Pokemon Recognition Algorithm

1. Algorithm

1.1 Introduction

The Pokemon Recognition algorithm described here, classifies Pokemons from different categories using Bag of Words model and SVM technique.
This algorithm also recognise CP, HP and Stardust using OCR techniques, where it performs morphological operations to extract digits and then detects extracted digits using Neural Network architecture. Additionally, it predicts Pokemon Level using pre-trained Neural Network model.

1.2 Setup and Initialization

1. Create 'model.mat' file to store codebooks and different trained models.
2. Configure Pokemon Recognition model for different parameters (Configuration.m).
   > Path for training, and testing images
   > Allowed image types or pattern filters
   > Standard resolution for preprocessed images
   > Parameters for MSER, HoG, SURF and Color features
   > Parameters for K-Means quantization
   > Parameters for SVM
   > Parameters for Pokemon ID, CP, HP, Stardust and Pokemon Level models

1.3 Algorithm for Pokemon ID

1.3.1 Training

1. Crop region of interest for Pokemon picture from training image.
2. Preprocess cropped training images,
   a. Convert image to single precision.
   b. Standardize image using fixed resolution for all images.
      e.g. (150, 150, 3)
3. Create a codebook of visual words.
   Note: This algorithm generates codebooks for two different types of features.
   - MSER
   - HoG

   a. MSER Codebook:
      - Detect MSER regions.
      - Extract MSER local features using SURF method.
      - Quantize MSER local features using K-Means algorithm.
      - Save MSER codebook into 'model.mat' file.

   b. HoG Codebook:
      - Detect Corners using FAST algorithm.
      - Extract HoG local features around Corner points.
      - Quantize HoG local features using K-Means algorithm.
      - Save HoG codebook in 'model.mat' file.

4. Generate image histograms over vocabulary of visual words.
   For every cropped training image,
      For every codebook(feature) type,
         a. Extract local features for the given image.
            e.g. MSER, HoG, SURF, etc.
         b. Assign every extracted local feature to respective closest cluster (visual word).
            Note: Here, algorithm refers codebooks generated in previous step.
         c. Compute visual word frequencies for a given image.
            Note: This step generates a histogram of visual word frequencies for a given image.
         d. Additionally, generate Color histogram using below steps,
            I. Transform image from RGB to HSV color space.
            II. Normalize intensity values as per the maximum buckets available for every color channel.
            e.g.
            h = int64(floor(hsv(:, :, 1) * 350));
            III. Count total number of pixels in every bucket for every color channel.
            IV. Combine all 3 color channels to generate Color histogram.
   Note: Here, we horizontally concatenate all histograms generated in the above steps.
5. Create a SVM model.
6. Train SVM model over histograms generated in the above steps.
7. Save trained SVM model into 'model.mat' file (i.e. POKEMON_MODEL).

1.3.2 Testing

1. Crop region of interest for Pokemon picture from testing image.
2. Preprocess cropped testing images using exactly same procedure as specified in Training section.
3. Extract local features from cropped testing images.
   Note: This algorithm extracts below local features as specified in Training section.
   - MSER
   - HoG
4. Generate histograms over vocabulary of visual words for the cropped testing images.
   Note: This algorithm generates histogram over visual words as specified in Training section.
5. Predict Pokemon ID for the cropped testing image using trained SVM model.

1.3.3 Accuracy Measurement

The algorithm computes accuracy simply based on total number of correct predictions against total number of test images.
e.g.

```
accuracy = sum(PredictedLabel==OriginalLabel) ./ Total test images
```

1.4 Algorithm for CP

1.4.1 Training

1. Crop region of interest for CP portion from training image, i.e. Text patch.
2. Perform below set of morphological operations over text patch,
    - Resize the text patch by keeping aspect ratio same.
    - Convert the text patch from RGB to Grayscale color space.
    - Perform Median Filtering to remove noise.
    - Dilate text patch with the disk structural element.
    - Erode text patch with the disk structural element.
    - Perform Edge Enhancement using Morphological Gradient.
    - Convert text patch to intensity image.
    - Perform two dimensional convolution over text patch.
    - Scale intensities between the range 0 to 1.
    - Convert text patch to binary form.
    - Perform thinning over text patch to ensure character isolation.
    - Select all the regions from text patch that are of pixel area more than 50.
    - Segment characters based on vertical histogram over text patch.
    - Segment characters based on horizontal histogram over text patch.
3. Crop number patch, i.e. region of interest and perform below set of operations on it,
    - Extract and label connected components in number patch.
    - Measure properties of the connected components.
4. Determine expected total digits from a given label.
5. Find the total number of valid digit components from number patch using pre-computed width and height thresholds.
6. If the total number of valid digit components from the number patch is equal to the expected digit count,
   Then,
    a. Preprocess digit component as below,
        - Convert digit to double precision.
        - Standardize digit using fixed resolution.
          e.g. (100, 100, 3)
    b. Add extracted valid digit components into training set of digits
7. Create a Pattern Recognition Network model.
   Note: Please refer 'Section 1.8' for the detail architecture of Pattern Recognition Network model.
8. Train a Pattern Recognition Network model over extracted digits.
9. Save trained model into 'model.mat' file (i.e. CP_MODEL).

1.4.2 Testing

1. Crop region of interest for CP portion from testing image, i.e. Text patch.
2. Perform set of morphological operations over text patch using exactly same procedure as specified in Training section.
3. Crop number patch and perform set of operations on it as specified in Training section.
4. Find the valid digit components from number patch using pre-computed width and height thresholds.

5. For every digit component,
    a. Preprocess digit component as below,
        - Convert digit to double precision.
        - Standardize digit using fixed resolution.
          e.g. (100, 100, 3)
    b. Predict exact digit using trained Pattern Recognition Network model.
6. And finally, predict CP by forming the whole CP number using predicted digits.

1.4.3 Accuracy Measurement

The algorithm computes accuracy simply based on total number of correct predictions against total number of test images.
e.g.

```
accuracy = sum(PredictedLabel==OriginalLabel) ./ Total test images
```

1.5 Algorithm for HP

1.5.1 Training

1. Crop region of interest for HP portion from training image.
2. Preprocess cropped training images,
    a. Convert image to single precision.
    b. Standardize image using fixed resolution for all images.
       e.g. (150, 150, 3)
3. Create a codebook of visual words.
   Note: This algorithm generates codebooks for three different types of features.
        - MSER
        - HoG
    a. MSER Codebook:
       Note: MSER features are useful to detect text/digits in images. Hence, using MSER features.
        - Detect MSER regions.
        - Extract MSER local features using SURF method.
        - Quantize MSER local features using K-Means algorithm.
        - Save MSER codebook into 'model.mat' file.
    b. HoG Codebook:
        - Detect Corners using FAST algorithm.
        - Extract HoG local features around Corner points.
        - Quantize HoG local features using K-Means algorithm.
        - Save HoG codebook in 'model.mat' file.
4. Generate image histograms over vocabulary of visual words.
   For every cropped training image,
   For every codebook(feature) type,
    a. Extract local features for the given image.
       e.g. MSER, HoG, etc.
    b. Assign every extracted local feature to respective closest cluster (visual word).
       Note: Here, algorithm refers codebooks generated in previous step.
    c. Compute visual word frequencies for a given image.
       Note: This step generates a histogram of visual word frequencies for a

given image.

Note: Here, we horizontally concatenate all histograms generated in the above steps.

5. Create a SVM model.
6. Train SVM model over histograms generated in the above steps.
7. Save trained SVM model into 'model.mat' file (i.e. HP_MODEL).

1.5.2 Testing

1. Crop region of interest for HP portion from testing image.
2. Preprocess cropped testing images using exactly same procedure as specified in Training section.
7. Extract local features from cropped testing images.
   Note: This algorithm extracts below local features as specified in Training section.
   - MSER
   - HoG
8. Generate histograms over vocabulary of visual words for the cropped testing images.
   Note: This algorithm generates histogram over visual words as specified in Training section.
9. Predict HP for the cropped testing image using trained SVM model.

1.5.3 Accuracy Measurement

The algorithm computes accuracy simply based on total number of correct predictions against total number of test images.

e.g.

```
accuracy = sum(PredictedLabel==OriginalLabel) ./ Total test images
```

1.6 Algorithm for Stardust

1.6.1 Training

1. Crop region of interest for Stardust portion from training image, i.e. Text patch.
2. Perform below set of morphological operations over text patch,
   - Resize the text patch by keeping aspect ratio same.
   - Convert the text patch from RGB to Grayscale color space.
   - Perform Median Filtering to remove noise.
   - Dilate text patch with the disk structural element.
   - Erode text patch with the disk structural element.
   - Perform Edge Enhancement using Morphological Gradient.
   - Convert text patch to intensity image.
   - Perform two dimensional convolution over text patch.
   - Convert text patch to binary image by thresholding.
   - Perform thinning over text patch to ensure character isolation.
   - Select all the regions from text patch that are of pixel area more than 50.
   - Segment characters based on vertical histogram over text patch.
   - Segment characters based on horizontal histogram over text patch.
3. Crop number patch, i.e. region of interest and perform below set of operations on it,
   - Extract and label connected components in number patch.
   - Measure properties of the connected components.
4. Determine expected total digits from a given label.

5. Find the total number of valid digit components from number patch using pre-computed width and height thresholds.
6. If the total number of valid digit components from the number patch is equal to the expected digit count,
   Then,
   c. Preprocess digit component as below,
      - Convert digit to double precision.
      - Standardize digit using fixed resolution.
        e.g. (100, 100, 3)
   d. Add extracted valid digit components into training set of digits
7. Create a Pattern Recognition Network model.
   Note: Please refer 'Section 1.8' for the detail architecture of Pattern Recognition Network model.
8. Train a Pattern Recognition Network model over extracted digits.
9. Save trained model into 'model.mat' file (i.e. STARDUST_MODEL).

1.6.2 Testing

1. Crop region of interest for Stardust portion from testing image, i.e. Text patch.
2. Perform set of morphological operations over text patch using exactly same procedure as specified in Training section.
3. Crop number patch and perform set of operations on it as specified in Training section.
4. Find the valid digit components from number patch using pre-computed width and height thresholds.
5. For every digit component,
   a. Preprocess digit component as below,
      - Convert digit to double precision.
      - Standardize digit using fixed resolution.
        e.g. (100, 100, 3)
   b. Predict exact digit using trained Pattern Recognition Network model.
6. And finally, predict Stardust by forming the whole Stardust number using predicted digits.

1.6.3 Accuracy Measurement

The algorithm computes accuracy simply based on total number of correct predictions against total number of test images.
e.g.

```
accuracy = sum(PredictedLabel==OriginalLabel) ./ Total test images
```

1.7 Algorithm for Pokemon Level

1.7.1 Training

1. Crop region of interest for Pokemon Level portion from training images.
2. For every cropped image,
   a. Preprocess image as below,
      - Convert digit to double precision.
      - Standardize digit using fixed resolution.
        e.g. (100, 100, 3)
   b. Add pre-processed image into training set.

3. Create a Pattern Recognition Network model.
   Note: Please refer 'Section 1.8' for the detail architecture of Pattern Recognition Network model.
4. Train a Pattern Recognition Network model over Pokemon Level patches.
5. Save trained model into 'model.mat' file (i.e. POKEMON_LEVEL_MODEL).

1.7.2 Testing

1. Crop region of interest from testing image, i.e. first half part of testing image.
2. Perform below set of morphological operations over region of interest,
   - Resize the patch by keeping aspect ratio same.
   - Convert the patch from RGB to Grayscale color space.
   - Perform Median Filtering to remove noise.
   - Dilate patch with the disk structural element.
   - Erode patch with the disk structural element.
   - Perform Edge Enhancement using Morphological Gradient.
   - Convert patch to intensity image.
   - Perform two dimensional convolution over patch.
   - Scale intensities between the range 0 to 1.
   - Convert patch to binary form.
   - Perform thinning over patch to ensure character isolation.
   - Select all the regions from patch that are of pixel area more than 50.
3. Mask the middle portion of level patch, which is a potential Pokemon picture.
4. Detect all possible circles from level patch that are within specified radius range.
5. Modify center locations of detected circles to original X-Y scale.
6. Run trained Pokemon Level model over all extracted circle patches and find out probability of being level patch.
6. Pick the circle which has the maximum probability of being Level patch and accordingly record corresponding center location.
7. And finally, predict location of Pokemon Level as per detected center in a given image of Pokemon.

1.8 Pattern Recognition Neural Network

This algorithm uses Pattern Recognition Neural Network for detecting CP, Stardust and Pokemon Level.

1.8.1 Neural Network Architecture

Pattern Recognition Neural Network used in this algorithm has one Input Layer (input size [20000]), five Hidden Layers (20, 40, 80, 40, 20 activation units respectively) and one output layer (output class [3]).

Splitting training set:

Training data used by this architecture is split into Training, Validation and Testing sets as below:

```
PatternNet.divideFcn      = 'dividerand';
PatternNet.divideMode = 'sample';
PatternNet.divideParam.trainRatio = 75/100;
PatternNet.divideParam.valRatio     = 20/100;
```

PatternNet.divideParam.testRatio = 5/100;

Training function:

Here, we are using Scaled Conjugate Gradient backpropagation.

PatternNet.trainFcn = 'trainscg';

Performance check:

For checking performance of neural network, crossentropy technique is used.

PatternNet.performFcn = 'crossentropy';

Tuning parameters:

1. max_fail: Maximum validation failures (Here, it is set to 8).
2. epochs: Maximum number of epochs to train (Here, it is set to 100).

Regularization:

Regularization parameter is set for generalizing Pattern Recognition Network and for avoiding overfitting.

PatternNet.performParam.regularization = 0.5;

2. Experiments

The experiments are done with described algorithm for classifying ~150 testing images of different categories.

2.1 Pokemon ID

2.1.1 Training Statistics

Parameter	Value
Training images	~1100 images from 150 different categories
Codebook Types	1. MSER (100 Features) 2. HoG (200 Features) 3. Color (400 Features)
Standard resolution	[150, 150]
K-Means Cluster Points	700
Machine Learning model	SVM (RBF Kernel)

Note: For more details, please refer 'Pokemon' structure in 'Configuration.m' file.

2.1.2 Testing Accuracy

43.26%

2.2 CP

2.2.1 Training Statistics

Parameter	Value
Training images	~1100 images from 150 different categories
Standard resolution	[100, 100]
Machine Learning model	Pattern Recognition Network

Note: For more details, please refer 'CP' structure in 'Configuration.m' file.

2.2.2 Testing Accuracy

60.28%

2.3 HP

2.3.1 Training Statistics

Parameter	Value
Training images	~1100 images from 150 different categories
Codebook Types	1. MSER (200 Features) 2. HoG (200 Features)
Standard resolution	[150, 150]
K-Means Cluster Points	400
Machine Learning model	SVM (RBF Kernel)

Note: For more details, please refer 'HP' structure in 'Configuration.m' file.

2.3.2 Testing Accuracy

46.81%

2.4 Stardust

2.4.1 Training Statistics

Parameter	Value
Training images	~1100 images from 150 different categories
Standard resolution	[100, 100]
Machine Learning model	Pattern Recognition Network

Note: For more details, please refer ‘Stardust’ structure in ‘Configuration.m’ file.

2.4.2 Testing Accuracy

40.43%

2.5 Pokemon Level

2.5.1 Training Statistics

Parameter	Value
Training images	~1100 images from 150 different categories
Standard resolution	[100, 100]
Machine Learning model	Pattern Recognition Network

Note: For more details, please refer ‘PokemonLevel’ structure in ‘Configuration.m’ file.

2.5.2 Testing Accuracy

~ 7/10

3. Improvements

3.1 Using SIFT and SURF

Accuracy of Pokemon and HP model can further be increased by including SIFT and SURF features.

3.2 Hierarchical Clustering

Hierarchical clustering can be used to increase speed of algorithm.

3.3 Sliding Window technique

Sliding window technique can be used to extract digits from images for training purpose. It will improve performance of an algorithm and there will be no need of manually tuning patches.

3.4 Tuning Neural Network

Neural Network architectures used in CP, Stardust and Pokemon Level models should be tuned further for improvising performance.

4. References

1. http://yann.lecun.com/exdb/publis/pdf/jackel-95.pdf
2. http://www.slideshare.net/HiraRizvi/final-project-presentation1-28562412?next_slideshow=1

5. Repository

https://github.com/chetanborse007/Pokemon-Recognition-Algorithm.git