RANCANG BANGUN SISTEM PENGENALAN WAJAH BERBASIS RESIDUAL NETWORK

Design of Face Recognition Base on Residual Network

Muhammad Sya'roni Mujahidin[1] , Dr. Misbahuddin, ST.,MT. [2] , Bulkis Kanata,ST.,MT. [3]

Jurusan Teknik Elektro Universitas Mataram

[1]msmujahidin2018@gmail.com, [2]misbahuddin@unram.ac.id , [3] uqinata@unram.ac.id

ABSTRAK

Deep Learning adalah sebuah bidang keilmuan baru dalam bidang *Machine Learning* yang akhir-akhir ini berkembang karena perkembangan teknologi *GPU accelaration*. *Deep Learning* memiliki kemampuan yang sangat baik dalam visi komputer. Salah satunya adalah pada kasus pengenalan objek pada citra. Dengan mengimplementasikan salah satu metode *Machine Learning* yang dapat digunakan untuk pengenalan citra objek yaitu *CNN*. *Convolutional Neural Networks* (*CNN*) merupakan saat ini diklaim sebagai model terbaik yang saat ini diklaim sebagai model terbaik untuk memecahkan masalah *object recognition* dan *detection*. Dalam implementasinya, algoritma *CNN* menggunakan input data berupa sampel *training* menggunakan gambar yang diberi label berupa teks. Pada penelitian ini peneliti merancang sistem pengenalan wajah dengan menggunalan algoritma *CNN ResNet*. Hasil dari penelitian ini adalah 92 % untuk meningkatkan akurasi pengenalan wajah.

Kata Kunci : *Residual Network, CNN,* Pengenalan Wajah

ABSTRACT

Deep Learning is a new scientific field in the field of Machine Learning which has recently developed due to the development of GPU accelaration technology. Deep Learning has excellent skills in computer vision. One of them is in the case of object recognition in the image. By implementing one of the Machine Learning methods that can be used for object image recognition, CNN. Convolutional Neural Networks (CNN) is currently claimed to be the best model that is currently claimed to be the best model for solving object recognition and detection problems. In its implementation, the CNN algorithm uses input data in the form of training samples using images that are labeled in the form of text. In this study, researchers designed a face recognition system using the CNN ResNet algorithm. The results of this study were 92% to improve the accuracy of face recognition.

Keywords: Residual Network, CNN, Face Recognition.

PENDAHULUAN

Pengenalan wajah adalah salah satu metode *biometerik* yang cukup popular. Wajah lebih sulit untuk ditiru, dimodifikasi, atau dicuri jika dibandingkan dengan kunci atau *password* pada keamanan *non-biometrik*. Pada umumnya, metode *biometric* membutuhkan perangkat khusus untuk mengumpulkan data. Misalnya, *fingerprint* untuk mendeksi sidik jari, *scanner* untuk memindai sebuah bentuk dan *palmprint scanner* untuk mendeksi jari di handphone namun pengguna harus menyentuh alat tersebut secara fisik untuk mendapatkan data. Pada pengenalan wajah, wajah akan dideteksi secara otomatis tanpa memerlukan sentuhan wajah pada perangkat pendeteksinya

Sistem pengenalan wajah menjadi topik yang sering dipelajari dibidang *computer vision* dalam beberapa dekade ini. Sistem ini telah diaplikasikan dalam beberapa bidang, contohnya

pada *smartphone* untuk *facelock,* imigrasi, dan juga di media sosial untuk mengatasi *face tagging*. Pengenalan wajah sendiri terdiri dari tahap deteksi dan klasifikasi. Kedua tahap tersebut begitu cepat dilakukan oleh manusia tetapi butuh waktu yang lama bagi komputer. Kemampuan manusia itulah yang ingin diduplikasi oleh para peneliti dalam beberapa tahun belakangan ini sebagai teknologi biometrik dalam bidang *computer vision* degan tujuan membentuk suatu model untuk pengenalan citra wajah pada komputer.

Penelitian yang pernah dilakukan sebelumnya oleh Abhirawa (2017) merancang sistem pengenalan wajah dengan menerapkan *Convolutional Neural Network (CNN)* dengan membuat beberapa layer *conv2D.* Akan tetapi, penelitian tersebut memiliki akurasi pengenalan wajah yang rendah, selain itu *Convolutional Neural Network (CNN)* yang digunakan hanya membuat beberapa layer saja, sehingga jaringan syaraf tiruan yang dibuat tidak dalam.

Oleh karena itu pada penelitian ini akan dibuat sistem pengenalan wajah dengan menggunakan metode *Convolutional Neural Network (CNN)* dengan mengaplikasikan *Residual network (Resnet)* untuk meningkatkan akurasi pada sistem pengenalan wajah.

TINJAUAN PUSTAKA

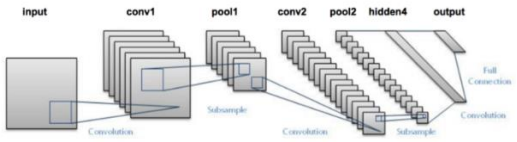
1. Convolulutional Neural Network (*CNN*)

Convolutional Neural Network (*CNN*) adalah pengembangan dari *Multilayer Perceptron (MLP)* yang didesain untuk mengolah data dua dimensi. *CNN* termasuk dalam jenis *Deep Neural Network* karena kedalaman jaringan yang tinggi dan banyak diaplikasikan pada data citra.

Macam-macam *CNN* arsitektur yaitu :

a. LeNet-5(1998)

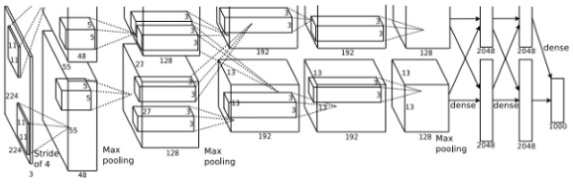
LeNet-5, jaringan konvolusional 7-level perintis oleh LeCun et al pada tahun 1998, yang mengklasifikasikan digit, diterapkan oleh beberapa bank untuk mengenali nomor tulisan tangan pada cek (cek) yang didigitalkan dalam input input skala abu-abu 32x32 pixel. Kemampuan untuk memproses gambar dengan resolusi lebih tinggi membutuhkan lapisan yang lebih besar dan lebih convolutional, sehingga teknik ini dibatasi oleh ketersediaan sumber daya komputasi.



Gambar 2.1. LeNet-5 arsitektur

b. AlexNet (2012)

AlexNet secara signifikan mengungguli semua pesaing sebelumnya dan memenangkan tantangan dengan mengurangi kesalahan top-5 dari 26% menjadi 15,3%. Tingkat kesalahan top-5 tempat kedua, yang bukan variasi *CNN*, adalah sekitar 26,2%.



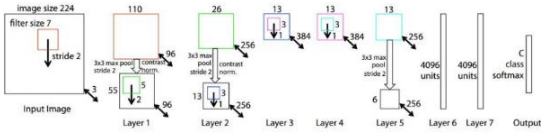
Gambar 2.2. Alexnet arsitektur

Jaringan ini memiliki arsitektur yang sangat mirip dengan LeNet oleh Yann LeCun et al tetapi lebih dalam, dengan lebih banyak filter per lapisan, dan dengan lapisan convolutional yang bertumpuk. Itu terdiri dari 11x11, 5x5,3x3, konvolusi, *Max Pool*ing, *dropou*t, augmentasi data,

aktivasi *Relu, SGD* dengan momentum. Ini melekat aktivasi *Relu* setelah setiap lapisan konvolusional dan terhubung sepenuhnya. AlexNet dilatih selama 6 hari secara bersamaan pada dua GPU Nvidia Geforce GTX 580 yang merupakan alasan mengapa jaringan mereka dipecah menjadi dua jaringan pipa. AlexNet dirancang oleh grup SuperVision, yang terdiri dari Alex Krizhevsky, Geoffrey Hinton, dan Ilya Sutskever.

c. ZFNet (2013)

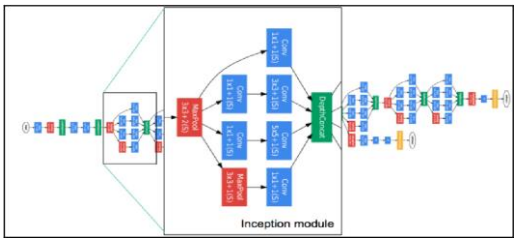
Pemenang ILSVRC 2013 juga merupakan *CNN* yang kemudian dikenal sebagai ZFNet. Ini mencapai tingkat kesalahan top-5 14,8% yang sekarang sudah setengah dari tingkat kesalahan non-saraf yang disebutkan sebelumnya. Itu sebagian besar merupakan sebuah pencapaian dengan men-tweak parameter-hyper dari AlexNet sambil mempertahankan struktur yang sama dengan elemen-elemen Belajar Dalam tambahan



Gambar 2.3. ZFNet arsitektur

d. GoogLeNet/Inception (2014)

Pemenang kompetisi ILSVRC 2014 adalah GoogLeNet (alias Inception V1) dari Google. Itu mencapai tingkat kesalahan top-5 dari 6,67%! sangat dekat dengan kinerja tingkat manusia yang sekarang dipaksa oleh panitia tantangan untuk dievaluasi. Ternyata, ini sebenarnya agak sulit dilakukan dan membutuhkan beberapa pelatihan manusia untuk mengalahkan ketepatan GoogLeNets. Setelah beberapa hari pelatihan, pakar manusia (Andrej Karpathy) mampu mencapai tingkat kesalahan top-5 sebesar 5,1% (model tunggal) dan 3,6% (ansambel). Jaringan menggunakan *CNN* yang terinspirasi oleh LeNet tetapi menerapkan elemen novel yang dijuluki modul awal. Ini digunakan normalisasi batch, distorsi gambar dan RMSprop. Modul ini didasarkan pada beberapa konvolusi yang sangat kecil untuk mengurangi jumlah parameter secara drastis.



Gambar 2.4. GoogLeNet arsitektur

e. VGGNet(2014)

Runner-up di kompetisi ILSVRC 2014 dijuluki VGGNet oleh komunitas dan dikembangkan oleh Simonyan dan Zisserman. VGGNet terdiri dari 16 lapisan konvolusional dan sangat menarik karena arsitekturnya yang sangat seragam.Mirip dengan AlexNet, hanya konvolusi 3x3, tetapi banyak filter. Dilatih dengan 4 GPU selama 2-3 minggu. Saat ini pilihan paling disukai di komunitas untuk mengekstraksi fitur dari gambar. Konfigurasi berat VGGNet tersedia untuk umum dan telah digunakan di banyak aplikasi dan tantangan lainnya sebagai ekstraktor fitur dasar. Namun, VGGNet terdiri dari 138 juta parameter, yang bisa sedikit menantang untuk ditangani

Gambar 2.5. VGGNet arsitektur

f. Residual Network (2015)

Residual networks (Resnet) Ini adalah arsitektur pemenang ILSVRC2015 dengan 152 lapisan. Kontribusi utamanya adalah menggunakan normalisasi batch dan koneksi loncatan khusus untuk melatih arsitektur yang lebih dalam. *Resnet* dengan 1000 lapisan dapat dilatih dengan teknik-teknik tersebut. Namun, secara empiris menemukan bahwa *Resnet* biasanya beroperasi pada blok kedalaman yang relatif rendah (20 - 30 lapisan), yang bertindak secara paralel, daripada secara serial mengalir ke seluruh panjang jaringan.

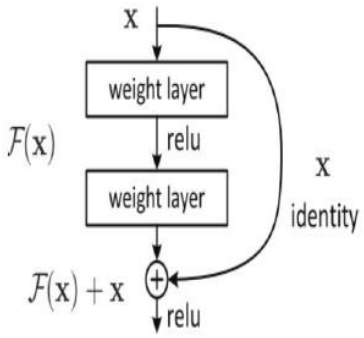
Deep residual networks (*Resnets*) terdiri dari banyak "*Residual Units"* yang ditumpuk. Setiap unit dapat dinyatakan dalam bentuk umum:

$$\boldsymbol{yl} = \boldsymbol{h}(\boldsymbol{xl}) + \boldsymbol{F}(\boldsymbol{xl}, \boldsymbol{wl}) , \boldsymbol{xl} + \mathbf{1} = \boldsymbol{f}(\boldsymbol{yl}) \text{.....(1)}$$

xl dan xl + 1 adalah input dan *output* dari unit ke-l, dan F adalah fungsi residual. Dalam, h (xl) = xl adalah pemetaan identitas dan f adalah fungsi *Relu*. *wl* adalah satu **R** set bobot (dan bias) yang dikaitkan dengan *Unit Residual* ke-l. *Resnets* yang lebih dari 100 lapisan telah menunjukkan akurasi *state-of-the-art* untuk beberapa tugas pengenalan yang menantang *di ImageNet* dan *MS COCO* kompetisi.

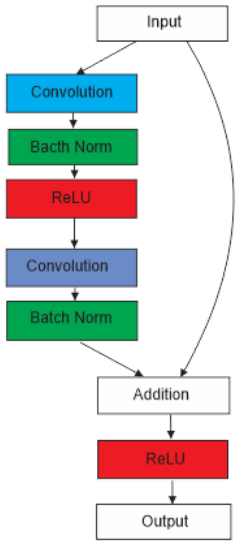
Gagasan utama *Resnets* adalah untuk mempelajari fungsi residual aditif F sehubungan dengan h (xl), dengan pilihan kunci menggunakan pemetaan identitas h (xl) = xl. Ini diwujudkan dengan melampirkan koneksi lewati identitas ("jalan pintas").

Ide inti dari *Resnet* adalah memperkenalkan apa yang disebut "koneksi pintas identitas" yang melompati satu atau lebih lapisan, seperti yang ditunjukkan pada gambar berikut:



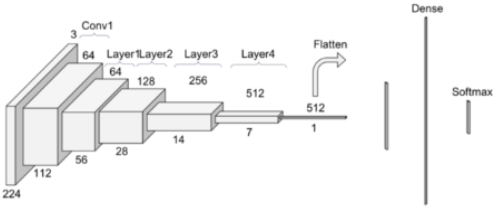
Gambar 2.6. Proses pemintasan blok

Para peneliti berpendapat bahwa susunan lapisan seharusnya tidak menurunkan kinerja jaringan, karena *resnet* dapat menumpuk pemetaan identitas (layer yang tidak melakukan apa-apa) pada jaringan dan arsitektur yang dihasilkan akan melakukan hal yang sama. Ini menunjukkan bahwa model yang lebih dalam seharusnya tidak menghasilkan kesalahan pelatihan yang lebih tinggi dari pada model yang lebih rendah.



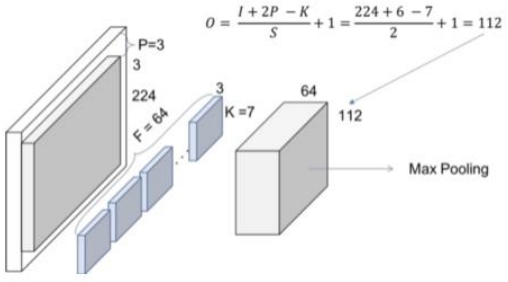
Gambar 2.7. Pemintasan *Resnet*

Resnet bekerja secara parallel yaitu pada saat diberi inputan jalur yang dilalui ada 2 yaitu pertama lurus ke proses *convolutional, Batch Norm, Fungsi Activation Relu*, dilanjutkan dengan *Convolution* lagi dan *Bacth Norm* dan yang kedua menuju addition pada proses ini dilakukan proses menggabungkan layer input yang dimasukan dan digabungkan dengan proses *convolutional* kemudian diteruskan kembali oleh *Relu* dan mengeluarkan hasil berupa *output*(faris, 2018).



Gambar 2.8. *layer resnet*

Langkah pertama pada *Resnet* sebelum memasuki perilaku lapisan umum adalah blok - disebut di sini Conv1 - terdiri dari konvolusi + normalisasi batch + operasi penyatuan maks Jadi, pertama ada operasi konvolusi. Pada Gambar 2.3 dapat dilihat bahwa *resnet* menggunakan ukuran kernel 7, dan ukuran peta fitur 64.



Gambar 2.9. *konvolusi*

Dapat disimpulkan bahwa *resnet* telah diisi dengan nol 3 kali pada setiap dimensi. Dengan mempertimbangkan hal ini, dapat dilihat pada Gambar 4 bahwa ukuran *output* dari operasi itu adalah volume (112x122). Karena setiap filter konvolusi (dari 64) menyediakan satu saluran dalam volume *output*, selanjutnya berakhir dengan volume *output* (112x112x64) - perhatikan ini bebas dari dimensi bets untuk menyederhanakan penjelasan.

2. Python

Python adalah bahasa pemrograman model skrip (*scripting language*) yang berorientasi obyek. *Python* dapat digunakan untuk berbagai keperluan pengembangan perangkat lunak dan dapat berjalan di berbagai *platform* sistem operasi. *Python* merupakan bahasa pemrograman yang freeware atau perangkat bebas dalam arti sebenarnya, tidak ada batasan dalam penyalinannya atau mendistribusikannya, lengkap dengan *source* codenya, debugger dan profiler, antarmuka yang terkandung di dalamnya untuk pelayanan antarmuka, fungsi sistem, *GUI* (antarmuka pengguna grafis), dan basis datanya (Kumar, 2015).

Adapun *Library python* yang digunakan dalam penelitian ini adalah :

a. *Keras*

Keras adalah jaringan jaringan saraf tingkat tinggi, yang ditulis dengan *Python* dan mampu berjalan di atas *TensorFlow* , CNTK , atau Theano. Ini dikembangkan dengan fokus pada memungkinkan eksperimen cepat. Mampu pergi dari ide ke hasil dengan penundaan yang

paling mungkin adalah kunci untuk melakukan penelitian yang baik. Pada tahun 2017, tim *TensorFlow* Google memutuskan untuk mendukung *Keras* di perpustakaan inti *TensorFlow* . Chollet menjelaskan bahwa *Keras* dipahami sebagai antarmuka daripada kerangka pembelajaran mesin mandiri. Ini menawarkan rangkaian abstraksi yang lebih tinggi dan lebih intuitif yang membuatnya mudah untuk mengembangkan model pembelajaran mendalam terlepas dari backend komputasi yang digunakan.

b. *NumPy (Numerical Python)*

Numpy adalah pustaka aljabar linier dalam *Python*. Ini adalah pustaka yang sangat penting tempat hampir setiap sains data atau mesin mempelajari paket *Python* seperti SciPy (Scientific *Python*), Mat − plotlib (plotting library), Scikit-learning, dll tergantung pada batas yang wajar. NumPy sangat berguna untuk melakukan operasi matematika dan logis pada Array. Ini menyediakan banyak fitur berguna untuk operasi pada n-array dan matriks dalam *Python*. (aigiomawu, 2018)

c. *Cv2* atau *OpenCV*

Cv2 adalah sebuah *library* fungsi pemrograman yang ditujukan untuk *computer vision.* Awalnya dikembangkan oleh pusat penelitian Intel di Nizhny Novgorod (Rusia), kemudian didukung oleh Willow Garage dan sekarang dikelola oleh Itseez. *Library OpenCV* di bawah lisensi BSD *open-source* gratis dan cross-platform untuk digunakan. Didalamnya terdapat ratusan algoritma *computer vision.*

d. *Matplotlib*

Matplotlib adalah pustaka penyusunan 2D untuk bahasa pemrograman *Python* yang menghasilkan angka kualitas publikasi dalam berbagai format hardcopy serta lingkungan interaktif di seluruh *platform. Matplotlib* mencoba untuk membuat hal-hal mudah mudah serta hal-hal yang sulit dan menyediakan API berorientasi objek untuk menanamkan plot ke dalam aplikasi menggunakan toolkit GUI tujuan umum seperti wx*Python*, Qt, atau GTK +. *Matplotlib* terutama ditulis oleh John D. Hunter, memiliki komunitas pengembangan aktif, dan didistribusikan di bawah lisensi gaya BSD (Kumar, 2015)

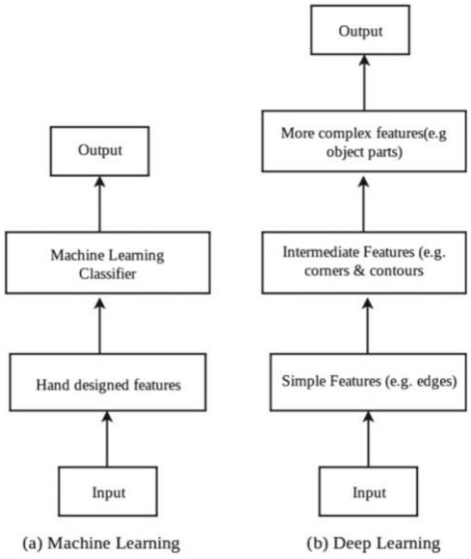
2.2.1. *Deep Learning*

Deep Learning adalah area baru dari *machine learning* yang semakin populer pada dekade ini. Istilah *Deep Learning* merujuk pada arsitektur ANN yang terdiri dari beberapa *hidden layers* (*Deep Networks*) yang berfungsi untuk mengenali fitur-fitur yang berbeda pada *input*. algoritma *Deep Learning* bertujuan untuk mencari struktur yang tidak diketahui untuk mendapatkan representasi yang baik dari *input* (Wani et al., 2018).

Metode *machine learning* yang konvensional terbatas dalam cara memproses data mentah. Dalam waktu yang lama untuk dapat mengenal pola menggunakan metode *machine learning* membutuhkan keahlian khusus dalam bidang-bidang tertentu dan membutuhkan keahlian rekayasa yang sangat hati-hati sesuai kasus yang dihadapi untuk dapat mengekstraksi fitur dari sebuah *input*.

Namun dengan *Deep Learning* proses pengenalan pola dan rekayasa secara manual yang membutuhkan keahlian dalam domain tertentu tersebut dapat dilakukan secara otomatis selama proses pembelajaran. Dalam proses pembelajaran (*training*) metode *Deep Learning* dapat mengenali fitur-fitur yang ada pada data mentah secara otomatis dengan hasil yang lebih baik dari cara manual (Wani et al., 2018).

Metode pembelajaran fitur pada Deep Learning dilakukan dengan membentuk layer hirarki dimana setiap layer dibangun diatas layer yang lain. Layer yang paling bawah dari model ini bertanggung jawab untuk mempelajari representasi dasar dari masalah, dan layer-layer diatasnya bertanggung jawab untuk membentuk konsep yang lebih kompleks dari data. Sebagai contoh dalam kasus pengenalan wajah pada gambar, setiap pixel yang ada pada gambar akan dimasukkan ke hirarki layer. Setiap hidden layer pada struktur hirarki ini kemudian akan mengekstrak fitur-fitur dari gambar input. Pada layer pertama hirarki ini akan mendeteksi tepi-tepi dari wajah, kemudian pada pada layer kedua akan dipelajari garis-garis pipi, alis, cekungan pada dagu, pada layer berikutnya garis-garis yang telah dipelajari akan membentuk gambaran yang lebih abstrak dari gambar. Semua pembelajaran yang dilakukan pada setiap layer ini dilakukan secara otomatis tanpa campur tangan manusia. Secara visual perbedaan tehnik *machine learning* konvesional dan *deep learning* bisa dilihat pada gambar 2.6 .



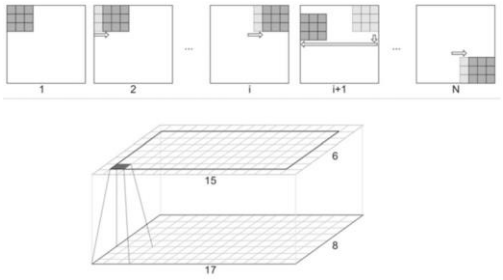
Gambar 2.11. Perbedaan *Machine Learning* dan *Deep Learning*

Adapun layer ekstraksi fitur yang bisa digunakan pada arsitektur *Deep Learning* adalah sebagai berikut :

Layer Konvolusi

Layer Konvolusi (*Convolutional Layer*) adalah layer operasi konvolusi 2D, dimana dengan operasi konvolusi setiap blok wilayah pada gambar dapat dipelajari dan menjadi fitur dari gambar tersebut. Komponen dari operasi konvolusi adalah matriks persegi yang disebut filter/kernel umumnya berdimensi 3x3, filter/kernel pada operasi konvolusi berfungsi untuk mengekstrak fitur setiap wilayah yang ada pada gambar. Dalam proses *training* filter/kernel akan menjadi *learnable parameter* yaitu setiap nilai matriks kernel akan dipelajari pada saat *training*. Pada setiap layer konvolusi jumlah kernel/filter bisa lebih dari satu sesuai kompleksitas dan jumlah fitur yang ingin dipelajari dari input gambar. Sebagai contoh jika pada sebuah gambar dengan satu kanal warna ingin dipelajari 64 jenis fitur maka pada

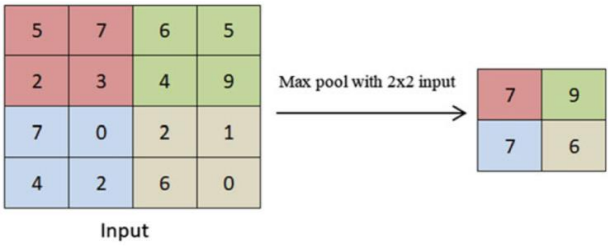
layer *convolutional* bisa memiliki 64 jenis kernel/filter dengan dimensi $R \times C$ sehingga hasil dari operasi layer *convolutional* adalah 64 gambar atau juga disebut 64 kanal.(Skansi, 2018) Ilustrasi proses konvolusi bisa dilihat pada gambar 2.7 dibawah.



Gambar 2.12. Ilustrasi Proses Konvolusi

Max Pool

Layer *Max Pool* pada lapisan jaringan ANN berfungsi untuk mereduksi ukuran gambar dengan metode *down sampling*. Layer *Max Pool* akan menyimpulkan nilai tertinggi pada gambar dalam ukuran region tertentu (Wani et al., 2018). Sebagai contoh gambar 2.8 dibawah operasi *Max Pool* akan mengambil nilai tertinggi pada gambar dalam setiap 2x2 wilayah pada gambar.



Gambar 2.13. Ilustrasi Operasi *Max Pool*

Batch Normalization

Proses *Batch Normalization* (BN) pada lapisan ANN berfungsi untuk menormalisasi skala nilai *input*. Proses *training* menambahkan lapisan BN dapat meningkatkan generalisasi pada model sehingga model dapat belajar dengan baik tanpa *overfit* (Ioffe & Szegedy, 2015). Proses BN bisa dilihat dalam persamaan dibawah.

$$x_i^* = x_i \frac{M(x)}{\sqrt{V(x)}} \quad \text{.....(2)}$$

Dimana:

x_i^* adalah hasil proses BN

x_i adalah data yang akan dinormalisasi

$M(x)$ adalah nilai rata-rata pada *batch*

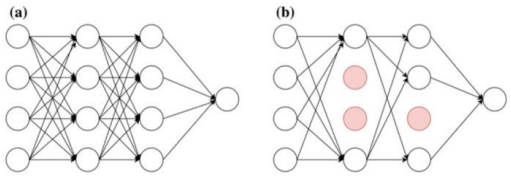
$V(x)$ adalah nilai variansi dalam batch

Menumpuk layer Konvolusi dengan *Max Pool* menjadi beberapa layer disebut dengan *CNN* (*Convolutional Neural Network*).

Dropout

Lapisan jaringan ANN yang terdiri dari beberapa lapis yang kompleks dapat membuat model mempelajari fitur yang rumit pada data, setiap fitur yang dipelajari akan tersambung ke sejumlah *neuron* atau disebut *Fully Connected Layer* (FC) untuk membuat kesimpulan dari fitur yang didapat. Layer FC akan saling terhubung dengan layer FC yang lain dan sangat mudah mengalami *overfit*. *Overfit* adalah keadaan dimana model mempelajari data *training* dengan sangat baik namun gagal dalam memprediksi kesimpulan dari data baru, untuk mengatasi hal ini layer *dropout* dapat ditambahkan pada lapisan ANN. Layer *dropout* akan memutus secara acak sambungan FC pada saat *training* sehingga hanya sambungan yang masih terhubung yang akan

dilatih pada saat *training* (Wani et al., 2018). Ilustrasi proses *dropout* bisa dilihat pada gambar 2.9 dibawah.



Gambar 2.14. Ilustrasi Proses *Dropout*

3. *Training*

training dalam *Deep Learning* adalah proses melatih model untuk mempelajari data untuk mendapatkan hasil yang diinginkan. Dalam proses *training* ada beberapa komponen yaitu:

a. Epoch

Epoch dalam *training* merujuk pada berapa kali semua data *training* akan digunakan untuk belajar. Menggunakan data *training* dalam melatih model tidak cukup hanya sekali, dibutuhkan beberapa kali agar model dapat belajar dengan baik.

b. Batch Size

Batch Size dalam *training* merujuk pada jumlah data yang akan diproses untuk latihan dalam satu waktu. Metode *deep learning* pada umumnya membutuhkan data dalam jumlah besar agar dapat belajar dengan baik. Dalam proses belajar, memasukkan keseluruhan data ke dalam jaringan dalam waktu bersamaan akan memakan banyak sumber daya memory dan operasi komputasi. Agar dapat diproses dengan baik data yang dalam jumlah banyak akan dipecah menjadi beberapa bagian yang disebut dengan *mini batch* dimana jumlah data setiap *mini batch* disebut dengan *Batch Size* dan pemrosesan setiap *batch* disebut dengan iterasi. Umumnya ukuran *Batch Size* yang digunakan adalah kelipatan 2 misal 32, 64, 128 jumlah data.

4. Learning Rate

Tingkat belajar adalah salah satu parameter hiper paling penting ketika datang ke pelatihan jaringan saraf. Ini menentukan besarnya pembobotan (atau parameter)
pembaruan. Ini juga merupakan parameter tersulit yang harus ditetapkan karena dapat berdampak signifikan pada kinerja model. Semakin rendah nilainya, semakin lambat *resnet* melakukan pengecekan data. Nilainya berkisar antara $0 < lr < 1$.

Jika tingkat pembelajaran disetel terlalu rendah, kemajuan pelatihan tidak efisien karena memakan waktu dengan pembaruan bobot yang kecil. Jika tingkat pembelajaran ditetapkan terlalu tinggi, itu dapat menyebabkan perilaku yang berbeda dalam *loss function* (Janocha, 2017)

5. AI Framewk

a. TensorFlow

TensorFlow adalah platform *open source end to end* untuk pembelajaran mesin. Ini memiliki ekosistem alat, perpustakaan, dan sumber daya komunitas komprehensif yang fleksibel, yang memungkinkan para peneliti mendorong teknologi mutakhir dalam ML dan pengembang dengan mudah membangun dan menggunakan aplikasi bertenaga ML.

Fungsi *Tensor Flow* adalah :

Ini berguna untuk membuat dan bereksperimen dengan arsitektur pembelajaran yang mendalam, dan formulasinya nyaman untuk integrasi data seperti memasukkan grafik, tabel SQL, dan gambar secara bersamaan.

itu didukung oleh Google yang menjamin itu akan tetap ada untuk sementara waktu, maka masuk akal untuk menginvestasikan waktu dan sumber daya untuk mempelajarinya.

TF beroperasi dengan grafik perhitungan statis **.** Yaitu, pertama-tama mendefinisikan grafik, selanjutnya menjalankan perhitungan dan, jika perlu membuat perubahan pada arsitektur dan melatih kembali model. Pendekatan seperti itu dipilih demi efisiensi, tetapi banyak alat jaringan saraf modern dapat memperhitungkan perbaikan dalam proses pembelajaran tanpa kehilangan yang signifikan dalam kecepatan belajar. Dalam hal ini, pesaing utama *TensorFlow* adalah ***PyTorch*** .

b. Keras

Keras adalah pustaka neural-network open-source yang ditulis dengan Python . Itu mampu berjalan di atas *TensorFlow* , Microsoft Cognitive Toolkit,Theano, atau PlaidML. Dirancang untuk memungkinkan eksperimen cepat dengan jaringan saraf yang dalam , *Keras* berfokus untuk menjadi ramah pengguna, modular, dan dapat dikembangkan. *Keras* berisi banyak implementasi dari blok-blok pembangun jaringan-saraf yang biasa digunakan seperti lapisan, tujuan , fungsi aktivasi , pengoptimal , dan sejumlah alat untuk membuat bekerja dengan data gambar dan teks lebih mudah. Selain jaringan saraf standar, *Keras* memiliki dukungan untuk jaringan saraf konvolusional dan berulang . Ini mendukung lapisan utilitas umum lainnya seperti dropout, normalisasi batch, dan pengumpulan.

c. PyTorch

PyTorch adalah perpustakaan pembelajara nmesin sumberterbuka berdasarkan perpustakaan T orch , digunakan untuk aplikasi seperti visi komputer dan pemrosesan bahasa alami. Ini terutama dikembangkan oleh kelompok riset kecerdasan buatan Facebook . Ini adalah perangkat lunak bebas dan sumber terbuka yang dirilis di bawah lisensi BSD yang imodifikasi . Meskipun antarmuka Python lebih dipoles dan fokusutam apengembangan , *PyTorch*j uga memiliki antarmuka C++ Selanjutnya, perangkat lunak bahasa pemrograman probabilistik Uber Pyro menggunakan *PyTorch* sebagai backend. (Pytorch n.d, 2019)

d. MXNet

MXNet adalah framework mendalam yang sangat skalabel dan dapat digunakan pada berbagai perangkat. Meskipun tampaknya tidak banyak digunakan dibandingkan dengan *TensorFlow*, pertumbuhan MXNet kemungkinan akan didorong dengan menjadi proyek Apache

METODOLOGI PENELITIAN

pada bab ini pertama dilakukan studi literatur untuk mengetahui prinsip kerja dari Residual network (Resnet). Selanjutnya dilakukan analisa kebutuhan sistem yaitu analisa kebutuhan perangkat keras dan analisa kebutuhan perangkat lunak untuk mengetahui kebutuhan system dan menghindari hal yang tidak diinginkan selain itu dalam bab ini akan dibahas bagaimana model perancangan Resnet serta bagaimana implementasinya pada training data yang diharapkan dapat meningkatkan akurasi pengenalan wajah pada dataset Yale dan yang terakhir dilakukan pengujian system untuk mengetahui seberapa besar akurasi peninggalan wajah yang didapatkan.

PEMBAHASAN

Pada penerapan *ResNet* akan dibahas tentang langkah-langkah pembuatan dan hasil pengujian sehingga menghasilkan aplikasi sistem pengenalan wajah

4.1. Konfigurasi Hadware dan Sofware

Proses *training* pada penelitian ini menggunakan layanan dari *IBM CLOUD* dapat diakases di https://www.labs.cognitiveclass.ai dengan konfigurasi s*oftware* dan hadware sebagai berikut:

SOFTWARE	Jupyter lab
	Python v3.7.1
	KERAS Framework
	Anaconda Library
HADWARE	PC ARMAGEDON
	RAM 16 GB
	CORE I7

4.2. Pengolahan Data

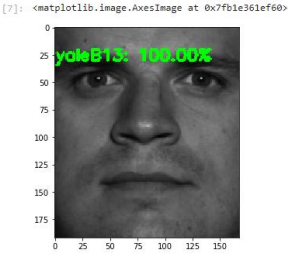
Gambar 4.1. Resize gambar

Pada penelitian ini dilakukan beberapa tahapan sebelum melakukan proses pelatihan yaitu proses *preprosessing* dengan mengubah *size* gambar menjadi 32x32 pixel. Hasil gambar yang diubah selanjutnya dilakukan pembagian dengan perbandingan 80% untuk data *training* dan 20% untuk data *testing* untuk mendapatkan nilai akurasi yang baik.

Gambar 4.2. Pembagian dataset

Terdapat 2 perbedaan akurasi yaitu akurasi pada data *training* dan akurasi pada data *testing*. Akurasi pada data *training* adalah akurasi yang bertujuan untuk mengetaui seberapa baik kinerja dari algoritma yang dibuat dan akurasi pada data *testing* bertujuan untuk memvalidasi seberapa baik algoritma yang sudah dilatih mendapatkan hasil pengenalan wajah yang baik. Selanjutnya data yang sudah dibagi dilakukan normalisasi data untuk mendapatkan nilai pixel dengan rentang 0 dan 1.

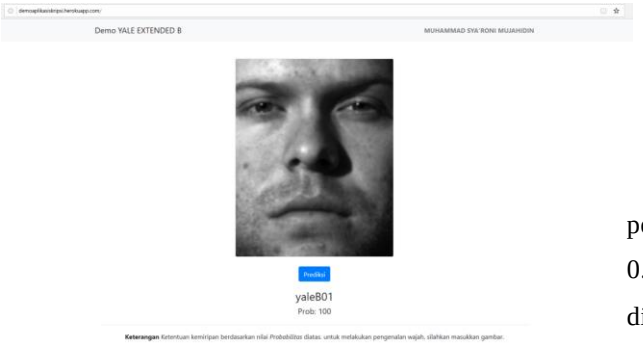
Nilai pixel dari hasil normalisasi data selanjutnya akan dimasukkan pada layer *neural network* yaitu menggunakan *resnet.* Tahap akhir dilakukan pengaturan parameter yaitu mengubah laju pembelajaran untuk mengetahui laju pembelajaran yang sesuai dan *training* dapat dilakukan dengan mengatur epoch atau jumlah belajar per keseluruhan data yang selanjutnya hasil training dapat dilakukan *testing* dengan menggunakan data pada folder *testing.* hasil dari testing dilakukan di lokal komputer dan dapat dilihat pada gambar 4.3



Gambar 4.3 Testing pada lokal komputer

4.3. Antarmuka Aplikasi

Dalam pembuatan aplikasi berbasis web ini menggunakan bahasa pemrogramman python dengan framework Flask serta menggunakan BootStrap sebagai Front-end untuk mempermudah dalam pengaturan tata letak text dan gambar. Tampilan aplikasi pengenalan wajah dapat dilihat pada gambar 4.4.



Gambar 4.4. Antarmuka aplikasi

Pada gambar 4.4 dilakukan pendeteksian gambar menggunakan dataset dengan label yaleB01. dari gambar diatas dapat disimpulkan aplikasi sistem pengenalan wajah dapat mengenali secara baik gambar di inputkan kedalam aplikasi dengan probabilitas 100%. aplikasi ini bekerja dengan memanfaatkan model yang sudah di training dengan format .h5 yang menyimpan bobot dari data yang sudah di *training*

4.4. Hasil Pengujian

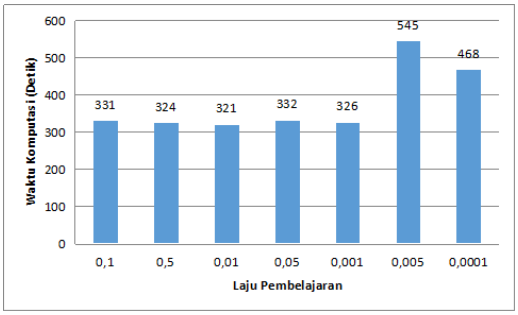
Pelatihan (*training)* adalah peroses melatih data untuk dapat memahami informasi yang terdapat didalam data. Untuk memperoleh hasil pengenalan wajah terbaik maka dilakukan pengaturan parameter yaitu laju pembelajaran dan jumlah iterasi. hasil yang didapatkan pada perubahan parameter yaitu dapat dilihat pada tabel 4.1

Tabel 4.1 Hasil pengujian algoritma *resnet* dengan epoch 50

Laju Pembelajaran	Waktu Komputasi (Detik)	Akurasi (%)
0.1	331	2
0.5	324	3
0.01	331	99
0.05	332	99
0.001	326	99
0.005	545	99
0.0001	564	92

Pada tabel 4.1 dilakukan perubahan laju pembelajara (*learning rate)* yaitu 0.1, 0.5, 0.01, 0.05, 0.001, 0.05 dan 0.0001. selanjutnya dilakukan pelatihan (*training)* data dengan jumlah belajar 50 *iterasi* (*epoch)* dan jumlah data latihan yang digunakan 1600 data. Langkah pertama yaitu memahami informasi pada data *training* yaitu dengan mengubah laju pembelajaran dan membandingkan hasil laju pembelajaran terhadap waku komputasi, dan laju pembelajaran terhadap akurasi (*acc).* Hal ini bertujuan untuk mendapatkan laju pembelajaran yang tepat sehingga didapatkalah akurasi terbaik.

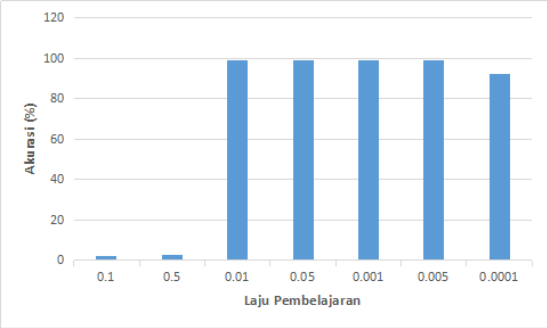
4.4.1. Laju Pembelajaran Terhadap Waktu Komputasi



Gambar 4.5. Diagram laju pembelajaran terhadap waktu komputasi

Pada gambar 4.5 adalah diagaram perbandingan laju pembelajaran terhadap waktu komputasi. Dari diagram diatas dapat disipmulkan perubahan laju pembelajaran mempengaruhi waktu komputasi yang dapat dilihat dari laju pembelajara 0.0001 memerlukan waktu komputasi yang cukup lama dibandingkan dengan laju pembelajan 0.005. Waktu komputasi semakin cepat seiring dengan semakin melambatnya laju pembelajaran yaitu dari 0.005 ke laju pembelaran 0.01 tetapi pada laju pembelajaran 0.1 terjadi perubahan waktu komputasi yaitu 331 detik. hal ini disebabkan oleh data *training* yang cukup lama diperores oleh algoritma karena nilai pixel yang diperoses oleh laju pembelajaran memperlambat pelatihan pada data *training.*

4.4.2. Laju Pembelajaran Terhadap Akurasi (%)



Gambar 4.6. Laju pembelajaran terhadap akurasi

Gambar 4.6 adalah diagram perbandinan laju pembelajaran terhadap akurasi pada data *trainng*. Dari gambar diatas dapat disimpulkan bahwa dengan semakin melambatnya laju pembeajaran yaiu dimula dari laju pembelajaran 0.0001 menuju 0.005 didapatkan nilai akurasi pengujian algoritma pada data *training* yaiu semakin meningkat dari 92% menuju 99% dan pada laju pembelajaran 0.005 menuju 0.01 didapatkan akurasi yang konstan sebesar 99% dapat disimpulkan bahwa akurasi pengenalan yang baik berada pada lerning 0.005-0.01 karena mendaatkan akurasi sebesar 99%. Dari hasil peruban learning rate diatas dapat disimpulkan bahwa pengujin algoritma yang terbak berada pada laju pembelajaran 0.01 dengan persentase kesalahan yaitu 1.1% dan akurasi yaitu 99%. **Maka dapat disimpulkan dari keseluruhan hasil pengujian. Parameter yang terbaik yaitu dengan Laju pembelajaran 0.001, waktu komputasi 326 detik dengan akurasi dalam memahami data 99%**

4.5. *Testing*

Tahap selanjutnya adalah melakukan *testing* data yang sudah di latih. Pada tahap ini dataset dibagi menjadi 8 pengujian dengan merubah jumlah perbandingan data *testing* terhadap data *training* . perbandingan dataset dibagi menjadi total jumlah dikali dengan persentase 15% ,20%, 25%,30%,35%,40%,45% dan 50 % . hal ini bertujuan untuk melihat apakah terdapat pengaruh pembagian data terhadap akurasi pada data *testing* yang selanjutnya dari hasil perubahan jumlah dataset akan diperoleh akurasi pengenalan wajah..

Tabel 4.2. Tabel Validation (*Testing)*

No urut Pengujian	Data Pengujian		Akurasi (%)
	Jumlah/per sentase	Persentase (%)	
1	240	15	87
2	315	20	92
3	400	25	86
4	480	30	84
5	560	35	84
6	640	40	80
7	720	45	78
8	800	50	76

Pada tabel 4.2 dilakukan 8 kali pengujian dengan parameter jumlah data yang dirubah mulai

dari 15% sampai 50 %. Dari tabel diatas dapat disimpulkan jumlah datas jumlah data akan mempengaruhi akurasi pengenalan wajah yaitu pada nomor urut pengujian 2. dapat dilihat dari hasil pengujian yang dilakukan 8 kali. Pada saat *testing* berkurang secara otomatis data *training* akan meningkat. Peningkatan data *training* memberikan dampak akan semakin banyak data yang dilatih sehingga *resnet* dapat mempelajari data semakin baik. Akan tetapi pada saat data *testing* bertambah jumlahnya, hasil akurasi yang didapatkan menurun. Hal ini dikarenakan data yang dilatih berkurang sehingga dapat dapat disimpulkan perubahan data pasti berpengaruh terhadap hasil akurasi pengenalan wajah dan **akurasi pengenalan wajah pada data testing adalah 92%**

PENUTUP

5.1. Kesimpulan

1. Hasil perubahan laju pembelajaran terhadap parameter waktu komputasi adalah perubahan laju pembelajaran mempengaruhi waktu komputasi yang dapat dilihat dari laju pembelajara 0.0001 memerlukan waktu komputasi yang cukup lama dibandingkan dengan laju pembelajan 0.005. Waktu komputasi semakin cepat seiring dengan semakin melambatnya laju pembelajaran
2. Hasil validasi pengenalan wajah dengan merubah variasi jumlah data adalah jumlah data akan mempengaruhi akurasi pengenalan wajah yaitu pada nomor urut pengujian 2. dapat dilihat dari hasil pengujian yang dilakukan 8 kali. Pada saat *testing* berkurang secara otomatis data *training* akan meningkat. Peningkatan data *training* memberikan dampak akan semakin banyak data yang dilatih sehingga *resnet* dapat mempelajari data semakin baik.
3. Perbandingan hasil akurasi pada data testing. Penelitian sebelumnya menggunakan *Convolutionan neural network* mendapatkan akurasi pengenalan wajah sebesar 75,79 % dan pada penelitian ini menggunakan *Residual Network* mendapatkan akurasi pengenalan wajah sebesar 92% sehingga dapat disimpulkan algoritma *Residual Network* dapat meningkatkan akurasi pengenalan wajah

5.2. Saran

1. Dapat dikembangkan menjadi sebuah aplikasi yang dapat digunakan di kehidupan nyata.
2. Kedalaman layer dapat dimaksimalkan hingga kedalaman 152 layer
3. Pelatihan data dapat menggunakan *cloud* yaitu Google Colab untuk mengantisipasi kerusakan *hadware*.
4. Dataset yang digunakan dapat berupa dataset yang dibuat sendiri untuk menguji algoritma *resnet* dengan dataset berbeda

DAFTAR PUSTAKA

Abhirawan, H., Jondri, & Arifianto, A. (2017). Pengenalan Wajah Menggunakan *Convolutional Neural Networks* (*CNN*). *Universitas Telkom*, *4*(3), 4907–4916.

Faris (2018). *Implementasi Algoritma CNN resnet untuk pengenalan tanda tangan*. Tugas Akhir. Program Studi Teknik Informatika Universitas Bumigora Mataram.

Indra. (2012). Sistem Pengenalan Wajah Dengan Metode Eigenface Untuk Absensi Pada PT Florindo Lestari. *Journal Pendidikan*

Teknologi-Informasi & Komunikasi Terapan, *2012*(Semantik), 138.

Ioffe, S., & Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network training by Reducing Internal Covariate Shift. Retrieved from http://arxiv.org/abs/1502.03167

Janocha, Katarzyna and Wojciech Marian Czarnecki. 2017. "On Loss Functions for Deep *Neural Network*s in Classification." 1–10.

Pytorch. (n.d.). torchvision.models — PyTorch master documentation. Retrieved July19,2019,from,https://pytorch.org/docs/stable/torchvision/models.html?highlight=resnet \

Qiong Cao, Shen, L., Weidi Xie, Parkhi, O. M., & Zisserman, A. (2018). VGGFace2: A dataset for recognising faces across pose and age, 826.

Ranjan, R., Castillo, C. D., & Chellappa, R. (2017). L 2 -constrained Softmax Loss for Discriminative Face Verification.

Schroff, F., Kalenichenko, D., & Philbin, J. (2015). FaceNet: A unified embedding for face recognition and clustering. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *07-12-June*, 815–823. https://doi.org/10.1109/CVPR.2015.7298682

Schroff, F., & Philbin, J. (n.d.). FaceNet : A Unified Embedding for Face Recognition and Clustering.

Sehman. (2015). Penerapan Face Recognition dengan Metode EigenFace pada Intelligent Car Security. *Seminar Nasional Inovasi dalam Desain dan Teknologi*, 342–348.

Sebastian Raschka. (2015). Single-Layer Neural Networks and Gradient Descent. Retrieved July 27, 2019, from https://sebastianraschka.com/Articles/2015_singlelayer_neurons.html

Skansi, S. (2018). *Introduction to deep learning: Part 1*. *Chemical Engineering Progress* (Vol. 114). https://doi.org/10.1007/978-3-319-73004-2

Soares, F., & Souza, A. M. F. (2016). *Neural network programming with Java : unleash the power of neural networks by implementing professional Java code*.

Udacity Course. (2018). Secure and Private AI Scholarship Challenge - Udacity. Retrieved July 27, 2019, from https://classroom.udacity.com/nanodegrees/nd185/parts/3fe1bb10-68d7-4d84-9c99-9539dedffad5/modules/28d685f0-0cb1-4f94-a8ea-2e16614ab421/lessons/d9869c40-de54-4395-9d5f-fa13c8254277/concepts/70526adf-40d3-4446-ac32-d3f798739745

Wani, M. A., Bhat, F. A., Afzal, S., & Khan, A. I. (2018). *Advances in Deep Learning on Graphs*.

Zhang, K., Zhang, Z., Li, Z., Member, S., Qiao, Y., & Member, S. (2016). Joint Face Detection and Alignment using Multi-ta, (1), 1–5. https://doi.org/10.1109/LSP.2016.2603342