Data-Centric AI Competition — Tips and Tricks of a Top 5% Finish

Techniques of a top-ranked submission in the competition organized by Andrew Ng and DeepLearning.AI

Kenneth Leung 2 days ago · 7 min read ★

Photo by Kalen Emsley on Unsplash

the recent **Data-Centric AI Competition** organized by Andrew Ng and DeepLearning.AI.

In this article, I share the techniques of my **Top 5%** submission (~84% accuracy), methods that worked and did not work, and learning points from experiments performed by myself and other high-ranked participants.

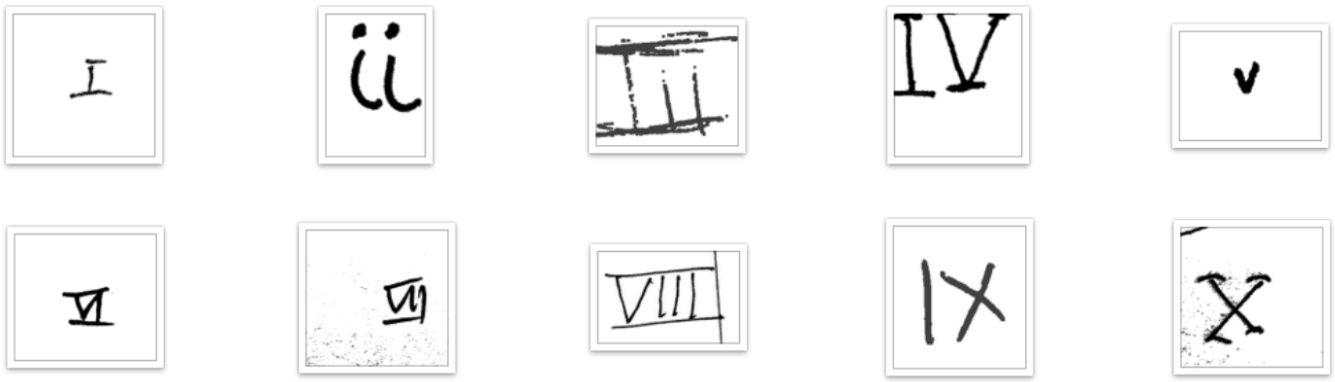
Contents

Photo by Pietro Mattia on Unsplash

highly performant advanced model architectures.

This competition breaks from tradition by **fixing the model** (ResNet50) instead of the dataset. Hence, participants need to improve the data by using techniques such as label correction and data augmentation.

The dataset contains 2,880 images of handwritten Roman numerals ranging from 1 to 10 (i.e., *I* to *X)*. The data is separated into training and validation sets, where each contains subfolders corresponding to the ten numerals.



Sample of the ten types of Roman numeral images | Compiled by author from public competition data

The goal is to **improve the image dataset** such that the model (when evaluated against a hidden test set) can identify the Roman numeral labels with the highest possible **accuracy**.

(2) Experiment Tracking

Instead of using advanced ML tracking tools (e.g., MLFlow, W&B, etc.) for experiment management, I went with the good old **Microsoft Excel** as it was well-suited for these straightforward tasks.

The aim is to document the effects of every change and use it to guide subsequent experiments. Here is a snapshot of the spreadsheet template I used:

	• Data cleaning initiated			
2	• Start data shuffle of train and val raw folders, and then finally do a random split 70/30 to train/val after augmentation complete • Switch to imgaug library instead of Torchvision • Implement imgaug's proposed simple augmentation sequence (https://imgaug.readthedocs.io/en/latest/source/examples_basics.html#a-simple-and-common-augmentation-sequence) • Perform random horizontal flip for labels i, ii, iii, v, x	70/30	0.8157	+0.026
3	• Repeat steps (e.g. data shuffle and baseline imgaug basic augmentation, train test split 70/30) from Experiment 2 • Introduce random vertical flips for i,ii, iii, ix, x	70/30	0.82107	+0.005
4	• Perform extensive manual review of data (i.e. review past deleted images and add back if looks valid, while also reviewing existing ones to check that they are correctly labeled • Add sample label book data into training set • Adjust train/val split from 70/30 to 80/20	80/20	0.83471	+0.014
5	• Remove random order of augmentation (i.e. keep augmentation steps sequential in pre-specified order)	80/20	0.83967	+0.005

Spreadsheet template for experiment tracking | Image by author

(3) Techniques Of My Best Submission

You can find the codes for my best submission in the '*Full_Notebook_Best_Submission.ipynb*' Jupyter notebook within this **GitHub repo**, so check it out for a comprehensive step-by-step walkthrough.

Instead of inundating you with details of every step, I will focus on the **four** most impactful techniques.

(i) Combining train and validation data

During data exploration, I found that images in the validation set appeared **cleaner and more representative** of the numerals than those in the train set.

Therefore I **combined images from both train and validation sets** into a single folder for a **more balanced distribution**. This shuffled dataset then undergoes augmentation before being split into *train/val* sets at the end.

There was also a separate `label_book` folder containing five 'best' examples for each label. I also placed these ideal samples into the dataset to **enrich the dataset**.

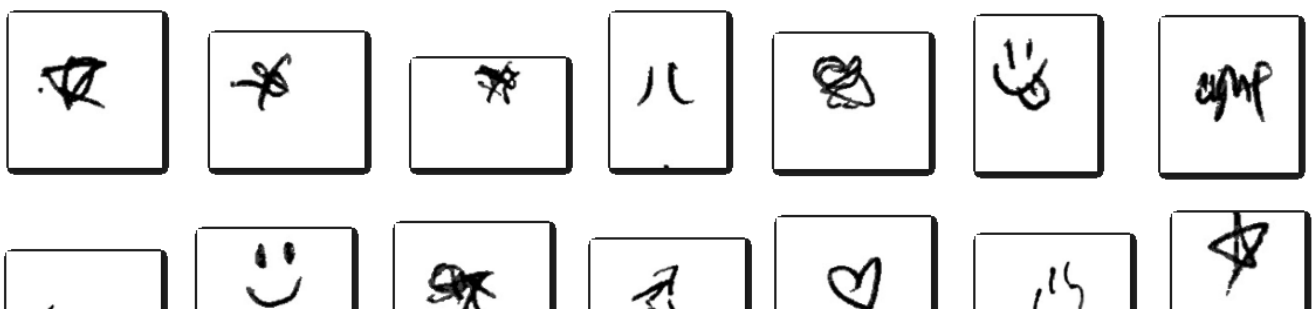
distributions between them. You can find the UMAP visualization here.

Photo by Tim Mossholder on Unsplash

(ii) Focus on data cleaning

Given that this is a data-centric competition, one can expect a significant amount of work to go into data cleaning. There are two main types of issues to handle in the process:

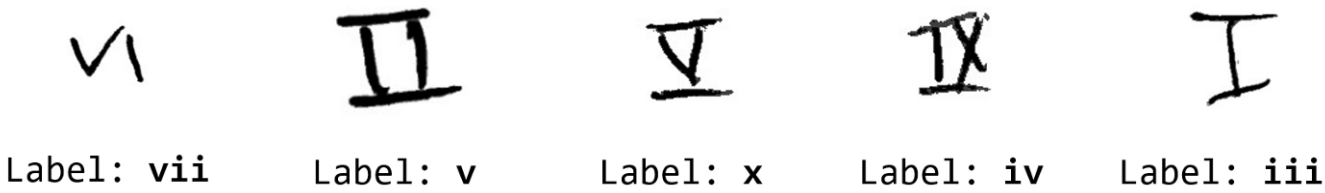
- ***Irrelevant images***



Examples of images unrelated to the ten Roman numerals | Compiled by author from public competition data

These noisy irrelevant examples were moved in a separate folder **excluded** from model training. Although this shrinks the dataset size, it is crucial to remember that **data quality trumps data quantity**.

- ***Mislabeled images***

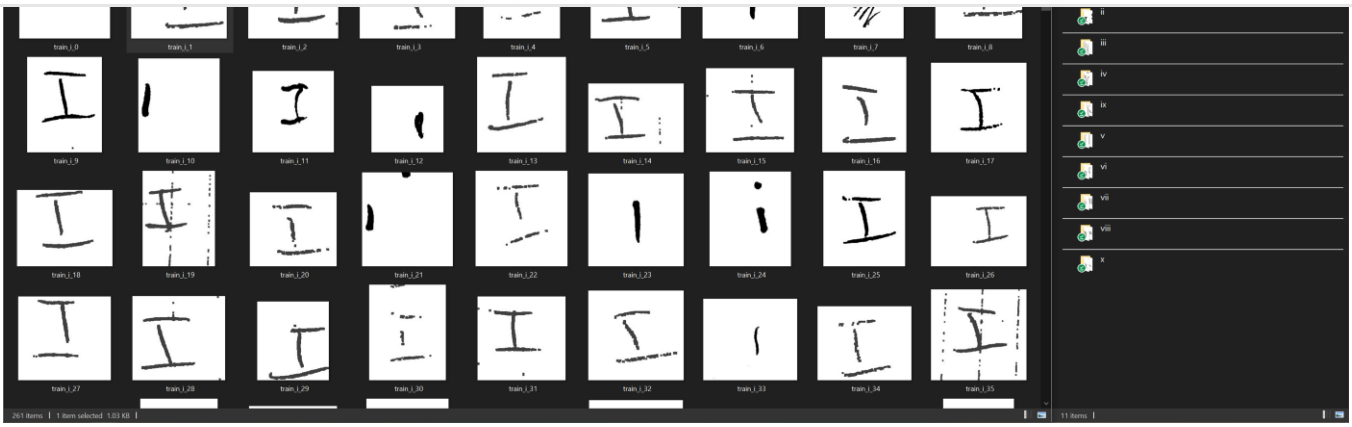


Examples of mislabeled images | Compiled by author from public competition data

I shifted the mislabeled images into the appropriate folders based on my judgment. The benefit of doing data cleaning by myself is that it **introduces consistency in the relabeling process**, and this is key as numerous examples were highly ambiguous.

I kept the execution of data cleaning as simple as possible by using a **drag-and-drop approach with two windows open side-by-side**.

The (wider) left window displays the images prominently, while the right window lists the different folders into which I can shift the images.

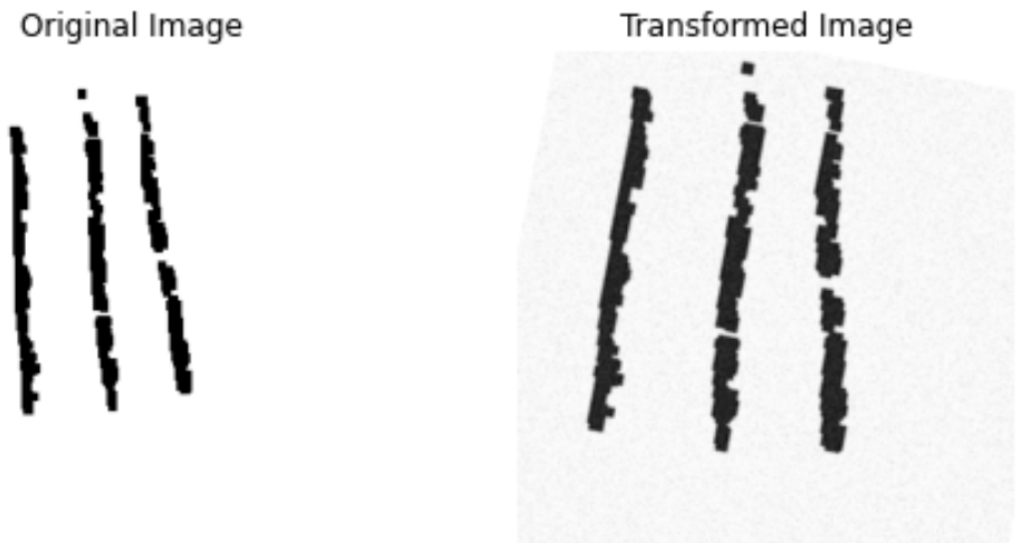


My drag-and-drop approach to deleting or relabeling images. My widescreen monitor certainly made this easier for me to execute. | Image by author

(iii) Targeted flipping

I utilized ***imgaug*** for data augmentation as I find that it has a broader range of augmenter options. A sequence of common augmentations involving the following transformations was applied:

- Resize and Crop
- Gaussian Blur and Additive Gaussian Noise
- Linear Contrast and Multiply (for random contrast adjustments)
- Affine transformations like scaling, translation, rotation, and shearing



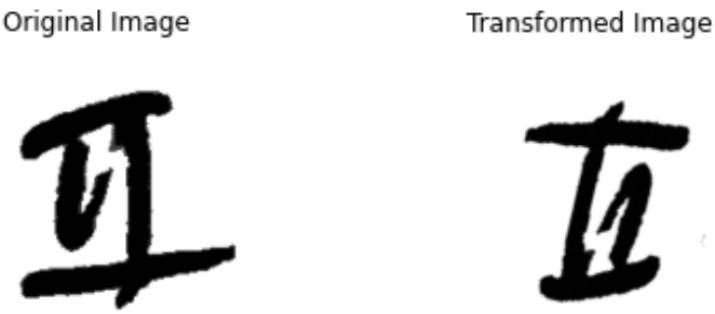
The above sequence is pretty standard, so there isn't much to discuss. (*Details can be found in the **GitHub repo**)*

The more intriguing part is that I adopted a **targeted random flipping** approach. Depending on the label, I applied either **horizontal** flipping (`fliplr`), **vertical** flipping (`flipud`), **both**, or **none**.

Amidst all these flips, the vital thing is ensuring that the resultant augmented images still resemble the original Roman numerals.

- ***Horizontal and Vertical Flip:*** **i, ii, iii, x**

These four labels (i, ii, iii, x) are suitable for both types of flips because the resultant images still look like the original numerals.



Example of a horizontal + vertical flip on an image labeled `**ii**` | Image by author

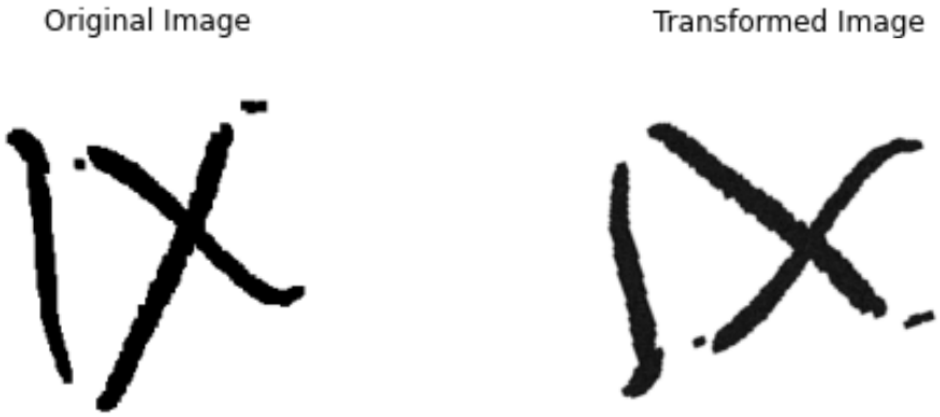
- ***Horizontal Flip only:*** **v**

For the label **v**, only horizontal flips make sense.

Example of a horizontal flip on an image labeled `**v**` | Image by author

- ***Vertical Flip only:*** **ix**

For the label **ix**, only vertical flips make sense.



Example of a vertical flip on an image labeled `**ix**` | Image by author

- ***No Flipping:*** **iv, vi, vii, viii**

For the remaining labels, performing any form of flipping would result in unrecognizable or incorrect numerals. As such, no flips were scripted for images with these labels.

(iv) Maximizing dataset size limits

size as much as possible.

I set a target size of 1,000 images for each of the ten labels for equal representation. This is done by topping up the data of each label with new images generated from random augmentation on the original images.

With the above techniques, I achieved a model accuracy of **83.97%** (Top 5%, Rank 24). This accuracy is significantly greater than the baseline (64.4%) and is only <2% away from the winning submission of 85.82%.

24	submission-v05	0.83967
Aug 05, 2021	*Kenneth Leung*	
	https://github.com/kennethleungty	

My final position on the leaderboard | Image by author

(4) What Did Not Work

Here are some techniques I tried which did **not** seem to improve the dataset:

- Increase the size of images (e.g., 500 x 500)
- Introduce random order within the sequence of augmentation steps
- Apply morphological operations (e.g., dilation/erosion, Otsu thresholding)
- Further increase the strength of augmentation, e.g., more drastic rotations, shearing, contrasting, etc.
- Ensure every image is augmented at least once when topping up the dataset to 10,000.

utilized. Here are some of the published examples:

- GoDataDriven created a Streamlit app to augment images sequentially.
- Johnson Kuan developed a Data-Boosting technique that adds the nearest neighbor augmented images (based on embeddings) to the training set.
- Pierre-Louis Bescond leveraged square cropping and background noise transfer functions as part of the augmentation.
- Walid Daboubi used TornadoAI (implementation of human-in-the-loop machine learning) for data labeling.

There is also a post-competition forum thread filled with brilliant ideas.

Do let me know if you come across other high-ranked innovative solutions, and I will be happy to add them to this list.

Photo by AbsolutVision on Unsplash

(3) Key Learning Points

- Simplicity does not equate to low quality. While my methods are simple compared to other submissions, they still yielded excellent results and outperformed many other innovative techniques. I hope this is encouraging for anyone keen to try out data-centric techniques.
- There are many solutions to a single problem in the field of data science, so don't be afraid to experiment with all your wildest ideas.
- It is important to balance benefits and costs. When I found that my remaining ideas did not improve the accuracy, I decided not to commit additional time and energy for experimentation given these diminishing returns.
- Data-centric approaches will undoubtedly grow in popularity, especially since modeling is just a small part of the entire ML pipeline.

Before you go

I welcome you to **join me on a data science learning journey!** Follow this Medium page and check out my GitHub to stay in the loop of more exciting data science content.

You can find the GitHub repo for this article **here**. Meanwhile, have fun adopting a data-centric approach!