184.702 Machine Learning 2021W

Exercise 3: Image Classification
Deep Learning vs. Traditional Approaches

Jakob Aichinger
11814579
Andreas Buchner
11811860
Samo Kolter
11810909

February 21, 2022
Group: 09

Contents

1 Introduction

In this work we try to extract information from images by using different feature extraction methods. With this data, we then train prediction models to classify new images. The main idea is to explore how representation and extraction from more complex media content, in this case, images, differ from extraction methods for traditional text and numerical-based data. Furthermore, we want to find out how various approaches, including simple methods (e.g. color histograms) and more complex deep learning approaches (e.g. convolutional neural networks) differ.

For this, we used two different datasets, namely:

1. CIFAR-10 [1]
2. The German Traffic Sign Recognition Benchmark (GTSRB) [2]

The report is divided into five sections. First, we will give an overview of how we structured our work and give insights on the experimental setup used for both datasets. Also, we will outline sources and example code used for both traditional and deep learning methods. Next, the two datasets main characteristics will shortly be summarized, followed by a detailed overview of results for the corresponding dataset. Lastly, a conclusion of the main findings of this work is given.

2 General Approach

2.1 Project Structure

In order to make our code less redundant and more structured, the steps performed in this work were split up into multiple Jupyter Notebooks, which can all be found in the folder './notebooks'. We also tried to follow a name convention, which firstly suggests the task's name and secondly the corresponding dataset.

Moreover, we implemented various utility functions/classes in helper script files, which can be found in './notebooks/helper_scripts'. These helper scripts include a generic 'data_loaders.py' script, allowing us to import data from either CIFAR-10 or GTSRB with custom parameters such as grayscale conversion, standardization, and specific image size.

Lastly, for reproducibility, we also included fitted deep learning models in the folder './model_fit_results'. The AlexNet model files were quite large (more than 200 MB) and therefore could not be included in the submission. However, they are available on Google Drive (for the next few weeks at least).

2.2 Experimental Setup

The aim of this exercise is to compare traditional feature extraction methods (namely Color Histogram and SIFT Bag of Visual Words) with Deep Learning approaches (Convolutional Neural Networks such as AlexNet and LeNet).

To extract the Color Histogram, we basically followed the provided tutorial in TUWEL. To extract the Histogram resulting from the BOVW of SIFT we followed several tutorials, thinking that we did something wrong because of the small number of extracted keypoints but later finding out that these are indeed all the keypoints in the pictures.

In order to get a baseline of performance for our subsequently to be tested Deep Learning approaches, we implemented a function that evaluated a given set of features by applying different

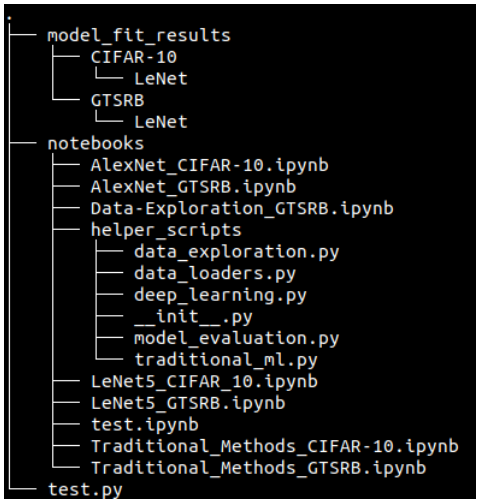

Figure 1: Project Structure

traditional classifiers, such as KNN, MLP, Decision Tree, Random Forest and Naive Bayes, on them.

For Deep Learning, we used two different Architectures: AlexNet and LeNet. We experimented with different numbers of Epochs, Batch Sizes and also tested Data Augmentation.

2.3 Performance Measures

The main Performance Measure was accuracy. Due to a relative equally distributed number of observations per class in the test set this is not a problem. Another Performance Measure was of course time. This includes Training/Fit Time as well as Prediction Time. Finally, we also compared the Confusion Matrix to get a visual result of which classes were often mistaken for another.

3 Dataset 1 - CIFAR10 [1]

The CIFAR10 Dataset is a labeled subset of the 80 million tiny images dataset. It contains images of airplanes, automobiles, birds, cats, deer, dogs, frogs, horse, ships and trucks. These classes are, of course, mutually exclusive.

We split our Data into three chunks: training (40.000 images), validation (10.000 images) and test (10.000 images).

3.1 Characteristics

- tiny images from different domains (animals, automobile, ...)
- 10 different classes
- 60,000 images in total
- 6,000 images per class
- original image size 32x32 pixels

3.2 Traditional Methods

3.2.1 Color Histogram

Extracting a histogram of colors from the images and using this as a feature resulted in an Accuracy of 32,96%. Which is not bad, considering we have 10 available classes and using only information about the distribution of the colors does not seem to be a reliable feature when wanting to e.g. distinguish between a plane and a ship.

Best classifier was a Random Forest. The prediction of 10.000 images (test data size) took 1,21 seconds. Fitting the optimal model took around 47 seconds.

3.2.2 SIFT BOVW

Extracting the histogram containing the Bag of Visual Words depending on the result of SIFT of an image proved to be more difficult. How exactly it was done can be read in the notebook attached to this submission.

Due to the small image size (32x32) SIFT did not find any keypoints at all in the provided images - so we upscaled our images to 50x50. Using these features showed that they tend to overfit largely. We were able to obtain a Cross Validation Score on the training set of 16%, when using it to evaluate on the test data we obtained an accuracy of only 10,03% - only roughly better than randomly guessing.

The best classifier here was a DecisionTree, predicting took 0,01 seconds and fitting took 0,33 seconds. So to conclude, at least it was very fast.

3.3 Deep Learning approaches

3.3.1 LeNet

LeNet takes as input a 32x32x1 Matrix which is perfect for this dataset as we do not need to rescale the data. The only thing we had to do for preprocessing was scaling and converting from RGB to grayscale.

First, we fitted our training data without using data augmentation. We used a batch size of 32 and trained for 300 epochs. Fitting took 35 minutes, predicting only one second, and the LeNet was able to achieve an accuracy of 51% on the test set. Quite an achievement compared to the previously mentioned traditional feature extraction methods.

When using Data Augmentation, our fitting time increased to roughly three hours and ten minutes, so quite a lot. The results however decreased, and we obtained an accuracy of only 45,59% on the test set. We used the same number of epochs as when fitting without data augmentation.

When visually comparing the confusion matrix, we can not see major differences in the distribution, but simply a better result when using the model which was trained without using data augmentation.

3.3.2 AlexNet

AlexNet is a more advanced architecture than the LeNet. One major difficulty using the AlexNet properly on this dataset was the required size of the images (277x277). Manually rescaling all the images proved to be impossible, as we quickly hit our RAM limits. We tried using a smaller, modified version of the AlexNet, but we were not quite happy with the results. So we kept trying and found a way to make the original AlexNet work using TensorFlow Datasets.

The whole process of fitting the model (running locally instead of in Colab this time) took roughly two hours. We only trained 50 epochs since our model showed to be overfitting to the training data when training for more epochs. We achieved an Accuracy of 77% and predicting the Test set took only twelve seconds.

4 Dataset 2 - GTSRB [2]

The German Traffic Sign Recognition Benchmark dataset (GTSRB) contains images of different traffic signs with different dimensions. In most cases, these traffic signs include a border, and the actual sign is not centered within the image. However, the dataset also includes annotations where the x and y coordinates of the bounding box is given.

4.1 Characteristics

- variety of unique German traffic signs
- 43 different classes
- over 50,000 images in total
- original image size between 15x15 to 250x250 pixels

4.2 Traditional Methods

4.2.1 Color Histograms

After extracting RGB color histograms of the traffic sign image, we applied our 'evaluate_feature' implementation to find the best classifier. With an accuracy of <20% on the test data, the results were quite bad. The best classifier for the test data was Random Forest with 100 estimators and an accuracy of 0.16 on the test data. The fit time was 32.39 seconds. One reason for the unsatisfying prediction results might be that traffic signs often share a common color distribution, e.g. speed signs, making it very hard to differ between 20, 50 or 100 km/h. Another problem is that we are working with many different classes.

4.2.2 SIFT BOVW

For extracting keypoints with SIFT, we first changed all our test images to grayscale and tested different dimensions for the pixel resolution. We also changed the standardized range of a pixel to 8 bit (0-255). One problem was that the original image size was relatively small, and OpenCV's implementation of SIFT could not always find keypoints. After some research, we concluded it might be a good idea to upscale our images (we tested 50x50 and 100x100) and also try to use dense sift by manually identifying the keypoints. Unfortunately, both attempts have not proved to be particularly successful. The extracted BOVW histograms were tested on different classifiers, including MLP. The best score was obtained with KNeighbors Classifier, with an accuracy of only 4% for the test data. The reason for this might be that SIFT seems to struggle to identify significant keypoints for traffic signs or, in some cases, not finding keypoints at all. Although upscaling helped for reducing the number of images where no keypoints could be obtained, the relatively small size of the original might still be a problem.

4.3 Deep Learning approaches

4.3.1 LeNet

The smaller architecture we tested, the LeNet, outperformed traditional approaches by a lot. Without Data Augmentation we achieved an accuracy of 76,64%, fitting took only 15 minutes and predicting was fast too, taking only one second. So when fit time is crucial, this model would in overall be our way to go.

Interestingly, using Data Augmentation worsened our accuracy by a lot - reduced to only 11%, so there might have been some issue there which would need further investigation.

4.3.2 AlexNet

AlexNet yielded superior results on this dataset compared to all other previously tested methods. Due to the large scale of the images and the architecture of the AlexNet the fit time was rather slow (2 hours 40 minutes) but the results (accuracy of 95,69%) are worth the additional fit time.

Data Augmentation unfortunately did not improve the AlexNet results. In the contrary, it made results worse. The exact results can be found in the table in the next chapter. Furthermore, fit and prediction times were decreased. Both things could be caused by programming errors on our side. Unfortunately, we were not able to identify them.

4.3.3 Additional remarks and experiments

We will investigate further why data augmentation did not improve results for any Deep Learning classifier. Probably we made some mistakes that we were not aware of. We also briefly experimented with transfer learning and an adaption of AlexNet that works with smaller images in the input layer, however the results were not noteworthy and are therefore not included in this submission.

5 Conclusion

Model/Feature	Accuracy	Fit Time	Prediction Time
Color Histogram	**32,96%**	**47 sec.**	**1,21 sec.**
SIFT BOVW	**10,03%**	**0,33 sec.**	**0,01 sec.**
LeNet	**51%**	**35 min.**	**1 sec.**
LeNet with Data Augmentation	**45,59%**	**3 h 10 min.**	**1 sec.**
AlexNet	**75%**	**47 min.**	**12 sec.**
AlexNet with Data Augmentation	**47%**	**1 h 15 min.**	**12 sec.**

Table 1: Performance on the CIFAR-10 dataset

The main conclusion of this assignment is that deep learning approaches can be quite convenient to use and yield superior results compared to traditional feature extraction methods. The only major downside of the deep learning approach is the fitting time, which is quite long. Regarding traditional feature extraction methods we found out that they can under certain circumstances work very well, but struggle to successfully identify key features for our datasets. This in particular applies to color histograms for e.g. traffic signs, where different classes such as speed limits (10, 20, or 100 km/h) have a very similar color distribution and the resulting

Model/Feature	Accuracy	Fit Time	Prediction Time
Color Histogram	**16,07%**	**32,39 sec.**	**1.12 sec.**
SIFT BOVW	**4,13%**	**0,01 sec. (lazy)**	**4,65 sec.**
LeNet	**76,64%**	**15 min.**	**1 sec.**
LeNet with Data Augmentation	**11%**	**58 min.**	**1 sec.**
AlexNet	**97%**	**40 min.**	**22 sec.**
AlexNet with Data Augmentation	**60%**	**2 hours 25 min.**	**41 sec.**

Table 2: Performance on the GTSRB dataset

histograms lack in uniqueness. We also noticed that for small images, the traditional methods seem to give very bad results in comparison to the more complex deep learning approaches.

All our main results for both datasets can be obtained from the Tables 1 and 2 - for details on the model parameters please have a look at the notebooks for the respective classifier. Looking at the table, it becomes very apparent that the Deep Learning classifiers performed very well. The fit time, however, is considerably higher than with traditional methods. It also shows that the more sophisticated architecture of AlexNet (it has a about 61 mio. parameters compared to the only about 60,000 parameters of LeNet 5) indeed makes a significant difference in performance. However, both fit and prediction time are also several orders of magnitude higher.

References

[1] Krizhevsky, A. Learning multiple layers of features from tiny images. Tech. rep., 2009.

[2] Stallkamp, J., Schlipsing, M., Salmen, J., and Igel, C. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural Networks*, 0 (2012), –.