Hand Gesture Recognition

Team members: Paul Woodward, Sahiti Tadepalli, Hien Nguyen, Annie Liang, Xinyi Zhang

-----Machine Learning , Winter 2018

Outline

- Project intro
- Design Overview
- Model Detail
- Results
- Morbitity and Mortality
  - tuning attemps

Project Intro:

Sebastien Marcel Static Hand Posture Database:
6 hand postures (a, b, c, point, five, v), about 10 persons.

Count of the number of images in each section:

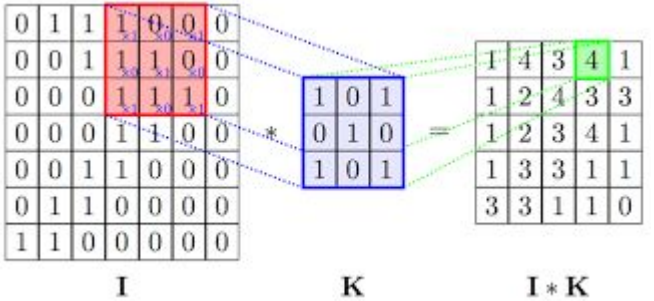
Found 4872 images belonging to 6 classes (training set)
Found 659 images belonging to 6 classes (test set)

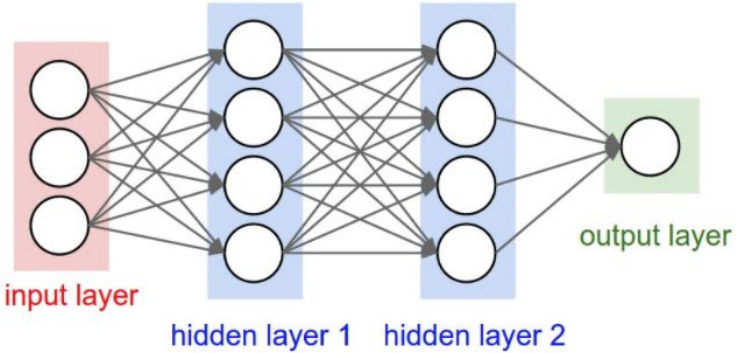
Originial Image sizes:
{'76,66,3': 1573, '82,70,3': 471, '84,72,3': 147, '90,78,3': 495, '288,384,3': 153, '155,155,3': 86, '320,240,3': 82, '88,76,3': 461, '80,68,3': 499, '66,56,3': 116, '78,66,3': 199, '74,64,3': 169, '66,76,3': 110, '68,80,3': 59, '76,88,3': 55, '70,82,3': 51, '72,84,3': 24, '64,80,3': 6, '66,80,3': 11, '78,90,3': 79, '187,194,3': 5}

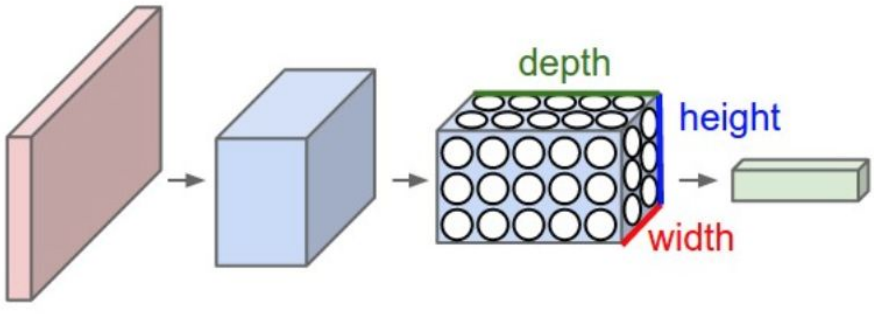
Pre-processed: 64 x 64 x 1 (also the input pixel values)
Filter/Receptive field: 3 x 3 x 1
CONV layer (hidden layer): slide over the image spatially, computing dot products

- stride(2,2)

$$\mathbf{W}^T\mathbf{x} + b$$



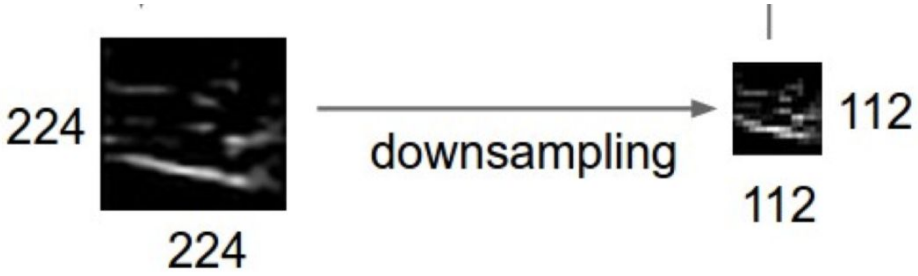


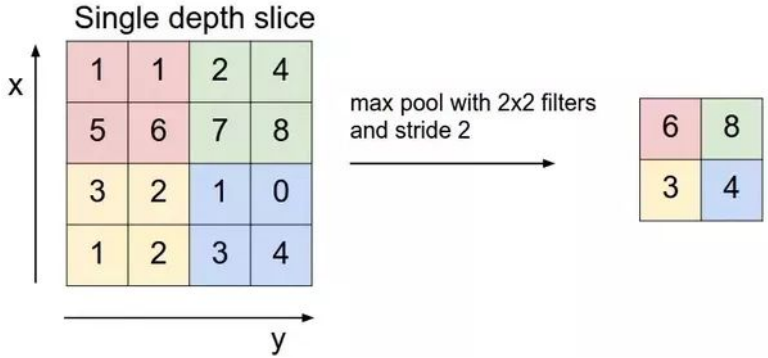


Output size is computed using (N-F)/S +1.

Here, we have 64x64 input and 3x3 filter with 2 stride gives us 32 x32.
This “new image” is the input for Maxpooling function that takes only max value per the grid.

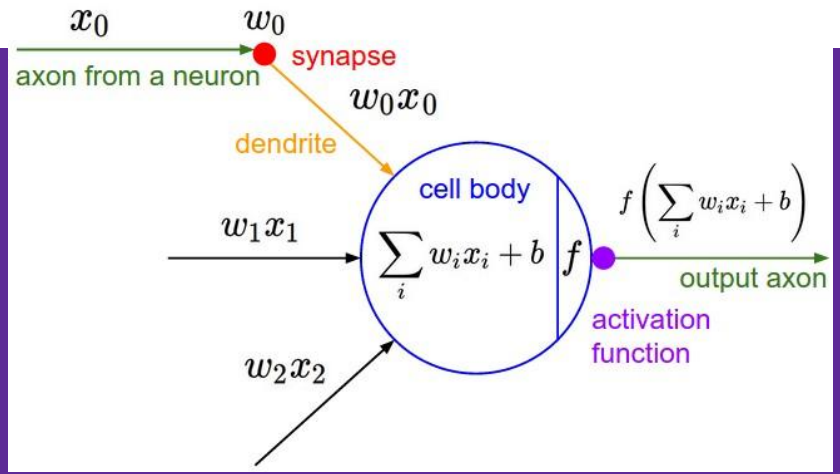
- dimension reduction. It only keeps the most intense pixels
  but also retain image information

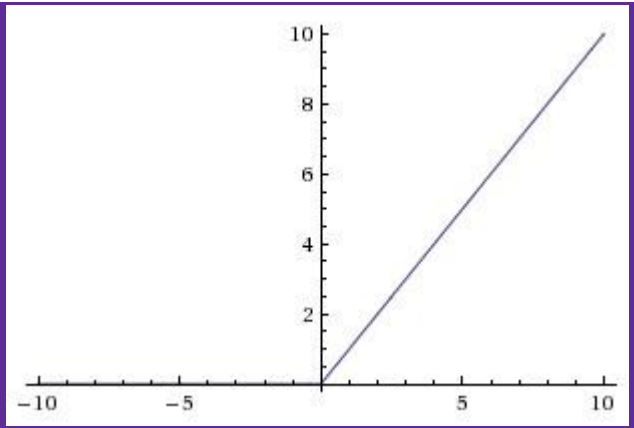




Activation Function

rectified linear unit (ReLU)





Model summary:

```
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 62, 62, 32)        320
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 31, 31, 32)        0
_________________________________________________________________
activation_1 (Activation)    (None, 31, 31, 32)        0
_________________________________________________________________
batch_normalization_1 (Batch (None, 31, 31, 32)        128
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 29, 29, 32)        9248
_________________________________________________________________
activation_2 (Activation)    (None, 29, 29, 32)        0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 14, 14, 32)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 6272)              0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               802944
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_2 (Dense)              (None, 6)                 774
=================================================================
Total params: 813,414
Trainable params: 813,350
Non-trainable params: 64
_________________________________________________________________
```

Model Parameters

Filter: size 3x3

Activation function:

- Relu: for Convolutional Layers, MaxPooling Layer and Dense except Fully Connected Layer.
- Softmax: for the fully Connected layer in Multiple Label classification. Softmax was built for the purpose of probabilistic classification.

Optimizer:

- SGD and learning rate start at 5e-3 and adjust over time using LearningRateSchedule function built in Keras

Results:

Model accuracy:

```
model.fit_generator(training_set, steps_per_epoch = TRAIN_IMG_CNT / batch_size, epochs = NUM_EPOCHS,
                    callbacks=callbacks,
                    validation_data = test_set, validation_steps = VALIDATION_CNT / batch_size)
model.save_weights('1st_run.h5')
```

```
Epoch 91/100
305/304 [==============================] - 10s 33ms/step - loss: 0.0743 - acc: 0.9736 - val_loss: 1.3238 - val_acc: 0.7754
Epoch 92/100
305/304 [==============================] - 10s 33ms/step - loss: 0.0737 - acc: 0.9713 - val_loss: 1.4556 - val_acc: 0.7557
Epoch 93/100
305/304 [==============================] - 10s 33ms/step - loss: 0.0789 - acc: 0.9721 - val_loss: 1.4525 - val_acc: 0.7451
Epoch 94/100
305/304 [==============================] - 10s 32ms/step - loss: 0.0664 - acc: 0.9742 - val_loss: 1.5095 - val_acc: 0.7329
Epoch 95/100
305/304 [==============================] - 10s 33ms/step - loss: 0.0744 - acc: 0.9721 - val_loss: 1.4269 - val_acc: 0.7542
Epoch 96/100
305/304 [==============================] - 11s 35ms/step - loss: 0.0697 - acc: 0.9734 - val_loss: 1.6327 - val_acc: 0.7238
Epoch 97/100
305/304 [==============================] - 10s 33ms/step - loss: 0.0710 - acc: 0.9746 - val_loss: 1.6378 - val_acc: 0.7269
Epoch 98/100
305/304 [==============================] - 10s 34ms/step - loss: 0.0698 - acc: 0.9748 - val_loss: 1.5895 - val_acc: 0.7329
Epoch 99/100
305/304 [==============================] - 10s 33ms/step - loss: 0.0623 - acc: 0.9768 - val_loss: 1.6094 - val_acc: 0.7344
Epoch 100/100
305/304 [==============================] - 10s 34ms/step - loss: 0.0727 - acc: 0.9721 - val_loss: 1.5964 - val_acc: 0.7375
```

Using the trained model to make some predictions:

In [29]:

```
for i in range(11,13):
    for ch in ['A','B', 'C']:
        filename = "Marcel-Test/" + ch + "/uniform/" + ch +(
        "-uniform" + str(i) + ".ppm")
        y = predict(filename)
        print(ch, y)
```

```
A [0]
B [1]
C [2]
A [3]
B [1]
C [2]
```

Morbitity and Mortality

64 32 128 6 - 70% kern=5x5
256 128 64 6 - slow
256 64 6 - slow, bad
128 64 6 - bad
32 128 6 - 57%
32 128 6 kern=3x3 - 45%
32 128 6 kern=7x7 - 50%
32 128 6 batch size = 32, kern = 5, opt=adam - 50%
64 128 6 batch 16 slow
16 128 6 - 55
del relu from first cov2d 50
32 32 128 6 kern=3 70 del white
32 32 256 6 - 65
32 32 192 6 - 70+
32 32 192 6 with poly_decay - 74
32 32 192 6 with more-train - meh
32 32 256 6 with marcel-train - 71
32 32 160 - 68
32 32 32! 160 6 - 61
32 16 16 64 - no
16 16 16 128 - no
64 16 16 128 - no
32 32 32 190 - no
32 32 192 - ok, 70
32 32 128 - good
24 24 190 - 70
24 24 190 no noise : overfit
24 24 190, 20% noise + 2 degrees: nope
10% and 0 degrees... - 73
32 32 192 2 degrees... 70
32 32 192 1 ... 73
32 32 128 1 ... 73

troubles.txt - Notepad

File Edit Format View Help

failed:
image processing to edges, position -- never started
tensor flow -- it never ran
aws -- no gpu for you
google cloud -- anomaly detected!
no noise -- overfitting
more noise -- no accuracy
running noise once -- overfitting
large epoch -- too slow
lots of nodes -- overfitting
even more layers -- no effect

???:
image size / scaling -- losing data going to 64x64?
monochrome vs color -- early tests favored grayscale
convolution & pooling on hidden layers (seems like image processing)

good:
installing Tensory flow GPU -- fast
keras -- intelligent defaults, got us going

Thank you