Experiment: Face Gender Recognition

Implementation of Deep Learning Framework Based on Convolutional Neural Networks and Pytorch

1. **Overview of experimental tasks**:
   - Download an image collection that includes the names and images of over 5000 people, categorized by folder.
   - Select a certain number of male and female images based on the correspondence between person names and gender provided in the lfw deepfunneled gender. txt document.
   - Use the selected 3500 male images and 1000 female images as the training set, and the remaining 500 male images and 200 female images as the test set.
   - In order to reduce memory consumption, each image is first captured with 200x200 pixels in the center, and then reduced to a 100x100 image for learning.
   - Based on Python, train a model that can recognize the gender of images using these images.
   - Based on Matplotlib, display the predicted image results.

2. **Experimental principle:**

In fact, based on the differentiated training set, a convolutional neural network is used to repeatedly use different and a certain number of filters on RGB three channel images to obtain multiple feature maps that match the number of filters. Then, convolution calculations are performed again to continuously extract different features of the input image, thereby increasing the network's expression ability and extracting richer and more diverse image features, Finally, the processing flow of the CNN is mapped to a fully connected layer with a certain number of neurons, which is actually the fusion and extraction of features. Then, the output results are classified and normalized, compared with the data labels, and the loss value is calculated.

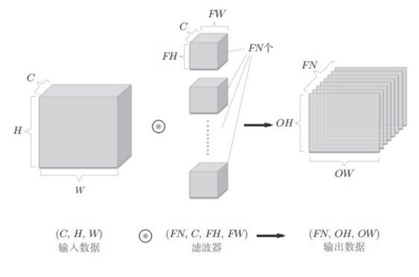


Figure 1: Example of convolution operation based on multiple filters

Among them, in order to prevent the activation value from being too biased towards a certain value, which may lead to the problem of gradient disappearance and limited expressiveness, batchnorm is used to standardize the data in a batch.

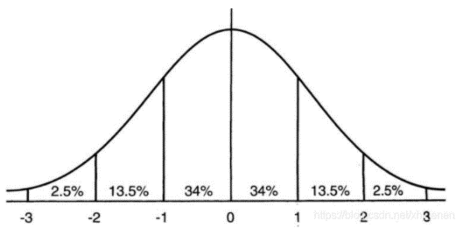


Figure 2: Average distribution of activation values after batchnorm operation

In addition, to prevent overfitting of training data by the model, dropout is used to discard neurons with a certain probability.

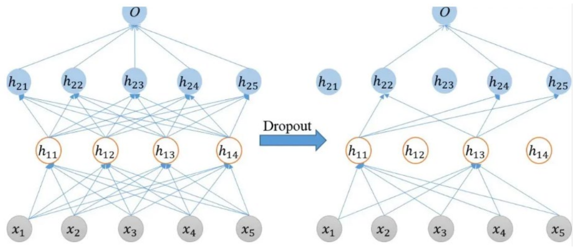


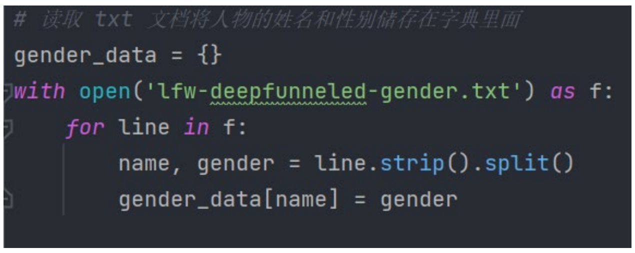
Figure 3: Dropout of neurons

3. **Overview of experimental steps:**
    a) Data collection: Collect facial image data for training and testing. These datasets can contain a large number of male and female facial images (without labels).
    b) Data preprocessing: Preprocess the collected image data to improve the accuracy of subsequent gender recognition. The preprocessing steps may include image cropping, etc.
    c) Feature extraction: Extract gender distinguishing features from each facial image. Common feature extraction methods include using image filters to extract texture features, or using convolutional neural networks (CNN) to learn advanced features from images.
    d) Feature selection: Select and reduce the dimensionality of features extracted from facial images based on different gender recognition algorithms and model requirements. This helps to reduce the dimensionality of features and improve the efficiency and accuracy of recognition algorithms.
    e) Training classifier: Train the classifier model using labeled facial image data to associate features with the corresponding gender.
    f) Performance evaluation: Use a backup labeled dataset to evaluate the performance of the trained classifier. This involves calculating losses using training sets and verifying recognition accuracy using test datasets.

4. **Specific experimental operations:**

   After downloading the complete dataset, the experiment will proceed as follows:

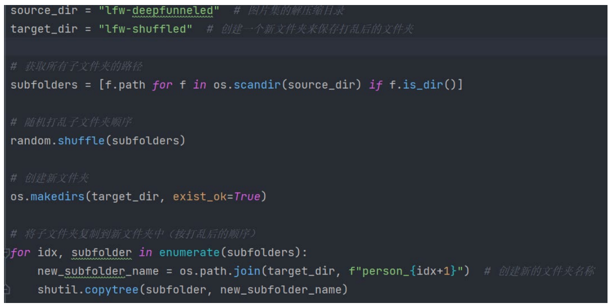
   ➢ **Preprocess the dataset (data_extract. py):**
      a) Firstly, read the data in the text document, store the names and genders of the characters in the dataset in the dictionary, and create a mapping relationship:



```python
# 读取 txt 文档将人物的姓名和性别储存在字典里面
gender_data = {}
with open('lfw-deepfunneled-gender.txt') as f:
    for line in f:
        name, gender = line.strip().split()
        gender_data[name] = gender
```

Figure 4: Constructing the relationship between person names and gender

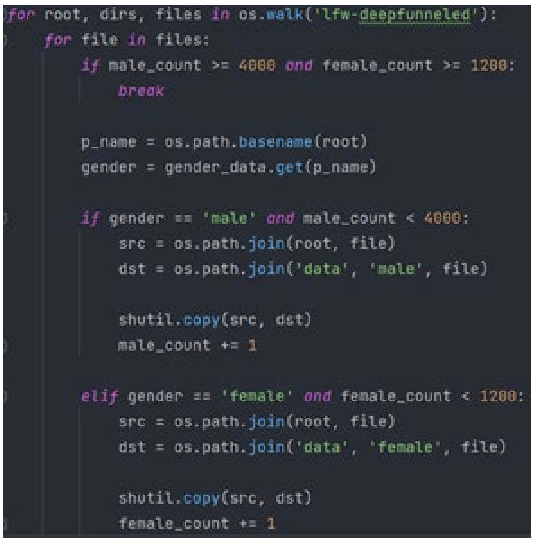
b) On the basis of the original dataset, shuffle the subfolders in the dataset files to achieve the function of randomly selecting 4000 male images and 1200 female images:



```python
source_dir = "lfw-deepfunneled"  # 图片集的解压缩目录
target_dir = "lfw-shuffled"  # 创建一个新文件夹来保存打乱后的文件夹

# 获取所有子文件夹的路径
subfolders = [f.path for f in os.scandir(source_dir) if f.is_dir()]

# 随机打乱子文件夹顺序
random.shuffle(subfolders)

# 创建新文件夹
os.makedirs(target_dir, exist_ok=True)

# 将子文件夹复制到新文件夹中（按打乱后的顺序）
for idx, subfolder in enumerate(subfolders):
    new_subfolder_name = os.path.join(target_dir, f"person_{idx+1}")  # 创建新的文件夹名称
    shutil.copytree(subfolder, new_subfolder_name)
```

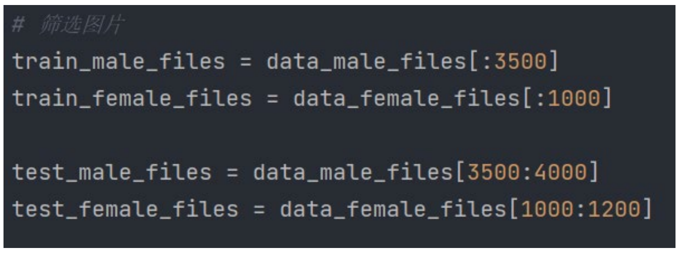
Figure 5: Disrupting sub folders in source data files

c) Classify the selected data according to gender, and then extract a certain amount of data from the dataset after gender classification as the training and testing sets:



```python
for root, dirs, files in os.walk('lfw-deepfunneled'):
    for file in files:
        if male_count >= 4000 and female_count >= 1200:
            break

        p_name = os.path.basename(root)
        gender = gender_data.get(p_name)

        if gender == 'male' and male_count < 4000:
            src = os.path.join(root, file)
            dst = os.path.join('data', 'male', file)

            shutil.copy(src, dst)
            male_count += 1

        elif gender == 'female' and female_count < 1200:
            src = os.path.join(root, file)
            dst = os.path.join('data', 'female', file)

            shutil.copy(src, dst)
            female_count += 1
```

Figure 6: Classifying the dataset by gender

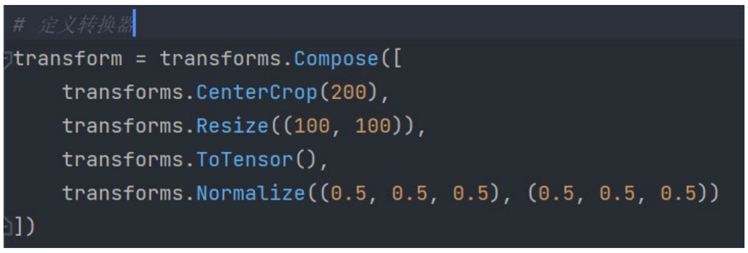


```python
# 筛选图片
train_male_files = data_male_files[:3500]
train_female_files = data_female_files[:1000]

test_male_files = data_male_files[3500:4000]
test_female_files = data_female_files[1000:1200]
```

Figure 7: Partitioning the training set and dataset

➢ **Image capture (Model. py):**

Take a 200x200 pixel image from the center and reduce it to a 100x100 image for learning.



```python
# 定义转换器
transform = transforms.Compose([
    transforms.CenterCrop(200),
    transforms.Resize((100, 100)),
    transforms.ToTensor(),
    transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))
])
```

Figure 8: Capture images as required

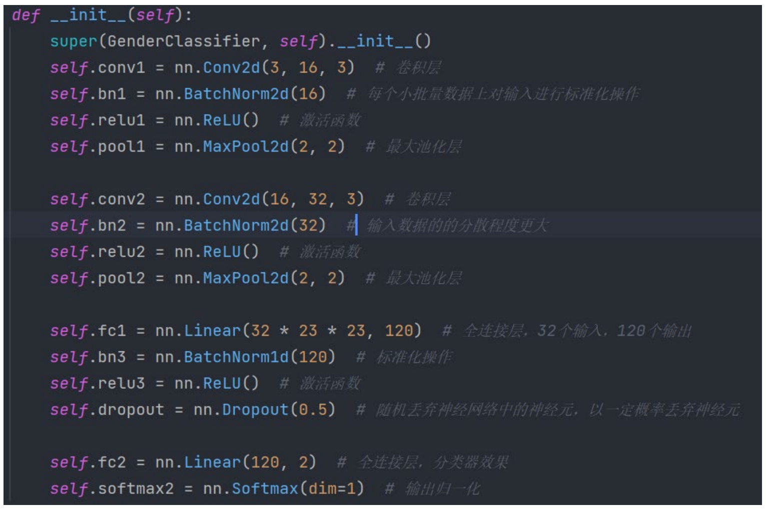
➢ **Construction of Convolutional Neural Networks (Model. py):**

a) The first convolutional layer: The input channel is 3, and the output is 16 feature maps. An appropriate number of channels can provide sufficient model expression ability to learn the features of the input data. However, selecting a larger number of channels may lead to overfitting, making the model more complex, and increasing the corresponding computational burden. At the same time, batchnorm was introduced to standardize the data.

b) The second layer of convolutional layer: Based on the previous layer of CNN data flow, the number of output channels in the convolutional layer is increased from 16 to 32 in order to increase the complexity and expressive power of the model.

c) The third fully connected layer: fusion and extraction of features. At the same time, dropout is introduced to discard neurons according to probability.

d) Output layer: classifier



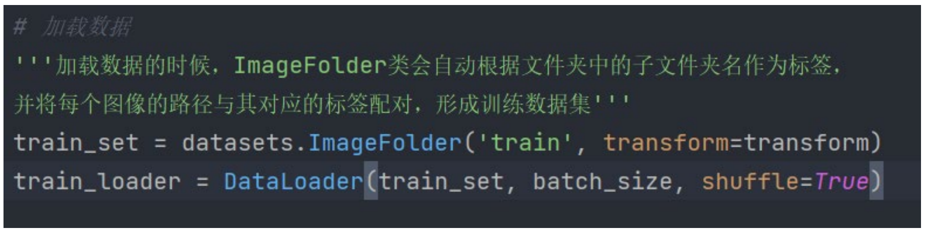
```python
def __init__(self):
    super(GenderClassifier, self).__init__()
    self.conv1 = nn.Conv2d(3, 16, 3)  # 卷积层
    self.bn1 = nn.BatchNorm2d(16)  # 每个小批量数据上对输入进行标准化操作
    self.relu1 = nn.ReLU()  # 激活函数
    self.pool1 = nn.MaxPool2d(2, 2)  # 最大池化层

    self.conv2 = nn.Conv2d(16, 32, 3)  # 卷积层
    self.bn2 = nn.BatchNorm2d(32)  # 输入数据的的分散程度更大
    self.relu2 = nn.ReLU()  # 激活函数
    self.pool2 = nn.MaxPool2d(2, 2)  # 最大池化层

    self.fc1 = nn.Linear(32 * 23 * 23, 120)  # 全连接层, 32个输入, 120个输出
    self.bn3 = nn.BatchNorm1d(120)  # 标准化操作
    self.relu3 = nn.ReLU()  # 激活函数
    self.dropout = nn.Dropout(0.5)  # 随机丢弃神经网络中的神经元, 以一定概率丢弃神经元

    self.fc2 = nn.Linear(120, 2)  # 全连接层, 分类器效果
    self.softmax2 = nn.Softmax(dim=1)  # 输出归一化
```

Figure 9: Model Building

➢ **Training and Testing (Model. py):**

a) Data loading:

Use the ImageFolder class to create an image dataset object, which is a class provided by PyTorch for processing image datasets with folder structures. This class will automatically use the names of sub folders in the folder as labels and pair the path of each image with its corresponding label. In other words, when preprocessing the data, the dataset has been divided into two sub datasets (folders) based on gender. When using this class to process images in a folder, it will automatically add labels (female: 0, female: 1) to the image data in the subfolders in the order of the subfolders.



```
# 加载数据
'''加载数据的时候，ImageFolder类会自动根据文件夹中的子文件夹名作为标签，
并将每个图像的路径与其对应的标签配对，形成训练数据集'''
train_set = datasets.ImageFolder('train', transform=transform)
train_loader = DataLoader(train_set, batch_size, shuffle=True)
```

Figure 10: Data Loading

b) Settings for loss function and optimizer:
Use cross entropy function to calculate losses, and use Adam parameter update algorithm to optimize and deeply learn parameters.

c) Training model:
Perform multiple iterations on the training set data, propagate the batch data forward, compare the expected results (index of larger data in the two-dimensional matrix) with the labels, calculate the loss value, and then propagate back to update the parameters.

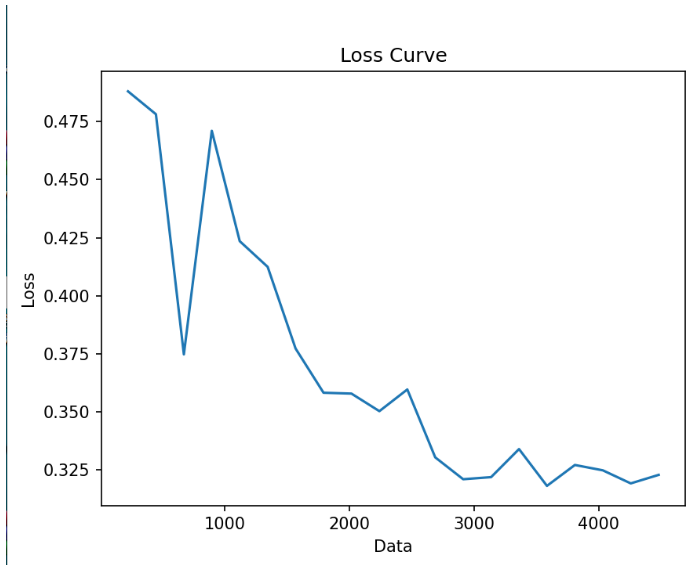


Figure 11: Loss value image of ten iterations of training

d) Test model:
Calculate recognition accuracy and evaluate the predictive ability of the model.

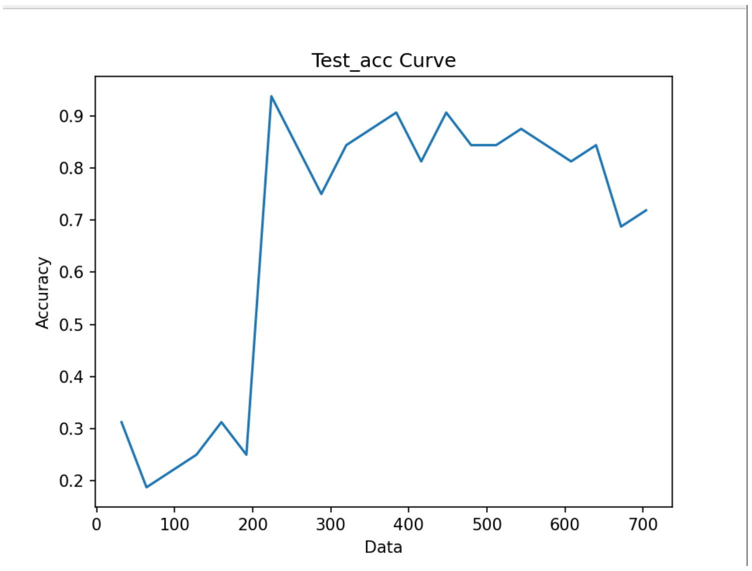


Figure 12: Test Set Test Results

5. Reference:

a) Figure 2, Batch Normalization (Deep Learning) Batch Standardization, https://blog.csdn.net/xhj_enen/article/details/89113034
b) Figure 1, Introduction to Deep Learning, Theory and Practice Based on Python, p212
c) Introduction to Pytorch CNN specific facial recognition in practice, https://blog.csdn.net/ziro_/article/details/124070187
d) Based on deep learning and facial gender recognition, https://blog.csdn.net/pythonyanyan/article/details/128697224
e) Facial Attribute Recognition - Pytorch lighting Multi label Practice, https://zhuanlan.zhihu.com/p/599885653
f) Figure 3, Recurrent Neural Network, https://m.zhangyue.com/readbook/11699255/8.html?p2=116764