OBJECT CLASSIFICATION IN IMAGES VIA DEEP LEARNING WITH OFFERUP

ZHENG HONG TAN, JUNNAN KOU, ZIQIAO XU, SANDEEP J RAMANATHAN
INDUSTRY MENTOR: ALEXANDRA TESTE
FACULTY MENTOR: PROFESSOR RANIA HUSSEIN

INTRODUCTION

OfferUp is a mobile based online shopping service provider with 44 million users that allows you to sell everything from clothing to cars. They are a C2C marketplace with emphasis on in-person transactions. It ranks top 10 of Pacific Northwest private companies (Geekwire 2016).

Project background

Sellers need to load images of the item and write a verbal description when posting. Buyers can find their desired items through search better if an image search function to find similar images based on the images.

Our team can use deep learning and image classification technology to classify images based on type (clothes vs shoes) and category (gender and shoe types).

Deep Learning

Deep Learning is a type of machine learning that trains a computer to perform human-like tasks, such as recognizing speech, classifying images or making predictions.

Major steps to build a model

1. Preparing the dataset
2. Training the Model
3. Deploying the model

Project overview

PREPARING THE DATASET

Generally, preparing the dataset is the most important step in the process of building a model. It is the base on which everything else is built. The clearer your dataset is, the better your model can perform.

1. Preliminary Steps

- Data collection and labeling
- Data preprocessing
- Train the model
- Test and update parameters
- Deploy the model

2. Project overview

Figure 1: The entire process of image classification

3. Data/Image preprocessing

Google AutoML

- Create a GCP account and a bucket on GCS.
- Load the original images for AutoML training on their respective category folders.

Transfer Learning

- Resize images to 224x224 pixels. Perform normalization. Divide the image pixel value by 255 to get a result in the range from 0 to 1.

TRAINING THE MODEL

When training the model, one of the metrics that we could use to evaluate how well the model performs on unseen data is the validation accuracy. The better the validation accuracy, the better the model can predict a correct category for which a new item belongs to.

1. Google AutoML

- We leveraged Google’s AutoML Neural Architecture Search (NAS) to train a model that distinguishes clothes from shoes.

2. Transfer Learning

- We decided to classify different shoe types to assist the gender model to deploy it on an iOS and Android app.

DEPLOYING THE MODEL

Once the model is ready for use, there are many ways to deploy it into production. We used a few methods to test out our model.

1. GCP Deployment: We first deployed the model onto GCP servers and used the User Interface to upload and test the accuracy of a few images.

2. Docker: We then deployed the model into a Docker container and made API calls to the model.

3. Streamlit: We then used Streamlit to make an interactive web application where users can upload multiple images and get the predictions.

4. iOS and Android apps: Finally we used the TensorFlow Lite (quantized) version of the model to deploy it onto an iOS and android app.

Figure 2: RegEx Python code for shoes and clothes classification

Figure 3: CSV file with location to images with their respective labels

Future Steps

- Keep improving the performance of the image classification models
- Implement an image search function to find similar images based on the models we trained.
- Use the shoes classifier model to label items and build text classifiers.
- Build the image classification completely from scratch instead of using transfer learning.

References


Figure 4: Images before and after resizing to 224x224 pixels.

Figure 5: GCP image upload

Figure 6: Images before and after resizing to 224x224 pixels.

Figure 7: Confusion Matrix and Precision vs Recall Curve of the Clothes/Shoes Model

Figure 8: Plot of the Validation Accuracy and Loss of the MobileNetV2 Model

Figure 9: Plots of the Validation Accuracy and Loss of the MobileNetV2 Model

Figure 10: The UI and prediction result on GCP

Figure 11: JSON Response

Figure 12: Prediction

Figure 13: MobileNetV2 accuracy on Kefas.

Figure 14: iOS App

Figure 15: Android App