Project 2 of Appied Data Science

Zihan Tang

November 2024

Abstract

This project focuses on classifying chemical perturbations among 2,867 bio-images and extract features. Two classification approaches were explored: (1) a Simple CNN applied directly to the original, labeled images and (2) a Simple CNN on the segmented images. As anticipated, the segmented-image model outperformed the direct classification approach, effectively distinguishing chemical perturbations. We evaluate on the loss-function and accuracy of train and test dataset and interpret corresponding meaning. This pipeline is useful in cell morphology analysis, which could detect meaningful patterns and features.

1 Introduction

1.1 Background

The study "Three Million Images and Morphological Profiles of Cells Treated with Matched Chemical and Genetic Perturbations" used machine learning and image processing to analyze microscopy images, identifying patterns and relationships between genetic modifications and drug treatments. This work sets a benchmark for measuring similarities and effects of cellular perturbations, showcasing the potential of image-based profiling for applications like drug mechanism discovery and functional genomics.

1.2 Goal

Building upon this foundational work, our project aims to replicate machine learning methodologies for image classification and analysis from scratch in Python.We explore two approaches: a Simple CNN applied to raw images and a refined pipeline using image segmentation followed by classification to enhance accuracy in detecting chemical perturbations.

1.3 Data

We used a subset of the original three-million-image dataset, focusing on 2,867 median-aggregated grays images in one batch, a robust dataset suited for downstream segmentation tasks. For testing, we created a smaller dataset of five images with specified chemical perturbation or non-perturbation labels. Label information is stored in the metadata file. The

training and testing images are stored in separate files without being categorized by label (perturbation status).

2 Methods

2.1 Dataset Preparation

- **Mapping image and label:** We wrote a CustomImageDataset function to assign labels to each image, storing mapping information in a DataFrame that contains the image names, file paths, and labels.
- **Splitting dataset:** We used stratified sampling to reserve 20
- **Transformation:** We resized each image to 256x256 pixels and normalized the pixel values to a mean of 0.5 and a standard deviation of 0.5, then converted them to tensor format.
- **Creating batches:** We set a batch size of 32 for the training process.

2.2 Data Segmentation

We used Cellpose to segment cells in the images and created masks. Due to the large size of the segmented dataset, we did not input it into the CNN model. Additionally, we utilized the OpenCV library to identify cell contours and segment images as a comparison. The segmented regions were assigned the same label as the original image.

2.3 Build a Convolution Neural Network Architecture

We built a two-layer CNN for classification and feature extraction, with two pipelines: one (SimpleCNN) that directly classifies the original images and another (SegCNN) that segments images before classification. Both use the same CNN architecture.

- **Convolutional Layer 1:** $nn.Conv2d(1, 64, kernel_size = 3, padding = 1)$, followed by ReLU activation and 2*2 Max-pooling.
- **Convolutional Layer 2:** $nn.Conv2d(64, 128, kernel\_size = 3, padding = 1)$, followed by ReLU activation and 2*2 Max-pooling.
- **Fully Connected Layer 1:** $nn.Linear(128*64*64, 512)$, followed by ReLU activation.
- **Fully Connected Layer 2:** $nn.Linear(512, 1)$, producing a single binary output, followed by Sigmond Activation.
- **Return:** Features and Classification results.

2.4 Train and Test the Network

2.4.1 Train process

- **Batch Size:** 32
- **Loss Function:**Binary Cross-Entropy Loss (BCELoss) was chosen to handle the binary classification of perturbation presence.
- **Optimizer:** The Adam optimizer was employed to train the network and adjust the weights efficiently.
- **Epoch Size:** 20

2.4.2 Test process

Model performance was primarily assessed based on classification accuracy.

2.5 SimpleCNN

Train results for SimpleCNN are presented in Table 1 and **test results** in Table 2

Epoch	Train Loss	Train Accuracy (%)
1	17.5276	82.25
2	17.7559	82.25
3	17.7786	82.25
4	17.7786	82.25
5	17.7331	82.25
6	17.8241	82.25
7	17.7559	82.25
8	17.7786	82.25
9	17.7786	82.25
10	17.7786	82.25
11	17.6877	82.25
12	17.7104	82.25
13	17.8241	82.25
14	17.7331	82.25
15	17.7104	82.25
16	17.7559	82.25
17	17.7559	82.25
18	17.7331	82.25
19	17.8241	82.25
20	17.7559	82.25

Table 1: Training Loss and Accuracy over 20 Epochs

Metrics: During the first few epochs, the model achieved a relatively high accuracy of 82.25

Features Extraction: We extract 21 features by using feature map.

Image	Predicted Label	Ground Truth	Correct/Incorrect
1	1	1.0	Correct
2	1	0.0	Incorrect
3	1	1.0	Correct
4	1	1.0	Correct
5	1	1.0	Correct

Table 2: Predicted Labels vs. Ground Truth for Test Images

The model achieved a test accuracy of 0.8 on a sample of five images, indicating that the model has learned to generalize, although there is still room for improvement. The model showed a tendency to predict the label "1" but performed well on images with ground truth label "1." The misclassification for Image 2 suggests that the model may not effectively distinguish "0," highlighting an area for further tuning or additional data.

2.6 SegCNN

After segmentation, we obtained 3,729 images. However, due to computational limitations, we could not fully test the model on this segmented dataset.

3 Conclusion

This project examined two CNN approaches for classifying chemical perturbations in bio-images: SimpleCNN, which classified raw images directly, and SegCNN, which applied segmentation prior to classification. SimpleCNN demonstrated initial success in identifying perturbations, though with fluctuating loss values, while SegCNN showed potential for improved accuracy in tasks requiring detailed morphological analysis.

Future work could focus on refining CNN parameters and reapplying the model on segmented images for more accurate feature extraction, enhancing classification accuracy. These improvements would support a more robust pipeline for bio-image analysis, benefiting applications like drug discovery and functional genomics.

4 Code Availability

The code used for this analysis is available on a public GitHub repository: `https://github.com/Zihan-Hazel/Cellimage_CNN_ADS`