CNN

許博竣

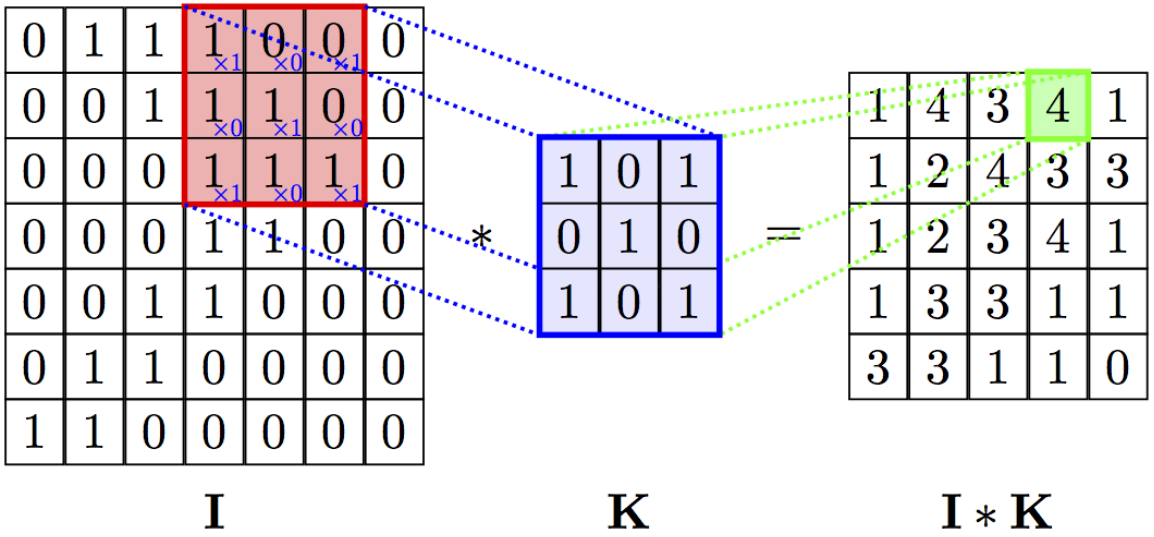
Before we start……

1. Handout and Sample code: https://ppt.cc/finj0x
2. Training Data:
   [NTU Space] https://ppt.cc/fOKTlx
   [Dropbox] https://ppt.cc/f7K7xx
3. ‘pip3 install –r requirements.txt’

Outline

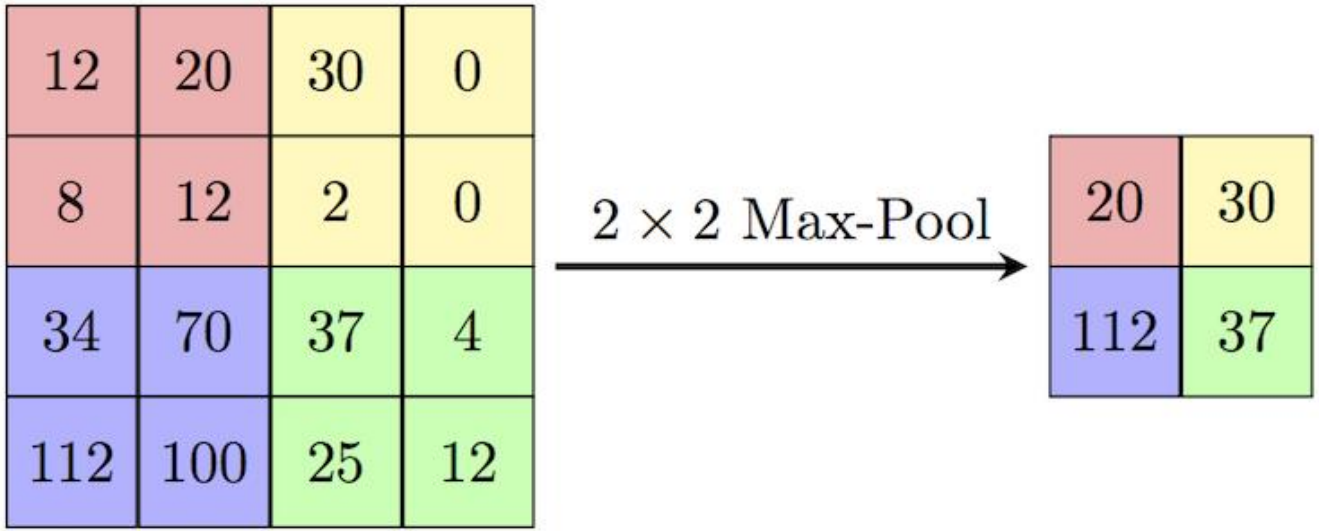
1. Introduction
2. Task
3. Data Format
4. Some Useful Keras Functions
5. Sample Code
6. Data Augmentation
7. Confusion Matrix

Introduction



Reference: https://ppt.cc/fR8pCx

Introduction



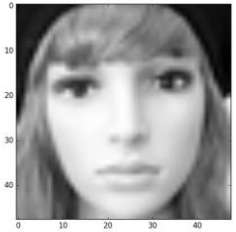
Reference: https://ppt.cc/f2gXjx

Task - Image Sentiment Classification

Task - Image Sentiment Classification

使用資料為網路上收集到的人臉表情資料，

經過特殊處理，每張圖片，均是人臉部份佔大部分

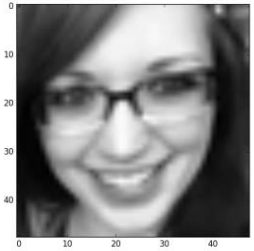


6(中立)

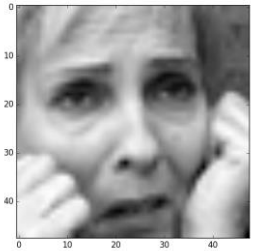
0(生氣)

1(厭惡)

2(恐懼)



3(高興)



4(難過)

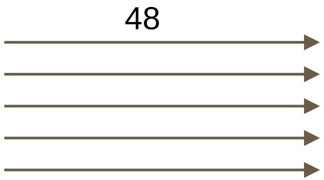
5(驚訝)

training data: about 28000 images

Data Format

[label], [48*48個灰階強度(intensity)值] (0為黑, 255為白)

<hint> use numpy.reshape function




```
toy.csv                                                                                          buffers
1 0,170 118 101 88 88 75 78 82 66 74 68 59 63 64 65 90 89 73 80 80 85 88 95 117 132 146 139 152 164 1
  81 182 194 195 162 142 147 151 148 158 127 113 112 97 117 117 112 131 124 168 125 108 94 95 77 94 8
  7 77 85 69 63 72 74 77 82 106 102 93 98 95 102 98 119 145 168 165 160 168 178 183 178 195 183 151 1
  39 145 144 154 143 115 106 122 107 100 111 124 133 173 134 113 101 94 85 104 89 88 91 69 66 75 85 9
  0 98 94 130 122 116 107 114 118 117 150 170 184 173 170 176 183 174 180 191 170 146 140 140 143 154
   118 119 119 103 91 103 114 128 176 145 115 106 90 95 107 90 99 91 72 72 75 87 101 116 105 118 148
  138 122 115 134 127 139 164 183 187 174 173 179 180 166 178 178 158 143 140 137 148 143 117 99 107
  104 91 104 116 178 151 119 111 90 105 105 95 109 87 79 85 86 83 102 129 125 121 147 156 148 121 129
   147 140 152 170 186 184 169 172 176 171 163 177 169 156 141 134 152 152 101 103 99 109 101 96 109
  181 154 125 109 94 111 102 102 108 79 85 91 93 87 87 122 146 128 147 160 164 149 123 143 149 150 15
  9 172 184 182 168 173 174 167 166 174 170 152 150 145 147 116 97 100 99 107 97 102 183 153 132 108
  101 110 100 113 99 79 91 97 98 98 95 100 144 146 139 170 166 168 144 132 149 149 156 164 173 184 17
  9 167 173 171 161 167 171 174 154 130 139 139 90 99 97 100 101 102 183 152 138 108 105 108 102 119
  89 84 94 100 103 104 102 99 121 157 147 159 181 167 172 148 139 146 145 150 163 175 185 176 166 172
   168 161 176 177 155 135 126 146 110 93 97 91 102 101 185 159 139 110 108 105 109 117 84 92 100 105
   109 109 111 107 108 140 162 151 172 182 170 174 149 146 148 143 150 167 180 186 173 164 168 168 16
  8 167 167 142 124 131 138 88 96 89 94 104 178 163 136 109 107 102 119 107 85 95 102 107 113 117 121
   117 113 118 153 165 159 178 182 169 169 163 159 158 156 159 170 179 187 177 164 166 166 164 167 15
  6 122 117 142 116 87 92 87 102 174 155 135 99 107 115 132 105 102 101 101 109 111 117 123 124 116 1
  18 126 157 169 164 179 186 171 170 171 169 169 169 158 161 170 180 180 171 169 167 159 162 139 103
  111 138 100 88 77 93 175 154 126 100 113 113 118 91 100 96 89 95 110 114 114 136 147 119 125 131 15
NORMAL  tony                                                                   10%  1/10 :  1
```

Some Useful Keras Functions

Sequential、Dense、Activation、Dropout、Adam

```
from keras.optimizers import Adam
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation

model = Sequential()
model.add(Dense(32, input_shape=(784,)))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(10, activation = 'softmax'))

model.compile(loss = 'categorical_crossentropy', optimizer = Adam(), metrics = ['accuracy'])
model.fit(x_data, y_data)
```

Some Useful Keras Functions

keras.utils.to_categorical

to_categorical

```
keras.utils.to_categorical(y, num_classes=None)
```

Converts a class vector (integers) to binary class matrix.

E.g. for use with categorical_crossentropy.

Arguments

- **y**: class vector to be converted into a matrix (integers from 0 to num_classes).
- **num_classes**: total number of classes.

Returns

A binary matrix representation of the input. The classes axis is placed last.

```
0 -> [1, 0, 0, 0, 0, 0, 0,]
1 -> [0, 1, 0, 0, 0, 0, 0,]
2 -> [0, 0, 1, 0, 0, 0, 0,]
3 -> [0, 0, 0, 1, 0, 0, 0,]
4 -> [0, 0, 0, 0, 1, 0, 0,]
5 -> [0, 0, 0, 0, 0, 1, 0,]
6 -> [0, 0, 0, 0, 0, 0, 1,]
```

Reference: https://keras.io/

Some Useful Keras Functions

keras.layers.Conv2D

Arguments

- **filters**: Integer, the dimensionality of the output space (i.e. the number of output filters in the convolution).
- **kernel_size**: An integer or tuple/list of 2 integers, specifying the height and width of the 2D convolution window. Can be a single integer to specify the same value for all spatial dimensions.
- **strides**: An integer or tuple/list of 2 integers, specifying the strides of the convolution along the height and width. Can be a single integer to specify the same value for all spatial dimensions. Specifying any stride value != 1 is incompatible with specifying any `dilation_rate` value != 1.
- **padding**: one of `"valid"` or `"same"` (case-insensitive). Note that `"same"` is slightly inconsistent across backends with `strides` != 1, as described here

Reference: https://keras.io/

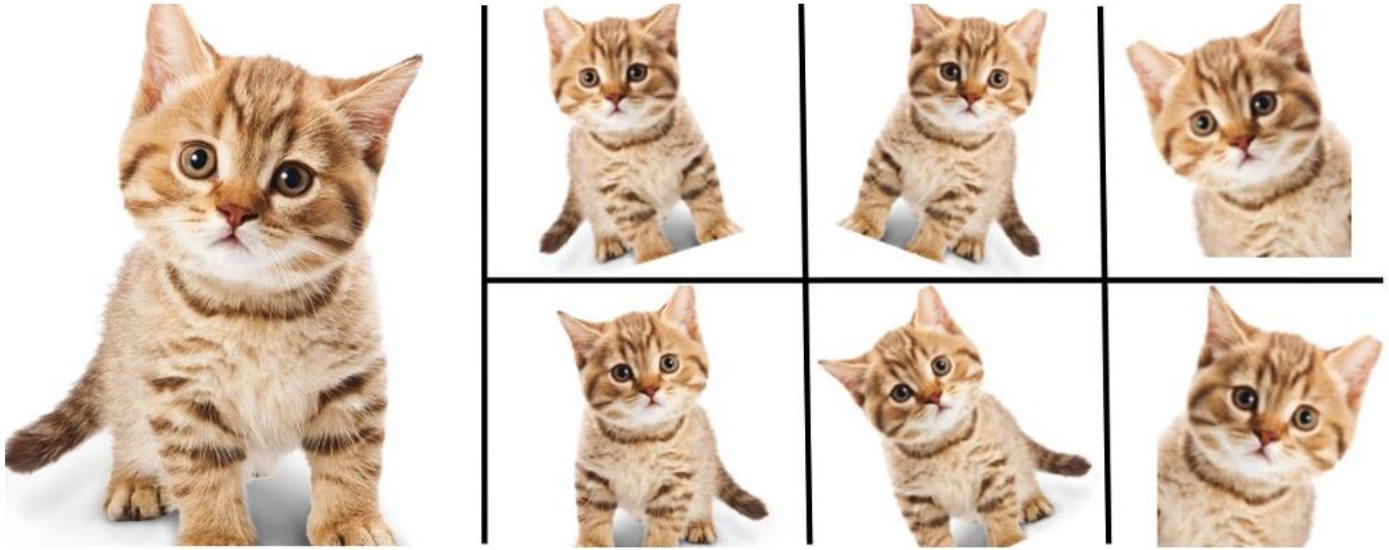
Sample Code

1. Sample Code
    - train.py -> main code
    - model.py -> model structure
    - utils.py -> data loader & preprocessor
2. Usage: ‘python3 train.py dnn’ or ‘python3 train.py cnn’
3. Make the network deeper!

Some Baseline - 1

Model	Simple DNN	Simple CNN
Training Acc	64%	98%
Validation Acc	43%	45%

Data Augmentation



Reference: https://ppt.cc/f8sGJx

Data Augmentation

keras.preprocessing.image.ImageDataGenerator

```
datagen = ImageDataGenerator(
    featurewise_center=True,
    featurewise_std_normalization=True,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    horizontal_flip=True)

model.fit_generator(datagen.flow(x_train, y_train, batch_size=32),
                    steps_per_epoch=len(x_train) / 32, epochs=epochs)
```

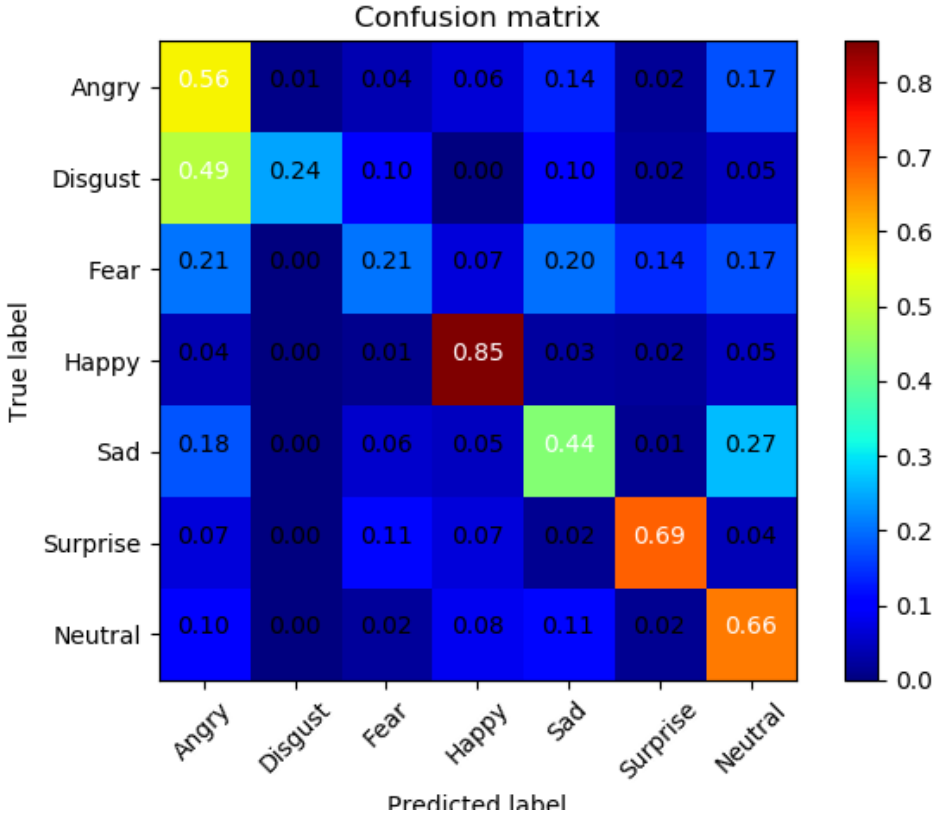
Data Augmentation

```
rotation_range: Int. Degree range for random rotations.
width_shift_range: Float, 1-D array-like or int.
height_shift_range: Float, 1-D array-like or int.
zoom_range: Float. Range for random zoom.
horizontal_flip: Boolean. Randomly flip inputs horizontally.
vertical_flip: Boolean. Randomly flip inputs vertically.
fill_mode: "constant", "nearest", "reflect" or "wrap".
```

Some Baseline - 2

Modelz	Simple DNN	Simple CNN	Deeper CNN
Training Acc	64%	98%	54%
Validation Acc	43%	45%	59%

Confusion Matrix



Confusion Matrix

1. Sample Code
    - matrix.py
2. Usage: ‘python3 matrix.py dnn’ or ‘python3 matrix.py cnn’
3. Load your model in ‘matrix.py’