CNN) *feature extractor* dan *Support Vector Regression* sebagai alat pemodelan.
3. Mendapatkan pemetaan hasil estimasi pengeluaran perkapita menggunakan *choropleth*.

1.4 Manfaat Penelitian

Informasi tentang distribusi tingkat kesejahteraan yang dihasilkan pada penelitian ini diharapkan dapat dimanfaatkan oleh suatu instansi atau lembaga pemerintah maupun non-pemerintah sebagai dasar kebijakan yang bertujuan untuk mengentaskan kemiskinan. Penelitian ini juga memberikan wawasan tentang penggunaan *Convolutional Neural Network* (CNN) dalam menyelesaikan permasalahan terkait pemrosesan gambar dan penggunaan *Support Vector Machine* (SVR) dalam pemodelan regresi. Selain itu, hasil estimasi akan divisualisasikan dalam *choropleth* (peta tematik) yang memudahkan pembaca dalam memahami distribusi tingkat kesejahteraan.

1.5 Batasan Masalah

Batasan masalah yang digunakan dalam penelitian ini adalah sebagai berikut.

1. Data yang digunakan merupakan hanya data Pulau Jawa yang melingkupi,
   a. Pengeluaran per kapita disesuaikan menurut kabupaten/kota tahun 2018 yang diambil dari BPS tingkat provinsi.
   b. Citra digital satelit yang diambil dari Google Earth yang diperbarui terakhir pada 13 September 2019.
2. Terdapat asumsi bahwa tidak ada perbedaan signifikan antara citra digital satelit dalam waktu satu tahun.
3. Banyaknya kabupaten/kota di Pulau Jawa berdasarkan file *shapefiles* yang digunakan adalah 117 kabupaten/kota.
4. Gambar yang diunduh memiliki resolusi 400 × 400 *pixel* dan mencakup area seluas 1 km^2.
5. Metode CNN yang digunakan adalah arsitektur VGG16 yang difungsikan sebagai *feature extraction* dan tidak ada parameter yang diubah. VGG16 dipilih karena arsitektur tersebut mencapai peringkat lima besar dalam kompetisi ImageNet 2014.
6. Metode SVR menggunakan kernel RBF, karena kernel tersebut memiliki performa yang baik secara umum.

BAB II
TINJAUAN PUSTAKA

Bab ini membahas mengenai statistika deskriptif, citra digital, CNN, *transfer learning*, VGG16, SVR, *k-fold cross validation*, kriteria pemilihan model, pengeluaran perkapita, dan Google Earth.

2.1 Statistika Deskriptif

Statistika deskriptif berhubungan dengan penyajian dan perhimpunan data. Ukuran pemusatan seperti rata-rata dan median, dan ukuran penyebaran seperti varians dan jangkauan (*range*) adalah statistika deskriptif. Jenis statistik ini merangkum informasi kuantitatif. Statistika deskriptif dapat digunakan untuk memberikan suatu acuan dan membandingkan berbagai kelompok data (Lee, Lee, & Lee, 2013).

Statistika deskriptif dapat berupa grafik. Grafik yang digunakan pada penelitian ini adalah *boxplot*, histogram, *scatterplot* dan peta tematik. *Boxplot* menonjolkan informasi dalam bentuk kuartil. Setengah bagian data, dari kuartil pertama hingga kuartil ketiga, diwakili oleh sebuah kotak dengan nilai tengah yang ditandai oleh suatu garis. Satu garis memanjang dari kuartil ketiga hingga nilai maksimum dan satu garis laiinya memanjang dari kuartil pertama hingga nilai minimum. Histogram adalah suatu grafik yang menggambarkan distribusi frekuensi menggunakan persegi panjang alih-alih garis. Persegi panjang tersebut berpusat pada suatu nilai tertentu dan tinggi persegi panjang mewakili frekuensi relatif. *Scatterplot* adalah suatu grafik yang menampilkan hubungan antara dua pengukuran pada sumbu dua dimensi. Pasangan nilai pengukuran tersebut diplot sebagai titik pada grafik tersebut (Johnson & Bhattacharyya, 2009). Peta tematik mudah dipahami karena wilayah yang dipetakan dapat dikenali dan menawarkan ilustrasi yang bagus. Nilai suatu variabel

pada peta dapat diwakili dengan *pie chart*, warna, dan simbol (Loonis & Bellefon, 2018).

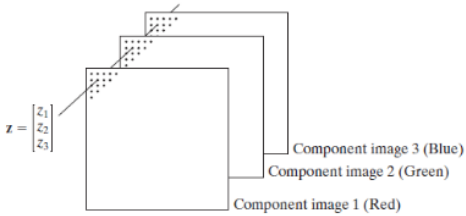
2.2 Citra Digital

Citra digital adalah data digital pada computer yang merepresentasikan sebuah citra. Sebuah citra yang diwakili oleh *f*(*a*, *b*) berbentuk matriks yang terdiri dari A baris dan B kolom, dimana perpotongan antara kolom dan baris inilah yang disebut *picture element*, *image element*, *pels*, atau *pixel*. Citra digital juga dapat digambarkan sebagai fungsi *f*(*a*, *b*) dengan dengan *a* dan *b* merupakan koordinat pada sebuah bidang datar yang merepresentasikan nilai suatu *pixel* dalam dua dimensi (Gonzalez & Woods, 2008).

Dalam penelitian tentang citra digital, terdapat tipe-tipe dasar citra digital yaitu citra RGB dan citra *grayscale*. Citra warna RGB biasanya disebut Citra *True Color* atau citra asli yang ditangkap oleh kamera. Setiap *pixel* pada citra warna RGB memiliki tiga komponen warna, yaitu merah (*Red* = R), hijau (*Green* = G), dan biru (*Blue* = B) dimana setiap komponen warna memiliki nilai minimum 0 dan nilai maksimum 255. Warna pada *pixel* ditentukan dari kombinasi ketiga komponen warna tersebut. Citra *grayscale* merupakan citra yang hanya memiliki satu nilai pada setiap *pixel*-nya, nilai tersebut disebut sebagai derajat keabuan yang memiliki nilai minimum 0 (warna hitam) dan nilai maksimum 255 (warna putih). Pada praktiknya, citra RGB direpresentasikan dalam susunan tiga fungsi yang ditulis seperti persamaan 2.1.

$$f(a,b) = \begin{bmatrix} r(a,b) & g(a,b) & b(a,b) \end{bmatrix} \tag{2.1}$$

Warna suatu gambar adalah sebuah fungsi, namun pada kasus ini, nilai *pixel* pada posisi (*a, b*) bukanlah nilai tunggal, namun merupakan vector yang memiliki tiga tingkat intensitas warna yang berbeda sesuai warnanya seperti Gambar 2.1 (Sewak, Karim, & Pujari, 2018).



Gambar 2.1 Gambar Bercitra RGB sebagai Vektor

Berikut ini adalah contoh warna beserta kode RGB-nya.

Tabel 2.1 Kode Warna RGB

Warna	Nama	Kode RGB
	red	(255, 0, 0)
	green, electric green	(0, 255, 0)
	blue	(0, 0, 255)
	brown	(165, 42, 42)
	gray, trolley grey	(128, 128, 128)
	silver	(192, 192, 192)
	white	(255, 255, 255)
	black	(0, 0, 0)

2.3 *Convolutional Neural Network* (CNN)

CNN didesain untuk mengenali pola visual dari suatu gambar secara langsung dengan proses minimal. CNN pada dasarnya adalah *neural network* dengan banyak *layer* (*multi-layer*

neural network). Tiap neuron menerima beberapa input dan melakukan perhitungan *dot product*. CNN memiliki *loss function* yang terletak di *fully connected layer* terakhir dan memiliki fungsi aktivasi. Nama "*convolutional neural network*" mengindikasikan bahwa *neural network* tersebut menggunakan operasi matematika yang disebut *convolution*. Goodfellow, Bengio, & Courville menyebut CNN sebagai *neural network* yang menggunakan *convolution* sebagai pengganti perkalian matriks umum di setidaknya satu *layer*. CNN mengatur neuron dalam bentuk tiga dimensi, yaitu lebar, panjang, dan *depth*. Secara umum, terdapat tiga *layer* utama dalam CNN, yaitu *convolutional layer*, *pooling layer*, dan *fully-connected layer* (Sewak et al., 2018).

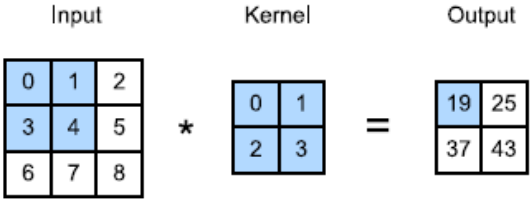
2.3.1 *Convolutional Layer*

Pada *layer* ini terdapat fungsi *convolution* yang berguna untuk mengekstrak suatu nilai dari *input* gambar. *Convolution* didefinisikan sebagai operasi matematika yang menjelaskan cara untuk menggabungkan dua set informasi. *Convolutional layer* membutuhkan *input*, kemudian mengaplikasikan *convolutional kernel*/*filter*, dan memberikan hasil berupa *feature map* sebagai *output*. Operasi *convolution* apabila *input* dan *kernel* berukuran dua dimensi ditulis pada persamaan (2.2).

$$FM_{a,b} = bias + \sum_{c}^{C}\sum_{d}^{D} Z_{c,d} \times X_{a+c-1,b+d-1} \tag{2.2}$$

dimana,

$FM_{a,b}$	:	*feature map* pada *pixel* ke-*a,b*
bias	:	*bias* pada *feature map*
$Z_{c,d}$	:	bobot pada *convolution kernel* ke-*c,d*
a	:	1, 2, …, *A*, dengan *A* merupakan panjang *pixel* pada *feature map*
b	:	1, 2, …, *B*, dengan *B* merupakan lebar *pixel* pada *feature map*
c	:	1, 2, …, *C*, dengan *C* merupakan panjang *pixel* pada *convolution kernel*
d	:	1, 2, …, *D*, dengan *D* merupakan lebar *pixel* pada *convolution kernel*



Gambar 2.2 Ilustrasi *Convolutional Layer*

Contoh perhitungan nilai *pixel output* yang terletak di baris pertama kolom pertama berdasarkan Gambar 2.2 dan persamaan (2.2) adalah sebagai berikut.

$$FM_{a,b} = bias + \sum_{c}^{C}\sum_{d}^{D} Z_{c,d} \times X_{a+c-1,b+d-1}$$

$$FM_{1,1} = bias + \sum_{c=1}^{2}\sum_{d=1}^{2} Z_{c,d} \times X_{1+c-1,1+d-1}$$

$$= 0 + \left(Z_{1,1} \times X_{1+1-1,1+1-1}\right) + \left(Z_{1,2} \times X_{1+1-1,1+2-1}\right) + \left(Z_{2,1} \times X_{1+2-1,1+1-1}\right) + \left(Z_{2,2} \times X_{1+2-1,1+2-1}\right)$$

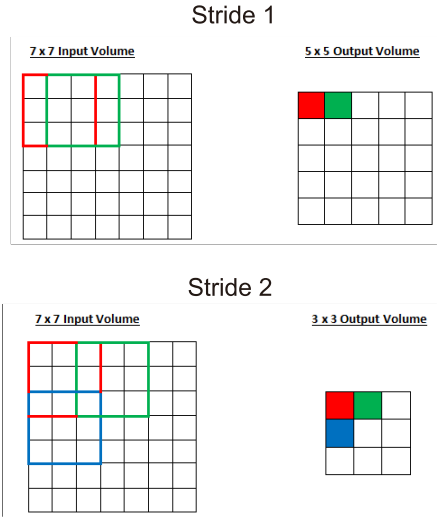
$$= 0 + \left(Z_{1,1} \times X_{1,1}\right) + \left(Z_{1,2} \times X_{1+,2}\right) + \left(Z_{2,1} \times X_{2,1}\right) + \left(Z_{2,2} \times X_{2,2}\right)$$

$$= \left(0 \times 0\right) + \left(1 \times 1\right) + \left(2 \times 3\right) + \left(3 \times 4\right)$$

$$FM_{1,1} = 19$$

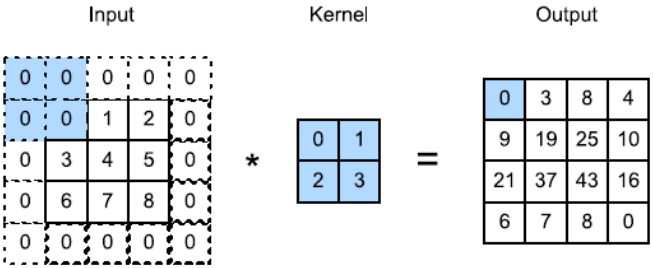
Perlu diketahui bahwa dimensi *output* lebih kecil daripada dimensi *input* seperti Gambar 2.2. Hal ini disebabkan oleh *kernel* yang memiliki lebar lebih dari 1 *pixel*, dan *convolution* hanya bisa dilakukan ketika dimensi *input convolutional* sama dengan dimensi *kernel*. Secara matematis, apabila dimensi *input* $A \times B$ dan dimensi *kernel* $a \times b$, maka dimensi *output* adalah $(A - a + 1) \times (B - b + 1)$. Untuk menghitung semua *output*, *kernel* digeser untuk setiap *pixel* (Zhang, Lipton, Li, & Smola, 2020). Demi meningkatkan efisiensi komputasi, *kernel* digeser lebih dari satu *pixel*. Banyaknya *pixel* yang digeser dalam input disebut *stride*. Secara umum, jika panjang *stride* adalah *sh* dan lebar *stride* adalah *sw*, maka dimensi

output menjadi $\lfloor (n_h - k_h + p_h + s_h)/s_h \rfloor \times \lfloor (n_w - k_w + p_w + s_w)/s_w \rfloor$. Perbedaan *stride* 1 dan *stride* 2 diilustrasikan pada Gambar 2.3.



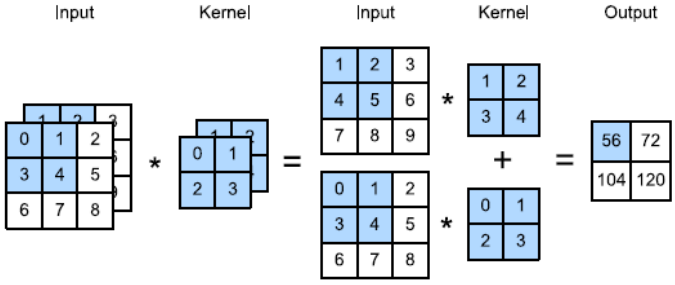
Gambar 2.3 Perbedaan Antara *Stride* 1 dan *Stride* 2

Salah satu masalah yang terjadi akibat pengaplikasian *convolutional layer* adalah kehilangan informasi *pixel* yang terletak di pojok gambar. Solusi yang mudah adalah menambah *pixel* tambahan di pinggir gambar, sehingga memperbesar dimensi gambar. Biasanya, *pixel* tambahan bernilai 0 (Zhang et al., 2020). Secara umum, jika ditambahkan baris *padding* sebanyak p_h (setengahnya untuk input bagian atas dan setengahnya untuk input bawah) dan kolom *padding* sebanyak p_w (setengahnya untuk input bagian kanan dan setengahnya untuk input kiri), maka *output* akan berukuran $(n_h - k_h + p_h + 1) \times (n_w - k_w + p_w + 1)$. Contoh ilustrasi *padding* terdapat pada Gambar 2.4.



Gambar 2.4 Ilustrasi *Padding* Berukuran 1 × 1

Jika input data mengandung banyak *channel*, maka perlu dibentuk *kernel* konvolusi yang memiliki *channel* sebanyak *channel* input sehingga dapat dilakukan perhitungan korelasi silang. Apabila disumsikan bahwa banyaknya *channel* input adalah c_i, maka *kernel* konvolusi harus sebanyak c_i pula. Jika *kernel* berukuran $k_h \times k_w$, maka ketika ukuran *kernel* konvolusi $c_i = 1$, dapat dikatakan bahwa *kernel* konvolusi berdimensi dua dengan panjang dan lebar $k_h \times k_w$. Jika $c_i > 1$, dperlukan *kernel* berukuran $k_h \times k_w$ untuk setiap *channel* input. Gabungan dari c_i akan menghasilkan *kernel* konvolusi berukuran $c_i \times k_h \times k_w$. Karena input dan *kernel* memiliki banyak *channel* yang sama, perhitungan korelasi silang dapat dilakukan. Korelasi silang dihitung dengan menjumlahkan operasi konvolusi untuk setiap channel, sehingga didapatkan output berdimensi dua.



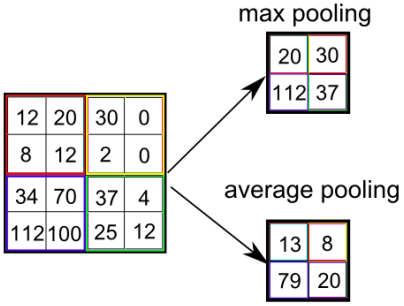
Gambar 2.5 Operasi Konvolusi dengan 2 *Input Channel*

Berdasarkan Gambar 2.5, sel output yang berwarna biru dihitung dengan cara menjumlahkan 2 operasi konvolusi yang berwarna biru, yaitu $(1 \times 1 + 2 \times 2 + 4 \times 3 + 5 \times 4) + (0 \times 0 + 1 \times 1 + 3 \times 2 + 4 \times 3) = 56$.

Terlepas dari banyaknya input *channel*, sejauh ini menghasilkan output dengan *channel* tunggal. Namun, arsitektur CNN yang popular umumnya memiliki output dengan banyak *channel*. Hal ini dilakukan untuk menyimpan lebih banyak informasi ketika *pooling* dilakukan. Untuk mendapatkan output *multi channel* sebanyak c_o, perlu digunakan *kernel* konvolusi dengan dimensi $c_i \times k_h \times k_w$ sebanyak c_o, sehingga dimensi *kernel* konvolusi yang terbentuk menjadi $c_o \times c_i \times k_h \times k_w$ (Zhang et al., 2020).

2.3.2 *Pooling Layer*

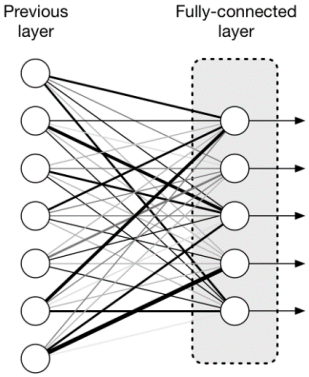
Pooling layer pada umumnya disisipkan diantara *convolutional layer*. *Pooling layer* setelah *convolutional layer* digunakan untuk mengurangi dimensi input. Tidak seperti *convolutional layer*, p*ooling layer* tidak memiliki parameter sehingga operasi *pooling* bersifat deterministik (Zhang et al., 2020). *Pooling layer* menggunakan fungsi max() atau mean() untuk mengurangi input data. Operasi ini dinamakan *max pooling* dan *average pooling*. Pada umumnya, *pooling layer* berukuran 2 × 2 dengan *stride* 2 (Patterson & Gibson, 2017). Contoh *pooling layer* diberikan pada Gambar 2.6.



Gambar 2.6 Ilustrasi *Pooling Layer*

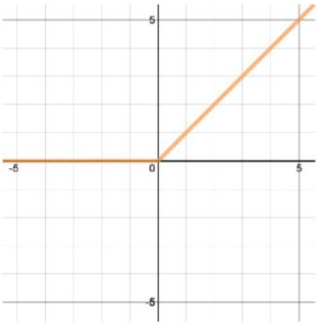
2.3.3 *Fully-connected Layer*

Fully connected layer (juga dikenal sebagai *dense layer*) ditambahkan pada bagian akhir arsitektur CNN. *Fully-connected layer* bekerja mirip seperti *Multi Layer Perceptron* (MLP) (Sewak et al., 2018). *Layer* ini mengkoneksikan semua neuron di satu *layer* ke semua neuron di *layer* lainnya. Pada pemrosesan gambar, data berdimensi dua, yang terdiri dari panjang dan lebar gambar ataupun berdimensi tiga, yang terdiri dari panjang, lebar, dan *depth*. Data ini akan diubah menjadi suatu vector (data berdimensi satu) sebelum masuk ke *fully-connected layer* karena input *layer* ini adalah suatu vector. Hal ini menyebabkan data kehilangan informasi spasialnya dan tidak reversible. Oleh karena itu, *layer* ini hanya diimplementasikan di akhir jaringan. Gambar 2.7 menunjukkan contoh *fully-connected layer*.



Gambar 2.7 llustrasi *Fully-connected Layer*

Layer ini memiliki *hidden layer*, fungsi aktivasi, *output layer*, dan *loss function.* Salah satu fungsi aktivasi adalah *Rectified Linear Unit* (ReLU) yang ditunjukkan pada Gambar 2.8. ReLU menerapkan fungsi max(0, x) terhadap input. Fungsi ReLU akan merubah nilai pixel pada *input* namun tidak merubah dimensi *input* sehingga dimensi *output* berukuran sama dengan *input* (Patterson & Gibson, 2017).



Gambar 2.8 Fungsi Aktivasi ReLU

Persamaan 2.3 merupakan persamaan dalam menghitung nilai *neuron hidden layer*. Kemudian pada *layer* ini diterapkan fungsi aktivasi ReLu.

$$W \times \vec{I} + \vec{B} = \vec{O} \tag{2.3}$$

dengan,

W : matriks bobot (*weights*)

$\vec{I}$: vektor *input*

$\vec{B}$: vektor bias

$\vec{O}$: vaktor *output*

2.4 *Transfer Learning*

Suatu arsitektur CNN yang sudah ada dilatih dapat digunakan pada data yang berbeda. Maka daripada membuat arsitektur CNN baru dan melatihnya dari awal, model CNN yang sudah dilatih tadi diadaptasi untuk data yang baru dengan teknik yang disebut *transfer learning*. *Transfer learning* adalah proses menyalin pengetahuan dari jaringan yang sudah ada ke jaringan yang baru untuk menyelesaikan masalah serupa (Sewak et al., 2018). Teknik ini diimplementasikan untuk mempercepat proses *training* dan meningkatkan performa model (Patterson & Gibson, 2017).

Suatu model yang telah dilatih disebut *pre-trained* model. Contoh *pre-trained* model adalah Xception, VGGNet, ResNet, Inception, MobileNet, DenseNet, NASNet, dan GoogleNet. *Pre-*

trained model tersebut dilatih pada dataset skala besar seperti ImageNet, CIFAR-10, dan MNIST. Dua penggunaan *pre-trained* model yaitu,

1. *Feature extractor*

   Fully-connected layer terakhir dihapus dan menggunakan sisanya sebagai *fixed feature extractor* untuk dataset yang lebih kecil.

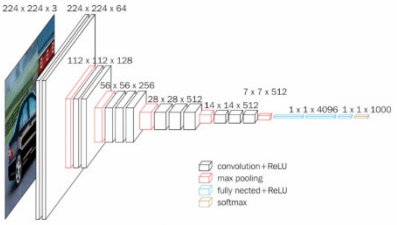
2. *Fine-tuning existing model*

   Melatih ulang *pre-trained* model dengan data baru dan melakukan *tuning* parameter (*weights*) dengan *backpropagation*.

2.5 VGG16

VGGNet diciptakan oleh *Visual Geometric Group* yang berasal dari Universitas Oxford. Model ini mampu mencapai peringkat lima besar pada kompetisi ImageNet 2014 dengan *error* sebesar 7.3% (Simonyan & Zisserman, 2014). VGGNet mempunyai dua versi, yaitu VGG16 dan VGG19. VGG16 adalah salah satu arsitektur CNN yang dianggap terbaik dalam hal *computer vision*.

Nama VGG16 berasal dari banyaknya *weight layer* pada arsitekturnya, yaitu 13 *convolutional layer* dan 3 *fully connected layer* (Sewak et al., 2018). VGGNet menggunakan *convolutional layer* dengan ukuran 3 × 3 dengan *stride* 1 *pixel* dan *padding* 1 dan input gambar RGB 224 × 224. *Pooling layer* pada model ini berukuran 2 × 2 *pixel* dengan *stride* 2. Semua *hidden layer* menggunakan fungsi aktivasi ReLU. VGG16 memiliki parameter sebanyak 138 juta. Arsitektur VGG16 diilustrasikan pada Gambar 2.9.



Gambar 2.9 Arsitektur VGG16

2.6 *Support Vector Regression* (SVR)

SVR merupakan pengembangan dari *Support Vector Machine* (SVM) yang diperkenalkan oleh Vapnik pada tahun 1992 di *Annual Workshop on Computational Learning Theory*. Sebelum itu, SVM dikembangkan untuk pengenalan pola. SVM adalah salah satu metode regresi nonlinier yang memberikan hasil menjanjikan. Ketika algoritma SV digeneralisasikan untuk mengestimasi kasus regresi, perlu suatu cara untuk tidak membuang *support vector*. Vapnik merancang sesuatu yang disebut *ε-incensitive loss function* yang ditulis dalam persamaan (2.3),

$$\left|y - f\left(x\right)\right|_{\varepsilon} = \max\left\{0, \left|y - f\left(x\right)\right| - \varepsilon\right\} \tag{2.4}$$

yang tidak memberikan penalty dibawah nilai $\varepsilon \geq 0$. Dalam pengenalan pola, ketika mengukur *error* yang terjadi, terdapat area besar yang menghasilkan nol *error*, yaitu ketika prediksinya benar dan tidak menyentuh *margin* sehingga tidak memberikan kontribusi *error* terhadap fungsi objektif. Maka dari itu, titik tersebut tidak memberikan informasi tentang hasil keputusan. Suatu *loss function* untuk estimasi regresi harus memiliki zona *incensitive*, maka dari itu digunakan *ε-incensitive loss*.

Algoritma regresi kemudian dikembangkan dengan analogi yang mirip dengan pengenalan pola. Vapnik mengestimasi fungsi linear menggunakan $\left\|w\right\|^2$ *regularizer*, dan menulis ulang semuanya untuk menggeneralisasi kasus nonlinear. Algoritma dasar SVR, yang selanjutnya dinamakan *ε*-SVR, mencari estimasi fungsi linear dengan,

$$f\left(x\right) = \left\langle w, x\right\rangle + b = \sum_{j=i}^{M} w_j x_j + b, y, b \in \mathbb{R}, x, w \in \mathbb{R}^M \tag{2.5}$$

$$f\left(x\right) = \begin{bmatrix} w \\ b \end{bmatrix}^T \begin{bmatrix} x \\ 1 \end{bmatrix} = w^T x + b, x, w \in \mathbb{R}^{M+1} \tag{2.6}$$

berdasarkan data independent dan identic dimana,

$$(x_1, y_1), \ldots, (x_m, y_m) \in \mathrm{H} \times \mathbb{R} \tag{2.7}$$

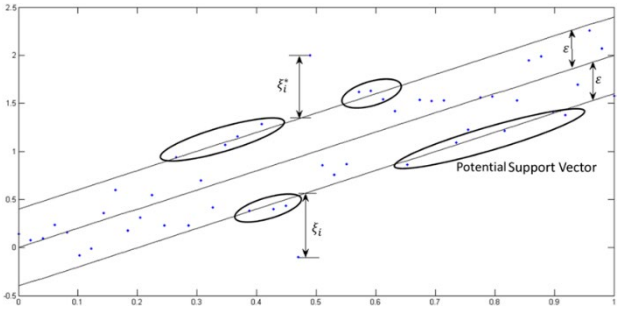
dimana H adalah *dot product space* yang memetakan *input* data. Tujuan dari proses ini adalah untuk menemukan suatu fungsi *f* dengan *error* kecil. Berdasarkan persamaan (2.5), *error* kecil diperoleh dari meninimalkan fungsi *regularized risk*,

$$\frac{1}{2}\|w\|^2 + C \bullet R_{emp}^{\varepsilon}[f] \tag{2.8}$$

dimana,

$$R_{emp}^{\varepsilon}[f] := \frac{1}{m}\sum_{i=1}^{m} |y_i - f(x_i)|_{\varepsilon} \tag{2.9}$$

mengukur kesalahan *training* ε-*incensitive*, dan *C* adalah konstanta yang menentukan *trade-off* dengan kompleksitas penalty $\|w\|^2$. Secara singkat, untuk meminimalkan persamaan (2.7) perlu kontrol terhadap kesalahan *training* dan kompleksitas model (Scholkopf & Smola, 2001).



Gambar 2.10 SVR Linear 1 Dimensi

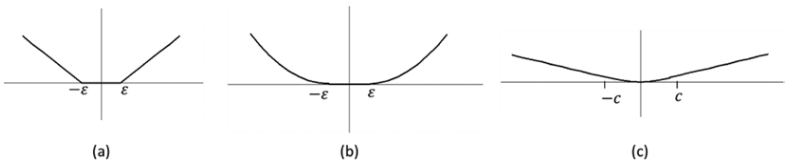
ε-*incensitive loss* yang digunakan memberikan penali kepada nilai prediksi yang berbeda terlalu jauh dari ε dari output yang diinginkan. Nilai ε menunjukkan lebar "*margin*", nilai yang kecil menunjukkan toleransi kecil terhadap *error* dan mempengaruhi banyaknya *support vector* yang menyebabkan

sedikit solusi yang didapatkan. Karena *ε-incensitive loss* kurang sensitive terhadap *noise*, maka *ε-incensitive region* menyebabkan model lebih *robust*. Beberapa *loss function* dapat diadaptasi adalah linear, kuadratik, dan Huber ε sebagaimana dituliskan dalam persamaaan (2.9), (2.10), dan (2.11). Gambar 2.11 menunjukkan bahwa fungsi kesalahan Huber lebih halus daripada kedua fungsi lainnya, namun memberikan penalty terhadap semua penyimpangan dari output yang diinginkan, dengan penalty yang semakin besar seiring meningkatnya *error*. Pemilihan *loss function* dipengaruhi oleh informasi awal mengenai distribusi *noise* pada sampel data., model *sparsity*, dan kompleksitas komputasi saat *training*. *Loss function* seharusnya berbentuk konveks untuk memastikan bahwa optimasi hanya memberikan solusi tunggal yang dapat ditemukan dalam langkah terbatas (Awad & Khanna, 2015).

$$L_\varepsilon\left(y, f\left(x,w\right)\right) = \begin{cases} 0 & \left|y - f\left(x,w\right)\right| \le \varepsilon; \\ \left|y - f\left(x,w\right)\right| - \varepsilon & lainnya, \end{cases} \tag{2.10}$$

$$L_\varepsilon\left(y, f\left(x,w\right)\right) = \begin{cases} 0 & \left|y - f\left(x,w\right)\right| \le \varepsilon; \\ \left(\left|y - f\left(x,w\right)\right| - \varepsilon\right)^2 & lainnya, \end{cases} \tag{2.11}$$

$$L_\varepsilon\left(y, f\left(x,w\right)\right) = \begin{cases} \left|y - f\left(x,w\right)\right| - \dfrac{c^2}{2} & \left|y - f\left(x,w\right)\right| > c; \\ \dfrac{1}{2}\left|y - f\left(x,w\right)\right|^2 & \left|y - f\left(x,w\right)\right| \le c, \end{cases} \tag{2.12}$$



Gambar 2.11 Tipe *Loss Function*: (a) Linear, (b) kuadratik, (c) Huber

Variabel *slack* ξ, ξ^* dapat ditambahkan untuk menjaga model terhadap *outlier*. Variabel ini menentukan banyaknya titik yang dapat ditoleransi diluar *margin* sebagaimana diilustrasikan pada Gambar 2.10. Fungsi tujuan pada optimasi ini ditulis pada persamaan (2.12)

$$\min \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N} \xi_i + \xi_i^* \tag{2.13}$$

dengan fungsi batasan,

$$\begin{aligned} y_i - w^T x_i &\le \varepsilon + \xi_i^* && i = 1,\ldots,N \\ w^T x_i - y_i &\le \varepsilon + \xi_i && i = 1,\ldots,N \\ \xi_i, \xi_i^* &\ge 0 && i = 1,\ldots,N \end{aligned}$$

$$\begin{aligned} L\left(w, \xi^*, \xi, \lambda, \lambda^*, \alpha, \alpha^*\right) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N} \xi_i + \xi_i^* + \\ \sum_{i=1}^{N} \alpha_i^* \left(y_i - w^T x_i - \varepsilon - \xi_i^*\right) + \\ \sum_{i=1}^{N} \alpha_i \left(-y_i + w^T x_i - \varepsilon - \xi_i\right) - \\ \sum_{i=1}^{N} \lambda_i \xi_i + \lambda_i^* \xi_i^* \end{aligned} \tag{2.14}$$

Nilai minimum dari persamaan (2.13) didapatkan dari turunan parsial terhadap variabel-variabel dan menyamakan nilainya dengan nol, berdasarkan kondisi *Karush-Kuhn-Tucker* (KKT). Optimasi *dual form problem* ditulis dengan persamaan (2.15) sebagai berikut,

$$\max_{\alpha,\alpha^*} -\varepsilon\sum_{i=1}^{N_{sv}}\left(\alpha_i+\alpha_i^*\right)+\sum_{i=1}^{N_{sv}}\left(\alpha_i^*-\alpha_i\right)y_i - \frac{1}{2}\sum_{j=1}^{N_{sv}}\sum_{i=1}^{N_{sv}}\left(\alpha_i^*-\alpha_i\right)\left(\alpha_j^*-\alpha_j\right)x_i^T x_j \tag{2.15}$$

dengan batasan,

$$\sum_{i=1}^{N_{SV}}\left(\alpha_i^*-\alpha_i\right)=0, \alpha_i, \alpha_i^* \in [0, C]$$

Untuk fungsi non-linear, data dapat dipetakan dalam dimensi yang lebih tinggi, yang disebut *kernel space*, untuk memperoleh akurasi tinggi menggunakan kernel yang memenuhi kondisi Mercer. Optimasi *dual form problem* dengan penambahan kernel ditulis pada persamaan (2.15).

$$\max_{\alpha,\alpha^*} -\varepsilon\sum_{i=1}^{N_{sv}}\left(\alpha_i+\alpha_i^*\right)+\sum_{i=1}^{N_{sv}}\left(\alpha_i^*-\alpha_i\right)y_i - \frac{1}{2}\sum_{j=1}^{N_{sv}}\sum_{i=1}^{N_{sv}}\left(\alpha_i^*-\alpha_i\right)\left(\alpha_j^*-\alpha_j\right)k\left(x_i x_j\right) \tag{2.16}$$

Persamaan estimasi regresi dibentuk oleh persamaan 2.17.

$$f(x)=\sum_{i=1}^{N}\left(\alpha_i^*-\alpha_i\right)k\left(x_i x_j\right)+b \tag{2.17}$$

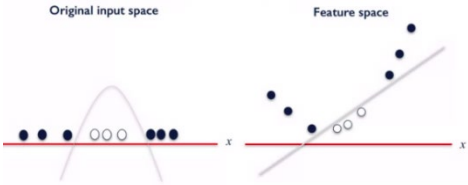
dengan,

i	:	1, 2, …, N dimana N merupakan banyaknya *support vector*
α_i^* dan α_i	:	*Langrange multipliers*
$k\left(x_i x_j\right)$	:	Fungsi kernel
b	:	*intercept*

Metode SVR dapat digunakan pada permasalahan data non-linear dengan menambahkan fungsi kernel. Kernel akan mentransformasi data ke dimensi ruang yang lebih tinggi sehingga

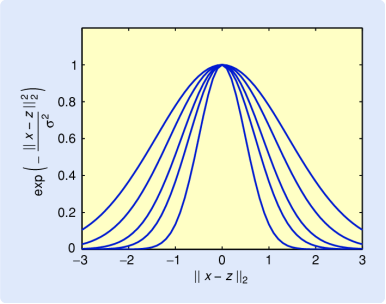
data dapat dipisahkan pada dimensi yang lebih tinggi. Umumnya terdapat tiga fungsi kernel yang digunakan. Gambar 2.14 menampilkan perbandingan garis prediksi tiap kernel. Kernel RBF dapat menangkap pola pada data lebih baik daripada kernel lainnya. Kernel RBF pada umumnya menghasilkan performa yang baik (Makridakis, Spiliotis, & Assimakopoulos, 2018). Berikut ini merupakan fungsi kernel yang umum digunakan:

1. Kernel Linear dengan $k\left(x_i x_j\right) = x_i^T x_j$
2. Kernel Polinomial dengan $k\left(x_i x_j\right) = \left(x_i^T x_j + 1\right)^n$ dimana i,j = 1,2,…, n. Kernel $k(x_i\, x)$ untuk $-\infty < x < +\infty$ diilustrasikan pada Gambar 2.12.

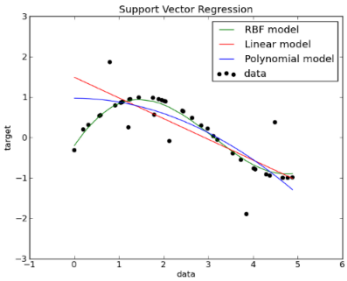


Gambar 2.12 Ilustrasi Kernel Polinomial

3. *Radial Basis Function* (RBF) dengan $k\left(x_i x_j\right) = \exp\left(-\gamma \left\| x_i - x_j \right\|^2\right)$. Jika $x_i = x$ dan $-\infty < z < +\infty$, maka kernel RBF $k(xz)$ dapat digambarkan seperti pada Gambar 2.13.



Gambar 2.13 Ilustrasi Kernel *Radial Basis Function*



Gambar 2.14 Ilustrasi Regresi Satu Dimensi Menggunakan Tiga Kernel SVR

Dalam proses training, SVR menentukan margin hyperplane dengan cara mengestimasi parameter α_i dan b (Härdle, Prastyo, & Hafner, 2014). Performa SVR juga ditentukan oleh parameter lainnya yang dinamakan hyperparameter, yaitu C, ε, dan γ jika menggunakan kernel RBF. Nilai C menentukan besarnya margin, ε adalah jarak minimum antara nilai prediksi dan nilai actual sehingga tidak diberikan penalty, dan γ digunakan untuk mengukur kesamaan antara dua titik observasi.

2.7 *K-fold Cross Validation*

Cross-Validation adalah suatu metode statistika untuk mengevaluasi dan membandingkan suatu algoritma dengan cara membagi data menjadi dua segmen, segmen pertama sebagai data *training* dan segmen kedua sebagai *testing*. Terdapat dua tujuan dalam *cross-validation*. Tujuan pertama dalah untuk mengestimasi performa model dari data yang tersedia menggunakan satu algoritma, atau mengukur generalisasi suatu model. Sedangkan tujuan keduanya adalah untuk membandingkan performa dari dua atau lebih algoritma dan mencari tahu algoritma terbaik untuk data yang ada (Liu & Özsu, 2018).

Bentuk dasar *cross-validation* adalah *k-fold cross-validation*. Dalam *k-fold cross-validation*, data dibagi menjadi *k* kemudian dilakukan iterasi sebanyak *k* kali. Setiap partisi akan berkesempatan untuk menjadi data *testing*. *K-fold cross validation* yang paling luas digunakan sebagai validasi adalah 10-*fold cross*

validation. Dalam 10-*fold cross validation*, data dibagi menjadi 10 partisi dimana 9 partisi akan menjadi data *training* dan partisi lainnya sebagai data *testing*. Data secara umum distratifikasi terlebih dahulu sebelum dipisah menjadi *k* bagian. Stratifikasi adalah proses menata ulang data untuk memastikan setiap bagian (*fold*) merupakan perwakilan yang baik dari keseluruhan data. Ilustrasi 10-*fold cross validation* terdapat pada Gambar 2.15.

Gambar 2.15 Ilustrasi 10-Fold Cross Validation

2.8 Ukuran Kebaikan Model

Berbagai ukuran digunakan untuk membandingkan nilai prediksi dan nilai actual, sehingga kinerja suatu model dapat dihitung. R^2 seringkali digunakan untuk mengukur kebaikan model regresi. R^2 digunakan untuk tujuan *explanatory* dan menjelaskan seberapa baik variabel independent dapat menjelaskan variabilitas dalam variabel dependen. R^2 dihitung dengan persamaan (2.16).

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y}_i)^2} = 1 - \frac{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y}_i)^2} \qquad (2.18)$$

dimana pembilang merupakan *mean square error* (MSE) dan penyebut merupakan varians dari variabel *y*. Semakin tinggi MSE mengakibatkan nilai R^2 semakin rendah dan model memiliki performa buruk (Ngestrini, 2019).

2.9 Pengeluaran Perkapita

Pengeluaran per kapita adalah biaya yang dikeluarkan untuk konsumsi semua anggota rumah tangga selama sebulan dibagi dengan banyaknya anggota rumah tangga. Data ini dapat mengungkapkan tentang pola konsumsi rumah tangga secara umum menggunakan indicator proporsi pengeluaran untuk makanan dan non makanan.

Komposisi pengeluaran rumah tangga dapat dijadikan ukuran untuk menilai tingkat kesejahteraan penduduk, makin rendah persentase pengeluaran untuk makanan terhadap total pengeluaran, makin membaik tingkat kesejahteraan (Badan Pusat Statistik, n.d.-a). Pengeluaran per kapita yang disesuaikan dihitung menggunakan persamaan 2.17 sedangkan pengeluaran per kapita harga konstan dihitung menggunakan persamaan (2.18).

$$Y^{**} = \frac{Y^{*}}{PPP} \tag{2.19}$$

$$Y^{*} = \frac{Y}{IHK} \times 100 \tag{2.20}$$

dimana,

Y^{**} : Pengeluaran perkapita yang disesuaikan
Y^{*} : Pengeluaran perkapita harga konstan

Y : Pengeluaran perkapita setahun
IHK : Indeks Harga Konsumen tahun dasar 2012

2.10 Google Earth

Google Earth pada awalnya dikenal sebagai Earth Viewer sebelum diambil alih oleh Google pada 2004. Google Earth adalah suatu *geobrowser* yang mampu mengakses satelit dan citra angkasa, kedalaman laut, dan data geografis lainnya melalui internet untuk menggambarkan Bumi sebagai globe tiga dimensi atau globe virtual. Google Earth menyediakan fitur pencarian dan

kemampuan untuk menggeser, memperbesar, memutar, dan memiringkan tampilan Bumi. Fitur-fitur yang terdapat pada Google Earth disertakan dalam Google Maps dan Google Earth API yang dapat digunakan untuk menyematkan peta statis atau dinamis dalam suatu situs (Science Education Resource Center, n.d.).

Google Earth dan Google Maps menggunakan satelit yang sama. Citra digital diperbarui secara regular namun kapan pembaruan selanjutnya terjadi tidak dapat diprediksi. Pembaruan terjadi sebulan sekali namun butuh waktu berbulan-bulan untuk diproses, diverifikasi, dan disiapkan sehingga dapat diakses secara public (Google Earth Blog, 2014).

(Halaman ini sengaja dikosongkan)

BAB III
METODOLOGI PENELITIAN

3.1 Sumber Data

Data yang digunakan dalam penelitian ini merupakan data sekunder yang diperoleh dari BPS dan gambar dari Google Maps. Data yang diperoleh dari BPS merupakan data pengeluaran per kapita yang disesuaikan per kabupaten/kota tahun 2018, namun hanya wilayah Jawa saja yang digunakan. Gambar yang diperoleh dari Google Maps sebanyak 229.262 file. Pembaruan terakhir saat data diambil terjadi pada 13 September 2019. Gambar yang diunduh diambil pada *zoom level* 16 dan berukuran 400 × 400 *pixel*. Gambar tersebut akan mencakup area seluas 1 km^2. Untuk menentukan lokasi gambar yang akan diunduh, perlu disediakan file shapefiles yang menentukan batas wilayah. Dalam file shapefiles yang digunakan, terdapat 117 kabupaten/kota untuk wilayah Jawa. Maps Static API aktif diperlukan untuk mengunduh gambar menggunakan link https://maps.googleapis.com/maps /api/staticmap?center=”Garis_Lintang”,”Garis_Bujur”&zoom=16 &size=400x500&maptype=satellite&key=”Kode _API”.

3.2 Variabel Penelitian dan Struktur Data

Variabel dependen yang digunakan dalam penelitian ini adalah pengeluaran perkapita yang disesuaikan, sedangkan variable independen yang digunakan adalah vector RGB tiap *pixel*.

Tabel 3.1 Variabel Penelitian

Simbol	Keterangan	Skala	Satuan
-	Daerah (Kabupaten/Kota)	Nominal	-
Y	Pengeluaran Perkapita	Rasio	Ribu Rupiah
$\overrightarrow{X}_{a,b}$	Vektor citra RGB pada *pixel* baris ke-a dan kolom ke-b	Interval	*Pixel*

Struktur data dari variabel-variabel yang digunakan dalam penelitian ini adalah ditampilkan pada Tabel 3.2. Struktur data yang tertera pada Tabel 3.2 merupakan struktur data sebelum tahap *feature extraction* dengan VGG16.

Tabel 3.2 Struktur Data Penelitian Sebelum *Feature Extraction*

Daerah	Y	**Gambar ke-**	$\vec{X}_{1,1}$	$\cdots$	$\vec{X}_{1,400}$	$\vec{X}_{400,1}$	$\cdots$	$\vec{X}_{400,400}$
		1	$\vec{X}_{1,1,1,1}$	$\cdots$	$\vec{X}_{1,1,1,400}$	$\vec{X}_{1,1,400,1}$	$\cdots$	$\vec{X}_{1,1,400,400}$
		2	$\vec{X}_{1,2,1,1}$	$\cdots$	$\vec{X}_{1,2,1,400}$	$\vec{X}_{1,2,400,1}$	$\cdots$	$\vec{X}_{1,2,400,400}$
1	Y_1	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
		n_1	$\vec{X}_{1,n_1,1,1}$	$\cdots$	$\vec{X}_{1,n_1,1,400}$	$\vec{X}_{1,n_1,400,1}$	$\cdots$	$\vec{X}_{1,n_1,400,400}$
		1	$\vec{X}_{2,1,1,1}$	$\cdots$	$\vec{X}_{2,1,1,400}$	$\vec{X}_{2,1,400,1}$	$\cdots$	$\vec{X}_{2,1,400,400}$
2	Y_2	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
		n_2	$\vec{X}_{2,n_2,1,1}$	$\cdots$	$\vec{X}_{2,n_2,1,400}$	$\vec{X}_{2,n_2,400,1}$	$\cdots$	$\vec{X}_{2,n_2,400,400}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
		1	$\vec{X}_{i,1,1,1}$	$\cdots$	$\vec{X}_{i,1,1,400}$	$\vec{X}_{i,1,400,1}$	$\cdots$	$\vec{X}_{i,1,400,400}$
i	Y_i	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
		n_i	$\vec{X}_{i,n_2,1,1}$	$\cdots$	$\vec{X}_{i,n_2,1,400}$	$\vec{X}_{i,n_2,400,1}$	$\cdots$	$\vec{X}_{i,n_2,400,400}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
117	Y_{117}	n_{117}	$\vec{X}_{117,n_{117},1,1}$	$\cdots$	$\vec{X}_{117,n_{117},1,400}$	$\vec{X}_{117,n_{117},400,1}$	$\cdots$	$\vec{X}_{117,n_{117},400,400}$

Keterangan:

n_i : Banyaknya gambar yang diunduh untuk daerah ke-i, dimana i=1, 2, 3, …, 117

$\vec{X}_{i,n_j,a,b}$: Vektor citra RGB pada daerah ke-i, gambar ke-j, baris ke-a, dan kolom ke-b

Tiap daerah bisa saja memiliki total gambar terunduh yang sama, atau berbeda. Apabila daerah i memiliki gambar sejumlah n_i dan daerah j memiliki gambar sejumlah n_j dimana $i = 1, 2, 3, \cdots, 117$,

$j = 1, 2, 3, \cdots, 117$, dan $i \neq j$, maka bisa jadi kedua daerah tersebut memiliki banyak gambar yang sama $(n_i = n_j)$ atau kedua daerah tersebut memiliki banyak gambar yang berbeda $(n_i \neq n_j)$.

Sedangkan struktur data setelah tahap *pre-processing* (ektraksi fitur VGG16) dicantumkan pada Tabel 3.3. Perubahan struktur data antara Tabel 3.2 dan Tabel 3.3 dapat dilihat pada Lampiran 2.

Tabel 3.3 Struktur Data Penelitian Setelah *Feature Extraction*

Daerah	Y	X_1	X_2	X_3	$\cdots$	X_{4096}
1	Y_1	$x_{1,1}$	$x_{2,1}$	$x_{3,1}$	$\cdots$	$x_{4096,1}$
2	Y_2	$x_{1,2}$	$x_{2,2}$	$x_{3,2}$	$\cdots$	$x_{4096,2}$
3	Y_3	$x_{1,3}$	$x_{2,3}$	$x_{3,3}$	$\cdots$	$x_{4096,3}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
123	Y_{123}	$x_{1,123}$	$x_{2,123}$	$x_{3,123}$	$\cdots$	$x_{4096,123}$

dengan $X_{i,j}$ merupakan rata-rata bobot pada *neuron* ke-*j* daerah ke-*i* hasil ekstraksi fitur VGG16.

3.3 Langkah Analisis

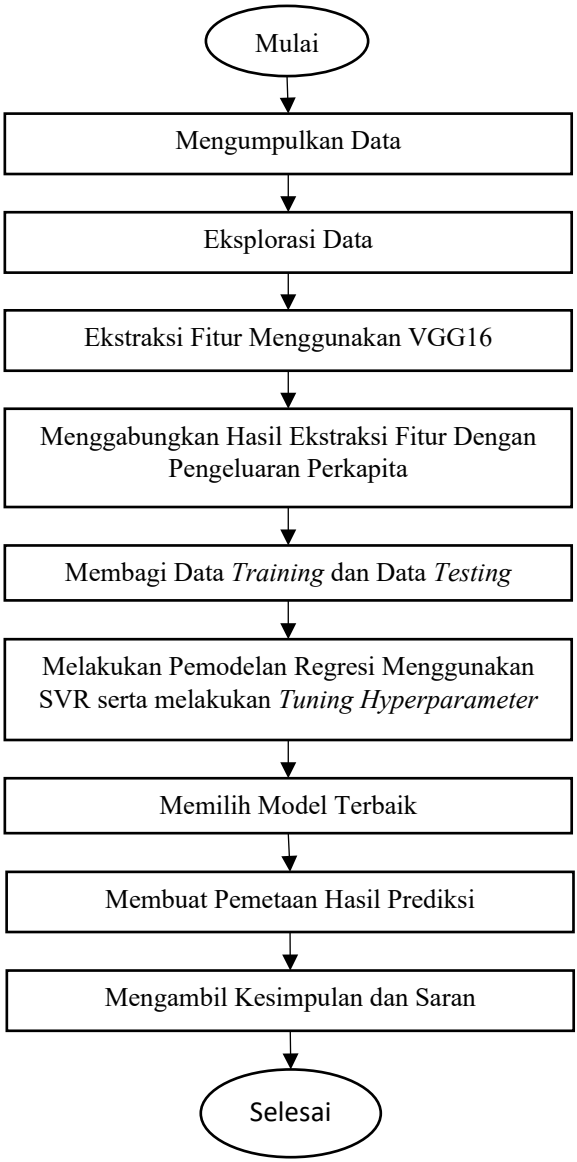
Langkah-langkah awal yang digunakan dalam penelitian ini adalah sebagai berikut:

1. Melakukan eksplorasi data pengeluaran perkapita kabupaten/kota tahun 2018 di Pulau Jawa.
   a. Menampilkan peta tematik untuk mengetahui persebaran pengeluaran perkapita 2018 berdasarkan subbab 2.1.
   b. Memvisualisasikan histogram dan boxplot untuk mengetahui distribusi data pengeluaran perkapita berdasarkan subbab 2.1.
2. Mengestimasi pengeluaran perkapita berdasarkan citra digital Google Earth menggunakan CNN dan SVR.
   a. Mereduksi dimensi gambar yang semula berukuran 400 × 400 *pixel* menjadi 224 × 224 *pixel* sebagai input dari VGG16.

b. Melakukan ekstraksi *feature* pada semua gambar yang telah diunduh menggunakan VGG16 yang dibahas pada subbab 2.4.
c. Menggabungkan data pengeluaran per kapita dengan 4096 variabel hasil ekstraksi berdasarkan koordinat kabupaten/kota.
d. Membagi data *training* dan data *testing* dengan 10-*fold cross validation* yang dibahas pada subbab 2.6.
e. Melakukan pemodelan regresi menggunakan metode SVR dengan kernel RBF dibahas pada subbab 2.5.
f. Melakukan *tuning hyperparameter* SVR menggunakan *10-fold cross validation* yang dibahas pada subbab 2.6.
g. Menampilkan 3D *scatter plot* untuk melihat hubungan antara *hyperparameter* SVR dengan R^2.
h. Memilih model terbaik berdasarkan parameter yang menghasilkan R^2 terkecil berdasarkan subbab 2.7.

3. Memvisualisasikan pemetaan hasil prediksi.
   a. Menampilkan *scatter plot* antara data aktual dan hasil prediksi berdasarkan subbab 2.1.
   b. Menggabungkan hasil prediksi pengeluaran perkapita dan *error* hasil prediksi dengan file *shapefiles*.
   c. Memvisualisasikan hasil prediksi pengeluaran perkapita dan *error* yang dihasilkan menggunakan peta tematik berdasarkan subbab 2.1.
   d. Menginterpretasikan hasil analisis, menarik kesimpulan dan saran berdasarkan hasil analisis.

Langkah-langkah di atas dapat digambarkan dengan diagram alir yang disajikan pada 3.1 gambar dibawah ini.



Gambar 3.1 Diagram Alir Penelitian

(Halaman ini sengaja dikosongkan)

BAB IV
ANALISIS DAN PEMBAHASAN

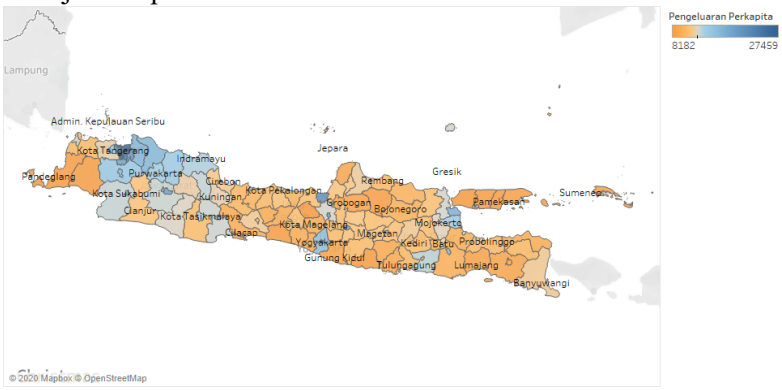
Pada penelitian ini membahas karakteristik pengeluaran perkapita di Pulau Jawa tahun 2018 dan serta memprediksinya berdasarkan citra digital Google Earth. Metode yang digunakan adalah *Feature Extraction* VGG16 dan *Support Vector Regression* (SVR) dengan 10-*folds Cross Validation*. Model dengan kombinasi *hyperparameter* yang menghasilkan R^2 paling tinggi akan digunakan untuk membuat prediksi pengeluaran perkapita.

4.1 Karakteristik Pengeluaran Perkapita Kabupaten/Kota di Pulau Jawa

Data kuantitatif yang dapat diandalkan tentang ekonomi suatu negara sangat diperlukan dalam oleh pemerintah dan para peneliti. Distribusi geografis tentang kesejahteraan penduduk digunakan sebagai dasar pengambilan keputusan tentang alokasi sumber daya dan sebagai faktor yang menentukan pertumbuhan ekonomi. Di Indonesia, mendapatkan data spasial pada tingkat area kecil merupakan tantangan bagi para pembuat kebijakan dan peneliti, karena data tersebut seringkali tidak tersedia secara publik. Data Susenas yang diambil pada tingkat rumah tangga diagregasi oleh BPS provinsi setempat pada tingkat kabupaten/kota. Salah satu produk Susenas adalah pengeluaran penduduk.

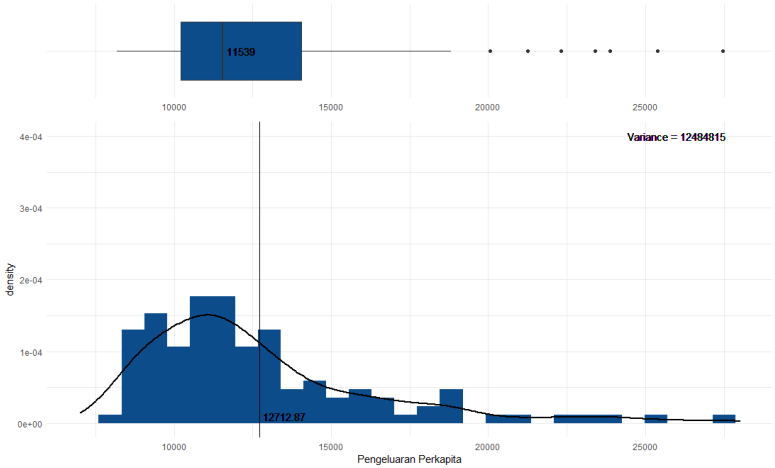
Pengeluaran perkapita menunjukkan rata-rata besarnya biaya konsumsi rumah tangga suatu daerah selama sebulan, baik konsumsi makanan maupun non-makanan. Provinsi DKI Jakarta merupakan provinsi tertinggi di Pulau Jawa dengan pengeluaran rumah tangga tertinggi, dengan rata-rata pengeluaran diatas 22 juta per tahun per orang, kecuali Daerah Administrasi Kepulauan Seribu. Di provinsi tersebut, Kota Jakarta Selatan merupakan kota dengan biaya hidup paling tinggi. Sedangkan daerah dengan

pengeluaran konsumsi paling rendah adalah Kabupaten Temanggung di Jawa Tengah dengan pengeluaran perkapita sekitar 8 juta per tahun per orang. Persebaran pengeluaran perkapita ditunjukkan pada Gambar 4.1.



Gambar 4.1 Persebaran Pengeluaran Perkapita 2018 di Pulau Jawa

Warna oranye pada peta menunjukkan pengeluaran perkapita yang berada dibawah rata-rata pengeluaran perkapita, sedangkan warna biru menandakan pengeluaran perkapita yang lebih tinggi dari nilai rata-rata. Semakin kuat warna oranye suatu daerah, maka biaya hidup yang dikeluarkan selama setahun semakin rendah, begitu pula sebaliknya jika berwarna biru. Rata-rata penduduk di mayoritas kabupetn/kota di Jawa Tengah, Daerah Istimewa Yogyakarta, dan Jawa Timur mengeluarkan uang konsumsi kurang dari 13 juta per tahun per orang. Di Provinsi Jawa Timur, hanya penduduk Surabaya, Sidoarjo, Kota Malang, Kota Blitar, dan Kota Madiun yang mengeluarkan biaya konsumsi lebih dari 13 juta per tahun. Sedangkan di Provinsi Jawa Tengah dan Yogyakarta, Surakarta (Solo), Salatiga, Kota Semarang, Kota Tegal, Sleman, dan Kota Yogyakarta memiliki pengeluaran perkapita diatas 13 juta. Semua penduduk Jawa Barat rata-rata menggunakan uang untuk kebutuhan konsumsi rumah tangga diatas 10 juta per tahun.



Gambar 4.2 *Boxplot* dan *Histogram* Pengeluaran Perkapita 2018 (ribu/tahun/orang)

Gambar 4.2 menunjukkan bahwa data pengeluaran perkapita di Pulau Jawa bersifat *skew* kanan, sehingga nilai rata-ratanya lebih besar daripada nilai mediannya. Rata-rata pengeluaran perkapita sebesar 12.712.870 sedangkan nilai tengahnya sebesar 11.539.000. Nilai varians dari pengeluaran perkapita cukup rendah karena nilai koefisien variansnya, yaitu 0,298 kurang dari 1. *Boxplot* menunjukkan adanya 7 titik *outlier* pada data. Kabupaten/kota dengan 7 nilai tertinggi tersebut adalah Jakarta Selatan, Jakarta Utara, Jakarta Timur, Jakarta Pusat, Jakarta Barat, Kota Semarang, dan Kota Bekasi. Ketujuh kota tersebut memiliki pengeluaran perkapita diatas 20 juta pada tahun 2018.

4.2 Prediksi Pengeluaran Perkapita Kabupaten/Kota di Pulau Jawa

Subbab ini menjelaskan mengenai pemodelan pengeluaran perkapita yang disesuaikan berdasarkan citra satelit menggunakan metode *convolutional neural network* dan *support vector regression*. Namun sebelum memodelkan menggunakan SVR, data

citra satelit harus diubah dulu sehingga dapat digunakan sebagai input *convolutional neural network*. Tahapan ini disebut *pre-processing* data. Tahapan setelahnya adalah pemodelan, dan pemilihan model terbaik.

4.2.1 *Pre-Processing* Citra Digital Satelit

Arsitektur CNN yang digunakan, VGG16 menerima input gambar dengan dimensi 224 × 224 × 3. Namun citra yang diunduh berdimensi 400 × 400 × 3. Oleh karena itu, dimensi citra satelit perlu diubah. VGG16 yang digunakan sebagai *feature extractor* kehilangan *fully-connected layer* terakhir. Tabel 4.1 menampilkan arsitektur VGG16 sebagai *feature extractor*.

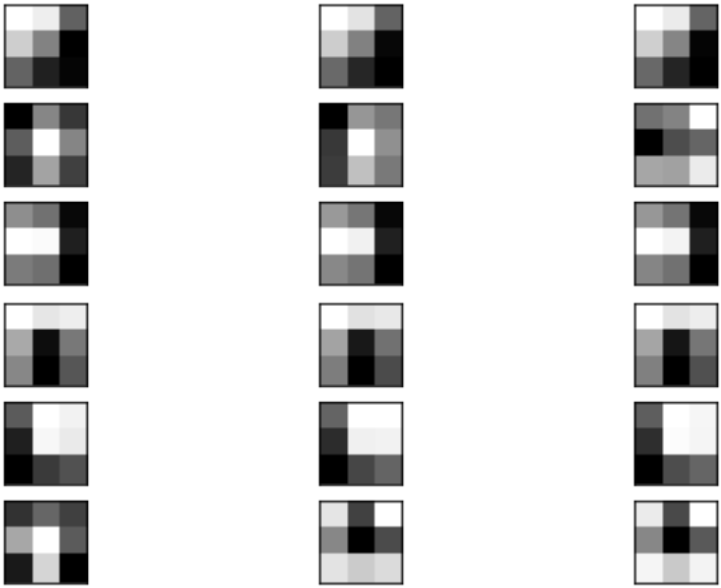
Tabel 4.1 Arsitektur VGG16 Sebagai Feature Extractor

Layer	Tipe	Dimensi Output
input_2	Input Layer	(None, 224, 224, 3)
block1_conv1	Conv2D	(None, 224, 224, 64)
block1_conv2	Conv2D	(None, 224, 224, 64)
block1_pool	MaxPooling2D	(None, 112, 112, 64)
block2_conv1	Conv2D	(None, 112, 112, 128)
block2_conv2	Conv2D	(None, 112, 112, 128)
block2_pool	MaxPooling2D	(None, 56, 56, 128)
block3_conv1	Conv2D	(None, 56, 56, 256)
block3_conv2	Conv2D	(None, 56, 56, 256)
block3_conv3	Conv2D	(None, 56, 56, 256)
block3_pool	MaxPooling2D	(None, 28, 28, 256)
block4_conv1	Conv2D	(None, 28, 28, 512)
block4_conv2	Conv2D	(None, 28, 28, 512)
block4_conv3	Conv2D	(None, 28, 28, 512)
block4_pool	MaxPooling2D	(None, 14, 14, 512)
block5_conv1	Conv2D	(None, 14, 14, 512)
block5_conv2	Conv2D	(None, 14, 14, 512)
block5_conv3	Conv2D	(None, 14, 14, 512)

Tabel 4.1 Arsitektur VGG16 Sebagai Feature Extractor (Lanjutan)

Layer	Tipe	Dimensi Output
block5_pool	MaxPooling2D	(None, 7, 7, 512)
flatten	Flatten	(None, 25088)
fc1	Dense	(None, 4096)
fc2	Dense	(None, 4096)

Setelah dimensi input disesuaikan, data akan diolah oleh konvolusi pertama pada blok pertama. Filter konvolusi pada *layer* ini mengubah citra gambar dan *channel* input gambar. Terdapat 64 × 3 filter konvolusi pada layer ini dan tiap filter konvolusi terdiri dari 3 *channel*, yang bertanggung jawab atas *channel* R, *channel* G, dan *channel* B. Ilustrasi *filter* konvolusi ditampilkan pada Gambar 4.3.

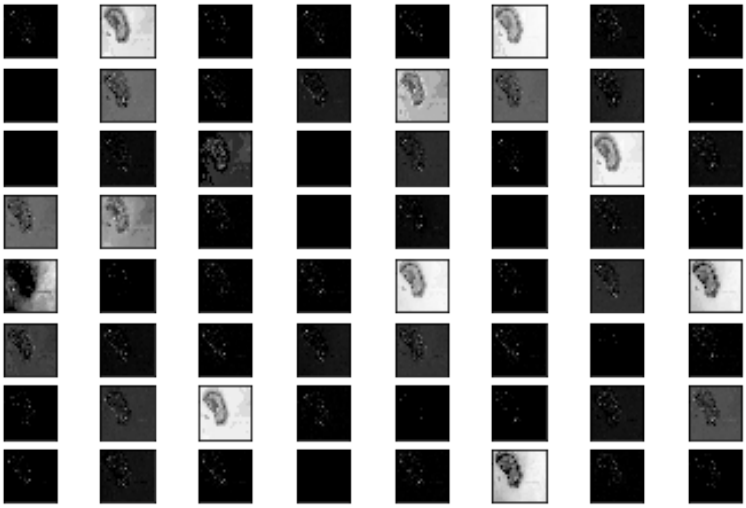


Gambar 4.3 Visualisasi 6 *Filter* Konvolusi Pertama Pada *Layer* Pertama

Gambar 4.3 menunjukkan visualisasi bobot (*weights* atau parameter) yang dinormalisasi pada tiap filter. Terdapat 6 baris yang terdiri dari 3 gambar, atau 18 gambar, satu baris untuk tiap

filter, dan satu kolom untuk tiap *channel*. Filter kedua dan filter keenam memiliki bobot yang berbeda pada ketiga *channel*-nya, sedangkan keempat filter lainnya memiliki bobot yang sama. Tiap gambar berukuran 3 × 3 *pixel*. *Pixel* yang berwarna gelap menunjukkan bobot yang kecil sehingga memblokir cahaya sedangkan *pixel* yang terang mewakili bobot yang besar sehingga merangsang cahaya. Sehingga filter pertama pada baris pertama akan mendeteksi gradien dari cahaya di kiri atas menjadi gelap di kanan bawah.

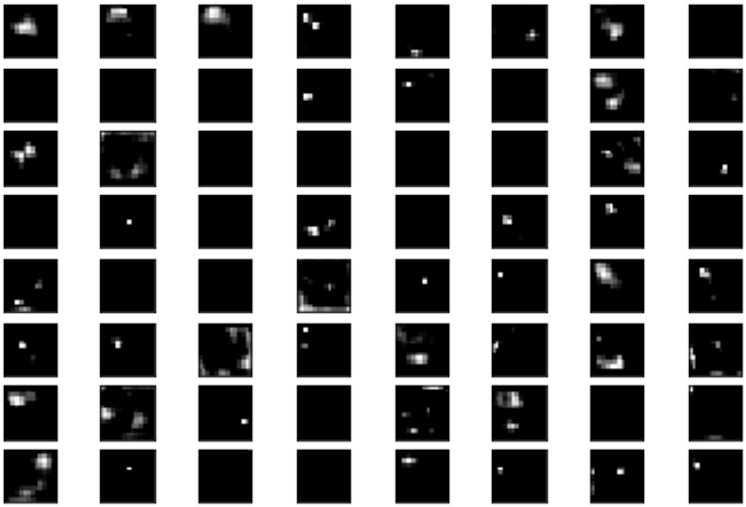
Hasil olahan input gambar yang ditangkap oleh filter disebut *feature maps*. Sesuai dengan model VGG16 pada Tabel 4.1, satu input gambar akan menghasilkan 64 *feature maps* setelah proses konvolusi pada *layer* conv1_block1 karena terdapat 64 filter pada tahap tersebut. Ke-64 *feature maps* divisualisasikan seperti pada Gambar 4.4.



Gambar 4.4 Visualisasi 64 *Feature Maps* Pertama Pada *Layer* block1_conv1

Hasil penerapan filter pada *layer* pertama menghasilkan versi gambar berbeda dengan beberapa fitur yang ditonjolkan. Ada

filter yang menonjolkan *background*, adapula yang menonjolkan *foreground*. *Feature maps* pada tahap terakhir, yaitu block5_conv3 atau filter konvolusi ketiga pada blok kelima, menghasilkan gambar yang tidak sedetail tahap conv1_block1 sebagaimana ditunjukkan pada Gambar 4.5. Gambar 4.5 hanya memvisualisasikan 64 *feature maps* dari 512 *feature maps* yang tersedia. Pada tahap ini, sulit untuk menafsirkan apa yang dihasilkan oleh *feature maps* tersebut. Namun pola ini memang diharapkan terjadi, karena model mengekstrak fitur dari gambar menjadi konsep yang lebih umum sehingga dapat digunakan dalam pemodelan.



Gambar 4.5 Visualisasi 64 *Feature Maps* Pertama Pada Layer block5_conv3

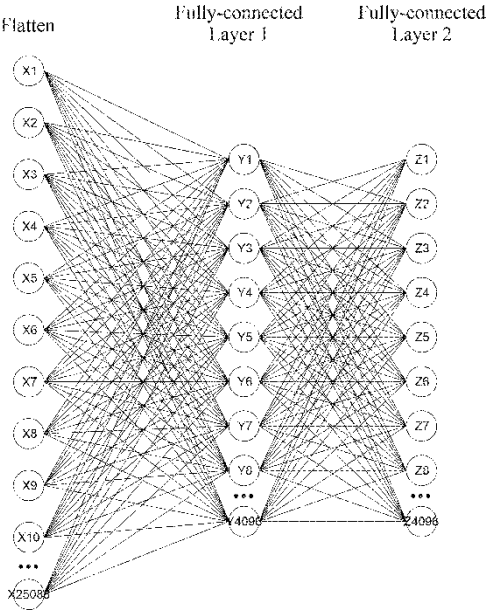
Tiap *feature maps* pada tahap block5_conv3 memiliki ukuran 14 × 14 *pixel*. Banyak bagian dari *feature maps* tersebut yang berwarna hitam sehingga bernilai nol apabila output *feature maps* dikeluarkan dalam bentuk angka, karena warna hitam dalam kode warna *grayscale* bernilai 0. Tabel 4.2 menunjukkan nilai tersebut dalam *grayscale* untuk gambar pada baris pertama kolom

pertama pada Gambar 4.5. Angka-angka bukan nol yang tertera pada Tabel 4.2 sesuai dengan gambar yang dimaksud dimana terlihat warna putih yang mengumpul di tengah gambar.

Gambar 4.6 *Output Feature Maps* Pertama pada *Layer* block5_conv3

0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	8	8	0	0	0	0	0	0	0
0	0	0	0	5	25	29	12	0	0	0	0	0	0
0	0	7	18	34	47	48	27	0	0	0	0	0	0
0	0	12	40	62	65	59	40	14	0	0	0	0	0
0	0	0	25	48	43	41	37	22	0	0	0	0	0
0	0	0	0	0	0	8	15	11	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0

Matriks pada Tabel 4.2 akan diubah menjadi suatu vektor pada *layer* flatten setelah direduksi ukurannya dengan proses *pooling* pada *layer* block5_pool. Karena ukuran output pada tahap block5_pool adalah $7 \times 7 \times 512$, maka panjang vektor output pada *layer* flatten adalah 25088. Vektor tersebut akan melalui proses *fully-connected layer* dua kali, masing-masing layer menghasilkan output vektor berukuran 1×4096. Proses ini diilustrasikan oleh Gambar 4.6. Vektor output pada *fully-connected layer* inilah yang akan menjadi variabel prediktor dalam pemodelan pengeluaran perkapita.



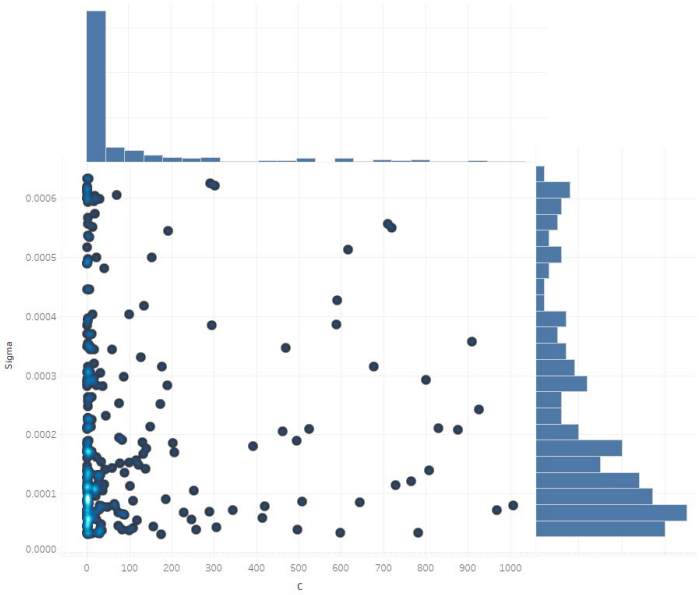
Gambar 4.7 Ilustrasi *Fully-connected Layer* VGG16

Karena input *layer* flatten banyak yang bernilai nol, maka wajar apabila output *fully-connected layer* kedua juga bernilai nol. Apabila terdapat satu variabel dari 4096 variabel hasil output *fully-connected layer* yang bernilai nol untuk semua observasinya, maka variabel tersebut tidak layak untuk dijadikan prediktor. Apabila persamaan $y = f(x)$ dimana x bernilai konstan atau nol untuk semua observasinya, maka nilai y akan sama juga untuk seluruh observasi. Dalam kasus ini, terdapat 19 variabel yang bernilai konstan (nol). Hal ini akan mempengaruhi performa model sehingga tidak perlu diikutsertakan dalam pemodelan.

4.2.2 Pemodelan SVR Berdasarkan *Output* VGG16

K-fold cross-validation pada pemodelan ini digunakan untuk melakukan *tuning hyperparameter* SVR, yaitu parameter *sigma* dan *cost*. Jumlah *folds* yang digunakan adalah 10. Nilai kedua *hyperparameter* tersebut dibangkitkan secara random oleh R

sebanyak 250 kombinasi. Gambar 4.8 menampilkan visualisasi kombinasi *hyperparameter* yang dibangkitkan secara acak.



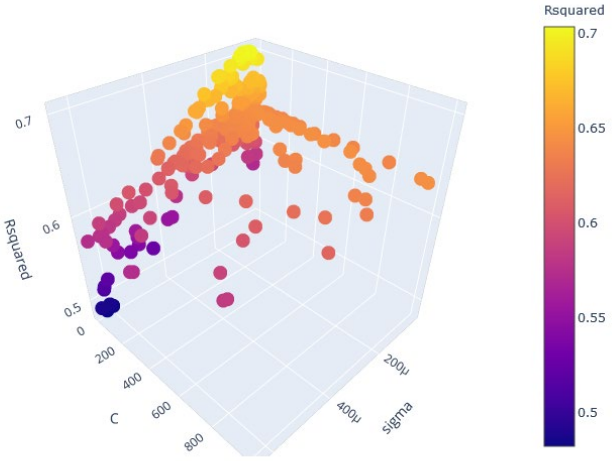
Gambar 4.8 *Density Scatter Plot Hyperparameter* Hasil Bangkitan

Density scatter plot memvisualisasikan pola pada *scatter plot* yang memiliki banyak titik yang tumpang tindih. Semakin banyak titik-titik yang tumpang tindih, warna densitasnya akan semakin kuning. Dua per tiga *hyperparameter cost* yang dibangkitkan berada bernilai antara 0 – 44, sedangkan *hyperparameter sigma* yang paling banyak dibangkitkan bernilai antara $2{,}8\times10^{-5}$ - $8{,}1\times10^{-5}$. Oleh karena itu, densitas tertinggi berada diantara kedua nilai interval tersebut. 10 kombinasi parameter yang menghasilkan rata-rata performa SVR terbaik dalam sepuluh *folds* ditampilkan pada Tabel 4.3.

Tabel 4.2 Kombinasi 10 Parameter Terbaik Berdasarkan R^2 *Testing*

Peringkat	sigma	C	R^2 *Training*	R^2 *Testing*
1	$8{,}24\times10^{-5}$	2,264	0,886	0,703
2	$8{,}60\times10^{-4}$	2,062	0,882	0,703
3	$6{,}15\times10^{-5}$	2,377	0,861	0,703
4	$9{,}36\times10^{-5}$	1,746	0,874	0,701
5	$5{,}41\times10^{-5}$	2,469	0,851	0,701
6	$7{,}26\times10^{-5}$	1,901	0,856	0,701
7	$9{,}61\times10^{-5}$	1,370	0,849	0,699
8	$9{,}91\times10^{-5}$	1,297	0,846	0,698
9	$9{,}43\times10^{-5}$	1,278	0,838	0,697
10	$5{,}48\times10^{-5}$	5,514	0,925	0,697

Nilai R^2 pada Tabel 4.3 diperoleh dengan merata-rata performa model dengan parameter yang sama namun dengan dataset (*fold*) yang berbeda. R^2 *training* dihitung dengan merata-rata performa model pada data *training*, sedangkan R^2 *testing* dihitung dengan merata-rata performa model pada data *testing*. Dari 10 *folds* yang dihasilkan, 9 diantaranya merupakan data *training* dan sisanya merupakan data *testing*. Dari 250 kombinasi parameter, didapatkan bahwa parameter yang menghasilkan R^2 terbaik secara keseluruhan adalah *hyperparameter sigma* $8{,}24\times10^{-5}$ dan *cost* 2,264. Model dengan parameter tersebut dapat menjelaskan variabilitas pengeluaran perkapita berdasarkan citra digital satelit sebesar 88,6% pada data *training* dan 70,3% pada data *testing*. Walaupun R^2 *training* yang dicapai bukan nilai yang paling tinggi, namun, kombinasi parameter yang diperoleh dengan cara *random* ini merupakan kombinasi parameter yang paling stabil performanya (*robust*) terhadap data baru. Kedua parameter ini akan digunakan untuk memprediksi keseluruhan dataset. Untuk melihat hubungan antara nilai kedua parameter dan kinerja modelnya diperlukan suatu grafik 3 dimensi. Gambar 4.8 menunjukkan hubungan tersebut.

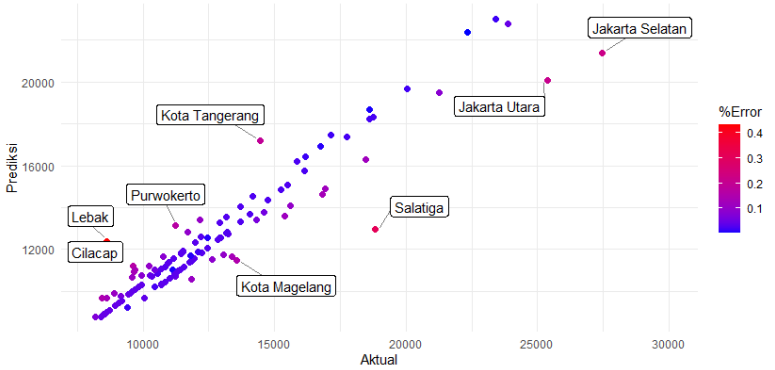


Gambar 4.9 3D *Scatter Plot* Hubungan Antara H*yperparameter* dan R^2

Berdasarkan Gambar 4.9, terdapat beberapa nilai kombinasi parameter dimana nilai parameter *cost* seolah-olah tidak berpengaruh terhadap kinerja model, yaitu ketika nilai R^2-nya berada di sekitar 0,64. *Hyperparameter sigma* yang bernilai kecil cenderung menghasilkan model dengan performa tinggi daripada nilai *sigma* yang besar.

4.3 Pemetaan Hasil Estimasi Pengeluaran Perkapita Kabupaten/Kota di Pulau Jawa

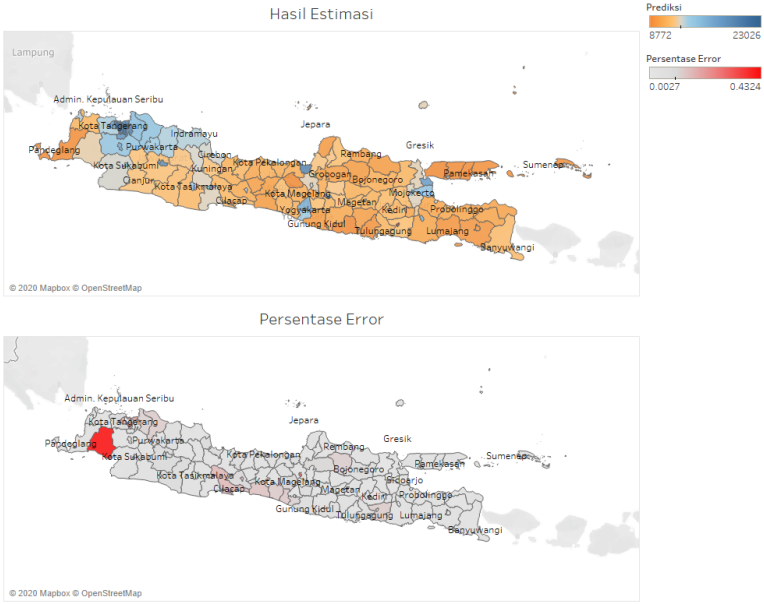
Prediksi pengeluaran perkapita kabupaten/kota di Pulau Jawa tahun 2018 dilakukan menggunakan SVR dengan *hyperparameter sigma* $8{,}24\times10^{-5}$ dan *cost* 2,264 menggunakan data dengan 117 kabupaten/kota. Perbandingan nilai actual pengeluaran perkapita dan estimasinya diilustrasikan pada Gambar 4.10.



Gambar 4.10 Perbandingan Nilai Aktual dan Estimasi Pengeluaran Perkapita

Model menghasilkan estimasi pengeluaran perkapita kabupaten/kota di Pulau Jawa tahun 2018 dengan cukup bagus, karena titik-titik pada Gambar 4.10 membentuk garis lurus, namun ada beberapa observasi yang memiliki persentase selisih antara nilai aktual dan estimasinya diatas 15%, yaitu Lebak, Salatiga, Jakarta Selatan, Jakarta Utara, Kota Tangerang, Purwokerto, Cilacap, dan Kota Magelang. Secara mengejutkan, hanya 3 diantara 7 observasi *outlier* yang memiliki persentase *error* lebih dari 15%, yaitu Jakarta Selatan, Jakarta Utara. Hal ini mengindikasikan bahwa model yang diciptakan dapat mengatasi sebagian *outlier*. Namun penyumbang persentase *error* terbesar berasal dari Lebak, yang merupakan salah satu daerah dengan 25% pengeluaran perkapita terendah di Pulau Jawa. Selain Lebak, daerah yang termasuk dalam 25% pengeluaran perkapita paling rendah adalah Cilacap. Berdasarkan data aktual, terdapat 29 daerah yang termasuk dalam 25% daerah dengan peneluaran terendah, dengan pengeluaran minimum sebesar 8,2 juta bagi penduduk Temanggung dan pengeluaran maksimum sebesar 10 juta untuk masyarakat Tuban. Secara keseluruhan, model ini memiliki rata-rata persentase *error* (*mean absolute percentage error*) 6%. Secara khusus, apabila hanya 25% kelompok terendah (kelompok dengan pengeluaran maksimal 10 juta) yang diukur, maka model ini memiliki rata-rata persentase *error* yang lebih tinggi daripada rata-

rata persentase *error* secara umum, yaitu sebesar hampir 8%. Peta hasil estimasi model yang diilustrasikan pada Gambar 4.11 kurang lebih memiliki warna yang hampir sama dengan peta aktual persebaran pengeluaran perkapita.



Gambar 4.11 Hasil Prediksi dan Persentase *Error* Model

Berdasarkan nilai estimasi, Bondowoso, Pemalang, Kabupaten Probolinggo, Rembang, Wonogiri, dan Banyumas termasuk daerah dengan 25% pengeluaran perkapita terendah, sedangkan menurut nilai aktual, Lebak, Kabupaten Magelang, Cilacap, Purworejo, Kulon Progo, dan Bojonegoro termasuk daerah dengan 25% pengeluaran terendah. Ke-12 daerah tersebut ditandai *font* berwarna merah pada Tabel 4.3. Hasil estimasi ini memberikan 6 daerah berbeda dengan nilai aktual pengeluaran perkapita berdasarkan nilai kuartil pertamanya. Dengan kata lain, terdapat 23 kabupaten/kota hasil estimasi yang sesuai dengan 25% kelompok pengeluaran perkapita terendah. Sehingga tingkat kesesuaian model adalah 23/29 atau sekitar 79,3%. Selain itu,

urutan kabupaten/kota dengan pengeluaran terendah antara nilai aktual dan nilai estimasi juga sedikit berbeda.

Tabel 4.3 29 Kabupaten/Kota dengan Pengeluaran Terendah

No	Aktual	Prediksi	No	Aktual	Prediksi
1	Temanggung	Temanggung	16	Jepara	Gunung Kidul
2	Bangkalan	Bangkalan	17	Trenggalek	Ponorogo
3	Kebumen	Pacitan	18	Ponorogo	Grobogan
4	Pacitan	Pamekasan	19	Purbalingga	Purbalingga
5	Pamekasan	Sampang	20	Magelang	Banjarnegara
6	Sampang	Pandeglang	21	Banjarnegara	Situbondo
7	Blora	Sumenep	22	Cilacap	Bondowoso
8	Pandeglang	Trenggalek	23	Purworejo	Malang
9	Lebak	Lumajang	24	Situbondo	Pasuruan
10	Sumenep	Kabupaten Jember	25	Kulon Progo	Probolinggo
11	Grobogan	Kota Jember	26	Malang	Rembang
12	Lumajang	Jepara	27	Bojonegoro	Pemalang
13	Kabupaten Jember	Tuban	28	Pasuruan	Wonogiri
14	Kota Jember	Kebumen	29	Tuban	Banyumas
15	Gunung Kidul	Blora			

(Halaman ini sengaja dikosongkan)

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan analisis dan pembahasan yang telah dilakukan pada bab 4, maka diperoleh kesimpulan sebagai berikut.

1. Jakarta Selatan merupakan kota dengan biaya konsumsi rumah tangga tertinggi. Daerah yang memiliki pengeluaran perkapita paling rendah adalah Kabupaten Temanggung dengan biaya per tahunnya hanya 8 juta. Mayoritas penduduk Jawa Tengah, D.I. Yogyakarta, dan Jawa Timur mengeluarkan uang kurang dari 13 juta per tahun. Distribusi pengeluaran perkapita tidak simetris dengan rata-rata 12.712.870 dan nilai tengah 11.539.000.
2. Pemodelan SVR menggunakan 10-*folds cross validation* menghasilkan kombinasi *hyperparameter* terbaik yaitu *sigma* $8{,}24\times10^{-5}$ dan *cost* 2,264. Kombinasi ini menghasilkan model dengan performa paling stabil walaupun R^2 *testing* yang dicapai hanya 0,703.
3. SVR dengan parameter terbaik menghasilkan prediksi yang hampir sama, dengan rata-rata persentase *error* (*mean absolute percentage error*) 6%. Hasil estimasi memberikan tingkat kesesuaian sebesar 79,3% terhadap 25% kelompok pengeluaran perkapita terendah (pengeluaran maksimum 10 juta) berdasarkan data aktual.

5.2 Saran

Berdasarkan kesimpulan yang diperoleh, dapat dirumuskan saran sebagai berikut.

1. Metode ini menunjukkan adanya potensi untuk memprediksi pengeluaran perkapita karena terdapat kesesuaian 25% kelompok pengeluaran perkapita terendah antara nilai aktual dan nilai prediksi. Apabila penelitian serupa dikembangkan, penelitian seperti ini diharapkan dapat menjadi salah satu dasar rekomendasi bagi pemerintah terkait penanganan kemiskinan.

2. Terdapat perbedaan kabupaten/kota antara data *file shapefiles* dan BPS, seperti tidak ada Bandung Barat, Pangandaran pada *file shapefiles* dan tidak ada Purwokerto pada data BPS. Sehingga diperlukan *file shapefiles* yang resmi dan paling mutakhir sehingga saling sinkron.
3. Menambah citra digital malam hari sebagai input sehingga terdapat informasi saat siang dan malam hari.
4. Menghapus gambar yang tidak diperlukan seperti permukaan laut karena tidak ada penduduk yang tinggal di permukaan laut sehingga tidak dapat dijadikan acuan dalam mengestimasi pengeluaran penduduk.
5. Menggunakan data ekonomi spasial pada tingkat yang lebih rendah. Data yang paling bermanfaat adalah data pada tingkat rumah tangga.
6. Membandingkan model tanpa *feature extraction* sehingga dapat diketahui seberapa bermanfaat tahap *feature extraction*.
7. Meng-*update* ulang parameter CNN (*fine tuning*) untuk dataset yang digunakan sehingga dapat meningkatkan performa model.
8. Perlu memahami statistika deskriptif dari suatu data sebelum memilih metode karena tiap metode memiliki kelebihannya masih-masing. Apabila pada data terdapat *outliers*, maka lebih baik menggunakan *Random Forest* daripada SVM.
9. Melakukan *tuning hyperparameter* SVR menggunakan metode optimasi seperti *Genetic Algorithm*, *Differential Evolution,* atau metode optimasi berbasis populasi lainnya.

DAFTAR PUSTAKA

Awad, M., & Khanna, R. (2015). *Efficient Learning Machines: Theories, Concepts, and Applications for Engineers and System Designers*. Apress Media.

Badan Pusat Statistik. (n.d.-a). Pengeluaran per Kapita. Retrieved January 22, 2020, from https://sirusa.bps.go.id/sirusa/index.php/indikator/197

Badan Pusat Statistik. (n.d.-b). Tentang BPS. Retrieved February 1, 2020, from https://www.bps.go.id/menu/1/informasi-umum.html#masterMenuTab1

Blumenstock, J., Cadamuro, G., & On, R. (2015). *Predicting Poverty and Wealth from Mobile Phone Metadata*. https://doi.org/10.1126/science.aac4420

Chen, X., & Nordhaus, W. D. (2011). Using Luminosity Data as a Proxy for Economic Statistics. *Proceedings of the National Academy of Sciences of the United States of America*, *108*(21), 8589–8594. https://doi.org/10.1073/pnas.1017031108

Gonzalez, R. C., & Woods, R. E. (2008). *Digital Image Processing* (3rd ed.). New Jersey: Prentice Hall.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. Massachusetts: The MIT Press.

Google Earth Blog. (2014). About Google Earth Imagery. Retrieved February 1, 2020, from https://www.gearthblog.com/blog/archives/2014/04/google-earth-imagery.html

Google Maps Platform. (2019). Maps Static API Usage and Billing. Retrieved February 5, 2020, from https://developers.google.com/maps/documentation/maps-static/usage-and-billing

Härdle, W. K., Prastyo, D. D., & Hafner, C. M. (2014). Support Vector Machines With Evolutionary Model Selection For Default Prediction. In *The Oxford Handbook of Applied Nonparametric and*

Semiparamtric Econometrics and Statistics (pp. 346–373). New York: Oxford University Press.

Head, A., Manguin, M., Tran, N., & Blumenstock, J. E. (2017). *Can Human Development be Measured with Satellite Imagery?* 11. https://doi.org/10.1145/3136560.3136576

Jean, N., Burke, M., Xie, M., Davis, W. M., Lobell, D. B., & Ermon, S. (2016). Combining Satellite Imagery and Machine Learning to Predict Poverty. *Science*, *353*(6301), 790–794. https://doi.org/10.1126/science.aaf7894

Jerven, M. (2013). Poor Numbers: How We Are Misled by African Development Statistics and What to Do About It. In *Journal of Chemical Information and Modeling* (Vol. 53). https://doi.org/10.1017/CBO9781107415324.004

Johnson, R. A., & Bhattacharyya, G. K. (2009). Statistics: Principles and Methods. In *Journal of Chemical Information and Modeling* (6th ed., Vol. 53). https://doi.org/10.1017/CBO9781107415324.004

Lee, C.-F., Lee, J. C., & Lee, A. C. (2013). Statistics for Business and Financial Economics. In *Statistics for Business and Financial Economics* (3rd ed.). https://doi.org/10.1142/9789812816214_0016

Liu, L., & Özsu, M. T. (2018). *Encyclopedia of Database Systems* (2nd ed.). Boston: Springer.

Loonis, V., & Bellefon, M.-P. de. (2018). *Handbook of Spatial Analysis*. https://doi.org/10.1007/s13398-014-0173-7.2

Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2018). Statistical and Machine Learning forecasting methods: Concerns and ways forward. *PLoS ONE*, *13*(3), 1–26. https://doi.org/10.1371/journal.pone.0194889

Ngestrini, R. (2019). *Predicting Poverty of a Region from Satellite Imagery using CNNs*. Utrecht University.

Noor, A. M., Alegana, V. A., Gething, P. W., Tatem, A. J., & Snow, R. W. (2008). Using Remotely Sensed Night-time Light as a Proxy for

Poverty in Africa. *Population Health Metrics*, *6*. https://doi.org/10.1186/1478-7954-6-5

Pandey, S. M., Agarwal, T., & Krishnan, N. C. (2018). Multi-task Deep Learning for Predicting Poverty from Satellite Images. *30th AAAI Conference on Artificial Intelligence, IAAI -8*, 7793–7798.

Patterson, J., & Gibson, A. (2017). *Deep Learning: A Practitioner's Approach*. California: O'Reilly Media, Inc.

Perserikatan Bangsa-Bangsa. (n.d.). Poverty Eradication. Retrieved January 21, 2020, from https://sustainabledevelopment.un.org/topics/povertyeradication

Riebeek, H. (2013). How to Interpret a Satellite Image: Five Tips and Strategies. Retrieved February 5, 2020, from https://earthobservatory.nasa.gov/features/ColorImage/page1.php

Scholkopf, B., & Smola, A. J. (2001). *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Massachusetts: The MIT Press.

Science Education Resource Center. (n.d.). What is Google Earth? Retrieved February 1, 2020, from https://serc.carleton.edu/introgeo/google_earth/what.html

Sewak, M., Karim, M. R., & Pujari, P. (2018). *Practical Convolutional Neural Networks*. Birmingham: Packt Publishing.

Simonyan, K., & Zisserman, A. (2014). *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 1–14.

World Bank. (2014). *Introduction to Poverty Analysis (English)*. https://doi.org/10.1016/B978-0-240-81203-8.00002-7

Zhang, A., Lipton, Z. C., Li, M., & Smola, A. J. (2020). *Dive into Deep Learning*. Retrieved from https://d2l.ai

(Halaman ini sengaja dikosongkan)

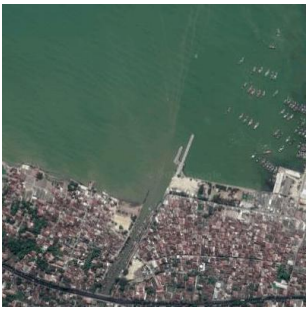
LAMPIRAN

Lampiran 1. Sampel *Input* Gambar

Monumen Nasional, Jakarta Pusat

Pulau Merak Besar, Cilegon

Jl. Tol Moh. Toha, Bandung

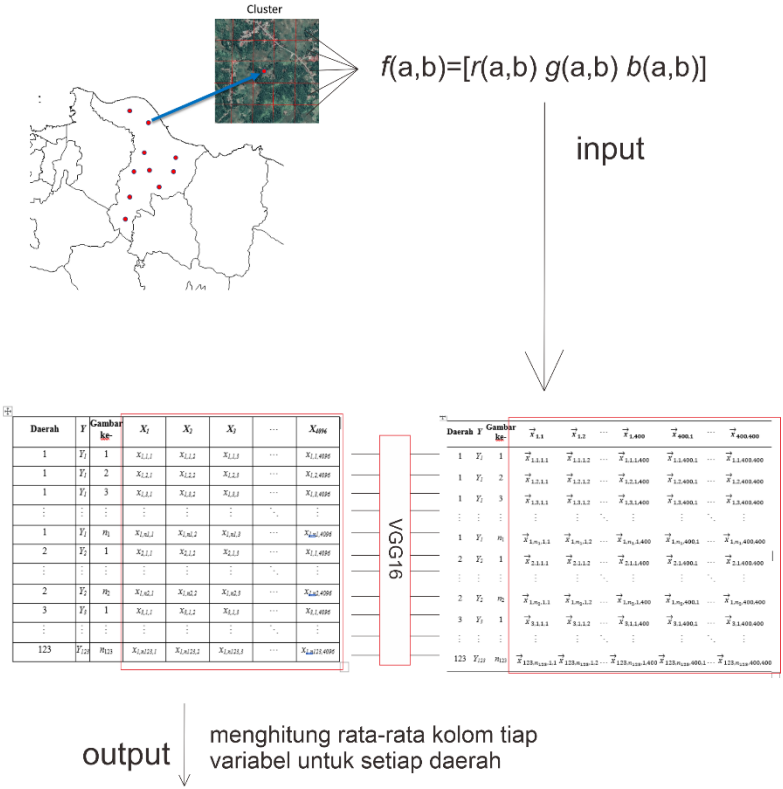


Dermaga Rembang, Rembang

UGM, Yogyakarta

ITS, Surabaya

Lampiran 2. Perubahan Gambar Menjadi 4096 Variabel



Daerah	Y	Gambar ke-	X_1	X_2	X_3	⋯	X_{4096}
1	Y_1	1	$x_{1,1,1}$	$x_{1,1,2}$	$x_{1,1,3}$	⋯	$x_{1,1,4096}$
1	Y_1	2	$x_{1,2,1}$	$x_{1,2,2}$	$x_{1,2,3}$	⋯	$x_{1,2,4096}$
1	Y_1	3	$x_{1,3,1}$	$x_{1,3,2}$	$x_{1,3,3}$	⋯	$x_{1,3,4096}$
⋮	⋮	⋮	⋮	⋮	⋮	⋱	⋮
1	Y_1	n_1	$x_{1,n1,1}$	$x_{1,n1,2}$	$x_{1,n1,3}$	⋯	$x_{1,n1,4096}$
2	Y_2	1	$x_{2,1,1}$	$x_{2,1,2}$	$x_{2,1,3}$	⋯	$x_{2,1,4096}$
⋮	⋮	⋮	⋮	⋮	⋮	⋱	⋮
2	Y_2	n_2	$x_{2,n2,1}$	$x_{2,n2,2}$	$x_{2,n2,3}$	⋯	$x_{2,n2,4096}$
3	Y_3	1	$x_{3,1,1}$	$x_{3,1,2}$	$x_{3,1,3}$	⋯	$x_{3,1,4096}$
⋮	⋮	⋮	⋮	⋮	⋮	⋱	⋮
123	Y_{123}	n_{123}	$x_{1,n123,1}$	$x_{1,n123,2}$	$x_{1,n123,3}$	⋯	$x_{1,n123,4096}$

Daerah	Y	Gambar ke-	$\vec{x}_{1,1}$	$\vec{x}_{1,2}$	⋯	$\vec{x}_{1,400}$	$\vec{x}_{400,1}$	⋯	$\vec{x}_{400,400}$
1	Y_1	1	$\vec{x}_{1,1,1,1}$	$\vec{x}_{1,1,1,2}$	⋯	$\vec{x}_{1,1,1,400}$	$\vec{x}_{1,1,400,1}$	⋯	$\vec{x}_{1,1,400,400}$
1	Y_1	2	$\vec{x}_{1,2,1,1}$	$\vec{x}_{1,2,1,2}$	⋯	$\vec{x}_{1,2,1,400}$	$\vec{x}_{1,2,400,1}$	⋯	$\vec{x}_{1,2,400,400}$
1	Y_1	3	$\vec{x}_{1,3,1,1}$	$\vec{x}_{1,3,1,2}$	⋯	$\vec{x}_{1,3,1,400}$	$\vec{x}_{1,3,400,1}$	⋯	$\vec{x}_{1,3,400,400}$
⋮	⋮	⋮	⋮	⋮	⋱	⋮	⋮	⋱	⋮
1	Y_1	n_1	$\vec{x}_{1,n_1,1,1}$	$\vec{x}_{1,n_1,1,2}$	⋯	$\vec{x}_{1,n_1,1,400}$	$\vec{x}_{1,n_1,400,1}$	⋯	$\vec{x}_{1,n_1,400,400}$
2	Y_2	1	$\vec{x}_{2,1,1,1}$	$\vec{x}_{2,1,1,2}$	⋯	$\vec{x}_{2,1,1,400}$	$\vec{x}_{2,1,400,1}$	⋯	$\vec{x}_{2,1,400,400}$
⋮	⋮	⋮	⋮	⋮	⋱	⋮	⋮	⋱	⋮
2	Y_2	n_2	$\vec{x}_{2,n_2,1,1}$	$\vec{x}_{2,n_2,1,2}$	⋯	$\vec{x}_{2,n_2,1,400}$	$\vec{x}_{2,n_2,400,1}$	⋯	$\vec{x}_{2,n_2,400,400}$
3	Y_3	1	$\vec{x}_{3,1,1,1}$	$\vec{x}_{3,1,1,2}$	⋯	$\vec{x}_{3,1,1,400}$	$\vec{x}_{3,1,400,1}$	⋯	$\vec{x}_{3,1,400,400}$
⋮	⋮	⋮	⋮	⋮	⋱	⋮	⋮	⋱	⋮
123	Y_{123}	n_{123}	$\vec{x}_{123,n_{123},1,1}$	$\vec{x}_{123,n_{123},1,2}$	⋯	$\vec{x}_{123,n_{123},1,400}$	$\vec{x}_{123,n_{123},400,1}$	⋯	$\vec{x}_{123,n_{123},400,400}$

Daerah	Y	X_1	X_2	X_3	⋯	X_{4096}
1	Y_1	$x_{1,1}$	$x_{2,1}$	$x_{3,1}$	⋯	$X_{4096,1}$
2	Y_2	$x_{1,2}$	$x_{2,2}$	$x_{3,2}$	⋯	$X_{4096,2}$
3	Y_3	$x_{1,3}$	$x_{2,3}$	$x_{3,3}$	⋯	$X_{4096,3}$
⋮	⋮	⋮	⋮	⋮	⋱	⋮
123	Y_{123}	$x_{1,123}$	$x_{2,123}$	$x_{3,123}$	⋯	$X_{4096,123}$

Lampiran 3. Data Penelitian

X_1	X_2	⋯	X_{4096}	Daerah	Pengeluaran
0.084	0.301	⋯	0.221	Admin. Kepulauan Seribu	17761,49
1.094	1.462	⋯	0.190	Bandung	13055.12
0.235	1.568	⋯	0.035	Bangkalan	8393
0.879	1.494	⋯	0.288	Banjar	14613,41
0.228	0.810	⋯	0.220	Banjarnegara	9598.356
1.742	2.211	⋯	0.064	Bantul	15386
0.441	1.040	⋯	0.293	Banyumas	11849,41
2.357	2.739	⋯	0.363	Banyuwangi	11828
0.578	0.920	⋯	0.487	Batang	10275,29
1.260	1.230	⋯	0.202	Batu	12466
1.704	1.753	⋯	0.539	Bekasi	16926,79
1.822	2.906	⋯	0.236	Kota Blitar	13391
1.148	1.928	⋯	0.167	Kabupaten Blitar	10327
1.186	1.167	⋯	0.116	Blora	8611,836
0.241	1.275	⋯	0.054	Bogor	14759,5
1.487	1.541	⋯	0.442	Bojonegoro	9926
2.642	2.039	⋯	0.705	Bondowoso	10429
1.075	1.144	⋯	0.023	Boyolali	11018,4
1.461	1.357	⋯	0.200	Brebes	10539,97
0.111	0.653	⋯	0.718	Ciamis	12951,94
0.030	0.516	⋯	0.255	Cianjur	11438,28
1.082	1.375	⋯	0.173	Cilacap	9635,772
0.644	1.020	⋯	0.300	Cilegon	12900
1.363	2.389	⋯	0.330	Cimahi	16761,8
1.090	2.264	⋯	0.165	Cirebon	11715,56
⋮	⋮	⋱	⋮	⋮	⋮
0.738	0.690	⋯	0.206	Tuban	10048
1.272	1.780	⋯	0.000	Tulungagung	10455
0.573	0.619	⋯	0.013	Wonogiri	10823,64
0.353	0.780	⋯	0.227	Wonosobo	11168,4
1.419	2.792	⋯	0.063	Yogyakarta	18629

Lampiran 4. Kombinasi *Hyperparameter* SVR Beserta Performa Model

Rank	**sigma**	**C**	**RMSE**	R^2	**MAE**
1	8.24×10^{-5}	2.263593	2145.837	0.703373	1675.059
2	8.6×10^{-5}	2.061925	2151.289	0.703073	1681.257
3	6.15×10^{-5}	2.377476	2164.506	0.702665	1694.257
4	9.36×10-5	1.746468	2167.256	0.70131	1695.145
5	5.41×10-5	2.469478	2174.809	0.701173	1703.282
6	7.26×10-5	1.9006	2181.389	0.700761	1708.223
7	9.61×10-5	1.369704	2206.609	0.698885	1732.305
8	9.91×10-5	1.296834	2212.987	0.698239	1738.122
9	9.43×10-5	1.277925	2222.618	0.697443	1746.284
10	5.48×10-5	5.513901	2140.651	0.696956	1657.518
11	5.39×10-5	1.836013	2225.557	0.695745	1751.809
12	4.61×10-5	1.987768	2236.254	0.694081	1759.74
13	0.000131	1.047587	2246.467	0.693851	1763.837
14	3.33×10-5	10.89255	2156.727	0.692774	1662.798
15	8.54×10-5	4.02062	2136.209	0.691994	1661.468
16	0.000171	1.507651	2167.457	0.687508	1698.944
17	0.000177	1.410293	2175.055	0.687059	1705.188
18	0.000139	2.497345	2142.916	0.685152	1670.858
19	0.000124	0.760483	2361.937	0.682119	1829.161
20	0.000134	0.748474	2366.046	0.680756	1832.084
21	0.000165	2.359855	2146.942	0.677453	1674.281
22	0.000168	2.616433	2143.417	0.674899	1665.559
23	0.000227	0.940683	2306.316	0.67458	1789.981
⋮	⋮	⋮	⋮	⋮	⋮
250	0.00062	0.047272	3698.921	0.482549	2589.841

Lampiran 5. Hasil Prediksi (dalam ribuan)

Daerah	Prediksi	Aktual	%Error
Admin. Kepulauan Seribu	17381.14	17761.49	0.021414
Bandung	11771.66	13055.12	0.098311
Bangkalan	8772.222	8393	0.045183
Banjar	13791.57	14613.41	0.056238
Banjarnegara	9977.248	9598.356	0.039475
Bantul	13597.32	15386	0.116253
Banyumas	10584.37	11849.41	0.10676
Banyuwangi	11447.03	11828	0.032209
Batang	10757.05	10275.29	0.046886
Batu	12085.66	12466	0.030511
Bekasi	14916.96	16926.79	0.118737
Kota Blitar	10707.19	10327	0.036815
Kabupaten Blitar	11649.7	13391	0.130035
Blora	9695.698	8611.836	0.125857
Bogor	14379.69	14759.5	0.025733
Bojonegoro	10760.18	9926	0.08404
Bondowoso	10199.68	10429	0.021989
Boyolali	10639.44	11018.4	0.034393
Brebes	10832.77	10539.97	0.02778
Ciamis	12574.03	12951.94	0.029178
Cianjur	11818.59	11438.28	0.033249
Cilacap	11228.18	9635.772	0.165261
Cilegon	13280.21	12900	0.029473
Cimahi	16912.14	16761.8	0.008969
Cirebon	12826.38	11715.56	0.094815
⋮	⋮	⋮	⋮
Tuban	9667.957	10048	0.037823
Tulungagung	11043.8	10455	0.056317
Wonogiri	10445.71	10823.64	0.034917
Wonosobo	11549.05	11168.4	0.034083
Yogyakarta	18247.9	18629	0.020457

Lampiran 6. Sintaks Mengambil Gambar Menggunakan Maps Static API Dengan Python

```
## Merge nightlight with IPM at kabupaten level
--------------------------------------------------------------------------
import os
import os.path
from osgeo import gdal, ogr, osr
from scipy import ndimage
from scipy import misc
from io import StringIO, BytesIO
gdal.UseExceptions()
import matplotlib as mpl
import matplotlib.pyplot as plt
from matplotlib import gridspec
%matplotlib inline
import urllib
import pandas as pd
import numpy as np
--------------------------------------------------------------------------
def read_raster(raster_file):
    """
    Function
    --------
    read_raster

    Given a raster file, get the pixel size, pixel location, and
pixel value

    Parameters
    ----------
    raster_file : string
        Path to the raster file
    Returns
    -------
```

Lampiran 6. Sintaks Mengambil Gambar Menggunakan Maps Static API Dengan Python (Lanjutan)

```
    x_size : float
        Pixel size
    top_left_x_coords : numpy.ndarray  shape: (number of
columns,)
        Longitude of the top-left point in each pixel
    top_left_y_coords : numpy.ndarray  shape: (number of
rows,)
        Latitude of the top-left point in each pixel
    centroid_x_coords : numpy.ndarray  shape: (number of
columns,)
        Longitude of the centroid in each pixel
    centroid_y_coords : numpy.ndarray  shape: (number of
rows,)
        Latitude of the centroid in each pixel
    bands_data : numpy.ndarray  shape: (number of rows,
number of columns, 1)
        Pixel value
    """
    raster_dataset = gdal.Open(raster_file, gdal.GA_ReadOnly)
    # get project coordination
    proj = raster_dataset.GetProjectionRef()
    bands_data = []
    # Loop through all raster bands
    for b in range(1, raster_dataset.RasterCount + 1):
        band = raster_dataset.GetRasterBand(b)
        bands_data.append(band.ReadAsArray())
        no_data_value = band.GetNoDataValue()
    bands_data = np.dstack(bands_data)
    rows, cols, n_bands = bands_data.shape
```

Lampiran 6. Sintaks Mengambil Gambar Menggunakan Maps Static API Dengan Python (Lanjutan)

```
    # Get the metadata of the raster
    geo_transform = raster_dataset.GetGeoTransform()
    (upper_left_x, x_size, x_rotation, upper_left_y, y_rotation,
y_size) = geo_transform

    # Get location of each pixel
    x_size = 1.0 / int(round(1 / float(x_size)))
    y_size = - x_size
    y_index = np.arange(bands_data.shape[0])
    x_index = np.arange(bands_data.shape[1])
    top_left_x_coords = upper_left_x + x_index * x_size
    top_left_y_coords = upper_left_y + y_index * y_size
    # Add half of the cell size to get the centroid of the cell
    centroid_x_coords = top_left_x_coords + (x_size / 2)
    centroid_y_coords = top_left_y_coords + (y_size / 2)

    return (x_size, top_left_x_coords, top_left_y_coords,
centroid_x_coords, centroid_y_coords, bands_data)

# Helper function to get the pixel index of the point
def get_cell_idx(lon, lat, top_left_x_coords,
top_left_y_coords):
    """
    Function
    --------
    get_cell_idx

    Given a point location and all the pixel locations of the raster
file,
    get the column and row index of the point in the raster
```

Lampiran 6. Sintaks Mengambil Gambar Menggunakan Maps Static API Dengan Python (Lanjutan)

```
    Parameters
    ----------
    lon : float
        Longitude of the point
    lat : float
        Latitude of the point
    top_left_x_coords : numpy.ndarray  shape: (number of
columns,)
        Longitude of the top-left point in each pixel
    top_left_y_coords : numpy.ndarray  shape: (number of
rows,)
        Latitude of the top-left point in each pixel

    Returns
    -------
    lon_idx : int
        Column index
    lat_idx : int
        Row index
    """
    lon_idx = np.where(top_left_x_coords < lon)[0][-1]
    lat_idx = np.where(top_left_y_coords > lat)[0][-1]
    return lon_idx, lat_idx
--------------------------------------------------------------------------------
raster_file = 'E:/Lomba/Gemastik
XII/F182013.v4c_web.stable_lights.avg_vis.tif'
x_size, top_left_x_coords, top_left_y_coords,
centroid_x_coords, centroid_y_coords, bands_data =
read_raster(raster_file)
```

Lampiran 6. Sintaks Mengambil Gambar Menggunakan Maps Static API Dengan Python (Lanjutan)

```
# save the result in compressed format - see
https://docs.scipy.org/doc/numpy/reference/generated/numpy.s
avez.html
np.savez('E:/Lomba/Gemastik XII/nightlight.npz',
top_left_x_coords=top_left_x_coords,
top_left_y_coords=top_left_y_coords, bands_data=bands_data)
-------------------------------------------------------------------------
# get nightlight features for each cluster
def get_nightlight_feature(sample):
    idx, wealth, x, y = sample
    lon_idx, lat_idx = get_cell_idx(x, y, top_left_x_coords,
top_left_y_coords)
    # Select the 10 * 10 pixels
    left_idx = lon_idx - 5
    right_idx = lon_idx + 4
    up_idx = lat_idx - 5
    low_idx = lat_idx + 4
    luminosity_100 = []
    for i in range(left_idx, right_idx + 1):
        for j in range(up_idx, low_idx + 1):
            # Get the luminosity of this pixel
            luminosity = bands_data[j, i, 0]
            luminosity_100.append(luminosity)
    luminosity_100 = np.asarray(luminosity_100)
    max_ = np.max(luminosity_100)
    min_ = np.min(luminosity_100)
    mean_ = np.mean(luminosity_100)
    median_ = np.median(luminosity_100)
    std_ = np.std(luminosity_100)
    return pd.Series({'id': idx, 'max_': max_, 'min_': min_,
'mean_': mean_,
                      'median_': median_, 'std_': std_, 'wealth':
wealth})
```

Lampiran 6. Sintaks Mengambil Gambar Menggunakan Maps Static API Dengan Python (Lanjutan)

```
clusters = pd.read_csv('E:/Tugas + Materi/Semester 8/Tugas
Akhir//Run April/jawa_117.csv')
data_all = clusters.apply(lambda x:
get_nightlight_feature([x['Nama'], x['PpKD'], x['longitude'],
x['latitude']]), axis=1)
data_all.to_csv('E:/Tugas + Materi/Semester 8/Tugas
Akhir/Run April/Jawa_Nightlight.csv', index=None)
--------------------------------------------------------------------------
## Download daytime satellite imagery
--------------------------------------------------------------------------
# Helper function to read a shapefile
def get_shp_extent(shp_file):
    """
    Function
    --------
    get_shp_extent

    Given a shapefile, get the extent (boundaries)

    Parameters
    ----------
    shp_file : string
        Path to the shapefile

    Returns
    -------
    extent : tuple
        Boundary location of the shapefile (x_min, x_max, y_min,
y_max)
    """
    inDriver = ogr.GetDriverByName("ESRI Shapefile")
    inDataSource = inDriver.Open(inShapefile, 0)
    inLayer = inDataSource.GetLayer()
```

Lampiran 6. Sintaks Mengambil Gambar Menggunakan Maps Static API Dengan Python (Lanjutan)

```
extent = inLayer.GetExtent()
    # x_min_shp, x_max_shp, y_min_shp, y_max_shp = extent
    return extent
---------------------------------------------------------------------------
# Helper functions to download images from Google Maps API

from retrying import retry
import imageio

@retry(wait_exponential_multiplier=1000,
wait_exponential_max=3600000)
def save_img(url, file_path, file_name):
    """
    Function
    --------
    save_img

    Given a url of the map, save the image

    Parameters
    ----------
    url : string
        URL of the map from Google Map Static API
    file_path : string
        Folder name of the map
    file_name : string
        File name

    Returns
    -------
    None
    """
```

Lampiran 6. Sintaks Mengambil Gambar Menggunakan Maps Static API Dengan Python (Lanjutan)

```
a = urllib.request.urlopen(url).read()
    b = BytesIO(a)
    image = imageio.imread(b, pilmode="RGB")
    # when no image exists, api will return an image with the
same color.
    # and in the center of the image, it said'Sorry. We have no
imagery here'.
    # we should drop these images if large area of the image has
the same color.
    if np.array_equal(image[:,:10,:],image[:,10:20,:]):
        pass
    else:
        imageio.imwrite(file_path + file_name, image[50:450, :,
:])
--------------------------------------------------------------------------
# Now read in the shapefile for Indonesia and extract the edges
of the country
inShapefile = "E:/Tugas + Materi/Pelatihan internal/xxx.shp"
x_min_shp, x_max_shp, y_min_shp, y_max_shp =
get_shp_extent(inShapefile)

left_idx, top_idx = get_cell_idx(x_min_shp, y_max_shp,
top_left_x_coords, top_left_y_coords)
right_idx, bottom_idx = get_cell_idx(x_max_shp, y_min_shp,
top_left_x_coords, top_left_y_coords)

banyak_cell = (right_idx-left_idx)*(bottom_idx-top_idx)
print(left_idx)
print(right_idx)
print(top_idx)
print(bottom_idx)
print(banyak_cell)
```

Lampiran 6. Sintaks Mengambil Gambar Menggunakan Maps Static API Dengan Python (Lanjutan)

```
key = 'Insert API Key Here'
m = 1

for i in range(left_idx, right_idx + 1):
    for j in range(top_idx, bottom_idx + 1):
        lon = centroid_x_coords[i]
        lat = centroid_y_coords[j]
        url =
'https://maps.googleapis.com/maps/api/staticmap?center=' +
str(lat) + ',' + \
              str(lon) +
'&zoom=16&size=400x500&maptype=satellite&key=' + key
        lightness = bands_data[j, i, 0]
        file_path = 'E:/Tugas + Materi/Semester 8/Tugas
Akhir/google_images/Jateng 6/' + str(lightness) + '/'
        if not os.path.isdir(file_path):
            os.makedirs(file_path)
        file_name = str(i) + '_' + str(j) +'.jpg'
        save_img(url, file_path, file_name)
        if m % 100 == 0:
            print(m)
        m += 1
-------------------------------------------------------------------------
## Use the keras library to use a basic CNN to extract
features of the daytime images
-------------------------------------------------------------------------
from keras.applications.vgg16 import VGG16
import numpy as np
from keras.preprocessing import image
from keras.models import Sequential
from keras.applications.vgg16 import preprocess_input,
decode_predictions
```

Lampiran 6. Sintaks Mengambil Gambar Menggunakan Maps Static API Dengan Python (Lanjutan)

```
from keras.layers.convolutional import Convolution2D,
AveragePooling2D
from keras.optimizers import SGD
from keras.layers.core import Activation
from keras.layers.core import Flatten
from keras.layers.core import Dense
from keras.layers import Dropout
from multiprocessing import Pool
import os
import time
import pandas as pd
import numpy as np
from keras.models import Model
--------------------------------------------------------------------------
images_name = {}
for i in range(64):
   dir_ = 'E:/Lomba/Gemastik XII/google_image/Jawa/' + str(i)
+ '/'
   image_files = os.listdir(dir_)
   for f in image_files:
      images_name[f] = i

def get_cell_idx(lon, lat, top_left_x_coords,
top_left_y_coords):
   lon_idx = np.where(top_left_x_coords < lon)[0][-1]
   lat_idx = np.where(top_left_y_coords > lat)[0][-1]
   return lon_idx, lat_idx

npzfile = np.load('E:/Lomba/Gemastik XII/nightlight.npz')
top_left_x_coords = npzfile['top_left_x_coords']
top_left_y_coords = npzfile['top_left_y_coords']
bands_data = npzfile['bands_data']
```

Lampiran 6. Sintaks Mengambil Gambar Menggunakan Maps Static API Dengan Python (Lanjutan)

```
# get image featuers
base_model = VGG16(weights='imagenet')
model = Model(inputs=base_model.input,
outputs=base_model.get_layer('fc2').output)

def get_input_feature(img_path):
    img = image.load_img(img_path, target_size=(224, 224))
    # img = image.load_img(img_path)
    x = image.img_to_array(img)
    x = np.expand_dims(x, axis=0)
    x = preprocess_input(x)
    features = model.predict(x)
    return features[0]

def get_daytime_feature(sample):
    idx, wealth, x, y = sample
    print(idx)
    lon_idx, lat_idx = get_cell_idx(x, y, top_left_x_coords,
top_left_y_coords)
    left_idx = lon_idx - 5
    right_idx = lon_idx + 4
    up_idx = lat_idx - 5
    low_idx = lat_idx + 4
    features_100 = []
    for i in range(left_idx, right_idx + 1):
        lon = centroid_x_coords[i]
        for j in range(up_idx, low_idx + 1):
            lat = centroid_y_coords[j]
            file_name = str(lon) + '_' + str(lat) + '.jpg'
            if file_name in images_name:
                luminosity = images_name[file_name]
```

Lampiran 6. Sintaks Mengambil Gambar Menggunakan Maps Static API Dengan Python (Lanjutan)

```
feature = get_input_feature('E:/Lomba/Gemastik
XII/google_image/Jawa/' + str(luminosity) + '/' + file_name)
            features_100.append(feature)
    if len(features_100) == 0:
        print('nononono: ' + str(idx))
        return np.asarray([np.nan] * 4096 + [wealth]).tolist()
    else:
        features_100 = np.asarray(features_100)
        return np.append(np.mean(features_100, axis=0),
wealth).tolist()

clusters = pd.read_csv('E:/Tugas + Materi/Semester 8/Tugas
Akhir/Run April/jawa_117.csv')
clusters['feature'] = clusters.apply(lambda x:
get_daytime_feature([x['Nama'], x['PpKD'], x['longitude'],
x['latitude']]), axis=1)

data_all = clusters['feature']
data_all = np.asarray([i for i in data_all])
data_all = data_all[~np.isnan(data_all).any(axis=1)]
np.savetxt('E:/Tugas + Materi/Semester 8/Tugas Akhir/Run
April/google117_image_features_cnn_v4.csv', data_all)
```

Lampiran 7. Histogram dan Boxplot Dengan R

```
# Set Working Directory and Load Data
setwd("E:/Tugas + Materi/Semester 8/Tugas Akhir/Run April")
data <- read.table("E:/Tugas + Materi/Semester 8/Tugas
Akhir/Run April/google117_image_features_cnn_v3.csv",
quote="\"", comment.char="")

# Load Package
library(ggplot2)
library(summarytools)
```

Lampiran 7. Histogram dan Boxplot Dengan R (Lanjutan)

```
library(cowplot)
library(gridExtra)
library(grid)

# Create Density Histogram
dfSummary(data$V4097)
data1<-data[,4096:4097]
var(data$V4097)
sd(data$V4097)/mean(data$V4097)
x_ref_line<-mean(data$V4097)+800
histo<-ggplot(data = data, aes(x=V4097,y=..density..))+
  geom_histogram(fill="#0c4c8a", bins = 30L) +
geom_density(size=0.75) + theme_minimal() +
  geom_vline(xintercept = mean(data$V4097)) +
  geom_text(aes(x=x_ref_line, label= "12712.87",y=0.00001))
+
  xlab("Pengeluaran Perkapita") +
  xlim(7000,28000) + geom_text(aes(label="Variance =
12484815",x=26000,y=0.0004))
histo
# Create Boxplot
y_ref_line<-median(data$V4097)+600
boxplt<-ggplot(data) + coord_flip() +
  aes(x = "", y = V4097) +
  geom_boxplot(fill = "#0c4c8a") +
  labs(y =NULL,x=NULL) +
  theme_minimal() +
geom_text(aes(y=y_ref_line,label="11539")) +
  ylim(7000,28000)
boxplt

# Combine Both Graph As One
plot_grid(boxplt,histo, ncol = 1, rel_heights = c(1, 3),
          align = 'v', axis = 'lr')
```

Lampiran 8 Sintaks Pemodelan SVR Dengan R

```
rm(list = ls())
# Set Working Directory and Load Data
setwd("E:/Tugas + Materi/Semester 8/Tugas Akhir/Run April")
data <- read.table("E:/Tugas + Materi/Semester 8/Tugas
Akhir/Run April/google117_image_features_cnn_v3.csv",
quote="\"", comment.char="")

# Show The First 10 Observations Of The Dependent Variable
data[1:10,4097]

# Load Library
library(e1071)
library(kernlab)
library(caret)
library(openxlsx)

# Remove Constant Data
nozero<-which(colSums(data)!=0)
data1<-data[,nozero]
data2<-subset(data1, select = -
c(V2809,V2794,V380,V172,V2945,V183))

# 10-Folds Using Caret Package, Randomly Generate 250
Parameter Combinations
set.seed(123)
train.control <- trainControl(method = "cv", number = 10,
search = "random")
set.seed(1234)
svm250<-train(V4097~., data = data2, method = "svmRadial",
          trControl=train.control, metric = "Rsquared",
          tuneLength = 250)
hasil_svm250<-svm250$results
```

Lampiran 8. Sintaks Pemodelan SVR Dengan R (Lanjutan)

```
hasil_svm250<-hasil_svm250[order(hasil_svm250$Rsquared,
decreasing = T),]
head(hasil_svm250)
ggplot(svm250)

# Save The Result
write.xlsx(hasil_svm250,"Hasil Randomized Search 250
Kombinasi.xlsx")

# Predict Using Whole Data
pred_svm250<-predict(svm250,data2[,-4078])
head(cbind(pred_svm250,data2$V4097))
write.xlsx(cbind(pred_svm250,data2$V4097),"Hasil
Prediksi.xlsx")

# Calculate Rsq Training
hasil_svm250$Rsq_train<-0
r2 <- rep(0,10)
for (i in 1:250) {
  for (j in 1:10) {
    a <- as.numeric(unlist(svm250[["control"]][["index"]][j]))
    training_fold <- data2[a,]
    regressor <- ksvm(V4097~.,data=training_fold,
kernel="rbfdot",
                      kpar=list(sigma=hasil_svm250$sigma[i]),
                      C=hasil_svm250$C[i])
    pred <- predict(regressor,training_fold[,-4078])
    rss <- sum((training_fold[,4078]-pred)^2)
    tss <- sum((training_fold[,4078]-
mean(training_fold[,4078]))^2)
    r2[j] <- 1 - rss/tss
  }
  hasil_svm250$Rsq_train[i] <- mean(r2)
}
```

Lampiran 8. Sintaks Pemodelan SVR Dengan R (Lanjutan)

```
write.xlsx(hasil_svm250,"Hasil Randomized Search 10 CV
250 Kombinasi.xlsx")

# Actual vs Pred
library(ggrepel)
vs <-
data.frame(pred_svm250,data2$V4097,abs(pred_svm250-
data2$V4097)/data2$V4097)
colnames(vs) <- c("Prediksi","Aktual","Persentase_Error")
jawa_117 <- read_excel("jawa_117.xlsx")
vs$Daerah<-jawa_117$Nama
a<-ggplot(vs) +
  aes(y = Prediksi, x = Aktual, label=Daerah) +
labs(col="%Error") +
  geom_point(size = 2, aes( colour = Persentase_Error)) +
theme_minimal() +
  scale_color_gradient(low = "blue", high = "red")+
xlim(8000,30000) +
geom_label_repel(aes(label=ifelse(Persentase_Error>0.15,Daer
ah,"")),
          box.padding = 0.35, point.padding = 0.5, segment.color
= "grey50")
a
```

Lampiran 9. Output Pemodelan SVR

```
Support Vector Machines with Radial Basis Func
tion Kernel

 117 samples
4077 predictors

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 105, 105, 106, 105, 1
06, 105, ...
Resampling results across tuning parameters:
```

Lampiran 9. Output Pemodelan SVR (Lanjutan)

```
sigma          C              RMSE      Rsquared
MAE
  3.107053e-05  4.576768e-01  2976.133  0.6344
750  2165.328
  3.108341e-05  1.740630e+02  2315.070  0.6379
136  1770.483
  3.230008e-05  7.824359e-02  3563.793  0.5820
559  2493.902
  3.291690e-05  2.955715e+01  2215.108  0.6665
364  1691.116
  3.326624e-05  1.089255e+01  2156.727  0.6927
738  1662.798
  3.348485e-05  7.816376e+02  2311.230  0.6383
298  1764.344
  3.365781e-05  5.968502e+02  2310.853  0.6383
696  1763.810
  3.384540e-05  6.324727e-01  2777.614  0.6485
589  2069.854
  3.413543e-05  2.379940e+01  2202.201  0.6703
041  1685.103
  3.526822e-05  1.113622e-01  3461.821  0.5978
611  2432.235
  3.574907e-05  2.901917e+01  2219.552  0.6645
025  1693.272
  3.748330e-05  3.469152e+01  2248.696  0.6568
059  1715.064
  3.835751e-05  9.863719e+01  2304.264  0.6390
956  1758.958
  3.875066e-05  4.962275e+02  2303.820  0.6391
351  1758.520
  3.947016e-05  8.187742e+01  2303.035  0.6392
322  1757.821
  3.980949e-05  2.569126e+02  2302.597  0.6392
673  1757.383
  4.175063e-05  1.081981e+02  2300.240  0.6395
343  1755.195
  4.251509e-05  2.996366e+01  2240.264  0.6578
167  1706.859
  ...           ...           ...       ...
...
  3.464921e-04  4.677940e+02  2329.806  0.6169
 553  1795.279
```

Lampiran 9. Output Pemodelan SVR (Lanjutan)

```
3.493082e-04  6.393531e+00  2332.272  0.6165
002  1797.123
 [ reached getOption("max.print") - omitted 50
rows ]

Rsquared was used to select the optimal model
using the largest value. The final values used
for the model were sigma = 8.241935e-05 and
C = 2.263593.
```

Lampiran 10. Sintaks 3D Plot Interaktif Dengan Python

```
import plotly.express as px
import pandas as pd
data = pd.read_excel("E:/Tugas + Materi/Semester 8/Tugas
Akhir/Run April/Hasil Randomized Search 250 10 CV
Kombinasi.xlsx")
fig = px.scatter_3d(data, x='sigma', y='C',
z='Rsquared',color='Rsquared')
fig.show()
fig.write_image("E:/Tugas + Materi/Semester 8/Tugas
Akhir/Run April/fig2.png")
```

Lampiran 11. Tahapan Perhitungan VGG16

1. Input gambar berdimensi 400 × 400 *pixel*.

Ilustrasi *Input* Berukuran 400 × 400 *pixel*

37 86 91	36 85 90	⋯	14 60 73
36 85 90	36 85 90	⋯	14 60 73
⋮	⋮	⋱	⋮
36 85 92	36 85 92	⋯	18 64 77

Lampiran 11. Tahapan Perhitungan VGG16 (Lanjutan)

2. Reduksi dimensi gambar menjadi 224 × 224 *pixel*.

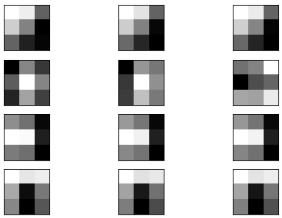
Ilustrasi Reduksi Dimensi *Input* Menjadi 224 × 224 *pixel*

37 86 91	36 85 90	⋯	14 60 73
34 83 88	35 84 89	⋯	14 60 73
⋮	⋮	⋱	⋮
36 85 92	37 86 93	⋯	18 64 77

3. Transformasi gambar pada *layer* block1_conv1

Terdapat *kernel* dengan ukuran 3 × 3 × 3 × 64 (panjang 3 *pixel*, lebar 3 *pixel*, 3 channel RGB, 64 buah *kernel*). Berikut adalah ilustrasi dari 4/64 *kernel* pada *layer* block1_conv1.

Ilustrasi Filter Konvolusi Pada *layer* block1_conv1

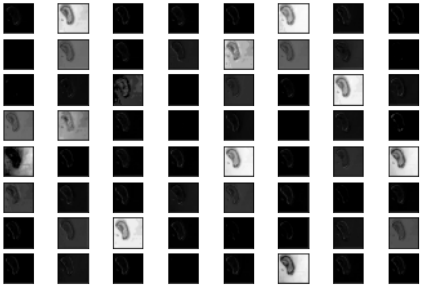


Nilai *pixel* pada *kernel* yang terletak di pojok kiri atas adalah (*kernel* ini mewakili *channel* R):

0,429	0,373	-0,036
0,275	0,039	-0,367
-0,057	-0,262	-0,350

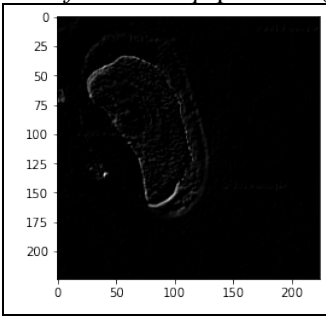
Berdasarkan subbab 2.3.1, apabila input gambar yang berdimensi 224 × 224 × 3 ditransformasi dengan tiap *kernel* dengan dimensi 3 × 3 × 3 × 64, akan menghasilkan *feature maps* (*output*) berdimensi 224 × 224 × 64 sebagai berikut.

Lampiran 11. Tahapan Perhitungan VGG16 (Lanjutan)



Ilustrasi Output *Layer* block1_conv1

Nilai *feature map* pertama (pojok kiri atas) adalah sebagai berikut.

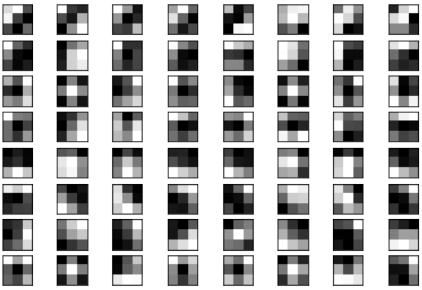


148,703	112,791	⋯	2,078
105,780	0	⋯	0
⋮	⋮	⋱	⋮
0	0	⋯	0

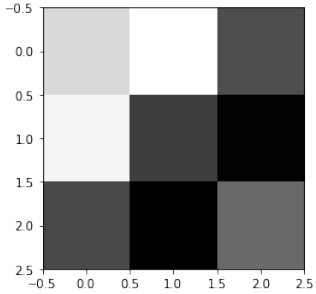
4. Transformasi gambar pada *layer* block1_conv2

Feature maps pada *layer* block1_conv1 akan menjadi *input* pada *layer* block1_conv2 sehingga memiliki dimensi 224 × 224 × 64. Karena *input* pada *layer* ini memiliki 64 *channel/depth*, maka *kernel* harus berjumlah 64 pula sebagaimana dibahas pada subbab 2.3.1. Oleh karena itu, *kernel* pada *layer* ini berdimensi 3 × 3 × 64 × 64 (panjang 3 *pixel*, lebar 3 *pixel*, 64 channel, 64 buah *kernel*). Namun, sangat tidah mudah untuk memvisualisasikan *kernel* tersebut, karena akan menghasilkan 64 × 64 gambar *kernel*. Berikut ini adalah ilustrasi *kernel* pertama dari 64 *kernel* berukuran 3 × 3 × 64.

Lampiran 11. Tahapan Perhitungan VGG16 (Lanjutan)



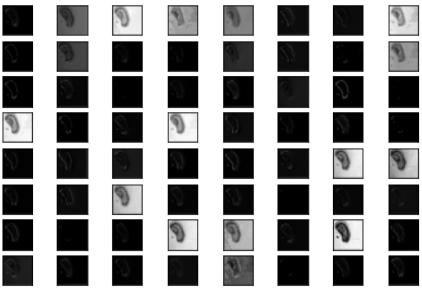
Ilustrasi Filter Konvolusi Pada *layer* block2_conv1



Nilai *pixel* pada *kernel* yang terletak di pojok kiri atas adalah (*kernel* ini *channel* pertama):

0,166	0,212	-0,006
0,199	-0,026	-0,098
-0,011	-0,102	0,027

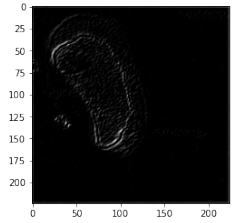
Input gambar yang berdimensi 224 × 224 × 64 ditransformasi dengan tiap *kernel* dengan dimensi 3 × 3 × 64 × 64, akan menghasilkan *feature maps* (*output*) berdimensi 224 × 224 × 64 sebagai berikut.



Ilustrasi Output *Layer* block2_conv1

Lampiran 11. Tahapan Perhitungan VGG16 (Lanjutan)

Nilai *pixel* pada *feature map* pertama (pojok kiri atas) adalah sebagai berikut.

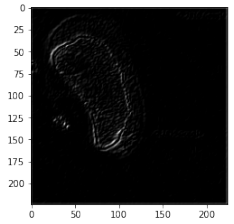


0	4,655	⋯	224,94
34,670	322,262	⋯	258,02
⋮	⋮	⋱	⋮
361,944	418,866	⋯	0,537

5. Transformasi gambar pada *layer* block1_pool

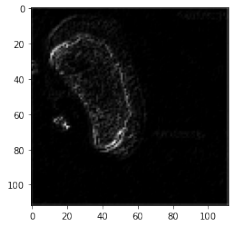
Dimensi i*nput* pada tahap ini adalah 224 × 224 × 64. *Layer* ini menerapkan fungsi *max pooling*, sehingga menghasilkan *output feature maps* berukuran 112 × 112 × 64. Berikut ini adalah contoh penerapan *max pooling* pada *layer* ini.

Input



0	4,655	⋯	224,94
34,670	322,262	⋯	258,02
⋮	⋮	⋱	⋮
361,944	418,866	⋯	0,537

Output



322,262	176,612	⋯	258,02
189,908	0	⋯	138,51
⋮	⋮	⋱	⋮
418,866	226,409	⋯	348,56

Lampiran 11. Tahapan Perhitungan VGG16 (Lanjutan)

6. Transformasi gambar hingga *layer* block5_pool

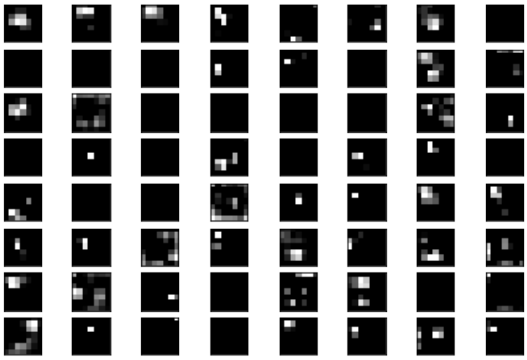
Transformasi gambar setelah *layer* block1_pool hingga *layer* block5_pool pada dasarnya sama seperti tahap 3-5.

Arsitektur VGG16 Beserta Input dan Outputnya

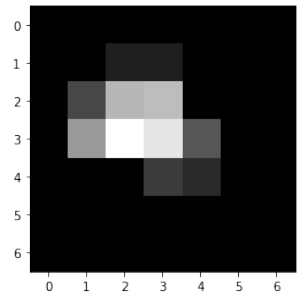
Layer	*Input*	*Output*
Input_2		224 × 224 × 3
Block1_conv1	224 × 224 × 3	224 × 224 × 64
Block1_conv2	224 × 224 × 64	224 × 224 × 64
Block1_pool	224 × 224 × 64	112 × 112 × 64
Block2_conv1	112 × 112 × 64	112 × 112 × 128
Block2_conv2	112 × 112 × 128	112 × 112 × 128
Block2_pool	112 × 112 × 128	56 × 56× 128
Block3_conv1	56 × 56× 128	56 × 56× 256
Block3_conv2	56 × 56× 256	56 × 56× 256
Block3_conv3	56 × 56× 256	56 × 56× 256
Block3_pool	56 × 56× 256	28 × 28× 256
Block4_conv1	28 × 28× 256	28 × 28× 512
Block4_conv2	28 × 28× 512	28 × 28× 512
Block4_conv3	28 × 28× 512	28 × 28× 512
Block4_pool	28 × 28× 512	14 × 14× 512
Block5_conv1	14 × 14× 512	14 × 14× 512
Block5_conv2	14 × 14× 512	14 × 14× 512
Block5_conv3	14 × 14× 512	14 × 14× 512
Block5_pool	14 × 14× 512	7 × 7 × 512

Lampiran 11. Tahapan Perhitungan VGG16 (Lanjutan)

Ilustrasi 64 *output* pertama pada *layer* block5_pool berdasarkan *input* gambar pada langkah pertama adalah sebagai berikut.



Nilai *feature map* pertama dari 512 *output* adalah sebagai berikut.



0	0	⋯	0
0	0	⋯	0
⋮	⋮	⋱	⋮
0	0	⋯	0

7. Konversi matriks menjadi vektor pada *flatten layer*

Feature map berdimensi 7 × 7 sebanyak 512 buah akan diubah menjadi vektor berdimensi 25088 (7 × 7 × 512) pada *layer* ini.

$$\begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix} \times \text{sebanyak 512 kali} \rightarrow \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Vektor inilah yang yang akan menjadi input *fully-connected layer*

Lampiran 11. Tahapan Perhitungan VGG16 (Lanjutan)

8. *Fully-connected Layer* 1

Output layer ini berupa vektor berukuran 4096 × 1 sedangkan *input* berukuran 25088 × 1. Karena semua *input* memiliki koneksi dengan semua *output*, maka matriks bobot (*W*) berukuran 25088 × 4096. Setelah *output* dihitung, diterapkan fungsi aktivasi ReLu, yaitu max(0,x).

$W \times \vec{I} + \vec{B} = \vec{O}$

$$\begin{bmatrix} 1,975\times10^{-5} & 3,531\times10^{-4} & \cdots & 0,003 \\ -0,00535 & -0,00157 & \cdots & 0,004 \\ 6,606\times10^{-4} & 0,0013 & \cdots & 1,708\times10^{-4} \\ \vdots & \vdots & \ddots & \vdots \\ -7,453\times10^{-4} & 0,003 & \cdots & -0,004 \\ 1,102\times10^{-5} & -6,471\times10^{-4} & \cdots & -0,001 \\ 6,675\times10^{-4} & 0,00183 & \cdots & -0,005 \end{bmatrix} \times \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 6,089 \\ 0 \end{bmatrix} + \begin{bmatrix} -0,189 \\ 0,127 \\ -2,996 \\ \vdots \\ -0,184 \\ 0,174 \\ -0,167 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0,99 \\ 0 \\ 0 \end{bmatrix}$$

9. *Fully-connected Layer* 2

Baik *input* dan *output layer* ini berupa vektor berukuran 4096 × 1, sehingga matriks bobot *W* berukuran 4096 × 4096. Setelah *output* dihitung, diterapkan fungsi aktivasi ReLu, yaitu max(0,x).

$W \times \vec{I} + \vec{B} = \vec{O}$

$$\begin{bmatrix} 0,004 & -0,002 & \cdots & 0,002 \\ 0,005 & -0,002 & \cdots & 0,005 \\ -0,003 & -8,607\times10^{-5} & \cdots & -0,006 \\ \vdots & \vdots & \ddots & \vdots \\ -0,004 & -4,144\times10^{-4} & \cdots & 0,002 \\ 2,787\times10^{-4} & 0,009 & \cdots & 0,002 \\ 4,969\times10^{-4} & 0,002 & \cdots & 0,005 \end{bmatrix} \times \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0,99 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0,647 \\ 0,480 \\ 0,586 \\ \vdots \\ 0,502 \\ 0,418 \\ 0,666 \end{bmatrix} = \begin{bmatrix} 0 \\ 0,272 \\ 0 \\ \vdots \\ 0,188 \\ 2,748 \\ 0 \end{bmatrix}$$

Output *vektor* inilah yang akan menjadi variabel prediktor

Lampiran 12. Contoh Estimasi Regresi SVR Kernel RBF

Contoh Data:

X1	X2	Y
0	0	0
1	1	1
1	2	1
2	1	2

Persamaan yang didapat adalah:

- *Support vector* dan *Lagrange multiplier*

No	*Support Vector*		*Lagrange multiplier*
	X1	X2	$\alpha_i^* - \alpha_i$
1	0	0	-1
2	1	2	1

- γ : 1e-3
- b : 0,99700698

berdasarkan persamaan 2.17, maka estimasi dari [1 1] adalah

$$f([1 \quad 1]) = \sum_{i=1}^{2} (\alpha_i^* - \alpha_i) \exp\left(-\gamma \left\| x_i - [0 \quad 0] \right\|^2\right) + b$$

$$f([1 \quad 1]) = (\alpha_1^* - \alpha_1) \exp\left(-10^{-3} \left\| [0 \quad 0] - [1 \quad 1] \right\|^2\right) +$$

$$(\alpha_2^* - \alpha_2) \exp\left(-10^{-3} \left\| [1 \quad 2] - [1 \quad 1] \right\|^2\right) +$$

$$0.99700698$$

$$f([1 \quad 1]) = -\exp\left(-10^{-3} \left\| [-1 \quad -1] \right\|^2\right) + \exp\left(-10^{-3} \left\| [0 \quad 1] \right\|^2\right) +$$

$$0.99700698$$

$$f([1 \quad 1]) = -\exp\left(-10^{-3} \times 2\right) + \exp\left(-10^{-3} \times 1\right) +$$

$$0.99700698$$

$$f([1 \quad 1]) = -1(0,998002) + 0,9990005 + 0.99700698$$

$$f([1 \quad 1]) = 0,99800548$$

Lampiran 12. Contoh Estimasi Regresi SVR Kernel RBF (Lanjutan)

Estmasi SVR menggunakan *software*

```
In [89]: from sklearn import svm
         import math
         import numpy as np

         X = [[0, 0], [1, 1], [1,2], [1,2]]
         y = [0, 1, 1, 2]
         clf = svm.SVR(gamma=1e-3)
         clf.fit(X, y)
         Xtest = [[1,1]]
         print(clf.predict(Xtest))

         [0.99800548]
```

Lampiran 13. Surat Pernyataan Data

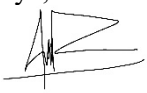
SURAT PERNYATAAN

Saya yang bertanda tangan di bawah ini, mahasiswa Departemen Statistika FSAD ITS,
Nama : Asva Abadila Rouhan
NRP : 062116 4000 0105
menyatakan bahwa data yang digunakan dalam Tugas Akhir ini merupakan data sekunder yang diambil dari ~~penelitian / buku / Tugas Akhir / Thesis / Publikasi /~~ lainnya yaitu :
Sumber : Data dari website

1. maps.googleapis.com/maps/api/staticmap?
2. BPS Provinsi
   a. https://jateng.bps.go.id
   b. https://yogyakarta.bps.go.id
   c. https://jakarta.bps.go.id
   d. https://jabar.bps.go.id
   e. https://jatim.bps.go.id
   f. https://banten.bps.go.id

Keterangan:
1. Data Citra Digital Siang Hari
2. Data Pengeluaran Perkapita Kabupaten/Kota

Surat pernyataan ini dibuat dengan sebenarnya. Apabila terdapat pemalsuan data maka saya siap menerima sanksi sesuai aturan yang berlaku.

Mengetahui, Pembimbing Tugas Akhir	Surabaya, 17 Juni 2020
	
Santi Puteri Rahayu, M.Si., Ph.D. NIP. 19820326 200312 1 004	Asva Abadila Rouhan NRP. 062116 4000 0105

BIODATA PENULIS

Penulis dilahirkan di Sleman, 1 Januari 1998 dengan nama lengkap Asva Abadila Rouhan, biasa dipanggil Asva. Penulis menempuh pendidikan formal di SDIT Salman Al-Farisi, SMPN 1 Depok, dan MAN 1 Yogyakarta. Kemudian penulis diterima sebagai mahasiswa Departemen Statistika ITS pada tahun 2016. Selama masa perkuliahan, penulis aktif di Divisi *Statistical Computer Course* (SCC) Himpunan Mahasiswa Statistika ITS (HIMASTA-ITS) sebagai staff Training & Development 2017-2018 dan manajer Training & Development 2018-2019. Selain itu, penulis juga aktif di Departemen Media Informasi Himpunan Mahasiswa Statistika ITS (HIMASTA-ITS) 2018-2019 sebagai Kabiro Kampanye Kreatif. Dalam kegiatan kepanitiaan, penulis berperan aktif dalam pada *big event* Statistika ITS, Pekan Raya Statistika (PRS) pada tahun 2017 sebagai Koordinator Media Informasi dan pada tahun 2018 sebagai Expert Committee Media Informasi. Selain itu selama menjalani masa perkuliahan penulis berkesempatan dalam menjalani program magang di Dinas Kependudukan dan Pencatatan Sipil Surabaya. Penulis tertarik dengan kompetisi statistika yang berbau komputasi walaupun selama 4 tahun kuliah hanya mampu menjadi finalis Data Analytics Competition FIND IT! 2019 dan finalis Data Mining Joints 2019. Bagi pembaca yang ingin berdiskusi, memberikan saran, dan kritik mengenai Tugas Akhir ini dapat disampaikan melalui nomor 087809737554 atau melalui email abadilaasva@gmail.com.