Increasing Data Diversity with Iterative Sampling to Improve Performance

Devrim Cavusoglu **Oğulcan Eryuksel** **Sinan Altinuc**

OBSS AI
{devrim.cavusoglu, ogulcan.eryuksel, sinan.altinuc}@obss.com.tr

Abstract

As a part of the Data-Centric AI Competition, we propose a data-centric approach to improve the diversity of the training samples by iterative sampling. The method itself relies strongly on the fidelity of augmented samples and the diversity of the augmentation methods. Moreover, we improve the performance further by introducing more samples for the difficult classes especially providing closer samples to edge cases potentially those the model at hand misclassifies.

1 Introduction

This paper serves as a summary documentation of methods for experiments studied under the Data-Centric AI competition DeepLearning.AI and Landing AI [2021]. The competition provides an MNIST LeCun et al. [1998] style dataset with Roman numerals and expects the competitors to increase the performance by only manipulating data and not the code or the model to emphasize the importance of data in machine learning models and enforce a data-centric approach. The competition holds a fixed model and training setup for all submissions.

2 Experimental Setup and Tools

The official training script[1] is used in all training phases of the experiments unless stated otherwise. The model is part of the ResNet50 He et al. [2016] as defined in the official training script. Without changing the training setup in the file, we added some utilities and helpers such as training accuracy/loss plots, classification reports and export of prediction outputs to ease the interpretation and exploration of the outputs.

The training loss/accuracy plot is added to track how well we develop and iterate the data set. To see how well the model performs on different classes, we added a classification report Pedregosa et al. [2011] to better understand and aid the data. To better discriminate the misclassified samples visually, we used FiftyOne package Moore and Corso [2020]. FiftyOne allowed us to go through the samples easily for which parts we wanted to investigate. We also utilize FiftyOne to compute similarities of embeddings for iterative sampling, which is discussed under subsection 3.4.

3 Evolution of Dataset(s) & Applied Methods

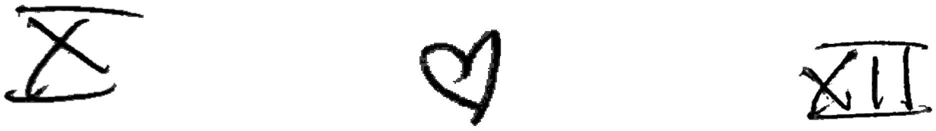
The information for all datasets used, manipulated, or generated during experiments are listed on Table 1. Our validation set is constructed from the original validation split of $Base_0$ by adding

[1] The official training script can be accessed here.

1st NeurIPS Data-Centric AI Workshop (NeurIPS DCAI 2021).

Dataset name	Equal	Avg. size per class (train)	Size per class (val)	Description
$Base_0$	✗	207.2	81.6	Original competition images
$Base_1$	✓	250	100	Cleaned $Base_0$ + Fixed per class size with DA
$Handcrafted$	✓	30	-	Handwritten Roman Numerals by People
$Label\_book$	✗	5.2	-	Label book provided by the competition
$Synthetic_1$	✗	78	-	Synthetic Handwriting Samples with $Style\_Set_1$
$Synthetic_2$	✗	124	-	Synthetic Handwriting Samples with $Style\_Set_2$
$Pool$	✓	6782.7	-	Augmented image samples from the combination of $Base_0$ + $Handcrafted$ + $Synthetic_1$ + $Synthetic_2$
$Dataset_{syn1}$	✗	328	100	$Base_1$ + $Synthetic_1$
$Dataset_{syn2}$	✗	452	100	$Dataset_{syn1}$ + $Dataset_{syn2}$
$Dataset_{syn\_aug}$	✓	900	100	$Dataset_{syn2}$ + Augmentations
$Dataset_{iter}$	✓	900	100	$Dataset_{syn\_aug}$ + Iterative Sampling
$Dataset_{uneven}$	✗	900	100	$Dataset_{iter}$ + Favoring difficult classes

Table 1: Description of datasets used during the competition. Column "Equal" indicates if training sample sizes are equal for each classes or not, note that for validation split this is always true except $Base_0$. Only train splits are manipulated and validation (and test) splits are held fixed except $Base_0$ which is raw dataset provided by the competition.



(a) Case 1

(b) Case 2

(c) Case 3

Figure 1: Three example cases for bad samples in $Base_0$. Sample **(a)** belongs to class "iv" whereas it actually belongs to class "x". Sample **(b)** is not a Roman numeral, but present under class "vii". Sample **(c)** belongs to class "vii" whereas it is distinguishably not a "vii".

augmented samples until each class has the same amount of samples, 100. We use $Label\_book$ given under the competition as "Label book" in the experiments and results.

Naturally, the development process for enhancing the data (specifically training set since validation and test sets are held fixed in our experiments) goes sequentially cumulative. That is, until reaching the hard limit for the competition submission, which is 10,000 (9,000 for training) samples in total, we tried to introduce each method with a soft limit on top of the previous one. This procedure is held for all datasets generated except for the final method, which is changing class sizes so that difficult classes have more training samples by visual inspection.

3.1 Cleaning the Raw Dataset

To create the $Base_1$ dataset, we removed samples that are non-numeral or too ambiguous between classes and corrected the labels for mislabelled examples. It can be observed in Figure 1 that some samples violate the consistency and validity of the dataset.

3.2 Synthetic Handwriting Generation

We used a synthetic handwriting generative model Graves [2013] to enrich the dataset diversity in the training split. We use the implementation of Vasquez [2018]. One drawback for this model is the tendency to add more characters than intended.

The generated samples from RNN are quite satisfying; however, some of the generated samples for some set of classes have discrepancies which can be seen in Figure 2. Some of these samples with flaws are corrected by adding a suffix to text and applying traditional post-processing to remove that

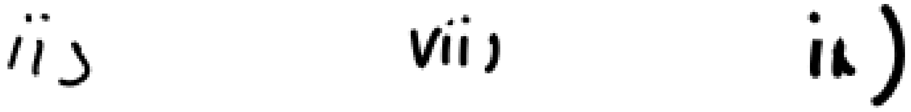


Figure 2: Samples with discrepancy that have trailing extra components.

suffix from the generated image, and to discard all flawed generated samples, the generated samples are visually inspected after the post-processing.

The implementation we use has a total of 13 styles and additional parameters such as $bias$ and $stroke\_width$. These parameters are either fixed to a certain value or set to a distribution where the distribution parameters are fixed by visually inspecting the generated samples. By using two different style sets, we generated $Synthetic_1$ and $Synthetic_2$ dataset. With these two synthetic datasets, we generated $Dataset_{syn1}$ and $Dataset_{syn2}$ by combining them with the cleaned dataset $Base_1$.

3.3 Data Augmentation

In order to both populate and enhance the diversity of the dataset, we applied data augmentation on $Dataset_{syn2}$. We selected some proper set of augmentations for these samples, which are grayscale and generally in small sizes (resolution). We used Augly package Bitton and Papakipos [2021] for our augmentation procedure. The set of augmentations we used are *HFLip, VFlip, ShufflePixels, Pixelization, Rotation, Blur, RandomAspectRatio, Noise* from Augly. We also used barrel and barrel inverse distortions from ImageMagick The ImageMagick Development Team [2021].

Note that we used these augmentations as single or as a composition of several. For these augmentations, we manually and in an iterative way selected a certain set of parameters for each of them carefully not to make the image unrecognizable. The potential danger zone here is that after an augmentation, the resulting image may become unrecognizable or may belong to a different class (horizontal flip of "vi" and "iv"). With the specified augmentation list, we combine the augmented images with $Dataset_{syn2}$ so that augmented images fill the gap until maximum value per class, *900* (maximum class size for equal class sizes). With this combination we get $Dataset_{syn\_aug}$.

3.4 Iterative Sampling with Augmentation

To increase diversity even more, we propose an algorithm that iteratively removes the similar samples and adds from a pool of augmented samples. We first created a pool of augmented images, $Pool$, with the augmentation set mentioned in subsection 3.3. $Pool$ contains only augmented samples and not the base images. Besides the augmented image pool, we also need embeddings for distance calculation. To get embeddings, we used the official model, trained on $Dataset_{syn\_aug}$, and we got the global average pooling outputs, $layer^{L-1}$ where $layer^{L}$ is the softmax layer. The embeddings for samples are vectors of $dim(1, 256)$.

Then, using algorithm 1 we iteratively replaced the similar samples in $Dataset_{syn\_aug}$ with the samples randomly drawn from $Pool$. To find similar samples, we used FiftyOne's $find\_duplicate$ function. In this experiment, we used Euclidean distance as a distance metric and $N = 10$. With this algorithm, $Dataset_{iter}$ is generated. Realize that the achievement of this procedure does not solely rely on the similarity computation by embeddings but also assuring the diversity of augmentations.

3.5 Favoring Difficult Classes

By observing the classification performance of the model for each class by f1 metric, we have seen that the classification performance for the classes "i", "v", and "x" are relatively higher. Thus, we have decided to imbalance the sample sizes towards more difficult classes as apparently "i", "v", and "x" are easier for the model to learn compared to other classes. Then, visually examining the misclassified examples through FiftyOne, we manually selected samples to the misclassified ones from $Pool$. With this approach, we retrieve $Dataset_{uneven}$.

Algorithm 1: Iterative sampling algorithm.

```
Data: $D$ dataset, $P$ augmented_data_pool, $N \geq 0$ number of iterations,
      $max\_sizes$ maximum size per class, $metric$ distance metric
Result: $D'$ new dataset
$C \leftarrow [i, ii, iii, iv, v, vi, vii, viii, ix, x]$;
for $c$ in $C$ do
    $D_c \leftarrow get\_class\_samples(D, class = c)$;
    $E_c \leftarrow get\_embeddings(D_c)$ ;                /* retrieve model embeddings */
    $P_c \leftarrow get\_class\_samples(P, class = c)$;
    $S \leftarrow find\_duplicates(E_c, metric)$;
    $n \leftarrow 1$;
    while $S \neq \varnothing$ or $n \leq N$ do
        $remove(D_c, S)$ ;                              /* remove $S$ from $D_c$ */
        $m \leftarrow max\_sizes_c - size(D_c)$;
        $P_{sub} \leftarrow select(P_c, n = m)$ ;       /* randomly get $m$ samples from $P_c$ */
        $add(D_c, P_{sub})$ ;                           /* add samples from $P_{sub}$ to $D_c$ */
        $remove(P_{sub}, P_c)$ ;                        /* remove $P_{sub}$ from $P_c$ */
        $S \leftarrow find\_duplicates(E_c, metric)$;
        $n \leftarrow n + 1$;
    end
end
```

Dataset	Train set Acc.	Validation set Acc.	Test set Acc.	Marginal Gain	Cumulative Gain
$Base_0$	99.66	67.53	59.62	-	-
$Base_1$	**1.00**	72.30	61.54	-	-
$Dataset_{syn1}$	99.75	77.80	67.30	5.5	5.5
$Dataset_{syn2}$	99.67	81.40	73.08	3.6	9.1
$Dataset_{syn\_aug}$	99.72	85.00	71.15	3.6	12.7
$Dataset_{iter}$	99.28	93.20	**98.08**	**8.2**	20.9
$Dataset_{uneven}$	99.31	**95.50**	**98.08**	2.3	**23.2**

Table 2: Results for iterations on datasets, accuracy scores are reported as percentages. $Label\_book$ is used for "Test set Acc." since submission scores are not available for all datasets. Marginal and cumulative gains are with respect to validation set. Note that the gain between $Base_0$ and $Base_1$ is not reported as the validation splits are different, however, it is fixed for the rest.

4 Conclusion

We took an iterative approach using the following methods in succession: data cleaning, generating synthetic data using RNN based handwriting generation, data augmentation, iterative sampling, creating an imbalance of classes favoring difficult ones. Each step provided a marginal improvement. However, it should be noted that it would not be very accurate to directly compare these improvements to each other as it gets harder to improve the accuracy as the overall accuracy increases. The results of the datasets can be seen in Table 2. Results of $Dataset_{syn1}$ and $Dataset_{syn2}$ Indicate that synthetic handwriting data generation strategy improved the results significantly providing a total accuracy improvement of 9.1 points. Marginally $Dataset_{iter}$ provides the most significant gain. Therefore, the proposed method of iterative sampling utilizing model embeddings improves the diversity of the dataset. The manipulated dataset $Dataset_{uneven}$ boosted the performance further and attained the highest score suggesting that having uneven class sizes favoring difficult classes can work well. Note that synthetic data generation and augmentation improve accuracy by increasing the number of samples; whereas, iterative sampling and favoring difficult classes increase performance without changing the total sample size.

This paper shows an approach to diversifying the dataset through iterative sampling, and also boost the performance even more with making class sizes uneven such that difficult classes have more samples. The results suggest that the approaches proposed here works well with the dataset (MNIST style with Roman Numerals) provided by the competition.

References

DeepLearning.AI and Landing AI. Data-centric ai competition, 2021. URL https://https-deeplearning-ai.github.io/data-centric-comp/.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

B. E. Moore and J. J. Corso. Fiftyone, 2020. URL https://github.com/voxel51/fiftyone.

Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

Sean Vasquez. Handwriting Synthesis, 02 2018. URL https://github.com/sjvasquez/handwriting-synthesis.

Joanna Bitton and Zoe Papakipos. Augly: A data augmentations library for audio, image, text, and video. https://github.com/facebookresearch/AugLy, 2021.

The ImageMagick Development Team. Imagemagick, 01 2021. URL https://imagemagick.org. version 7.0.10.