PENGECAMAN AKSARA CINA TULISAN TANGAN MENGGUNAKAN MODEL PEMBELAJARAN PEMINDAHAN BERASASKAN VGG16 YANG DIPERHALUSI

LIM ZI SIN
MOHD ZAMRI MURAH

Fakulti Teknologi & Sains Maklumat, Universiti Kebangsaan Malaysia

ABSTRAK

Bahasa mandarin telah menjadi satu daripada bahasa yang terkenal untuk dipelajari kerana pertumbuhan berterusan China di landasan untuk menjadi ekonomi nombor satu di pentas dunia menjelang tahun 2030. Di samping itu, pembelajaran bahasa mandarin juga akan memudahkan kita untuk berkomunikasi dengan orang lain pada tahap yang lebih mendalam terutamanya di Malaysia. Oleh itu, alat pengecaman aksara optik secara tidak langsungnya telah menjadi satu daripada alatan yang mampu meningkatkan penguasaan pengguna terhadap aksara Cina dalam perjalanan mempelajari bahasa Mandarin. Namun begitu, setiap orang mempunyai gaya penulisan tersendiri dan alat pengecaman aksara optik yang tradisional tidak dapat mengecamkan setiap gaya tulisan yang berbeza. Maka satu algoritma pembelajaran mendalam yang kompleks amat diperlukan dalam menyelesaikan masalah ini. Projek ini bertujuan untuk menggunakan kaedah pembelajaran pemindahan bagi membina beberapa model Rangkaian Neural Pengkonvolutan jenis VGG16 pada dataset CASIA-HWDB untuk mengecamkan input aksara Cina tulisan tangan daripada pengguna dengan mengenal pasti faktor saiz input, *dropout*, *batch normalization*, dan pengoptimum yang sesuai, seterusnya memilih satu model VGG16 yang mempunyai ketepatan yang paling tinggi untuk diintegrasikan ke dalam aplikasi sistem bagi tujuan demonstrasi. Model asas VGG16 telah mencapai ketepatan sebanyak 84.44% manakala model VGG16 yang diperhalusi dengan menambahkan lapisan batch normalization, lapisan dropout, pengoptimum RMSProp, serta mempunyai saiz input 128 x 128 piksel telah mencapai ketepatan paling tinggi iaitu 90.73%.

1 PENGENALAN

Sejak penubuhan Republik Rakyat China pada tahun 1949, Kerajaan China telah mempromosikan penyederhanaan aksara Cina bagi menyeragamkan aksara yang sedia ada. Tindakan tersebut bertujuan untuk mengurangkan bilangan sebutan yang terkandung di dalam aksara serta keseluruhan bilangan aksara. Sebanyak 2263 aksara kompleks telah digantikan dengan 2235 aksara yang telah dipermudahkan bersamaan dengan satu per tiga daripada jumlah bilangan aksara yang digunakan. Selepas dipermudahkan, bilangan sebatan yang digunakan dalam aksara Cina telah dikurangkan sebanyak hampir 50 peratus. Keadaan ini secara tidak langsungnya telah meningkatkan kelajuan kita untuk menulis menggunakan aksara Cina. Aksara Cina yang dipermudahkan (simplified Chinese) telah memudahkan orang ramai menulis serta diterima pakai oleh seluruh dunia. Sebagai contohnya, Persatuan Bangsa Bersatu (PBB) menggunakan aksara Cina yang dipermudahkan di dalam segala dokumen versi Cina mereka.

Apabila teknologi maklumat telah bertambah maju, maka teknik pengecaman aksara Cina tulisan tangan telah pun menjadi semakin popular dalam kalangan mereka yang kurang jelas dengan penggunaan Pinyin yang juga dikenali sebagai skema huruf fonetik Cina. Pengecaman aksara Cina tulisan tangan telah menjadi topik yang hangat dalam bidang penyelidikan dan juga untuk aplikasi praktikal. Namun begitu, HCCR di mana ketidakpastian tidak terhad kepada contoh latihan tertentu aksara tulisan tangan, masih merupakan masalah yang menimbulkan cabaran yang ketara (Du et al. 2014). Hal ini demikian kerana tahap kepelbagaian yang tinggi bagi kedua-dua gaya tulisan dan set aksara yang agak besar. Kedua-dua faktor ini telah menjadikan pengecaman aksara Cina tulisan tangan jauh lebih rumit daripada bahasa yang berasaskan sistem penulisan abjad Rom atau Latin (Hildebrandt Et Al.1993). Bukannya mudah bagi para penyelidik untuk membangunkan sistem pengecaman aksara Cina tulisan tangan. Antara cabaran yang dihadapi oleh HCCR termasuklah bilangan aksara Cina tulisan tangan yang terlalu banyak dan ini akan menyebabkan mesin mengambil masa yang agak lama untuk menjalani latihan. Seterusnya, kepelbagaian aksara Cina tulisan tangan juga telah menambahkan kerumitan perkara tersebut. Kerumitan akan bertambah lagi apabila variasi penulisan yang besar telah berlaku dari seorang ke seorang.

Bagi menangani imej tulisan tangan, Rangkaian Neural Pengkonvolutan (CNN) telah digunakan secara meluas kerana ia mempunyai prestasi yang lebih baik dalam pengecaman imej berbanding dengan kaedah tradisional yang lain. Terdapat banyak experimen telah membuktikan bahawa CNN berfungsi baik dalam pengecaman aksara tulisan tangan. Walau bagaimanapun, model CNN yang sedia ada tampaknya tidak begitu berkesan dalam masalah yang lebih rumit. Sebaliknya, kesemuanya memerlukan saiz yang besar untuk simpanan. Seperti yang dilaporkan, ATR-CNN26 memerlukan saiz simpanan sebanyak 2.46GB untuk kamus manakala AlexNet12 memerlukan kira- kira 60 juta parameter untuk dilatih (Bi et al. 2019).

2 PENYATAAN MASALAH

Secara umumnya, memandangkan setiap orang mempunyai gaya penulisan mereka sendiri, alat pengecaman aksara Cina tulisan tangan yang tradisional tidak dapat mengecamkan tulisan tangan setiap orang dengan tepatnya. Selain menggunakan teknologi *Computer Vision*, algoritma pembelajaran mendalam yang sangat kompleks dan rumit amat diperlukan untuk

mengenal pasti semua variasi ini dengan jayanya. Hal ini demikian kerana alatan pengecaman aksara Cina tulisan tangan sering menghadapi pelbagai cabaran. Sememangnya kualiti imej yang lebih tinggi adalah penting untuk tujuan pengecaman imej, namun penyelesaian OCR perlu menangani pelbagai jenis kualiti imej. Hal ini demikian kerana imej teks tulisan tangan mempunyai tahap kualiti yang berbeza berdasarkan kamera yang digunakan. Secara ringkasnya, pelbagai jenis gaya penulisan dan ciri kompleks aksara tulisan tangan menjadikannya satu cabaran untuk mengklasifikasikan aksara tulisan tangan dengan tepat. Projek ini telah mendapat inspirasi daripada projek YuHao Zhang yang bertajuk *Deep Convolutional Neural Network for Handwritten Chinese Character Recognition*. Beliau telah menggunakan Rangkaian Neural Pengkonvolutan dengan kedalaman dan bilangan penapis yang berbeza pada set data yang berkelas 200. Walau bagaimanapun, beliau tidak meneroka model yang lebih kompleks dan lebih mendalam seperti VGG16. VGG16 telah dicadangkan dalam projek ini kerana ia merupakan algoritma pengesanan dan pengelasan objek yang mampu mengklasifikasikan 1000 imej daripada 1000 kategori dengan ketepatan sebanyak 92.70%. Imej teks tulisan tangan juga akan menjalani pra-pemprosesan agar model pembelajaran mendalam ini dapat menampung pelbagai jenis kualiti imej.Selain itu, teknik pembelajaran pemindahan telah digunakan bagi membolehkan Rangkaian Neural Pengkonvolutan yang dilatih pada set data yang besar digunakan untuk tugas pengecaman yang mempunyai bilangan sampel data yang kecil. Maka, projek ini adalah berdasarkan penggunaan semua rangkaian terlatih untuk tugas pengecaman aksara Cina tulisan tangan. Namun begitu, daripada hanya menggunakan VGG16 untuk pengecaman aksara, penalaan halus telah dilakukan bagi mendapatkan ketepatan yang lebih tinggi.

3 OBJEKTIF KAJIAN

Objektif utama projek ini adalah untuk

A. Membangunkan model Rangkaian Neural Pengkonvolutan yang dapat mengecamkan aksara Cina tulisan tangan serta membuat perbandingan dan penilaian prestasi antara model tersebut.
B. Mengenal pasti faktor saiz input, *dropout*, *batch normalization*, dan pengoptimum yang sesuai bagi menghasilkan model berketepatan paling tinggi.

C. Membangunkan satu aplikasi sistem berintegrasikan model pembelajaran mendalam terlatih untuk tujuan demonstrasi

4 METOD KAJIAN

4.1 Pengumpulan set data

Set data yang digunakan untuk tujuan melatih model pembelajaran mendalam adalah dikumpulkan dari laman sesawang Kaggle yang bertajuk *Chinese Handwriting Recognition*: HSK 1. Set data ini adalah suntingan Pangkalan Data Tulisan Tangan Cina Dalam Talian dan Luar Talian CASIA dengan hanya 178 aksara yang dipaparkan dalam peperiksaan HSK 1. Namun begitu, hanya terdapat 20 jenis aksara Cina tulisan tangan yang telah dipilih secara rawak dalam set data tersebut bagi meneruskan projek ini atas faktor kesuntukan masa dan juga faktor perkakasan yang kurang memuaskan. Hal ini demikian kerana platform *Google Colab* hanya menyimpan fail dalam *Google Drive* dengan ruang kosong sebanyak 15 GB sahaja. Ini juga bermaknanya set data yang lebih besar akan memerlukan lebih banyak ruang, seterusnya menjadikannya lebih sukar untuk dilaksanakan. Selain itu, platform *Google Colab* hanya membenarkan penggunanya melatih model selama 12 jam sehari. Pengguna perlu mengakses versi berbayar iaitu *Google Colab Pro* sekiranya mereka ingin mengambil tempoh masa yang lebih lama iaitu 24 jam bagi membolehkan latihan model berlangsung. Namun begitu, *Google Colab Pro* masih tidak ada di kawasan negara Malaysia dan kita tidak dapat menikmati perkhidmatan tersebut seperti pengguna di negara lain. Tambahan pula, jikalau pengguna tidak berinteraksi dengan *Google Colab* selama 90 minit dan ia akan ditamatkan secara automatik. Maka, dengan adanya faktor-faktor yang diungkitkan tadi, sebanyak 20 jenis aksara Cina tulisan tangan bersamaan 20 kelas telah dipilih bagi melatih model pembelajaran mendalam kita.

```
os.mkdir('/content/drive/MyDrive/NewFYP/CASIA')
os.mkdir('/content/drive/MyDrive/NewFYP/CASIA_128px')
```

Rajah 4.1 Segmen kod bagi menyimpan imej, keputusan, dan model

Pertama sekali, folder baharu yang bernama CASIA, CASIA_128px, serta CASIA_64px akan dicipta bagi menyimpan fail-fail seperti imej, keputusan, dan model. Set data asal akan dimuat turun daripada laman sesawang Kaggle dan 20 jenis aksara Cina tulisan akan dipilih secara

rawak dalam kalangan 178 jenis aksara Cina tulisan tangan yang sedia ada. Seterusnya, set data terbaharu yang mempunyai 20 jenis aksara Cina tulisan tangan terpilih akan dimuat naik dalam bentuk fail zip ke dalam folder NewFYP di dalam *Google Drive*. Fail zip ini akan dinyahzipkan ke dalam sebuah dokumen yang bernama CASIA untuk tindakan seterusnya. Rajah di bawah menunjukkan 20 jenis aksara Cina tulisan tangan terpilih yang telah dinyahzipkan.

Rajah 4.2 Senarai aksara Cina tulisan tangan yang telah dinyahzipkan

Set data yang telah dimuat naik ke dalam dokumen CASIA masih belum menjalani proses train-valid-test split di mana ia merupakan satu teknik untuk menilai prestasi model pembelajaran mesin mendalam kita. Bagi tujuan latihan dan ujian model kita, set data kita seharusnya dibahagikan kepada tiga pembahagian set data yang berbeza iaitu set latihan (*training set*), set pengesahan (*validation set*), dan set pengujian (*testing set*). Set latihan merupakan set data yang digunakan untuk melatih dan membuat model mempelajari ciri-ciri serta corak-corak tersembunyi dalam data. Set pengesahan merupakan satu set data berasingan daripada set latihan yang digunakan untuk mengesahkan prestasi model kita ketika latihan model berlangsung (Baheti, P, 2022). Tujuan untuk membahagikan set data kepada set pengesahan adalah untuk mengelakkan model daripada menghadapi masalah *overfitting*, di mana model menjadi sangat baik dalam mengklasifikasikan sampel dalam set latihan tetapi tidak boleh mengklasifikasikan sampel dengan tepat dalam set pengesahan manakala set pengujian adalah set data berasingan yang digunakan untuk menguji model setelah tamat menjalankan latihan. Ia akan menyediakan metrik prestasi model dari segi ketepatan (*accuracy*), ketepatan (*precision*), dan sebagainya.

```
input_folder = '/content/drive/MyDrive/NewFYP/CASIA'

splitfolders.ratio(input_folder, output="/content/drive/MyDrive/NewFYP/CASIA_128px",
                   seed=42, ratio=(.7, .2, .1),
                   group_prefix=None)

Copying files: 2845 files [15:42,  3.02 files/s]
```

Rajah 4.3 Pembahagian set data kepada set latihan, set pengesahan, dan set pengujian

Maka dalam projek ini, pembahagian set data kepada set latihan, set pengesahan, dan set pengujian akan dilakukan secara manual dengan menggunakan splitfolder. Ketiga-tiga set ini yang dinamakan sebagai train, val, test akan dibahagikan dengan nisbah 70%, 20%, dan 10% masing-masing seterusnya dimasukkan ke dalam dokumen CASIA_128px. Seed adalah digunakan untuk memastikan keputusan boleh dihasilkan semula. Dalam erti kata lain, parameter ini akan memastikan bahawa sesiapa yang menjalankan semula kod tersebut akan mendapatkan output yang sama. Hal ini demikian kerana kebolehulangan (*reproducibility*) merupakan konsep yang sangat penting dalam sains data dan pembelajaran mesin (Bansal, J, 2022).

4.2 Pra-pemprosesan data

x_train	Tatasusunan numpy (*numpy arrays*) bagi imej set data latihan
y_train	Label set data latihan
x_val	Tatasusunan numpy (*numpy arrays*) bagi imej set data pengesahan
y_val	Label set data pengesahan
x_test	Tatasusunan numpy (*numpy arrays*) bagi imej set data pengujian
y_test	Label set data pengujian

Jadual 4.1 Senarai tatasusunan numpy dan label set data

Lazimnya, apabila kita ingin menggunakan set data Tensorflow atau Keras, kita akan mudah memperoleh nilai *x_train*, *y_train*, *x_test*, *y_test* ketika memuatkan set data itu sendiri. Nilai-nilai ini adalah penting untuk memahami cara label set data latihan dan pengesahan kita dikodkan serta mendapatkan laporan klasifikasi. Dalam projek ini, set data kita adalah dikategorikan sebagai set data tersuai, di mana imej set data kita disusun dengan kemas dalam folder secara manual dengan menggunakan *splitfolder*. Kita perlu menetapkan nilai-nilai

tersebut berdasarkan ketiga-tiga folder yang telah dicipta sebelum melakukan proses pra-pemprosesan data. Hal ini demikian kerana kita akan melatih model VGG-16 pada set data latihan tersuai bagi mengecamkan aksara Cina tulisan tangan dengan saiz input 64 x 64 piksel dan 128 x 128 piksel.

```
train_path="/content/drive/MyDrive/NewFYP/CASIA_128px/train"
test_path="/content/drive/MyDrive/NewFYP/CASIA_128px/test"
val_path="/content/drive/MyDrive/NewFYP/CASIA_128px/val"
```

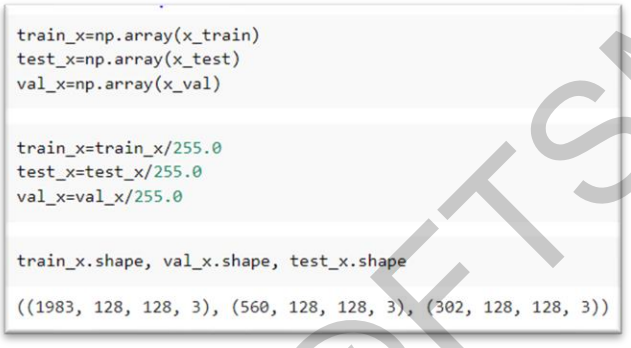
Rajah 4.4 Penyimpanan laluan folder dalam pembolehubah

```
x_train=[]

for folder in os.listdir(train_path):
    sub_path=train_path+"/"+folder

    for img in os.listdir(sub_path):
        image_path=sub_path+"/"+img
        img_arr=cv2.imread(image_path)
        img_arr=cv2.resize(img_arr,(128,128))
        x_train.append(img_arr)

x_test=[]

for folder in os.listdir(test_path):
    sub_path=test_path+"/"+folder

    for img in os.listdir(sub_path):
        image_path=sub_path+"/"+img
        img_arr=cv2.imread(image_path)
        img_arr=cv2.resize(img_arr,(128,128))
        x_test.append(img_arr)

x_val=[]

for folder in os.listdir(val_path):
    sub_path=val_path+"/"+folder

    for img in os.listdir(sub_path):
        image_path=sub_path+"/"+img
        img_arr=cv2.imread(image_path)
        img_arr=cv2.resize(img_arr,(128,128))
        x_val.append(img_arr)
```

Rajah 4.5 Segmen kod bagi mengubah saiz setiap imej dalam folder

Dalam fasa pra-pemprosesan data, setiap imej akan diubah saiz daripada saiz asal yang berbeza kepada saiz yang seragam iaitu 64 x 64 piksel serta 128 x 128 piksel. Di sini kita akan mempunyai dua jenis set data imej yang berbeza daripada saiz dimensi. Seterusnya, dalam Rajah 4.5, ketiga-tiga senarai (*Python lists*) iaitu *x_train* = [ ], *x_test* = [ ], dan *x_val* = [ ] akan dicipta bagi menyimpan berbilang imej dengan menggunakan .apend(). Kaedah ini akan

meletakkan imej baharu dalam senarai-senarai yang telah disediakan apabila proses mengubah saiz sekeping imej telah selesai dijalankan. Proses mengubah saiz imej akan dilakukan dengan menggunakan kaedah *OpenCV* di mana ia merupakan alat yang popular untuk pemprosesan imej serta melaksanakan tugas penglihatan komputer (*Computer Vision*). Kaedah *.imread*() telah digunakan untuk memuatkan imej daripada fail tertentu manakala kaedah *.resize*() digunakan untuk mengubah saiz imej kepada 64 x 64 piksel dan 128 x 128 piksel.



```
train_x=np.array(x_train)
test_x=np.array(x_test)
val_x=np.array(x_val)

train_x=train_x/255.0
test_x=test_x/255.0
val_x=val_x/255.0

train_x.shape, val_x.shape, test_x.shape

((1983, 128, 128, 3), (560, 128, 128, 3), (302, 128, 128, 3))
```

Rajah 4.6 Pertukaran senarai (*Python list*) kepada tatasusunan numpy

Seterusnya, ketiga-ketiga senarai akan ditukarkan kepada tatasusunan numpy iaitu *x_train*, *x_test*, dan *x_val*. Ketiga-tiga tatasusunan numpy ini perlu dibahagikan dengan 255.0 bagi tujuan pernormalan. Hal ini demikian kerana nilai piksel boleh berjulat dari nombor 0 hingga 255. Setiap nombor mewakili kod warna yang berbeza. Apabila kita ingin memasukkan imej ke dalam Rangkaian Neural Pengkonvolutan, pengiraan nilai angka yang tinggi barangkali menjadi lebih rumit ataupun kompleks. Bagi menyelesaikan masalah ini, kita boleh menormalkan nilai kepada julat dari 0 hingga 1. Dengan ini, nombor nilai akan menjadi kecil seterusnya pengiraan akan menjadi lebih mudah dan cepat. Dalam Rajah 4.6, kita juga dapat melihat bahawa bilangan imej yang sedia ada serta saiz dimensi imej bagi setiap tatasusunan numpy. Sebagai contohnya, dengan adanya *train_x.shape*, kita dapat mengetahui bahawa ia terdapat 1983 keping imej yang bersaiz 128 x 128 piksel. Nombor 3 mewakili kesemua imej tersebut adalah berwarna tetapi bukan dalam skala kelabu. Hal ini demikian kerana model VGG16 dalam projek ini hanya menerima imej yang berwarna sebagai syarat kemasukan imej ke dalam rangkaiannya.

```
train_datagen = ImageDataGenerator(rescale = 1./255,
                                   rotation_range = 20,
                                   brightness_range = (0.5,1.5),
                                   width_shift_range = 0.1,
                                   height_shift_range = 0.1,
                                   zoom_range = 0.2,
                                   fill_mode='constant', cval=255)
test_datagen = ImageDataGenerator(rescale = 1./255)
val_datagen = ImageDataGenerator(rescale = 1./255)

training_set = train_datagen.flow_from_directory(train_path,
                                                 batch_size = 32,
                                                 class_mode = 'sparse')

test_set = test_datagen.flow_from_directory(test_path,
                                            batch_size = 32,
                                            class_mode = 'sparse')

val_set = val_datagen.flow_from_directory(val_path,
                                          batch_size = 32,
                                          class_mode = 'sparse')

Found 1983 images belonging to 20 classes.
Found 302 images belonging to 20 classes.
Found 560 images belonging to 20 classes.
```

Rajah 4.7 Segmen kod bagi meningkatkan bilangan set data

Rajah 4.7 menunjukkan segmen kod untuk meningkatkan bilangan set data dengan menggunakan teknik penambahan imej. Teknik ini menggunakan kelas *Keras ImageDataGenerator* yang telah menyediakan cara yang cepat dan mudah untuk menambah imej. Faedah utama penggunaan kelas *Keras ImageDataGenerator* adalah kerana ia direka untuk menyediakan penambahan imej masa nyata. Dalam erti kata lain, ia akan menjana imej tambahan dengan cepat ketika model masih dalam proses latihan. Selain itu, kelebihan penggunaan kelas *Keras ImageDataGenerator* adalah ia memerlukan penggunaan memori yang lebih rendah. Hal ini demikian kerana tanpa menggunakan kelas ini, kita perlu memuatkan semua imej sekaligus. Sebaliknya, dengan adanya kelas ini, kita dapat memuatkan imej dalam kelompok seterusnya dapat menjimatkan banyak memori. Memang tidak dapat dinafikan bahawa terdapat banyak teknik penambahan imej dapat digunakan bagi meningkatkan saiz set data kita. Namun, bukannya semua teknik adalah sesuai digunakan dalam projek ini. Teknik seperti *horizontal_flip*, *vertical_flip*, dan lain-lain akan mengubah bentuk dan maksud asal imej terutamanya imej kita yang mengandungi aksara Cina tulisan tangan. Aksara Cina tulisan tangan adalah berbeza dengan imej objek yang lain. Contoh teknik yang telah diungkitkan tadi akan menyebabkan aksara Cina tulisan tangan hilang makna asalnya seterusnya menjadi tidak relevan dalam proses latihan nanti. Di samping itu, teknik penambahan imej hanya digunakan pada set latihan. Hal ini demikian kerana set pengesahan adalah digunakan semata-mata untuk penalaan hiperparameter manakala set pengujian adalah digunakan untuk menganggar prestasi akhir. Sekiranya kedua-dua set data ini menjalani proses penambahan imej, maka prestasi model akan terjejas di mana model tersebut akan sangat mudah mengklasifikasikan imej yang baharu

4.3 Pembangunan model Rangkaian Neural Pengkonvolutan

```
image_input = Input(shape=(128, 128, 3))

vgg16 = VGG16(input_tensor=image_input, weights='imagenet', include_top=False)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5
58892288/58889256 [==============================] - 0s 0us/step
58900480/58889256 [==============================] - 0s 0us/step
```

Rajah 4.8 Menetapkan saiz input model serta muat turun model VGG16

```
#do not train the pre-trained layers of VGG-16
for layer in vgg16.layers:
    layer.trainable = False
```

Rajah 4.9 Pembekuan lapisan-lapisan dalam model VGG16

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

Rajah 4.10 Formula fungsi pengaktifan *softmax*

Rajah 4.8 menunjukkan segmen kod bagi memuat turun model VGG16 serta menetapkan saiz input model. Argumen seperti *include_top* telah ditetapkan kepada *False*. Perkara ini membolehkan lapisan output sambungan sepenuhnya (*fully-connected output layer*) tidak dimuatkan, seterusnya membenarkan lapisan output baharu ditambah serta dilatih. Dalam projek ini, 18 jenis model VGG16 yang berbeza daripada segi hiperparameter telah dibina menggunakan pendekatan pembelajaran permindahan. Dengan menggunakan pendekatan ini, lapisan daripada model VGG16 yang telah dilatih sebelum ini akan dibekukan bagi mengelakkan daripada memusnahkan mana-mana maklumat yang terkandung dalam proses latihan model yang baharu. Model VGG16 juga boleh dikenali sebagai model pra-latihan (*pre-trained model*). Apabila menggunakan lapisan pra-latihan ini, kita boleh memutuskan untuk membekukan lapisan tertentu. Rajah 4.9 telah menunjukkan segmen kod yang menjadikan kesemua pemberat lapisan dalam model VGG16 daripada boleh dilatih kepada tidak boleh dilatih di mana proses tersebut dikenali sebagai pembekuan. Ini bermaknanya

kesemua lapisan di dalam model VGG16 kita telah dibekukan dan keadaan lapisan beku tidak akan dikemas kini serta dapat memastikan kita tidak akan melatih sebanyak 14 juta parameter dalam model VGG16 ketika proses latihan berlangsung.

Seterusnya, beberapa lapisan baharu yang bakal dilatih akan ditambahkan di atas lapisan yang telah dibekukan seperti lapisan *flatten*, lapisan *dense*, lapisan *dropout* dan lapisan *batch normalization*. Lapisan *flatten* digunakan untuk menukarkan data matriks dua dimensi kepada vektor satu dimensi sebelum membina lapisan bersambung sepenuhnya, ataupun juga boleh dikenali sebagai lapisan *dense*. Terdapat tiga lapisan *dense* telah digunakan. Lapisan *dense* yang pertama dan kedua mempunyai sebanyak 128 dan 64 nombor neuron masing-masing telah digunakan bersama dengan fungsi pengaktifan *Rectified Linear Unit* (*ReLu*). *ReLu* akan mengembalikan 0 untuk sebarang input negatif, dan nilainya sendiri sekiranya ia adalah positif. Operasi maksimum dalam *ReLu* membolehkannya untuk mengira lebih cepat daripada fungsi pengaktifan yang lain. Bagi membolehkan model kita membuat ramalan, lapisan *dense* terakhir ini memainkan peranan yang amat mustahak. Lapisan *dense* terakhir ini mempunyai sebanyak 20 neuron digunakan bersama dengan fungsi pengaktifan *softmax*. Penetapan 20 neuron bermaksud lapisan ini mempunyai 20 nilai pengaktifan yang akan mewakili kebarangkalian relatif 20 kelas aksara Cina tulisan tangan yang kita ingin ramalkan. Fungsi pengaktifan *softmax* boleh dinyatakan secara matematik dalam Rajah 4.10.

Seterusnya iaitu lapisan *dropout*. Ia merupakan satu teknik penyelarasan bagi mengelakkan *overfitting* dalam latihan model rangkaian neural. Dalam lapisan dropout projek ini, sebanyak 20% neuron telah disekat secara rawak semasa proses latihan. Ia dapat menghalang rangkaian daripada terlalu bergantung pada neuron tunggal, seterusnya akan memaksa semua neuron belajar untuk membuat generalisasi dengan lebih baik. Bagi lapisan *batch normalization*, secara berterusannya ia akan mengambil output dari lapisan sebelumnya seterusnya menormalkannya untuk mendapatkan nilai dalam julat antara 0 and 1 sebelum menghantarnya ke lapisan seterusnya.

Selain menggunakan lapisan-lapisan yang telah dibincangkan dalam perenggan sebelum ini, tiga jenis pengoptimim (*optimizer*) juga akan digunakan bagi menambahkan lagi variasi model yang bakal dibangunkan serta memastikan pengoptimum yang paling sesuai dalam projek ini. Pengoptimum merupakan satu fungsi ataupun algoritma yang mengubah suai atribut rangkaian neural. Oleh itu, ia dapat membantu dalam mengurangkan kehilangan keseluruhan serta meningkatkan ketepatan. Antara pengoptimum yang telah

digunakan adalah Adam (*Adaptive Moment Estimation*), RMSProp (*Root Mean Squared Propagation*), dan SGD (*Stochastic Gradient Descent*). Jadual di bawah menunjukkan senarai model VGG16 yang telah dibina berasaskan saiz input, *dropout*, *batch normalization*, dan pengoptimum.

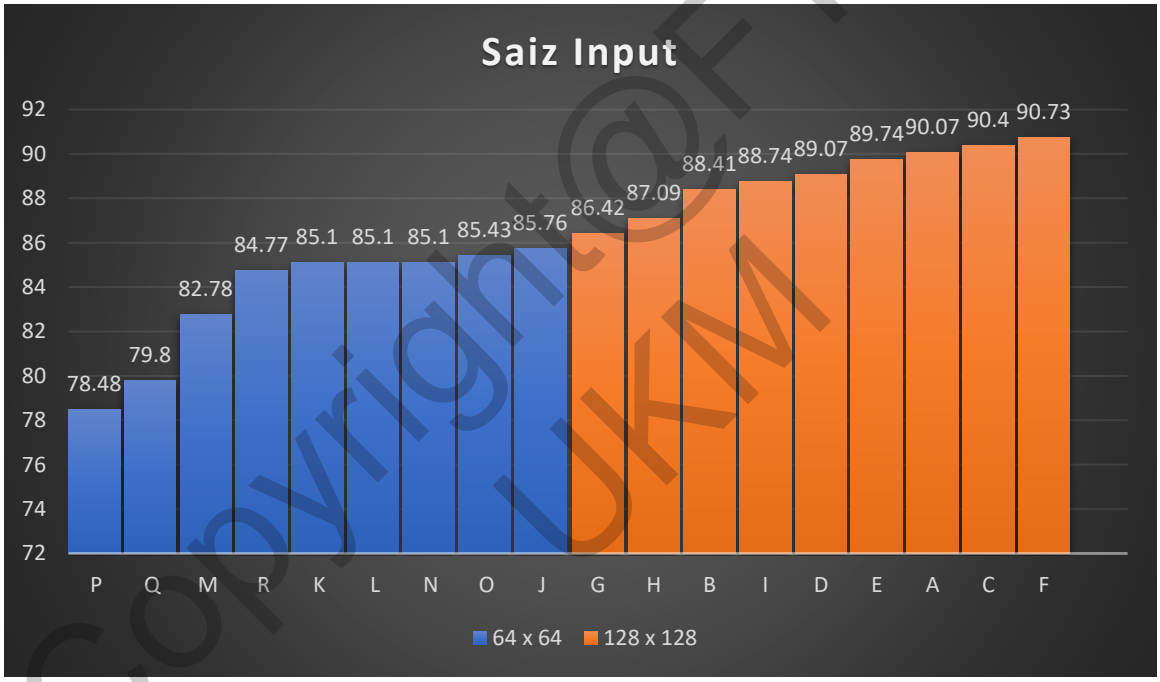
Model	Saiz input	Dropout	Batch Normalization	Pengoptimum
A	128 x 128	✓		Adam
B	128 x 128	✓		RMSProp
C	128 x 128		✓	Adam
D	128 x 128		✓	RMSProp
E	128 x 128	✓	✓	Adam
F	128 x 128	✓	✓	RMSProp
G	128 x 128	✓		SGD
H	128 x 128		✓	SGD
I	128 x 128	✓	✓	SGD
J	64 x 64	✓		Adam
K	64 x 64	✓		RMSProp
L	64 x 64		✓	Adam
M	64 x 64		✓	RMSProp
N	64 x 64	✓	✓	Adam
O	64 x 64	✓	✓	RMSProp
P	64 x 64	✓		SGD
Q	64 x 64		✓	SGD
R	64 x 64	✓	✓	SGD
Asas	224 x 224			Adam

Jadual 4.2 Senarai model VGG16 yang telah dibina berasakan saiz input, *dropout*, *batch normalization*, dan pengoptimum.

4.4 Pengujian dan penilaian model

Penilaian metrik telah dijalankan terhadap model pembelajaran mendalam yang telah dibangunkan bagi menentukan model yang paling sesuai seterusnya akan dipilih untuk dimasukkan ke dalam aplikasi sistem pengecaman aksara Cina tulisan tangan. Antara metrik penilaian yang telah digunakan adalah *accuracy*, *precision*, dan *recall*. Metrik penilaian tersebut boleh diperolehi melalui penjanaan laporan klasifikasi (*Classification Report*). Selain itu, graf lengkung ketepatan (*accuracy curve graph*) dan graf lengkung kerugian (*loss*

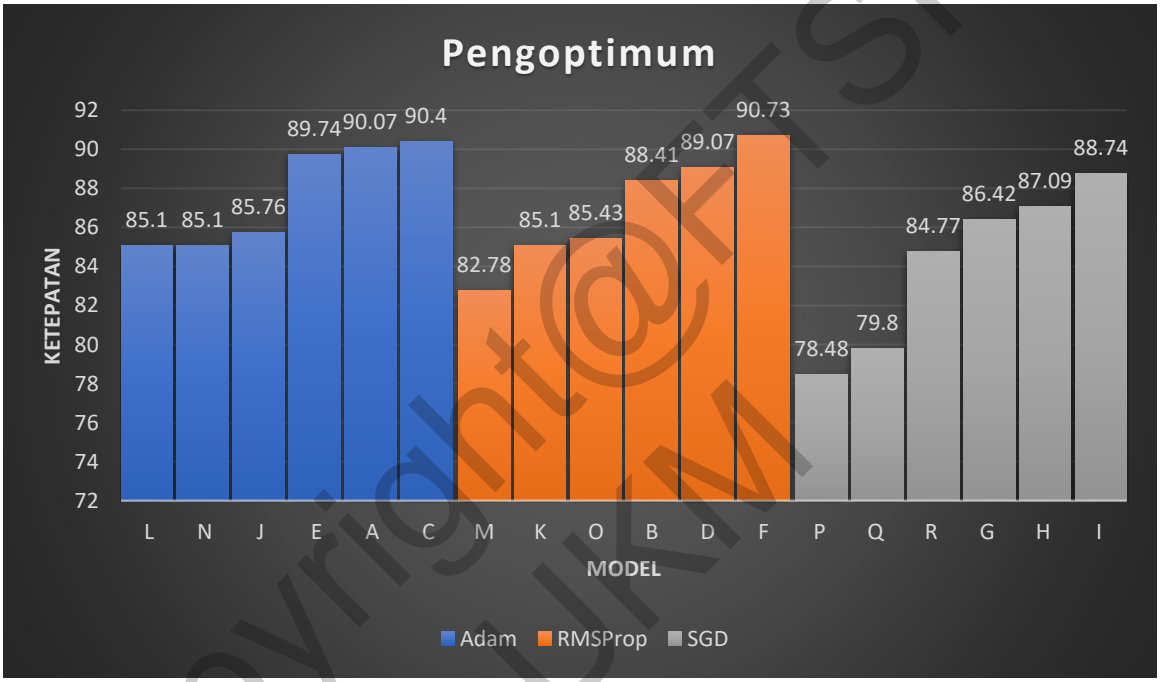
curve graph) setiap model akan juga dijana setelah tamat proses latihan agar dapat menentukan sama ada model tersebut mengalami masalah *overfitting* ataupun *underfitting*. Kita telah mendapati bahawa majoriti model mengalami masalah *overfitting*. Situasi *overfitting* berlaku apabila kerugian pengesahan adalah lebih tinggi daripada kerugian latihan, ataupun, ketepatan latihan lebih tinggi daripada ketepatan pengesahan. Ini bermakna majoriti model terlatih berfungsi dengan baik pada set data latihan tetapi tidak seperti dalam set data pengesahan di mana mereka tidak digeneralisasikan dengan baik pada pelbagai jenis data. Hal ini demikian kerana seni bina model yang agak rumit di mana kami meletakkan banyak lapisan tersembunyi di dalam model, seterusnya menyebabkan model tersebut mempunyai terlalu banyak pemberat.



Rajah 4.11 Graf perbandingan ketepatan model menggunakan saiz input yang berbeza

Berdasarkan Rajah 4.11, kita boleh mendapati bahawa ketepatan model yang mempunyai saiz input 128 x 128 piksel adalah lebih tinggi daripada ketepatan model yang mempunyai saiz input 64 x 64 piksel. Setahu kita, VGGNet mengambil saiz input 224 x 224 piksel kerana pencipta model telah memotong tampalan tengah 224 x 224 dalam setiap imej untuk memastikan saiz input adalah konsisten. Namun begitu, dimensi input aksara Cina tulisan tangan dalam projek ini adalah jauh lebih kecil daripada yang telah dilatih oleh VGGNet. Sekiranya kita meningkatkan saiz imej secara drastik, maka resolusi imej akan terjejas seterusnya mengakibatkan nilai ketepatan serta kerugian model terjejas.

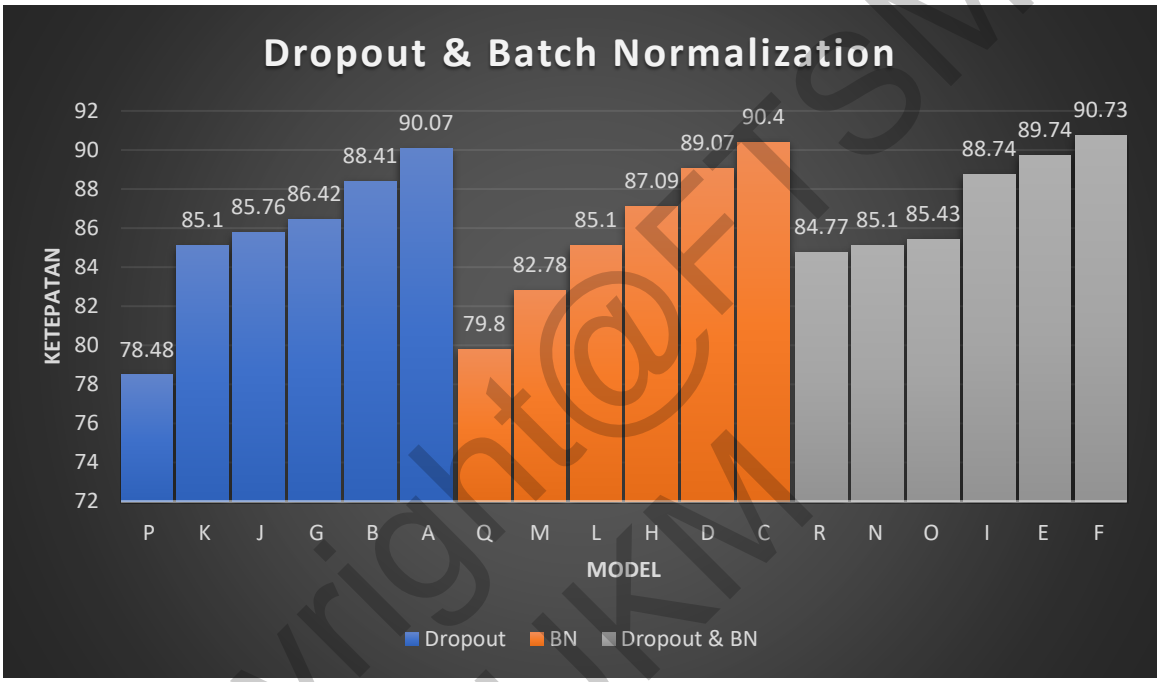
Penggunakan saiz input 224 x 224 piksel juga memakan masa proses latihan. Hal ini demikian kerana saiz input yang lebih besar memerlukan lebih banyak memori serta banyak parameter rangkaian. Sebagai contohnya, sekiranya kita ingin menggandakan saiz input maka kita perlu menggandakan jumlah memori yang diperlukan dalam GPU. Bukan itu sahaja, kita juga perlu menggandakan saiz penapis untuk mendapatkan medan penerimaan yang sama. Oleh itu, saiz input 128 x 128 piksel adalah saiz input yang lebih optimum dalam penalaan halus model pembelajaran mendalam kita berbanding dengan saiz input 64 x 64 piksel.



Rajah 4.12 Graf perbandingan ketepatan model menggunakan pengoptimum yang berbeza

Berdasarkan Rajah 4.12, kita boleh mendapati bahawa ketepatan model F yang menggunakan pengoptimum *RMSProp* adalah paling tinggi berbanding dengan model yang menggunakan pengoptimum yang lain. Walau bagaimanapun, adalah juga didapati bahawa majoriti model yang menggunakan pengoptimum *Adam* mempunyai ketepatan yang lebih tinggi berbanding dengan yang lain. Manakala model-model yang menggunakan pengoptimum *SGD* mempunyai ketepatan yang agak rendah. *SGD* merupakan algoritma yang sangat asas dan hampir tidak digunakan dalam aplikasi sekarang kerana kelajuan pengiraannya yang sangat perlahan. Bagi *RMSProp*, ia dapat menghalang algoritma daripada menyesuaikan terlalu cepat kepada perubahan parameter tertentu yang digunakan untuk mengklasifikasikan objek berbanding dengan parameter yang lain. Oleh itu, ia akan menumpu dengan cepat seterusnya menyebabkan kadar pembelajaran meningkat serta

hanya memerlukan penalaan yang lebih rendah berbanding dengan pengoptimum yang lain. Bagi *Adam* yang berperanan sebagai pengoptimum yang digunakan secara meluas, ia sering disesuaikan sebagai penanda aras untuk pembelajaran mendalam dan disyorkan sebagai pengoptimum utama. Hal ini demikian kerana ia mudah dilaksanakan, mempunyai masa berjalan yang lebih pantas, keperluan memori yang lebih rendah, serta memerlukan penalaan yang kurang daripada mana-mana algoritma pengoptimum yang lain.



Rajah 4.13 Graf perbandingan ketepatan model menggunakan lapisan *dropou*t & *batch normalization*

Berdasarkan Rajah 4.13, kita boleh mendapati bahawa ketepatan majoriti model yang mempunyai kombinasi antara lapisan *dropout* dengan lapisan *batch normalization* adalah lebih tinggi. Oleh disebabkan set data aksara Cina tulisan tangan merupakan set besar dengan nombor kelas yang berjumlah besar, maka *batch normalization* membantu untuk membentuk semula pengagihan input setiap lapisan. Manakala *dropout* membantu untuk meningkatkan keupayaan model dalam mengeneralisasikan set data yang baharu. Kombinasi antara *dropout* dengan *batch normalization* secara tidak langsungnya menjadikan rangkaian lebih pantas dan teguh dari segi kelajuan serta ketepatan prestasi. S. Ioffe and C. Szegedy. (2015) berpendapat bahawa rangkaian moden akan berprestasi lebih teruk dan tidak memuaskan apabila dilengkapi dengan lapisan *batch normalization* dan lapisan *dropout* bersama-sama. Hal ini demikian kerana *batch normalization* akan

menghapuskan keperluan *dropout* dengan menjangkakan bahawa *batch normalization* akan memberikan fungsi regularisasi yang sama seperti *dropout* secara intuitif, dan seterusnya menjejaskan ketepatan model. Walau bagaimanapun, situasi ini bukan sahaja tidak berlaku dalam projek ini bahkan juga telah membantu untuk meningkatkan ketepatan model dalam pengecaman aksara Cina tulisan tangan.

MODEL	ACCURACY	PRECISION	RECALL
A	90.07	0.89	0.89
B	88.41	0.89	0.89
C	90.40	0.91	0.91
D	89.07	0.89	0.89
E	89.74	0.90	0.90
F	90.73	0.91	0.91
G	86.42	0.87	0.87
H	87.09	0.87	0.88
I	88.74	0.89	0.89
J	85.76	0.86	0.86
K	85.10	0.85	0.86
L	85.10	0.85	0.86
M	82.78	0.83	0.84
N	85.10	0.85	0.85
O	85.43	0.85	0.86
P	78.48	0.79	0.80
Q	79.80	0.80	0.81
R	84.77	0.85	0.85
MODEL ASAS VGG16	84.44	0.87	0.84

Jadual 4.3 Keputusan pemodelan bagi set data pengujian

Model F mempunyai keputusan ketepatan yang paling tinggi iaitu 90.73% berbanding dengan model lain. Model F dibina dengan lapisan *batch normalization*, lapisan *dropout*, pengoptimum *RMSProp*, serta mempunyai saiz input 128 x 128 piksel.

4.5 Pembangunan aplikasi sistem

Aplikasi sistem telah dibangunkan dengan menggunakan *Flask* dan akan didemonstrasikan secara luar talian. *Flask* merupakan rangka kerja aplikasi web yang ditulis dalam bahasa *Python*. Ia menggunakan templat Jinjia untuk membina halaman HTML secara dinamik menggunakan konsep *Python*.

Rajah 4.14 Antara muka aplikasi sistem pengecaman aksara Cina tulisan tangan

Rajah 4.14 Antara muka aplikasi sistem mempamerkan keputusan ramalan

6 KESIMPULAN

Kesimpulannya, projek ini berjaya menghasilkan model pengecaman aksara Cina tulisan tangan dengan menggunakan pendekatan pembelajaran pemindahan berasaskan VGG16 yang diperhalusi yang mempunyai ketepatan sebanyak 90.73%. Ketepatan ini adalah lebih tinggi daripada ketepatan model asas VGG16 yang sebanyak 84.44%. Namun begitu projek ini telah menghadapi beberapa kekangan. Antaranya termasuklah kekurangan pengetahuan serta pemahaman tentang pengaturcaraan, algoritma pembelajaran mendalam serta pembangunan aplikasi sistem. Kekangan ini akan menyebabkan pembangun projek untuk mengambil masa bagi menyesuaikan diri dengan keperluan projek. Seterusnya, spesifikasi komputer yang

rendah secara tidak langsungnya akan mengambil masa yang agak lama untuk membangunkan model pembelajaran mesin. Selain itu, majoriti model terlatih mengalami masalah *overfitting* meskipun lapisan *dropout* dan lapisan *batch normalization* telah digunakan. Masalah ini barangkali berpunca daripada kekurangan set data aksara Cina tulisan tangan, seterusnya menyebabkan model tidak dapat mengklasifikasikan dengan tepat pada set data baharu. Selain itu, juga didapati bahawa model terlatih sukar mengecamkan imej input aksara Cina tulisan tangan daripada pengguna dengan tepat ketika demonstrasi aplikasi berlangsung

Bagi menambahbaikkan projek ini, terdapat beberapa cadangan yang barangkali boleh dilakukan pada masa akan datang iaitu, menambahkan bilangan set data. Walaupun bilangan set data yang kecil dapat diselesaikan dengan menggunakan teknik penambahan, namun variasi imej masih tidak mencukupi. Hal ini demikian kerana setiap orang mempunyai gaya tulisan yang berbeza. Seterusnya iaitu mengkaji balik penggunaan *dropout*, *batch normalization*, dan pengoptimum yang sesuai agar dapat mengelakkan model daripada mengalami masalah *overfitting* secara berkesan. Selain itu, kita boleh menggunakan saiz input yang lain seperti 256 x 256 piksel, 512 x 512 piksel, dan sebagainya. Apabila kita ingin mempraproses imej untuk proses latihan dan penilaian, banyak terdapat percubaan yang boleh kita lakukan berkaitan dengan saiz imej. Hal ini demikian kerana model kita tidak akan dapat belajar ciri-ciri tersendiri dalam sesuatu imej yang terlalu kecil. Sebaliknya, sekiranya imej terlalu besar, maka ia akan meningkat sumber pengiraan yang diperlukan oleh mesin, seterusnya menyebabkan mesin tidak cukup canggih untuk memproseskannya.

7 RUJUKAN

Baheti, P. 2022, June 20. Train Test Validation Split: How To & Best Practices [2022]. V7. shorturl.at/bI038 [1 Jun 2022]

Bi, N., Chen, J., & Tan, J. 2019. The Handwritten Chinese Character Recognition Uses Convolutional Neural Networks with the GoogLeNet. International Journal of Pattern Recognition and Artificial Intelligence 33(11): 2–3.

Du J, Hu J S, Zhu B, Wei S, Dai LR. 2014. A study of designing compact classifiers using deep neural networks for online handwritten Chinese character recognition. In: 2014 22nd international conference on pattern recognition (ICPR). IEEE, New York

Hildebrandt TH, Liu W. 1993. Optical recognition of handwritten Chinese characters: advances since 1980. Pattern Recognit 26(2): 205–225.

Zhang, Y. 2015. Deep convolutional network for handwritten Chinese character recognition. Computer Science Department, Stanford University.

Lim Zi Sin (A175835)
Mohd Zamri Murah
Fakulti Teknologi & Sains Maklumat,
Universiti Kebangsaan Malaysia.