ENHANCING MODEL PERFORMANCE: HANDWRITTEN ROMAN NUMERALS ANALYSIS

Ofri Hefetz 209028067, Shai Shani 206165318

Repository Link: https://github.com/OfriHefetz/Handwritten-Roman-Numerals

ABSTRACT

This project focuses on enhancing the performance of a pre-existing machine-learning model by manipulating a dataset of handwritten Roman numerals. The key objectives involve data cleaning, dataset enrichment through augmentation, and addressing class imbalance. First, through exploratory data analysis, labeling inconsistencies and unrelated noise were identified, leading to manual cleaning of the dataset. Next, additional images were collected from various sources, including friends, family, and AI tools, followed by data augmentation techniques. Finally, the class imbalance was addressed by balancing the dataset, and a split was done between training and validation sets. The results indicated that Augmenting the clean dataset and addressing class imbalance significantly improved performance on the given pre-existing machine-learning model. This project provides insights into improving model performance through data manipulation strategies.

Keywords: machine learning, data cleaning, dataset enrichment, data augmentation, class imbalance, handwritten Roman numerals, model performance.

1. INTRODUCTION

In recent years, data science has increasingly emphasized developing machine learning models that can achieve the best performance on given datasets. However, this project takes a unique approach by challenging us to improve the performance of a predetermined machine-learning model solely through actions on the dataset itself. In other words, we are tasked with enhancing the model's accuracy by manipulating the data, such as correcting incorrect labels, adding new examples for training, and appropriately dividing the dataset for training and validation.

The dataset provided for this project consists of images of handwritten Roman numerals, with each image labeled according to the numeral it represents. The dataset is divided into two folders: "train," containing all the training examples, and "val" containing a single example from each class for model performance validation. One necessary task is determining the division of examples between these folders to ensure effective model training and evaluation.

To accomplish our objectives, we are provided with the file "run_train_val.py" which implements the training and validation stages of the ResNet50 model, a convolutional neural network specifically designed for image recognition tasks.

This project's primary measure of success is the model's accuracy, as calculated by the code in the "run_train_val.py" file. This accuracy score will

serve as the main evaluation metric to assess our improvements to the model's performance.

The structure of this report will be designed to detail the nature and manner of our work, explaining the rationale behind each action performed and how it was implemented.

In the following sections, we will present our exploratory data analysis, which includes examining the initial dataset's characteristics and identifying incorrectly labeled data and noises. Then, we will describe our data cleaning process, discussing the methods employed to remove inaccurately labeled data and unrelated noises from the dataset. Furthermore, we will explain the steps to enrich the dataset by including augmentation and additional handwritten Roman numerals from friends and family members. Finally, we will also outline the process of collecting images from the internet, providing us with further examples of diversity in the dataset.

Subsequently, we will present the evaluation and performance analysis of our models. We will report the original data accuracy and the results achieved after cleaning the data and incorporating augmentation techniques. Finally, these results will be compared to demonstrate the performance improvements achieved through our actions on the dataset.

2. EXPLORATORY DATA ANALYSIS

In this section, we present our exploratory data analysis (EDA) findings on the provided handwritten Roman numerals dataset. The analysis aimed to gain insights into the dataset's composition, identify any labeling issues, and guide our subsequent data improvement strategies.

2.1 Dataset Overview

The dataset consists of images of handwritten Roman numerals containing ten classes: ['i', 'ii', 'iii', 'iv', 'ix', 'v', 'vi', 'vii', 'viii', 'x']. The original dataset was split into training and validation sets, with 2067 and 10 samples, respectively.

2.2 Labeling Distribution

We examined the distribution of labels within the training data and validation data separately.

The most common label in the training data (Figure 1) is 'iv', followed by 'i' and 'ix'. The remaining labels show relatively similar frequencies, with 'ii' being the least represented.

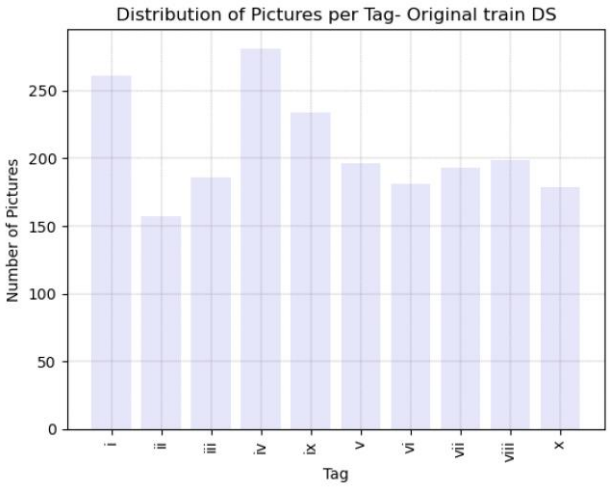

Figure 1: Distribution of Pictures per Tag – Original Dataset (Train)

In the validation data (Figure 2), each class is represented by one image, ensuring an equal distribution for validation purposes.

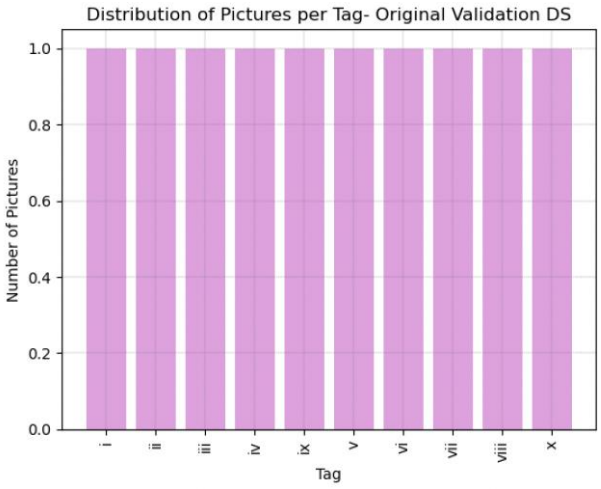

Figure 2: Distribution of Pictures per Tag – Original Dataset (Validation)

To provide a visual representation of each digit from every class, we included Figure 3. This display showcases examples of handwritten Roman numerals, offering a glimpse into the dataset's variety.

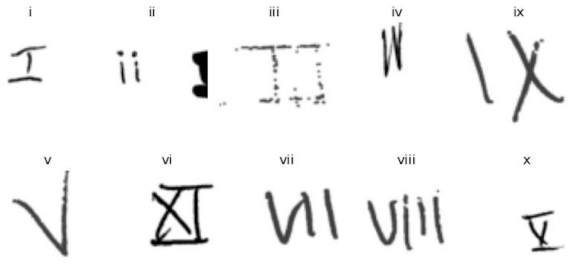

Figure 3: Representation of each Digit

During our exploratory data analysis, we encountered an instance that revealed a labeling inconsistency. Figure 4 showcases a collection of 20 randomly selected images from class 'IX'. Among these images, we discovered one specific instance that had been inaccurately labeled. This mislabeled image, essentially a drawing rather than a recognizable numeral, presents an example of the dataset's labeling inconsistency and need for more clarity. In addition, this finding raised concerns regarding the overall accuracy and reliability of the labeling process for this particular class and the overall dataset.

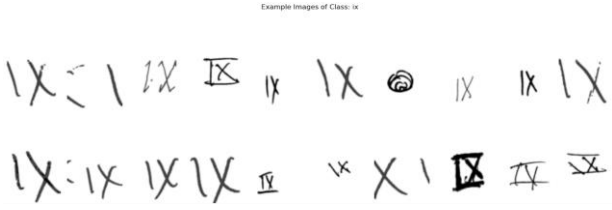

Figure 4: Collection of 20 Random Images from Class 'IX'

To address this issue, we recognized the need to use an alternative trained model to assess the confidence levels associated with the model's predictions. By incorporating a different model, we aim to evaluate the certainty or uncertainty with which the model assigns labels to images.

It is critical to note that the example in Figure 4 reflects only one occurrence in the dataset. However, we know more labeling inconsistencies or other significant concerns could occur. We are interested in improving the accuracy and efficacy of subsequent model training by proactively recognizing and addressing such instances.

3. DATA CLEANING, AUGMENTATION, AND BALANCING

This section delves into the crucial data cleaning, augmentation, and balancing steps in our data analysis and presentation project. By performing these steps, we lay the groundwork for an effective diverse dataset, improving our machine learning model's performance and generalization capabilities.

3.1 Data cleaning

As a first step in data processing, we focused on cleaning the dataset by addressing wrongly labeled data and removing unrelated noise. Due to the limited original data available, we performed this process manually.

It was essential for the model to learn to recognize images even if they were not flawless or properly written. However, we had to be cautious not to make the remaining images appear overly clean while selecting which ones to remove.

Here are some examples of images that required attention during the data cleaning process:

Figure 5: Images tagged as 'I'

As we can see from Figure 5, the first picture is a hoax, and the next two pictures were tagged incorrectly.

Figure 6: Images tagged as 'II'

Figure 7: Images tagged as 'IX'

Figure 8: Images tagged as 'VII'

Figure 9: Images tagged as 'VIII'

After completing this process, we combined the folders of the validation and training sets to handle all the data together. The resulting clean data is stored in the 'all_data' folder. The dataset now consists of 1,915 samples distributed among the different classes, as shown in Figure 10.

Figure 10: Distribution of Pictures per Tag – Clean Dataset (Train)

3.2 Gathering New Data

Following eliminating images during the data cleaning step, the dataset contained less than 2,000 images, below the maximum limit of 10,000 given to us. Therefore, we gathered additional images from various sources to address this issue and improve the dataset's diversity and robustness.

Our approach involved collecting images of the numbers manually written in different fonts. We also contacted friends and relatives, requesting their handwritten Roman numerals. Additionally, we generated images of letters using AI tools such as Midjourney and DALL·E, although the AI applications produced limited relevant images.

Separating the photos into individual numbers required unexpected time and effort, highlighting the challenges of collecting high-quality data. However, we managed to collect more than 48 photos for each number. Here are a few examples (see Figure 11):

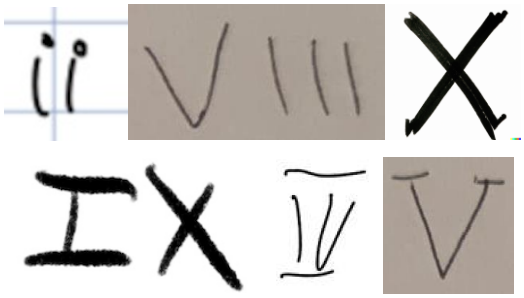

Figure 11: Images Collected Manually

The newly gathered data is stored in the extra_data_comb folder. With the addition of this data, the dataset now includes 2,430 samples distributed among the different classes, as shown in Figure 12.

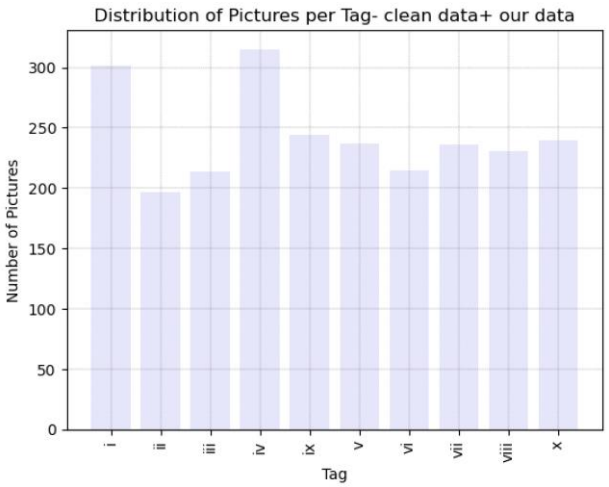

Figure 12: Distribution of Pictures per Tag – Clean Dataset (Train) and The New Images

3.3 Dataset Enrichment

Data set enrichment through data augmentation plays a crucial role in enhancing ML models' performance, robustness, and generalization capabilities by increasing the diversity, quantity, and quality of training data. Furthermore, it allows the models to learn more effectively and improve their ability to handle real-world scenarios and unseen data.

In our approach to data augmentation, each original image undergoes random modifications across four stages. At each stage, the image is modified five times and saved in the relevant subdirectory under 'aug_data'.

Here is a description of the operations performed in each stage:

Part A: Rotation of images at different angles (-x to x degrees, in our case 10, 20, 20). Since an angle greater than 30% can alter the meaning of the number We limited the angles to prevent significant alterations. For example, 'vi' could change into 'iv'.

Part B: This part includes five operations:

1. Randomly cropping a portion of the image within a range of 0-10% of the image size.
2. Affine Transformation (Shearing): Applying a shear transformation to the image, controlled by the shear parameter, ranging from -20 to 20 degrees.
3. Multiply multiplies the image's pixel values by a random value in the range of 0.8 to 1.2.
4. BlendAlpha: blends the image with an alpha mask, combining a foreground and background image. The alpha mask is randomly generated with values ranging from 0.0 to 1.0. The foreground image is created by adding 100 to the original image, while the background image is created by multiplying the original image by 0.2.
5. BlendAlphaElementwise: blends the image with an alpha mask. However, the blending is performed element-wise, meaning each pixel is independently blended.

Part C: This part contains three operations:

1. AdditiveGaussianNoise: Adding Gaussian noise to the image. The noise is randomly generated and added to each pixel.
2. LinearContrast: Adjusting the contrast of the image. The factor parameter determines the contrast adjustment range, randomly chosen between 0.75 and 1.25.
3. AddToBrightness: adjusts the brightness of the image. A positive value increases the brightness, while a negative value decreases it. In this case, the value is randomly chosen between -30 and 30.

Part D: This part includes two operations:

1. ScaleX: scales the width of the image. The scale parameter controls the scaling factor range. In this case, the scale factor between 0.5 and 1.5 is randomly chosen.
2. ScaleY: scales the height of the image. In this case, the scale factor is also randomly chosen between 0.5 and 1.5.

Additional information on augmentation options can be found here- augmenters.geometric - Affine.

After the data augmentation process, the dataset contains 12,083 samples distributed among the different classes, as shown in Figure 13.

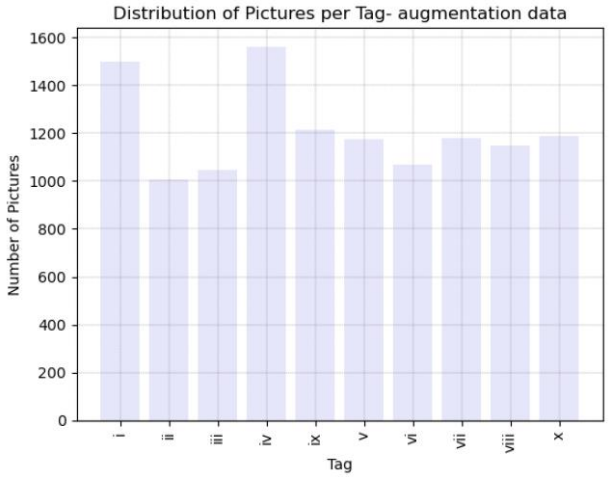

Figure 13: Distribution of Pictures per Tag – Augmentation Dataset (Train)

3.4 Data Balancing

Addressing class imbalance is critical for improving model performance, reducing biases, avoiding overfitting, ensuring resilience in real-world circumstances, and enabling reliable model assessment. Furthermore, it assures that the model may learn from and generalize to all groups in the dataset, resulting in a more accurate and equitable representation of the underlying data distribution.

Given our sample limit of up to 10,000, we will use 1,000 for each class. We will later divide the samples into training and validation sets.

3.5 Data Fragmentation

Splitting the data into training and validation sets is essential in machine learning for two main reasons:

1. Model Evaluation: The validation set is used to assess the performance and general capability of the trained model.
2. Preventing Overfitting: Splitting the data helps detect overfitting, which occurs when a model performs well on training data but fails to apply to new, unknown data. We may detect indicators of overfitting by monitoring the model's performance on the validation set during training and making relevant changes, such as changing the model architecture or using regularization techniques.

A common practice is to use a 70-30 or 80-20 split, where 70% or 80% of the data is used for training, and the remaining 30% or 20% is used for validation. We will examine what is the optimal distribution for our data set.

4. RESULTS

The results section displays the training results and evaluates the model on various datasets. By studying these results, we get important insights into the effectiveness of different data processing approaches and their effects on the model's accuracy and generalization capabilities.

4.1 Original data and original distribution accuracy (after 20 epochs)

- Training time: 213 minutes and 10 seconds
- Best validation accuracy: 0.9 (90%)
- Training loss: 0.0837
- Training accuracy: 0.9811 (98.11%)

These results indicate that when training on the original dataset without any modifications, the model achieved a high training accuracy of 98.11%. Furthermore, the best validation accuracy reached was 90%, suggesting that the model generalized well to unseen data.

4.2 Clean data (split 80-20) (after 20 epochs)

- Training time: 306 minutes and 23 seconds
- Best validation accuracy: 0.850254 (85.03%)
- Training loss: 0.1357
- Training accuracy: 0.9626 (96.26%)

In this case, the dataset was cleaned and split into an 80-20 ratio for training and validation. After training the model on this clean dataset, the best validation accuracy achieved was 85.03%. The training accuracy remained high at 96.26%, indicating that the model learned well from the cleaned dataset. However, the results are worse than the original data, so this approach is ineffective.

4.3 Clean data + augmentation+our data+ balancing (split 80-20) (after 20 epochs)

- Training time: 603 minutes and 46 seconds
- Best validation accuracy:0.981675 (98.167%)
- Training loss: 0.0542
- Training accuracy: 0.9844 (98.44%)

The results obtained after training the model on the clean dataset, augmented with additional data and balanced, show significant improvements in performance. The best validation accuracy achieved was 98.167%, indicating that the model learned effectively from the enriched dataset. The training accuracy remained high at 98.44%, demonstrating the model's ability to generalize well to unseen data. These results highlight the positive impact of data cleaning, augmentation, and addressing class imbalance on enhancing the model's accuracy and overall performance. The results can be shown in figures 14 and 15.

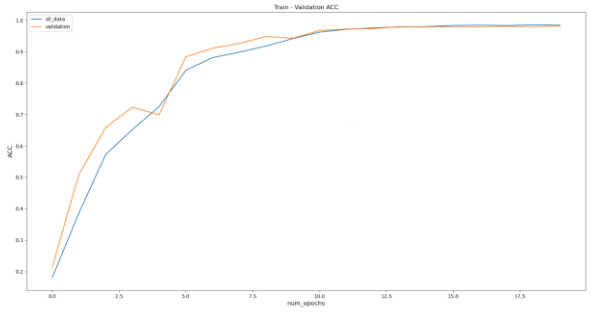

Figure 14: Train – Validation Accuracy

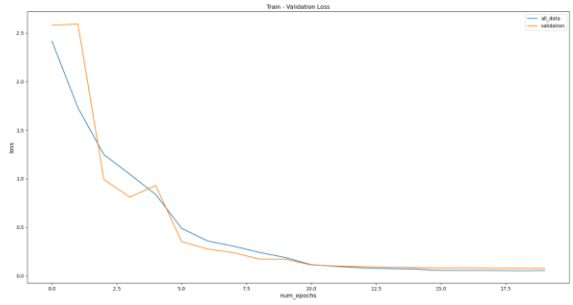

Figure 15: Train – Validation Loss

5. SUMMARY AND DISCUSSION

This project aimed to enhance the performance of a predetermined machine learning model by manipulating the provided handwritten Roman numerals dataset. The primary focus was data cleaning, data set enrichment through augmentation, and class imbalance.

The exploratory data analysis revealed the dataset's composition and labeling distribution, highlighting the need for data cleaning due to labeling inconsistencies and unrelated noise. Manual cleaning addressed wrongly labeled data while maintaining the dataset's diversity.

To augment the dataset, images of handwritten Roman numerals were collected from various sources, including friends, family, and AI tools. Data augmentation techniques, such as rotation, cropping, affine transformation, noise addition, and contrast adjustment, were applied.

Class imbalance was addressed by balancing the dataset, ensuring each class had an equal number of samples. The data was then split into training and validation sets to evaluate model performance and prevent overfitting.

The results indicated that different data processing approaches had varying effects on the model's accuracy and generalization capabilities. Training on the original dataset without modifications resulted in a high training accuracy of 98.11% and a validation accuracy of 90%, indicating good generalization. However, cleaning the dataset led to a decrease in performance, suggesting a loss of relevant patterns. Augmenting the clean dataset and addressing class imbalance significantly improved performance, with a validation accuracy of 98.167% and a training accuracy of 98.44%. These results highlight the positive impact of data augmentation and addressing class imbalance on enhancing accuracy and generalization.

In summary, data cleaning, augmentation, and addressing class imbalance are crucial for improving machine learning model performance. Cleaning and enriching the dataset with additional data enhance learning effectiveness and accuracy, while balancing the dataset prevents bias and ensures fair representation, leading to improved overall performance. These findings emphasize the significance of thoughtful data manipulation strategies in optimizing model outcomes. This project demonstrated the importance of data cleaning, augmentation, and addressing class imbalance in improving model performance, providing valuable insights for future efforts to enhance accuracy through data manipulation.