Artificial Intelligent Theory

FACIAL EMOTION RECOGNITION IN THE WILD

Prepared by:

Tran Nguyen Quynh Tram (198507)

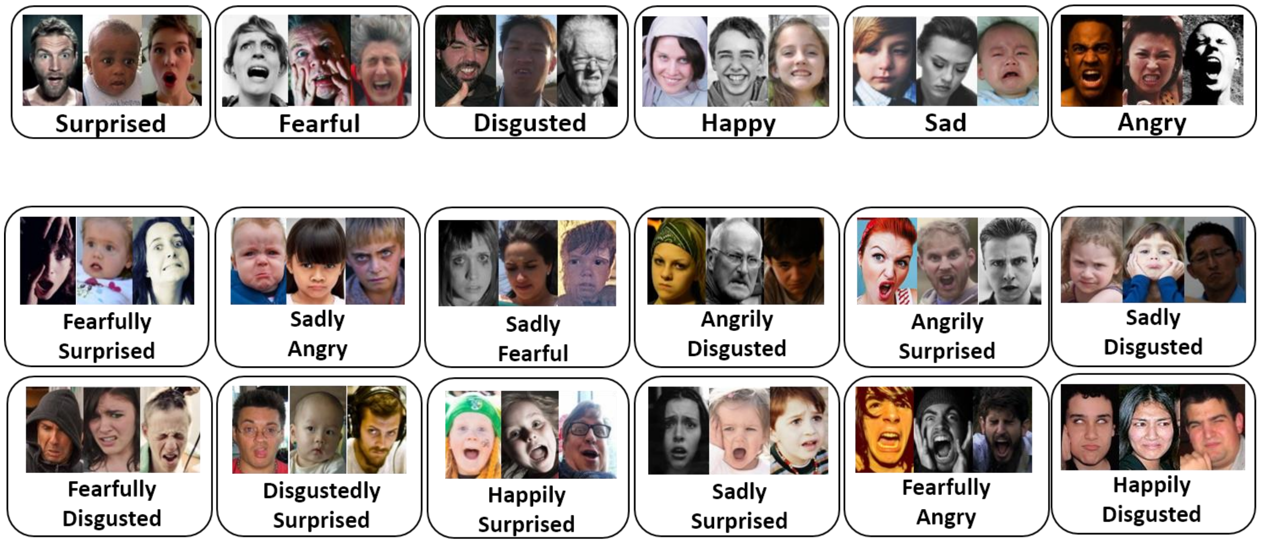
June, 11th, 2019

Agenda

1. Introduction

Facial Emotion Recognition

- ***Detect seven facial expressions*** with universal meaning: ***anger, disgust, fear, happiness, sadness, surprise and neutral***

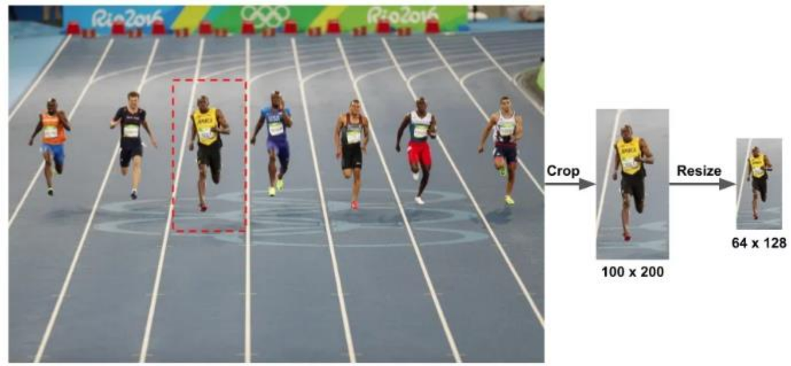


- Challenging: complex and dynamic properties
  - changing over time,
  - mixing with other factors, and
  - inherently multimodal in behavior, physiology, and language

2. Related Research and Key Technology

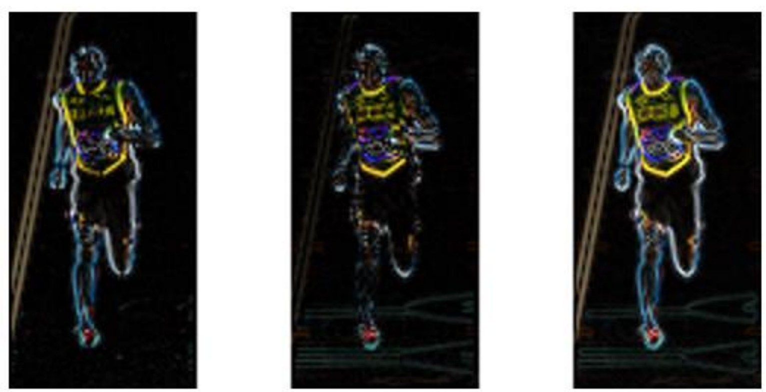
HOG Feature Extraction

- **Step 1) Pre-processing**: normalize gamma and color



Original Image : 720 x 475

- **Step 2) Calculate the Gradient Images**



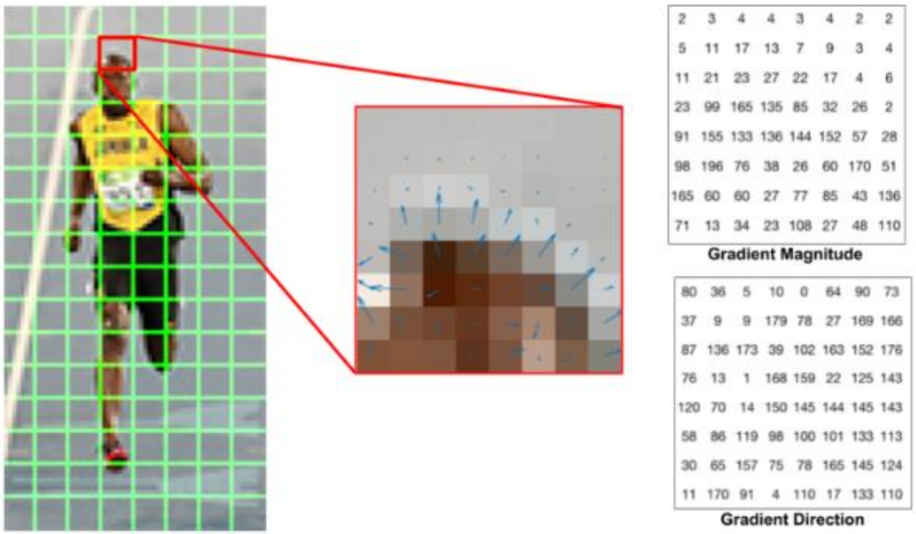
Left : Absolute value of x-gradient. Center : Absolute value of y-gradient.
Right : Magnitude of gradient.

[1] Dalal, Navneet, and Bill Triggs. "***Histograms of oriented gradients for human detection***." *international Conference on computer vision & Pattern Recognition (CVPR'05)*. Vol. 1. IEEE Computer Society, 2005.
[2] https://www.learnopencv.com/histogram-of-oriented-gradients/

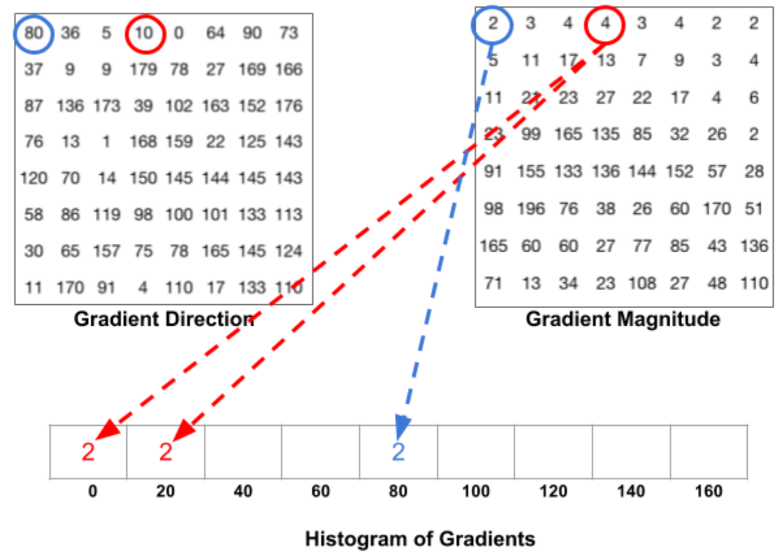
2. Related Research and Key Technology

HOG Feature Extraction

- **Step 3 : Calculate Histogram of Gradients in 8×8 cells**



Center : The RGB patch and gradients represented using arrows. Right : The gradients in the same patch represented as numbers

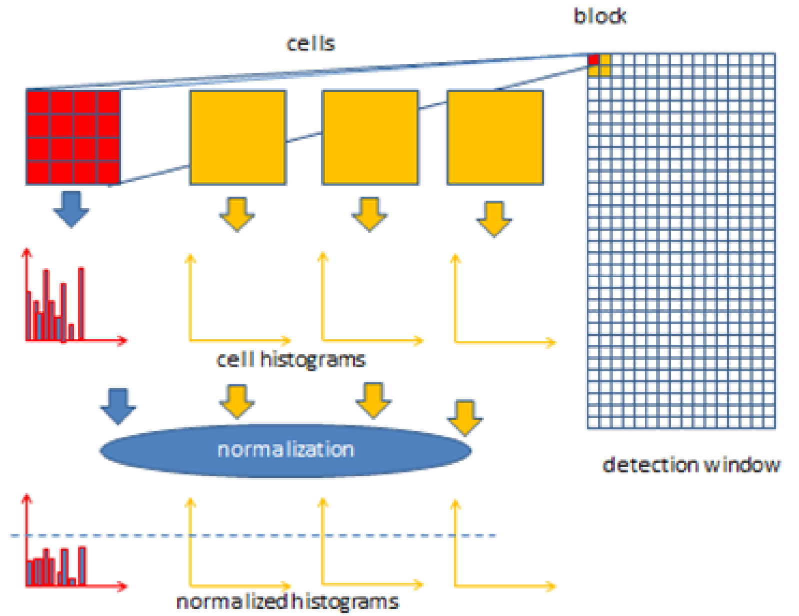


[1] Dalal, Navneet, and Bill Triggs. "***Histograms of oriented gradients for human detection***." *international Conference on computer vision & Pattern Recognition (CVPR'05)*. Vol. 1. IEEE Computer Society, 2005.
[2] https://www.learnopencv.com/histogram-of-oriented-gradients/

2. Related Research and Key Technology

HOG Feature Extraction

- **Step 4 : Block Normalization:** Invariant with illumination changes

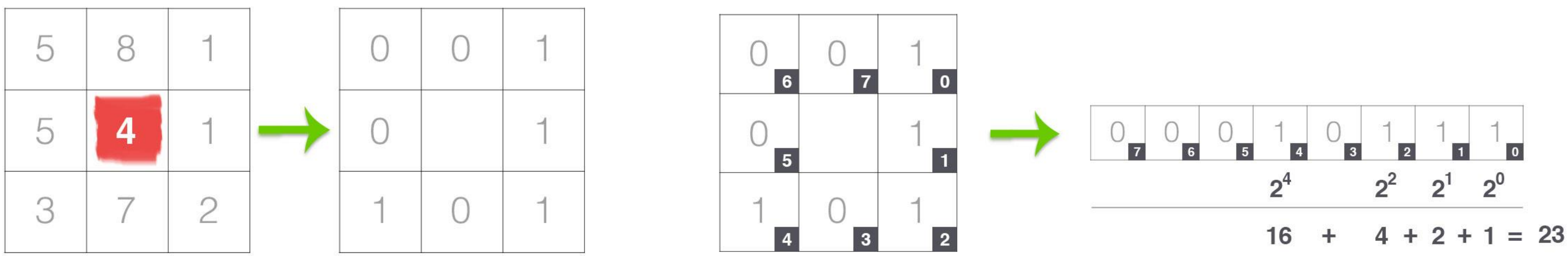


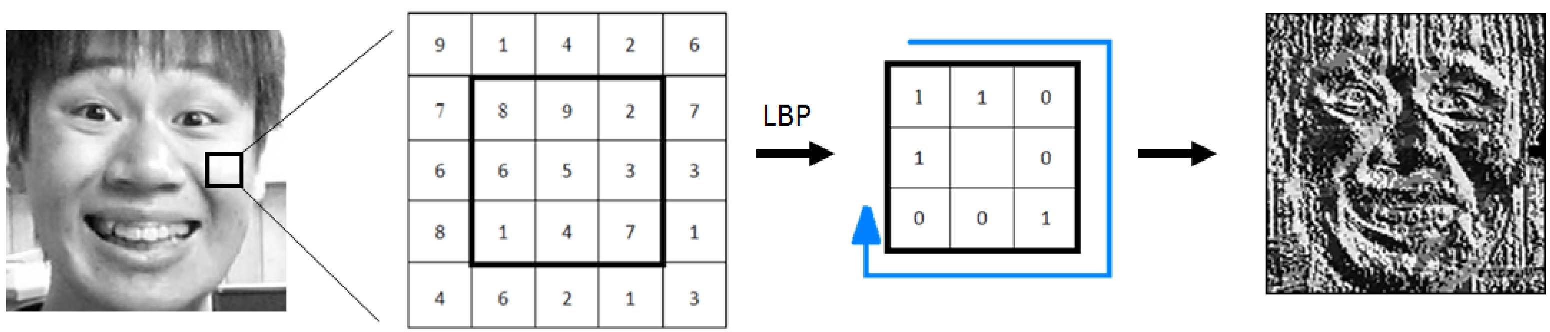
- **Step 5 : Calculate the HOG feature vector**
  - To calculate the final feature vector for the entire image
  - Image 64 x 128, cell_size 8x8, block_size 16x16, orientation 9
  - Sliding block 16x16 → 7 horizontal and 15 vertical positions → 105 positions
  - Each 16×16 block is represented by a 36×1 vector (due to block normalization 4 x 9 = 36 x 1)
  - One gaint vector w36×105 = 3780 dimensional vector

2. Related Research and Key Technology

LBP FEATURES

- **Local Binary Patterns**
  - constructed by comparing each pixel with its surrounding neighborhood of pixels
  - intensity of the center pixel is >= to its neighbor, then set 1; otherwise, to 0



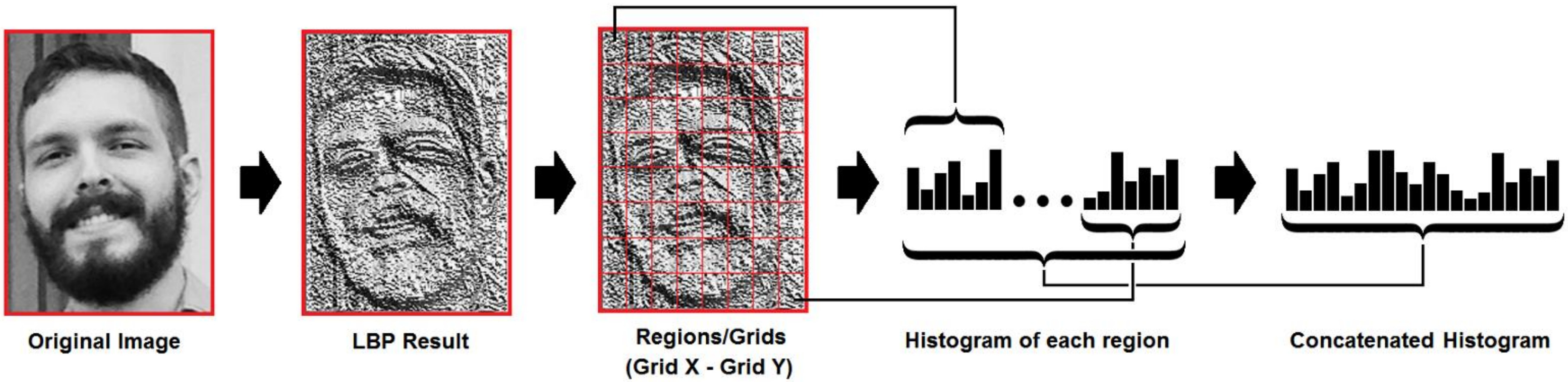


[1] T. Ojala, M. Pietikäinen, and T. Mäenpää, "***Multiresolution gray-scale and rotation invariant texture classification with local binary patterns***," IEEE Transactions on Pattern Analysis & Machine Intelligence, no. 7, pp. 971–987, 2002.

2. Related Research and Key Technology

LBP FEATURES

- **LBP Feature Extraction**
  - Calculate LBP Images
  - Divide into blocks
  - Calculate LBP patch histograms, and normalization
  - Concate all histograms

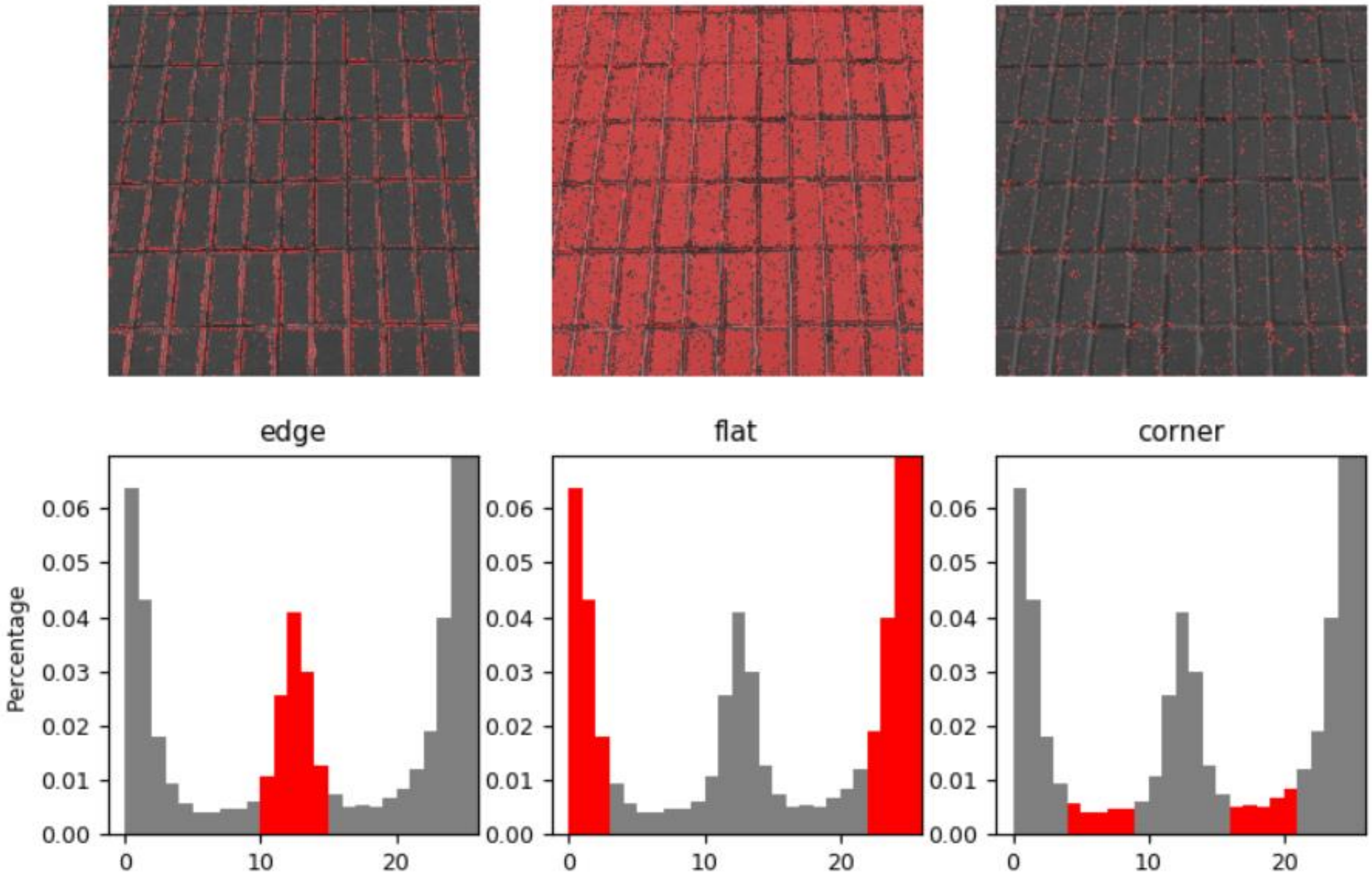


[1] T. Ojala, M. Pietikäinen, and T. Mäenpää, "***Multiresolution gray-scale and rotation invariant texture classification with local binary patterns***," IEEE Transactions on Pattern Analysis & Machine Intelligence, no. 7, pp. 971–987, 2002.

2. Related Research and Key Technology

LBP FEATURES

- **Robust with Texture Classification**

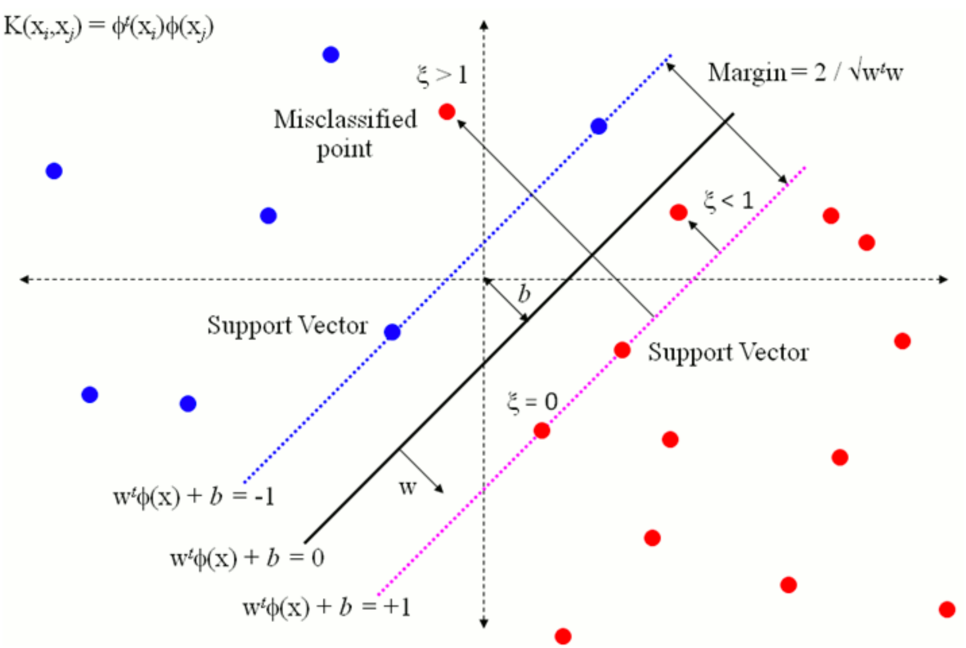


https://scikit-image.org/docs/dev/auto_examples/features_detection/plot_local_binary_pattern.html

2. Related Research and Key Technology

SVM Classification

- Constructs a hyperplane or set of hyperplanes in a high-dimensional space, which can be used for classification
- A good separation is achieved by the hyperplane that has the largest distance to the nearest training-data pointof any class.

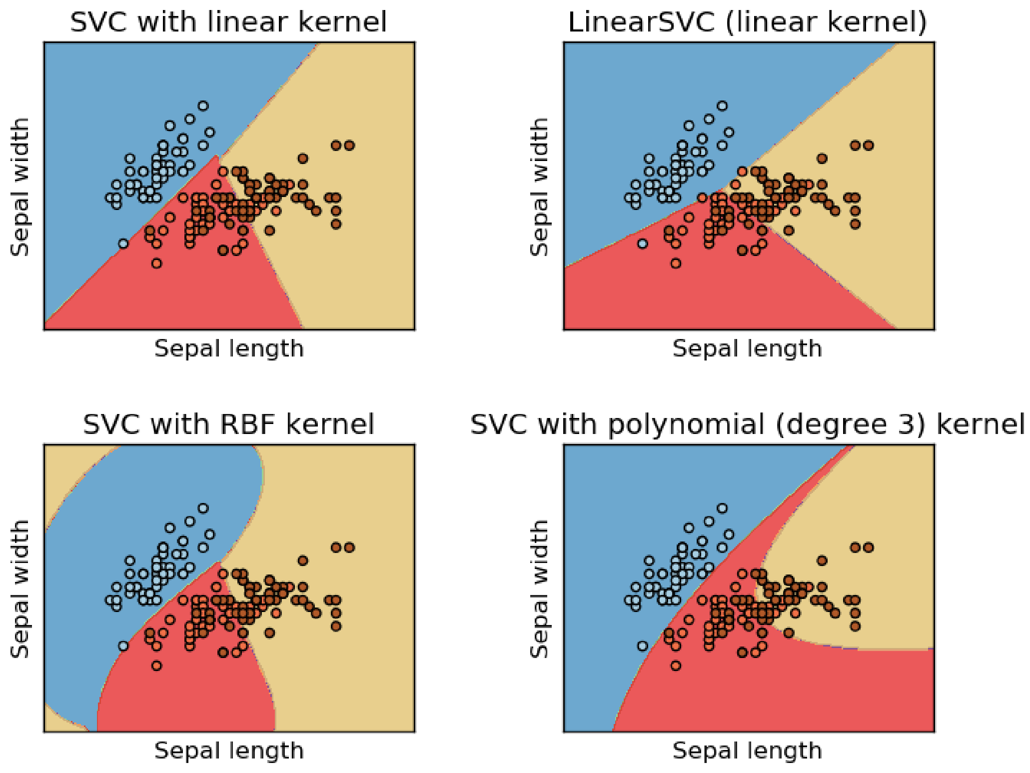


[1] J. A. K. Suykens and J. Vandewalle, "***Least squares support vector machine classifiers***," *Neural processing letters*, vol. 9, no. 3, pp. 293–300, 1999.

2. Related Research and Key Technology

SVM Kernel Parameter

- Kernel parameters selects the type of hyperplane used to separate the data.
  - Kernel = Linear, RBF, Polynominal

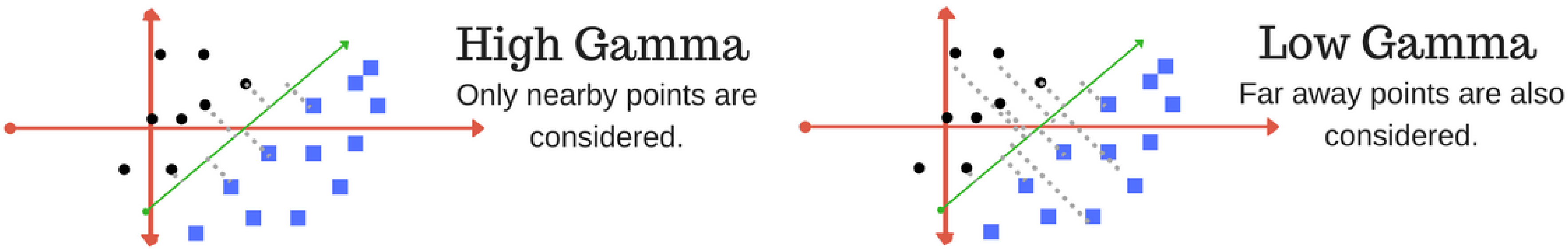


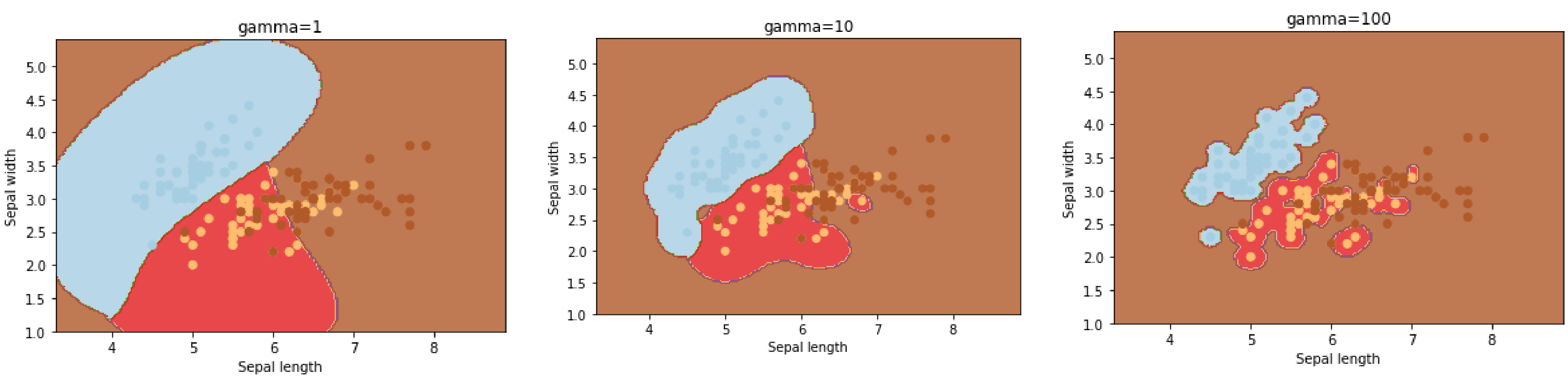
https://medium.com/all-things-ai/in-depth-parameter-tuning-for-svc-758215394769

2. Related Research and Key Technology

SVM Gama Parameter

- gamma is a parameter for non linear hyperplanes. The higher the gamma value it tries to exactly fit the training data set



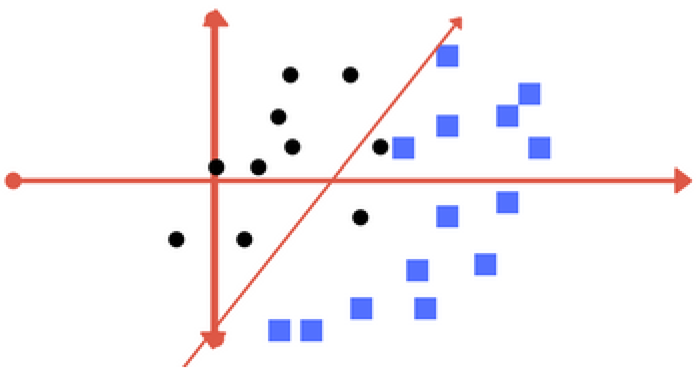


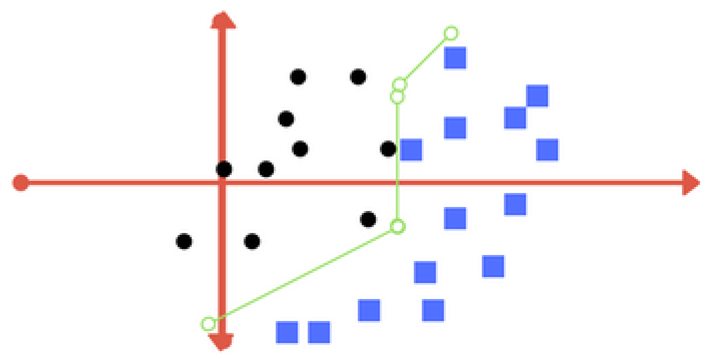
https://medium.com/all-things-ai/in-depth-parameter-tuning-for-svc-758215394769

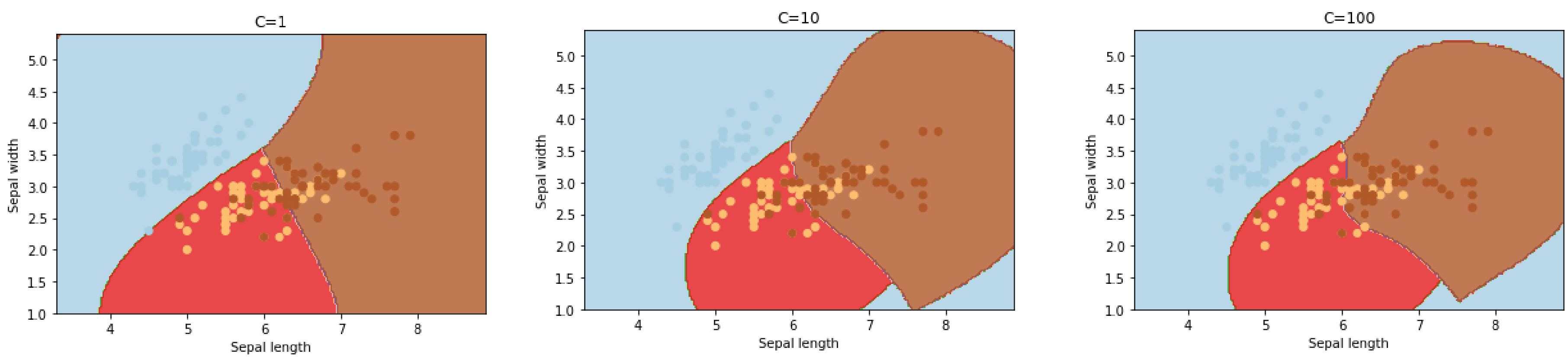
2. Related Research and Key Technology

SVM Regulazation Parameter

- C is the penalty parameter of the error term. It controls the trade off between smooth decision boundary and classifying the training points correctly. ***Increasing C values may lead to overfitting the training data***.







https://medium.com/all-things-ai/in-depth-parameter-tuning-for-svc-758215394769

3. Project Contents and Algorithm

Data preprocessing

- Indexing data from RAF-DB Dataset

BUILD RAF-DB INDEXING DATABASE

[1]:
```
# %load_ext autoreload
%reload_ext autoreload
%autoreload 2

from rafdb_lib import build_rafdb_basic
```

[4]:
```
build_rafdb_basic(rafdb_basic_dir = "./data/rafdb/basic", save_name = "rafdb_basic.hdf5", table_name = "data")
```

```
Step1. Read train emotion data
Read train: 100%|████████████████████████████████████████| 12271/12271 [00:57<00:00, 216.73it/s]
Step2. Read test emotion data
Read test: 100%|████████████████████████████████████████| 3068/3068 [00:14<00:00, 214.13it/s]
Step3. Build RAF-DB Basic
C:\Anaconda3\envs\cur35\lib\site-packages\pandas\core\generic.py:1996: PerformanceWarning:
your performance may suffer as PyTables will pickle object types that it cannot
map directly to c-types [inferred_type->mixed,key->block1_values] [items->['id', 'type', 'image_aligned', 'image_original', 'bbox', 'eye1', 'eye2', 'nose', 'mouth1', 'mouth2', 'landmarks']]

  return pytables.to_hdf(path_or_buf, key, self, **kwargs)
```

[4]:

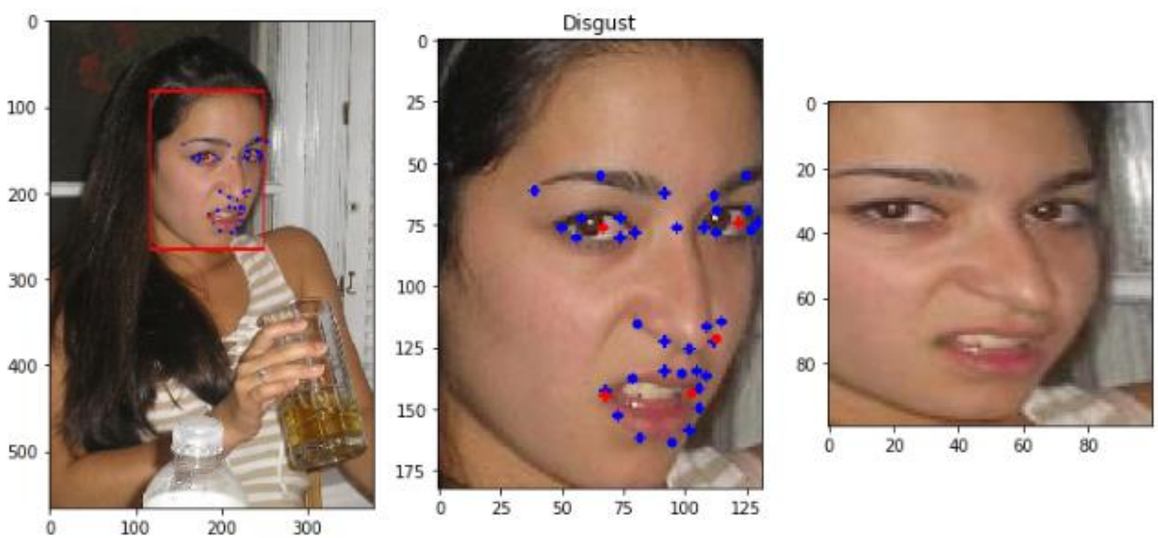
	id	type	emotion	image_aligned	image_original	bbox	eye1	eye2	nose	mouth1	mouth2	gender	race	age	landmarks
0	train_00001	train	5	Image/aligned/train_00001_aligned.jpg	Image/original/train_00001.jpg	[213.92009, 199.133484, 484.33071900000004, 52...	[270.17, 330.696]	[374.086, 302.67]	[325.068, 401.369]	[312.03, 445.782]	[399.256, 424.842]	1	0	2	[[228.0, 313.0], [256.0, 299.5], [291.0, 301.0...
1	train_00002	train	5	Image/aligned/train_00002_aligned.jpg	Image/original/train_00002.jpg	[407.148193, 265.544525, 912.439697, 1029.660645]	[532.365, 573.652]	[753.071, 574.156]	[604.688, 702.167]	[563.467, 861.359]	[736.37, 850.951]	1	0	2	[[471.0, 512.0], [517.0, 494.0], [576.0, 493.0...
2	train_00003	train	4	Image/aligned/train_00003_aligned.jpg	Image/original/train_00003.jpg	[418.01532000000003, 345.02063, 772.46106, 872...	[463.469, 554.446]	[596.902, 515.023]	[472.566, 621.163]	[475.599, 748.531]	[604.484, 718.206]	1	2	2	[[425.0, 525.0], [438.0, 486.0], [475.5, 485.7...

3. Project Contents and Algorithm

Data preprocessing

- Reviewing face annotations

```
[71]: plt.figure(figsize=(12, 12))
plt.subplot(1,3,1)
image_id = "test_0011"
rafdb_read_view_image(df_rafdb, rafdb_info["root_dir"], image_id = image_id, get_only_face = False,
                      draw_landmarks = True, draw_bbox = True, draw_points = True,
                      verbose = 1, fill_width = 2, line_width = 2);
plt.subplot(1,3,2)
rafdb_read_view_image(df_rafdb, rafdb_info["root_dir"], image_id = image_id, get_only_face = True,
                    draw_landmarks = True, draw_bbox = False, draw_points = True, verbose = 1, fill_width = 2, line_width = 2);
plt.title("%s"%label_mapping[df_rafdb.query("id=='%s'"%(image_id))["emotion"].values[0]])

plt.subplot(1,3,3)
read_aligned_image(df_rafdb, rafdb_info["root_dir"], image_id = image_id, verbose=1)
plt.show()
```



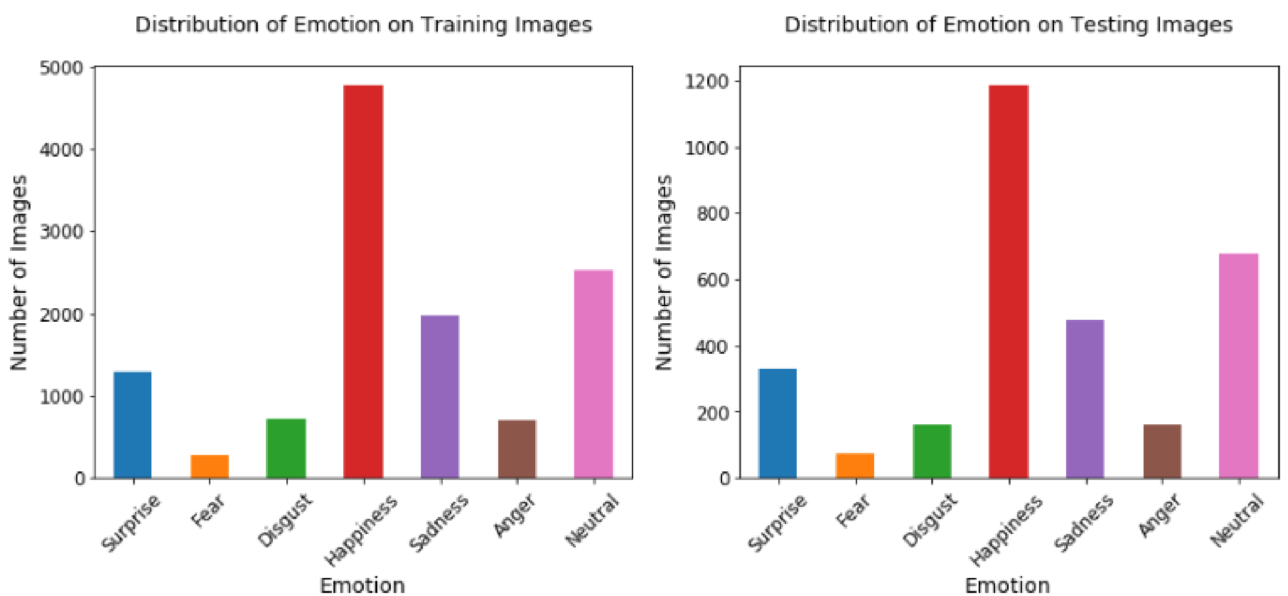
```python
def read_view_image(db, root_dir, image_id = "train_00010", get_only_face = True,
                    draw_landmarks = True, draw_bbox = True, draw_points = True,
                    verbose = 1, fill_width = 5, line_width = 5):
    """
    Read image from dataset & annotation
    + read_view_image(df_rafdb, rafdb_info["root_dir"], get_only_face = True, draw_annotation = True, verbose = 1)
    """
    db_data = db.set_index(keys=["id"], drop=False)

    image_path = os.path.join(root_dir, db_data.loc[image_id]["image_original"]).replace("\\", "/")
    image = cv2.imread(image_path)
    image_info = db_data.loc[image_id].copy()

    image_info["bbox"][0] = np.clip(image_info["bbox"][0], 0, image.shape[1] - 1)
    image_info["bbox"][1] = np.clip(image_info["bbox"][1], 0, image.shape[0] - 1)
    image_info["bbox"][2] = np.clip(image_info["bbox"][2], image_info["bbox"][0], image.shape[1] - 1)
    image_info["bbox"][3] = np.clip(image_info["bbox"][3], image_info["bbox"][1], image.shape[0] - 1)
    image_info["bbox"] = image_info["bbox"].astype(dtype=np.int)


    if draw_landmarks == True:
        # Draw 37 landmarks
        for point in image_info["landmarks"]:
            cv2.circle(image, (int(point[0]), int(point[1])), fill_width, (255, 0, 0), -1)
            pass
        # for
    # if

    if draw_points == True:
        # Draw 5 basic alignment points
        cv2.circle(image, (int(image_info["eye1"][0]), int(image_info["eye1"][1])), fill_width, (0, 0, 255), -1)
        cv2.circle(image, (int(image_info["eye2"][0]), int(image_info["eye2"][1])), fill_width, (0, 0, 255), -1)
        cv2.circle(image, (int(image_info["nose"][0]), int(image_info["nose"][1])), fill_width, (0, 0, 255), -1)
        cv2.circle(image, (int(image_info["mouth1"][0]), int(image_info["mouth1"][1])), fill_width, (0, 0, 255), -1)
        cv2.circle(image, (int(image_info["mouth2"][0]), int(image_info["mouth2"][1])), fill_width, (0, 0, 255), -1)
    # if

    if draw_bbox == True:
        # Draw bounding box
        cv2.rectangle(image, (int(image_info["bbox"][0]), int(image_info["bbox"][1])), (int(image_info["bbox"][2]), int(image_info["bbox"][3])), (0, 0, 255), line_width)
    # if

    if get_only_face == True:
        image = image[int(image_info["bbox"][1]): int(image_info["bbox"][3]), int(image_info["bbox"][0]): int(image_info["bbox"][2])]
    # if

    if verbose == 1:
        plt.imshow(image[...,::-1])
    pass

    return image, image_info
# read_view_image
```

3. Project Contents and Algorithm

Data analysis

- Emotion Distribution for train/test images

[8]:
```python
# Draw the distribution
plt.figure(figsize=(14, 5))

ax = plt.subplot(1,2,1)
df_hist_emotion.query("type=='train'").plot(x="emotion", y = "id", kind='bar', legend = False, ax = ax)
plt.xticks(np.arange(len(label_mapping)), list(label_mapping.values()), fontsize = 12, rotation = 45)
plt.yticks(fontsize = 12)
plt.title("Distribution of Emotion on Training Images\n", fontsize = 14)
plt.ylabel("Number of Images", fontsize = 14), plt.xlabel("Emotion", fontsize = 14)

ax = plt.subplot(1,2,2)
df_hist_emotion.query("type=='test'").plot(x="emotion", y = "id", kind='bar', legend = False, ax = ax)
plt.xticks(np.arange(len(label_mapping)), list(label_mapping.values()), fontsize = 12, rotation = 45)
plt.yticks(fontsize = 12)
plt.title("Distribution of Emotion on Testing Images\n", fontsize = 14)
plt.ylabel("Number of Images", fontsize = 14), plt.xlabel("Emotion", fontsize = 14)

plt.show()
```



3. Project Contents and Algorithm

Data analysis

- Review images on Dataset:
  - unconstrain, many poses and illumination changes

3. Project Contents and Algorithm

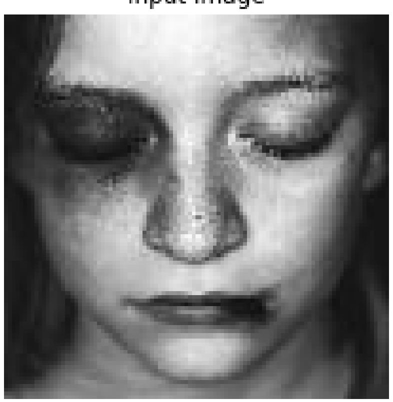
HOG Feature Extraction

- Image size 100x100, pixels_per_cell 8x8, cells_per_block 3x3, orientation 9
  - Feature dimension 8100
  - Good representation for facial emotion regions (nose, mouth, eye, forehead)

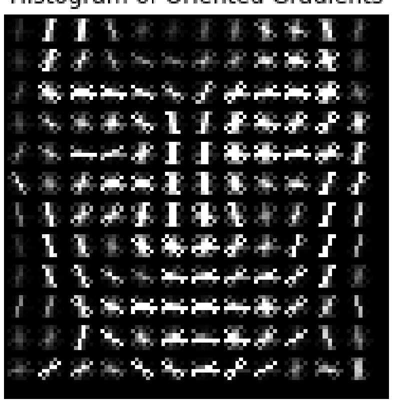
[14]:
```
# image, info = rafdb_read_view_image(db = df_rafdb, image_id = "train_00010", **default_read_settings)
image, info = read_aligned_image(db = df_rafdb, root_dir =  rafdb_info["root_dir"], image_id = "train_01010", verbose = 0)
# image_resized = cv2.resize(image, image_size)

feature = hog_feature_extraction(image[...,::-1], pixels_per_cell=(8, 8), cells_per_block=(3, 3), orientations=9, verbose = 1)
```

```
C:\Anaconda3\envs\cur35\lib\site-packages\skimage\feature\_hog.py:150: skimage_deprecation: Default value of `block_norm`==`L1` i
0.15. To supress this message specify explicitly the normalization method.
  skimage_deprecation)
Feature Shape:  (8100,)
Feature Visualize:
```

Input image



Histogram of Oriented Gradients



3. Project Contents and Algorithm

HOG Feature Extraction

```
def hog_feature_extraction(rgb_image, orientations=9, pixels_per_cell=(8, 8), cells_per_block=(3, 3), verbose = 1):
    feature, hog_image = hog(rgb_image, orientations=orientations, pixels_per_cell=pixels_per_cell, cells_per_block=cells_per_block, visualize=True, multichannel=True)

    if verbose == 1:
        print("Feature Shape: ", feature.shape)

        print("Feature Visualize: ")
        fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(8, 4), sharex=True, sharey=True)

        ax1.axis('off')
        ax1.imshow(rgb_image, cmap=plt.cm.gray)
        ax1.set_title('Input image')

        # Rescale histogram for better display
        hog_image_rescaled = exposure.rescale_intensity(hog_image, in_range=(0, 10))

        ax2.axis('off')
        ax2.imshow(hog_image_rescaled, cmap=plt.cm.gray)
        ax2.set_title('Histogram of Oriented Gradients')
        plt.show()
    # if

    return feature
# hog_dense_feature_extraction
```

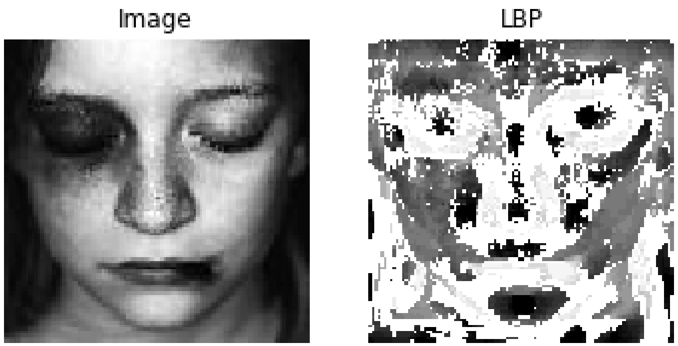
3. Project Contents and Algorithm

LBP Feature Extraction

- Image size 100x100, num_points = 24, radius = 8
  - Feature dimension 26 for entire features

```
[9]: # image, info = rafdb_read_view_image(db = df_rafdb, image_id = "train_00010", **default_read_settings)
image, info = read_aligned_image(db = df_rafdb, root_dir =  rafdb_info["root_dir"], image_id = "train_01010", verbose = 0)
image_resized = cv2.resize(image, image_size)
image_resized = cv2.cvtColor(image_resized, cv2.COLOR_BGR2GRAY)

feature1 = lbp_feature_extraction(image_resized, num_points = 24, radius = 8, entire_image = True, verbose = 1)
```

```
LBP in entire image
(100, 100) (26,)
```



3. Project Contents and Algorithm

LBP Feature Extraction

```
def lbp_feature_extraction(gray_image, num_points, radius, entire_image = True, verbose = 1):

    if entire_image == True:
        lbp = local_binary_pattern(gray_image, num_points, radius, method="uniform")

        # histogram & features
        (hist, _) = np.histogram(lbp.ravel(), bins=np.arange(0, num_points + 3), range=(0, num_points + 2))
        # normalize the histogram
        hist = hist.astype("float")
        hist /= (hist.sum() + 0.0001)

        if verbose == 1:
            print("LBP in entire image")
            print(lbp.shape, hist.shape)
            plt.subplot(1,2,1), plt.imshow(np.uint8(gray_image), cmap='gray'), plt.axis("off"), plt.title("Image")
            plt.subplot(1,2,2), plt.imshow(np.uint8(lbp), cmap='gray'), plt.axis("off"), plt.title("LBP")
            plt.show()
        # if

        return hist
    else:
```

3. Project Contents and Algorithm

LBP Feature Extraction

- Image size 100x100, num_points = 24, radius = 8, patch 10x10
  - Feature dimension 1700 for patch features

```
feature2 = lbp_feature_extraction(image_resized, num_points = 15, radius = 3, entire_image = False, verbose = 1)
```

```
LBP in patch images
(1700,)
```

3. Project Contents and Algorithm

LBP Feature Extraction

```
else:
    image_size = gray_image.shape[0:2]
    image_patch_height = int(gray_image.shape[0] / 10) # height = 10
    image_patch_width  = int(gray_image.shape[1] / 10) # width = 10

    image_patches = []
    for i in range(10):
        for j in range(10):
            # i * image_patch_height: (i + 1) * image_patch_height, j * image_patch_width: (j + 1) * image_patch_width
            # i: 0:10, 10:20, 20:30, 30:40, 40:50, 50:60, 60:70, 70:80, 80:90, 90:100
            # j: 0:10, 10:20, 20:30, 30:40, 40:50, 50:60, 60:70, 70:80, 80:90, 90:100
            image_patch = gray_image[i * image_patch_height: (i + 1) * image_patch_height, j * image_patch_width: (j + 1) * image_patch_width]
            image_patches.append(image_patch)
        # for
    # for

    patch_lbps   = []
    patch_hists  = []
    for patch_image in image_patches:
        patch_lbp = local_binary_pattern(patch_image, num_points, radius, method="uniform")

        (patch_hist, _) = np.histogram(patch_lbp.ravel(), bins=np.arange(0, num_points + 3), range=(0, num_points + 2))
        patch_hist  = patch_hist.astype("float")
        patch_hist /= (patch_hist.sum() + 0.0001)

        patch_lbps.append(patch_lbp)
        patch_hists.append(patch_hist)
    # for
    patch_total = np.hstack([patch for patch in patch_hists])
```

3. Project Contents and Algorithm

SVM Classification

Preparing Data

[3]:
```
with open("./data/rafdb_hog_aligned_features.pkl", "rb") as f:
    rafdb_hog_aligned_features = cPickle.load(f)
with open("./data/rafdb_hog_aligned_labels.pkl", "rb") as f:
    rafdb_hog_aligned_labels = cPickle.load(f)
with open("./data/rafdb_hog_aligned_type.pkl", "rb") as f:
    rafdb_hog_aligned_type = cPickle.load(f)
```

[166]:
```
train_data = []
train_label= []
for image_id in df_train["id"].values:
    train_data.append(rafdb_hog_aligned_features[image_id])
    train_label.append(rafdb_hog_aligned_labels[image_id])
# for
train_data = np.array(train_data)
train_label = np.array(train_label)
```

[167]:
```
test_data = []
test_label= []
for image_id in df_test["id"].values:
    test_data.append(rafdb_hog_aligned_features[image_id])
    test_label.append(rafdb_hog_aligned_labels[image_id])
# for
test_data = np.array(test_data)
test_label = np.array(test_label)
```

3. Project Contents and Algorithm

SVM Classification

Training

```
[81]: %%time
if os.path.exists("./data/rafdb_hog_classification.pkl") == False:
    classifier  = SVC(C = 100.0, gamma='scale', kernel='sigmoid')
    classifier.fit(train_data, train_label)

    with open("./data/rafdb_hog_classification.pkl", "wb") as f:
        cPickle.dump(classifier, f)
else:
    with open("./data/rafdb_hog_classification.pkl", "rb") as f:
        classifier = cPickle.load(f)
# if
```

```
Wall time: 23min 15s
```

```
[82]: %%time
classifier.score(train_data, train_label)
```

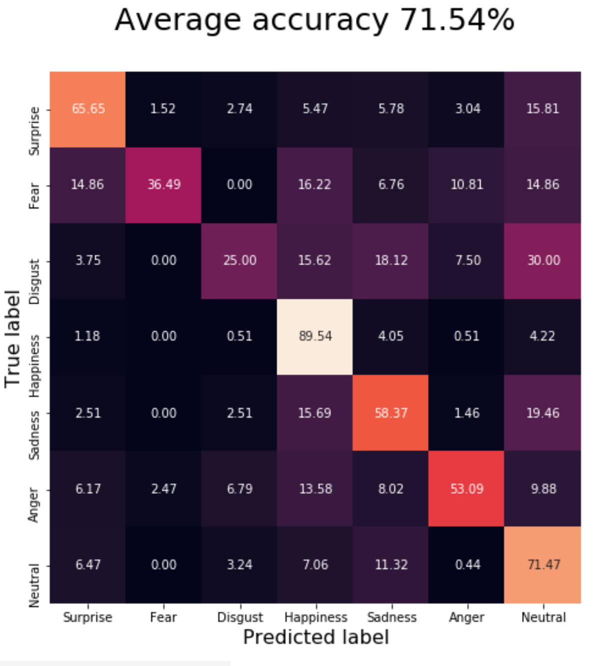
```
Wall time: 18min 8s
[82]: 0.8202265504033901
```

Testing

```
[89]: %%time
test_predict = classifier.predict(test_data)
```

```
[ ]: %%time
classifier.score(test_label, test_predict)
```

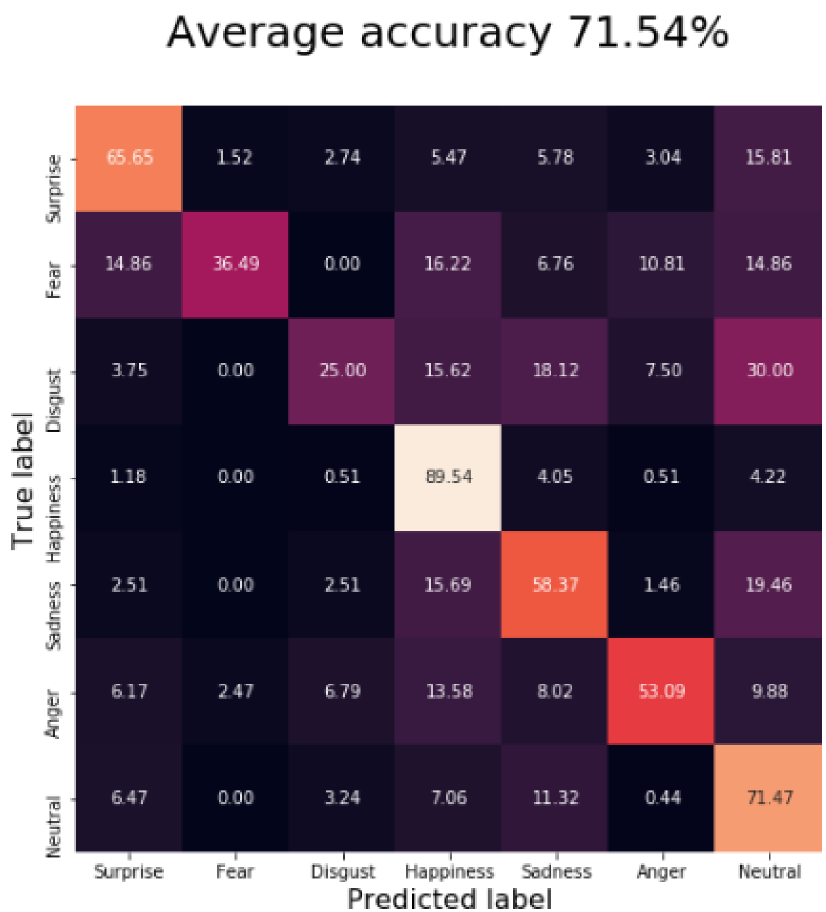
```
[189]: plot_confusion_matrix(test_label, test_predict, verbose = 0, classes = list(label_mapping.values()))
```



4. Experiment and Result

HOG Features

- Confusion Matrix



4. Experiment and Result

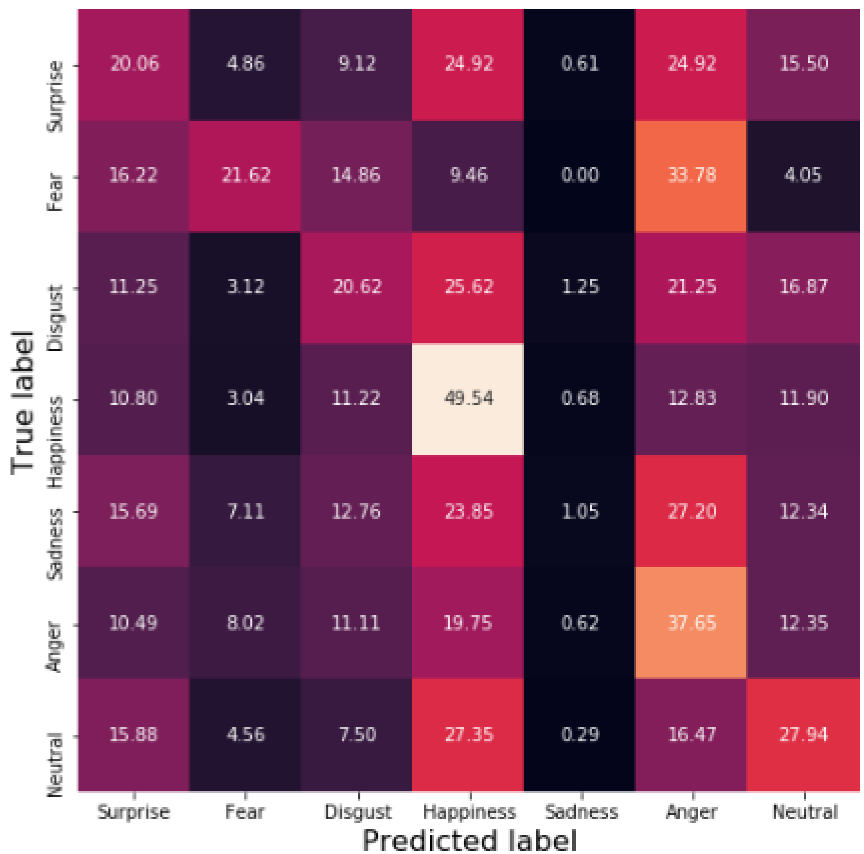
HOG Features

- Fail Cases

4. Experiment and Result
LBP Entire Features

- Confusion Matrix

Average accuracy 31.23%



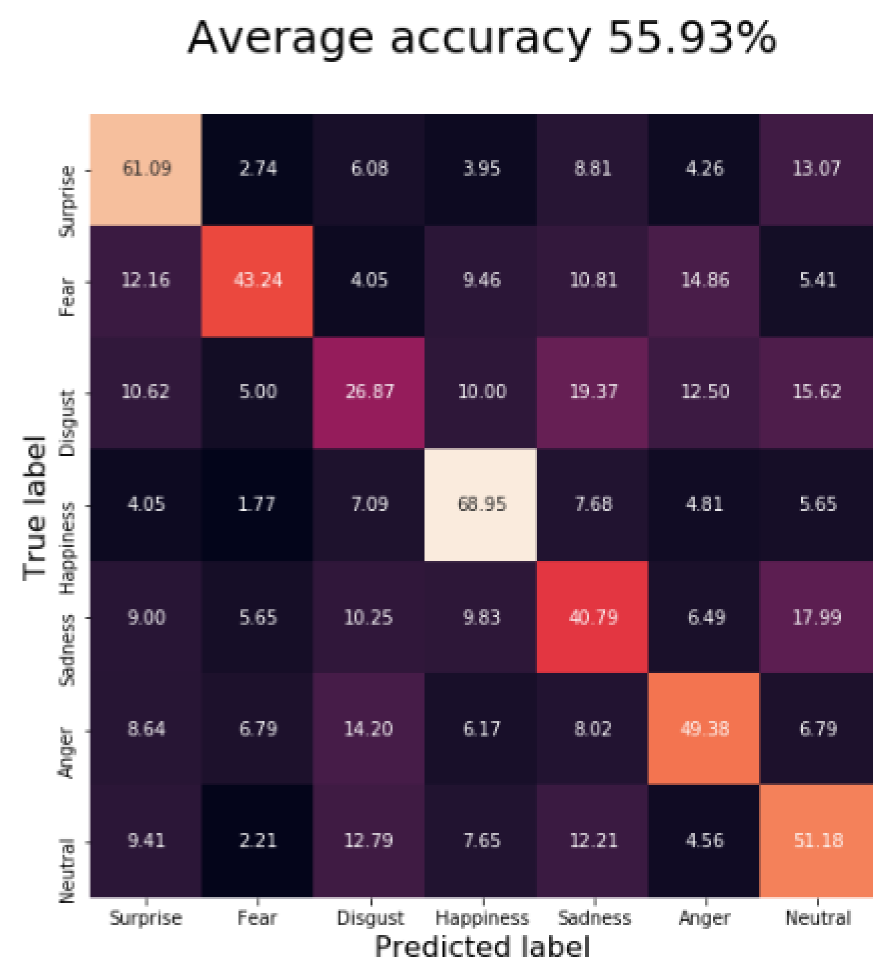
4. Experiment and Result

LBP Entire Features

- Fail Cases

4. Experiment and Result
LBP Patch Features

- Confusion Matrix



4. Experiment and Result

LBP Patch Features

- Fail Cases

5. Conclusion

- Dacial emotion recognition using traditional feature and classification.
- Dataset in-the-wild to recognize
- Good accuracy on HOG features using SVM classification

THANKS FOR LISTENING!