<u>Table of Context</u>

Explanatory Data Analysis

To better understand the data at hand, we decided to analyse the data while asking ourselves: what can affect the training process of the given model?

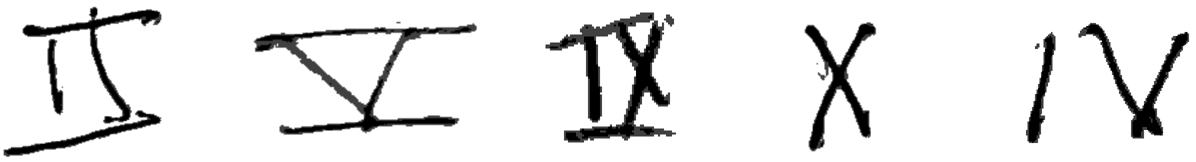
The given train set contains 2,068 images of handwritten roman numerals. The class distribution is unbalanced (see figure 1 to the right), which can cause a model bias towards learning the representation of the common classes and not learning a sufficient generalisation of the smaller represented classes.

The class representing the number 4 (iv) contains 281 images, 13.59% of all the dataset. In contrast, the class representing the number 2 (ii) contains only 157 images, 7.59% of all the dataset. This means there's a 1.79x more representation of iv in the dataset over ii.

Class	Count	%
iv	281	13.59%
i	261	12.62%
ix	234	11.32%
viii	199	9.62%
v	196	9.48%
vii	193	9.33%
iii	187	9.04%
vi	181	8.75%
x	179	8.66%
ii	157	7.59%
Total	2,068	

Figure 1
Class distribution in trainset

We manually examined a random portion of the data set and found there to be 2 main issues. The first issue was incorrectly labelled train data (see figure 2 below).



Label: 1
Correct Label: 2

Label: 2
Correct Label: 5

Label: 4
Correct Label: 9

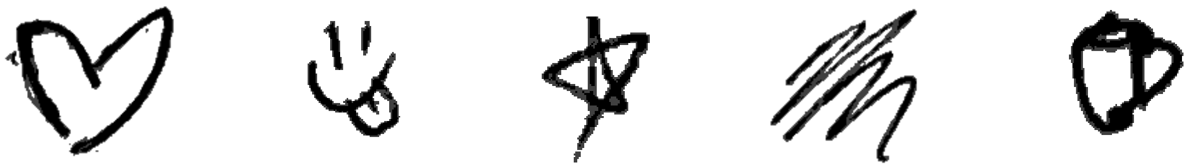
Label: 6
Correct Label: 10

Label: 8
Correct Label: 4

Figure 2
Examples of incorrectly labelled train data

Incorrectly labelled train data can mislead the model to learning a false representation of the concept we want it to learn, for instance – if all the ones (i) are twos (ii), the model will classify an image as a one (i), even though it is really a two (ii) after the training process.

The second issue found in the train data is the presence of noise (see figure 3 below).



Label: 3
Correct Label: Heart?

Label: 8
Correct Label: Emoji?

Label: 8
Correct Label: Star?

Label: 6
Correct Label: Scribble?

Label: 6
Correct Label: Mug?

Figure 3
Examples of noise within the train data

Noise which doesn't represent the labels we would like the model to learn, have no place in the training dataset. There should be a pre-processing stage removing images which are not part of the label class. On the other hand, if we would have the ability to modify the model, we could add a new class called "other", which classifiers all the images which are not over a chosen threshold to be classified as a roman numeral.

Manual Cleaning

After the mini-EDA we conducted, we decided to manually clean and filter the dataset as the first step. We manually removed all the "noisy" data, containing unrecognisable numerals. In addition, we relabelled all the labels which were incorrectly labelled as shown in figure 2 above. The description of the dataset after the manual cleaning process is shown in figure 4 to the right.

class	count	%	items removed / added
iv	260	13.72%	-21
i	244	12.88%	-17
ix	214	11.29%	-20
v	187	9.87%	-9
vii	178	9.39%	-15
viii	177	9.34%	-22
x	172	9.08%	-7
vi	164	8.65%	-17
iii	153	8.07%	-34
ii	146	7.70%	-11
Total	1,895		-173

Figure 4

Class distribution after manually cleaned.

Data Augmentation

Our goal is to improve the models' performance by only being able to configure the data the model will have at hand to train on.

To begin, we researched the topic in the literature to get a better understanding of the commonly used techniques in the field of data augmentation and data enrichment, with a focus on image classification in deep learning models. This would help us understand which methods are likely better to use then others.

In general, the main conclusion was that expanding the training dataset by generating diverse variations of the original images allows the model to learn a more generalized representation of the data. This will lead to a model which will be better generalized on real world data and will not tend to overfit over the train dataset. Interesting papers we came across worth pointing out include "Improving Deep Learning with Generic Data Augmentation"[1] (2018) and "Random Erasing Data Augmentation"[2] (2020).

With all this in mind, we set out to expand our data by using augmentations, once we had enough data to split the data into a balanced train set without exceeding the 10,000-image limit constraint, we would start looking into the best train validation split for the assignment.

Random Cropping

Random cropping is when we create a random subset of an original image. This was the technique which yielded the largest improvement in the paper stated above, as stated:

"The cropping scheme combined with occlusion yielded the most benefits, achieving a 9% improvement over a benchmark task performance." [3]

See figure 5 below as an example to a successful cropping[4]. When we started implementing the

[1] Taylor, L., & Nitschke, G. (2018, November). Improving deep learning with generic data augmentation. In *2018 IEEE symposium series on computational intelligence (SSCI)* (pp. 1542-1547). IEEE.

[2] Zhong, Z., Zheng, L., Kang, G., Li, S., & Yang, Y. (2020, April). Random erasing data augmentation. In *Proceedings of the AAAI conference on artificial intelligence* (Vol. 34, No. 07, pp. 13001-13008).

[3] Taylor, L., & Nitschke, G. (2018, November). Improving deep learning with generic data augmentation. In *2018 IEEE symposium series on computational intelligence (SSCI)* (pp. 1542-1547). IEEE.

[4] https://mxnet.apache.org/versions/1.5.0/tutorials/gluon/data_augmentation.html

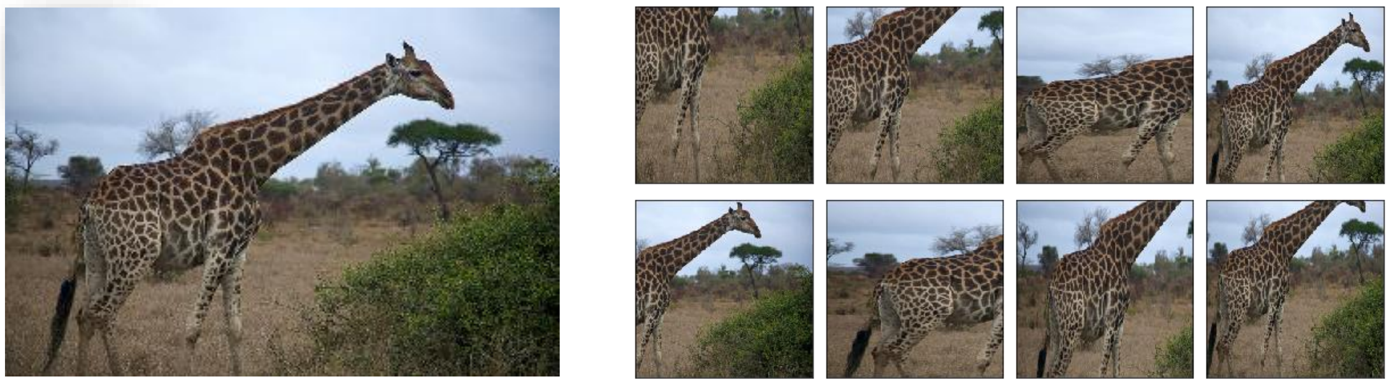


Figure 5

Random cropping augmentation. On the left is the original image and on the right is the randomly cropped images.

cropping augmentation, we found that in our specific assignment the cropping method can potentially create many mislabelled images (see figure 6 below). Therefore, we decided not to use random cropping methods on our data.

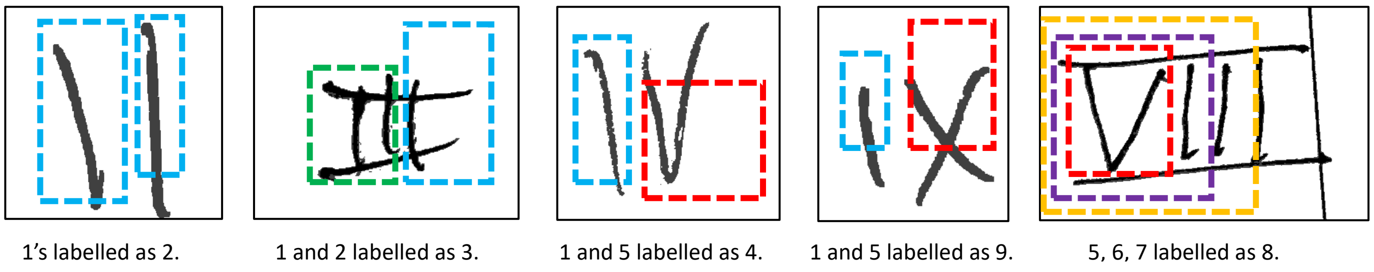


Figure 6

Demonstrating the mislabelled data issue caused by random cropping in classifying roman numerals.

Random Erasing

As we found in the literature[5], Random erasing, places a randomly positioned filled rectangle on the image, essentially erasing a part of the image. This method was found to be very interesting and useful in our use case and is very similar to performing Dropout. Dropout is performed on the weights of a neural network to force the model to not rely on specific concepts it has learnt, resulting in improved generalisation abilities. We are not allowed to configure the models architecture, therefore, finding an augmentation which can perform like Dropout, was very useful (figure 7 below).

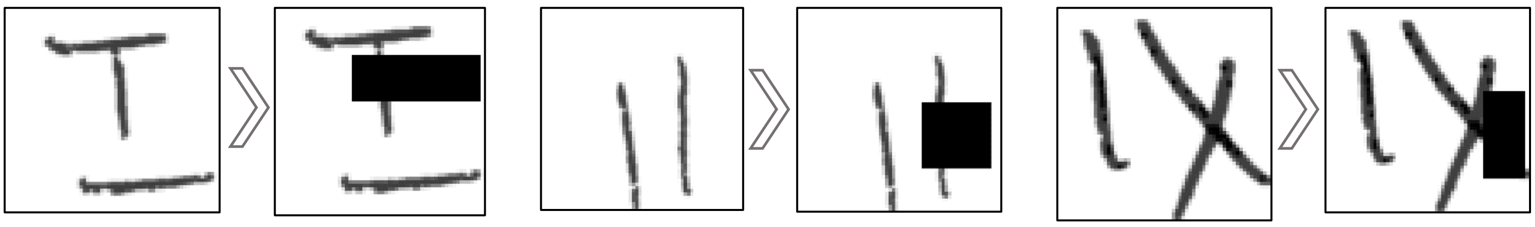


Figure 7

Random erasing examples from the train data.

[5] Zhong, Z., Zheng, L., Kang, G., Li, S., & Yang, Y. (2020, April). Random erasing data augmentation. In *Proceedings of the AAAI conference on artificial intelligence* (Vol. 34, No. 07, pp. 13001-13008).

Random Rotation

At first, we believed the numerals angle have meaning, which lead us to decide to randomly rotate the images only between -45° and 45°. But after reading in depth the idea behind the geometric augmentation of images, we decided to perform the rotations over all [-180°,180°] (we created a rotate right [15, 165°] and rotate left [-165°, -15], the randomness is in the rotation degree chosen). The best example to explain the concept is to think of an image of a dog, if we rotate the image 180°, we will get the same dog, just it would be on its head. With all the numerals, if we rotate 180°, we might have an upside down 5 (rotate v), but it hasn't changed the numeral it represents. The exception is for ix, which when rotated over 90° can be perceived as an 11 (xi). Due to the fact we don't have any class of xi in the data, this small interpretation wont cause any incorrectly labelled data within the dataset. Rotation is a very commonly used augmentation; the model is trained over many orientations of the image. See figure 8 below.

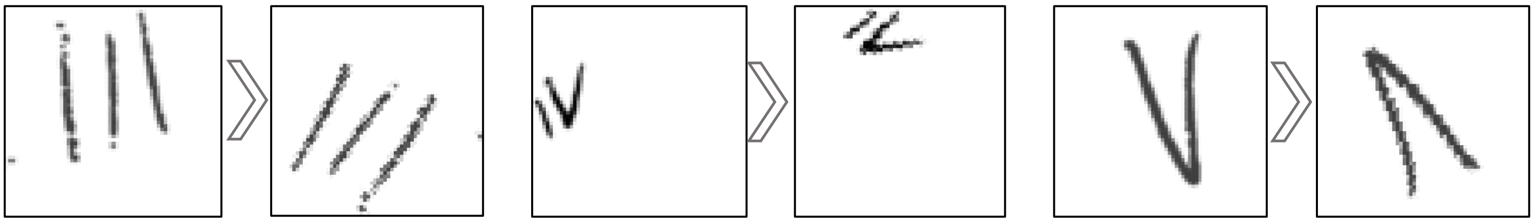


Figure 8

Random rotation examples from the train data.

Random Flip (Vertical & Horizontal)

Flipping is used as part of the geometric augmentations, very similar to the rotation explained above. The only exception here is that there are certain numerals which once they are flipped horizontally– they no longer belong to their original class. For instance: when iv (=4) is flipped horizontally it becomes vi (=6). To utilize these classes which are flipped and become part of another class – we decided to change the class of these classes to the new class after the flip, instead of ignoring them totally. Regarding the horizontal flips, it is like the 180° rotation explained above. See figure 9 below.

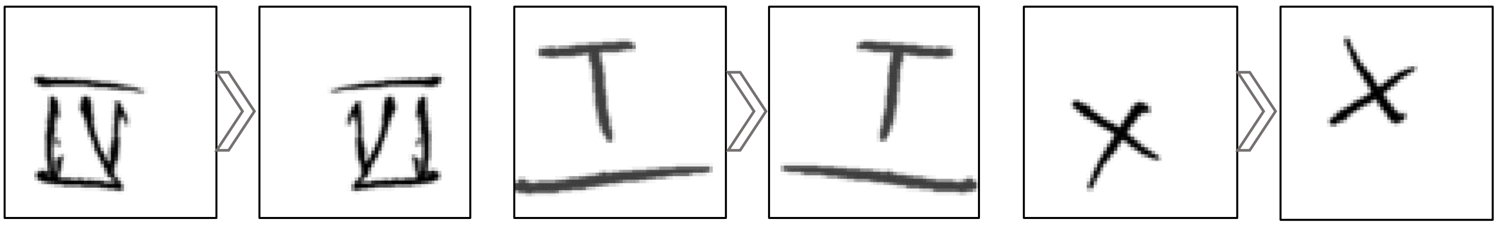


Figure 9

Horizontal and vertical flip examples from the train data. First two (iv, i) examples are horizontal flips, the third (x) is a vertical flip.

Gaussian Blur

Randomly blurring an image using a Gaussian distribution is the last data augmentation technique we decided to use. Intuitively, blurring images for Data Augmentation could lead to higher resistance to motion blur during testing. In general, the data is constructed of both clearer and blurrier images, sing Gaussian Blur will result in a blurrier representation for more images in the data[6]. See figure 10.

[6] Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. Journal of big data, 6(1), 1-48.

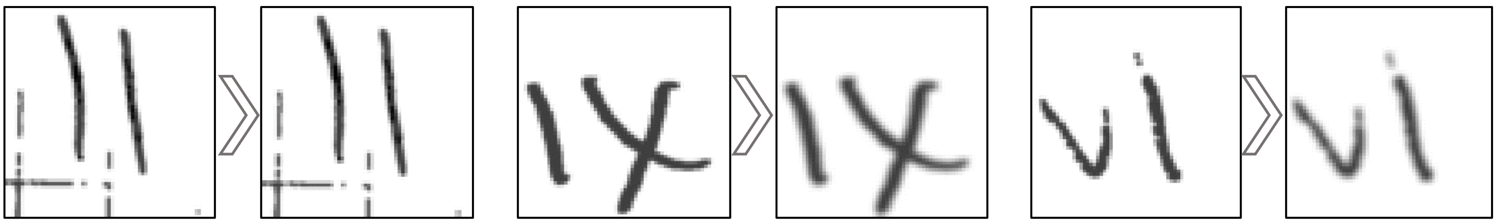


Figure 10

Gaussian Blur examples from the train data.

Data set after augmentation process

We performed all the augmentations we presented above that were applicable to the manually cleaned data we created. As a result, the data distribution was as shown in figure 11 below. The class with the least representation was ii, with 1,022. Fortunately, this was above 1,000 items, therefore we were able to randomly select 1,000 items from each class and split each class into 90/10 train and validation data. See figures 12 and 13 below. We now have a balanced train set of 9,000 items and a balanced validation set of 1,000 items, the sampling and the split was done randomly.
This will be used as our benchmark dataset and split.

Data Augmentation label balance

class	count	%
iv	1,724	13.00%
i	1,708	12.88%
ix	1,498	11.29%
v	1,309	9.87%
vii	1,246	9.39%
vi	1,244	9.38%
viii	1,239	9.34%
x	1,204	9.08%
iii	1,071	8.07%
ii	1,022	7.70%
Total	1,895	

Figure 11

Balanced Train Set

class	Count	%
i	900	10%
ii	900	10%
iii	900	10%
iv	900	10%
ix	900	10%
v	900	10%
vi	900	10%
vii	900	10%
viii	900	10%
x	900	10%
Total	9,000	

Figure 12

Balanced Validation Set

class	count	%
i	100	10%
ii	100	10%
iii	100	10%
iv	100	10%
ix	100	10%
v	100	10%
vi	100	10%
vii	100	10%
viii	100	10%
x	100	10%
Total	1,000	

Figure 13

Model Performance & Error Analysis

Attempt 1 *(Dataset size: 10,000, split: 0.9, 100 epochs)*

Best validation accuracy: 0.867. 100 epochs required 4.2 hours on the virtual machine allocated to us with a GPU. From examining the loss and accuracy plots per epoch (see figure 14 below) there are 2 important takeaways. First, there seems to be no change in the results after approximately 25 epochs, therefore, we decided to run 25 epochs for each following attempt. Second, even though the split was random to the train and validation folders, it appears there's some concept / representation in the data which the model is not successfully learning, this is causing the gap between the accuracy in the train and validation score (0.99 vs 0.86). This results in the model slightly overfitting on the train data.

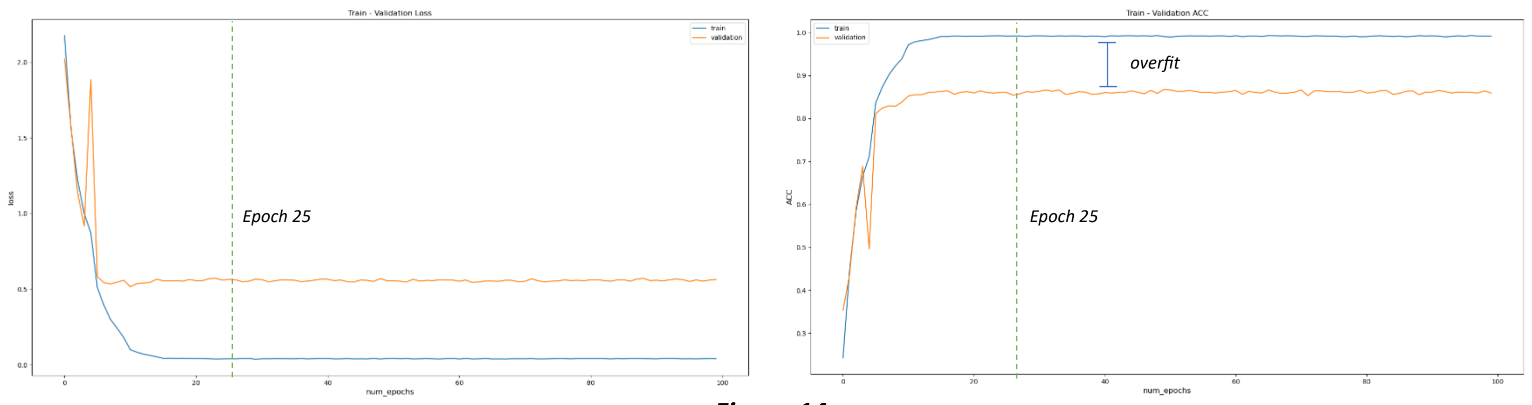


Figure 14

Loss per epoch (left), Accuracy per epoch (right) for the model trained on the data created in attempt 1.

Attempt 2 *(Dataset size: 10,000, split: 0.9, 25 epochs)*

When augmenting the data, we were curious to know if the fill color used in the random erasing influenced the performance of the model. To put this question to the test we only changed the fill color to white from black.

The results were slightly inferior to attempt 1, with a best validation accuracy of 0.847. This is an insignificant change. We were looking to achieve a stronger impact to the models' performance.

Attempt 3 *(Dataset size: 8,850, split: 0.95, 25 epochs)*

In this attempt we changed the horizontal and vertical classes we flipped. In the first attempt we used all the data augmentations to utilize the 10,000-size data limit. Now we wanted to start testing if all the augmentations had an advantageous effect on the training process. We allowed flips only on the classes which the flips resulted in an easily identified number (x was kept, viii vertical and horizontal flips were removed). The smallest class representation allowed us to randomly sample 885 images from each class to keep the data perfectly balanced. Although the train accuracy was able to reach 0.9971, the best validation accuracy was not able to surpass 0.837.

Attempt 4 *(Dataset size: 8,850, split: 0.9, 25 epochs)*

The random rotation augmentation was also creating numeral which are not explicitly defined as a roman numeral at the specific orientation created. So, we tested the affect of limiting the rotation angle randomly chosen to [-60°, -15°] on left rotations and [15°,60°] on right rotations. This attempt yielded a best validation accuracy of 0.888 and the train accuracy for the same epoch was 0.9959 (see figure 15 below).

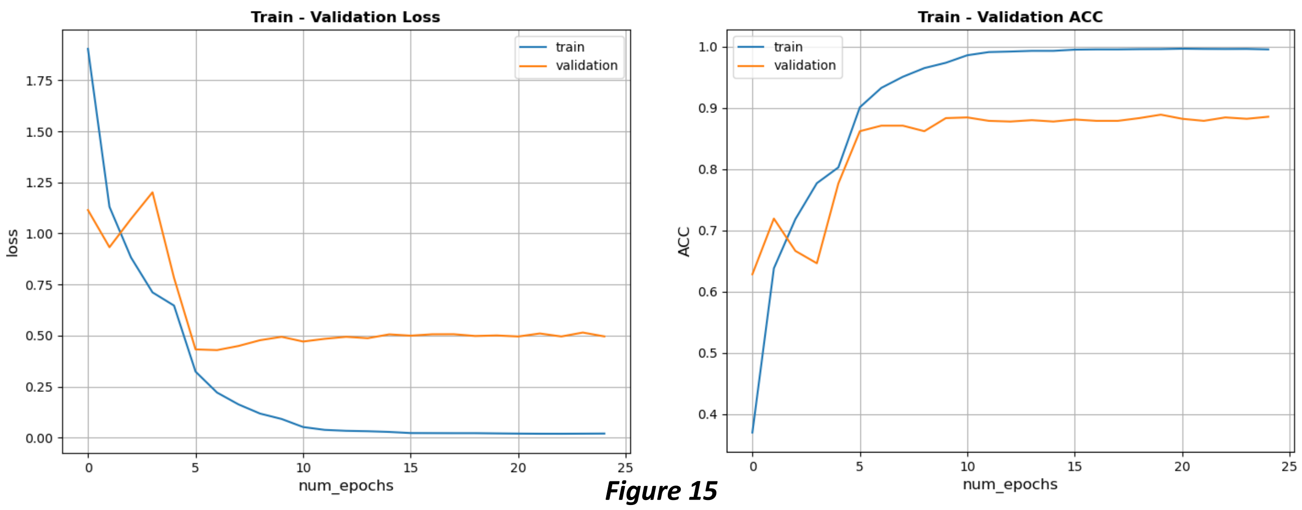


Figure 15

Loss per epoch (left), Accuracy per epoch (right) for the model trained on the data created in attempt 4.

The model was still overfitting, and we were not satisfied with the validation accuracy, so we decided to perform an error analysis on the results of the fourth attempt. Hopefully this would help us better understand the models' performance. Which images is the model having difficulties predicting correctly? Can we augment more of these images into the train data (boost train data with specific images which will improve validation score)?

Error Analysis

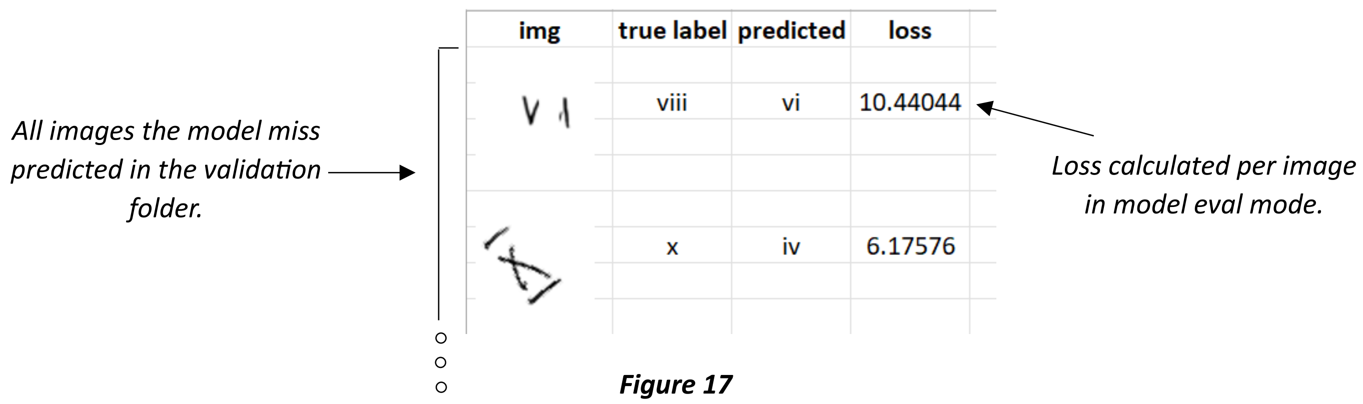
We used 2 tools to perform the error analysis. The first was to look at the distribution of wrongfully predicted images in the validation data. This approach is very straight forward yet might reveal that the model is only wrongfully predicting a single label (see figure 16 to the right).

The second tool we created was much more insightful. We exported the wrongfully predicted images by the model to an Excel sheet sorted descending over the loss of each individual image. This allowed us to see the worst images predicted (by loss) and better understand the reason the model miss labelled the image based on the prediction and the ground truth (see figure 17 below).

class	count	%
iii	18	15.79%
iv	13	11.40%
v	13	11.40%
vii	13	11.40%
x	13	11.40%
ii	12	10.53%
vi	12	10.53%
ix	10	8.77%
i	5	4.39%
viii	5	4.39%
Total	114	

Figure 16

Distribution of the image label's the model from attempt 4 incorrectly predicted.



All images the model miss predicted in the validation folder.
Loss calculated per image in model eval mode.

img	true label	predicted	loss
	viii	vi	10.44044
	x	iv	6.17576

Figure 17

Excel export tool created to analyse the miss labelled prediction. Sorted by descending order by loss on each individual image. See full page in appendix 1.

There are 2 main takeaways we decided to improve on after the error analysis. The first was the difficulty of the model to predict well on the letter X (in the number 10 and 9). From figure 16 above we can see that the wrong predictions over numerals containing a 'X' accounted for 23 errors.

The second takeaway was an issue the random erasing caused in the validation data. When the erased pixels aligned on numeral where it would change the number to the human eye – the label was incorrectly predicted in the validation, even though the model did a great job predicting the "new" label. See example to this phenomenon in figure 18 below.

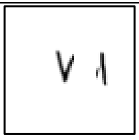
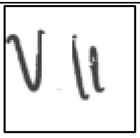
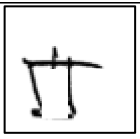
Image	**Ground Truth**	**Predicted**	**Loss**
	vii	vi	10.44
	viii	vii	4.10
	vi	ii	3.47

Figure 18

Random erasing causing the validation data to become miss labelled. In all 3 examples above, the model is predicting the label which really seems to be the correct label. In all examples the random erasing erased a crucial line in the image.

To assist the model overcoming these 2 issues, we added 50 hand drawn images labelled 9 and 50 labelled 10 (we drew 10 of each and used 4 augmentations to achieve 50 images from each label, see figure 19 below). In addition, we made a made a condition in the validation data creation which did not allow images from the randomly erased augmentation.

When further researching the field, we came across a new field under the name of Data Centric AI[7]. The main take away would be to use the augmentations in a more of a quality over quantity manner. Smart augmentations which reserve the main concepts of the data will yield the most improvement in the model result.

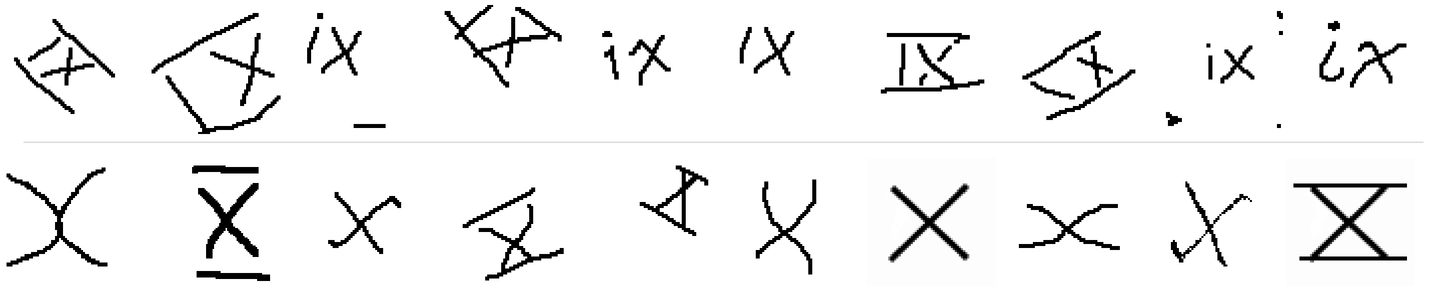


Figure 19

10 hand drawn images for label 9 and 10. Thanks to pixilart.com.

[7] Mazumder, M., Banbury, C., Yao, X., Karlaš, B., Rojas, W. G., Diamos, S., ... & Reddi, V. J. (2022). Dataperf: Benchmarks for data-centric ai development. arXiv preprint arXiv:2207.10062.

Attempt 5 *(Train size: 7,400, Validation size: 700 , 25 epochs)*

Train data size breakdown: 700*0.9*10 (700 per label, 0.9 split, 10 labels) = 6,300 + 100*10 (added 100 from each label augmented with the random erasing into the train data alone) = 7,300 + 50*2 (added 50 images from 10 handwritten and 40 augmented from the 10, for the x & ix labels) = 7,400.

Best validation accuracy 0.897, with a training accuracy of 0.977. The overfitting gap was reduced from 0.1079 in attempt 4 to just 0.08, a 20% improvement. We would have liked to see a better validation accuracy, but as we are allowed to configure the validation data as we wish, we could have made a very small dataset, constructed by data the model predicts well and result in a great accuracy. This is the reason we focused on reducing the overfit gap between the train and validation.

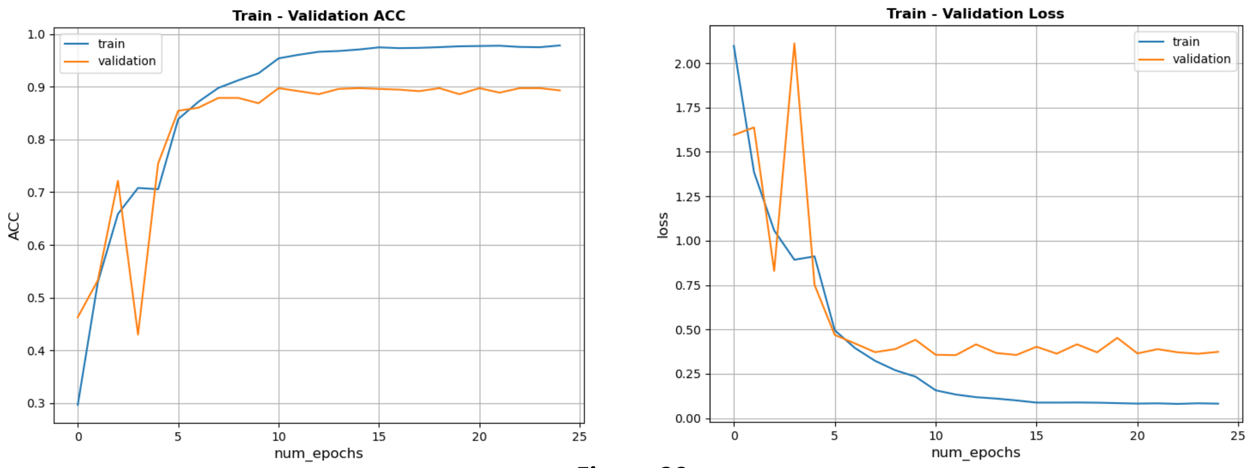


Figure 20

Attempt 5 results (final run)

<u>**Conclusions & Future Research**</u>

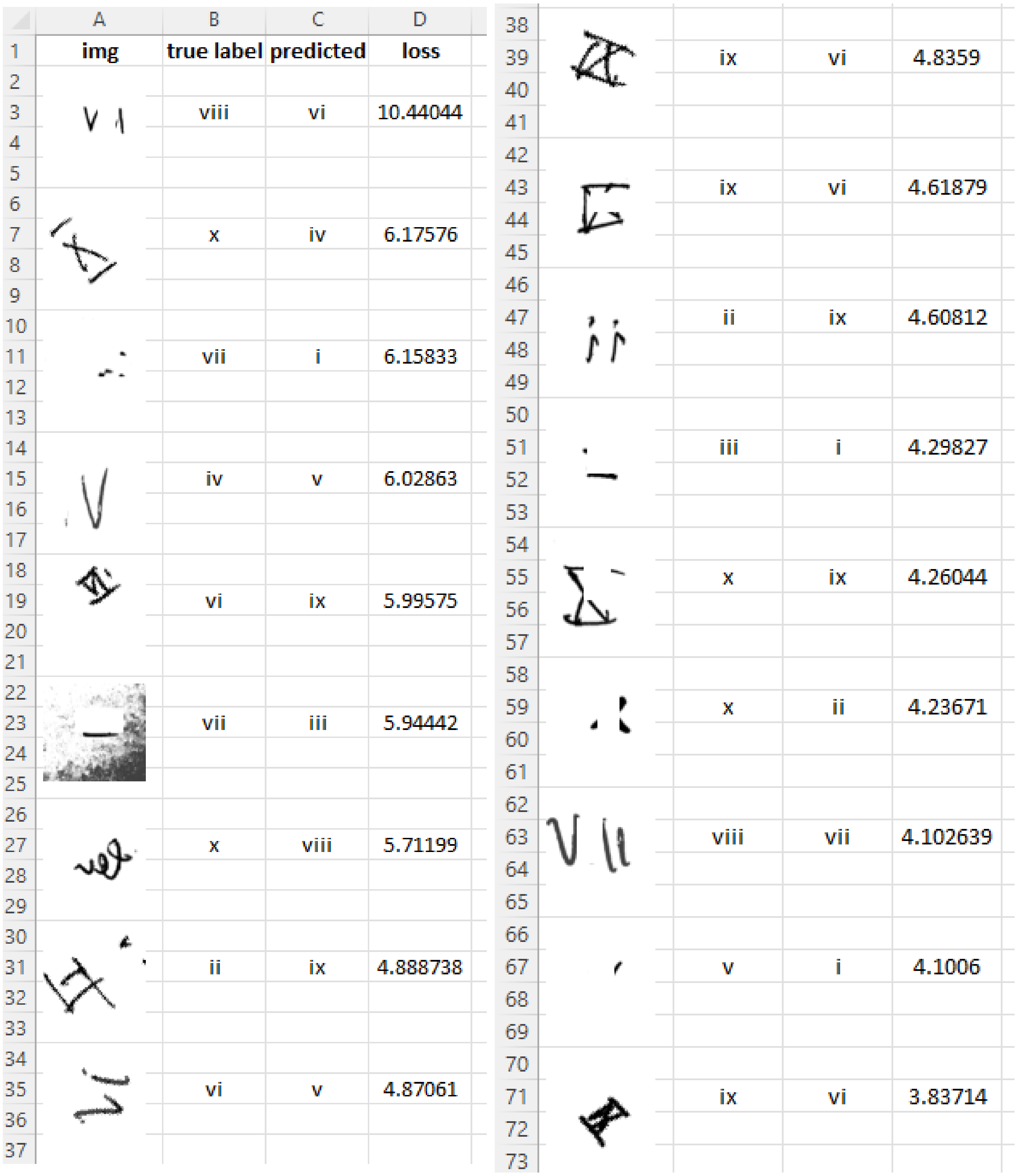
By researching the field of Data Centric machine learning, we were fascinated by the immense impact non model related configuration can have on the final accuracy score of a classification problem. From manually cleaning the data at first, to remove unclear data and relabel miss labelled data. Using several augmentations which make sense in the domain, to enrich the data with more perspectives can make a vast impact on the size of the dataset, allowing models which relay on large datasets to excel. But with all the enhancing in mind, it is important to remember that the quality of the data is the most important, adding garbage data, will just set us back to step one where we need to manually clean the data again.
When done right, error analysis will shed light on the weaknesses of the model, and by extracting meaningful insights, followed by finding a solution to strengthen these weaknesses, one can improve the model even more. We used the insights found from our error analysis to draw roman numerals for 2 classes and limit a certain augmentation from existing in the validation data.

Many more intelligent methods are already existing in the field or may be waiting to be found. To further research this assignment, we would propose to perform an additional error analysis to extract actionable insights to improve the model performance. Furthermore, we would like to see the impact of each augmentation by itself on the results. Learning which augmentation is the best for each task will help us understand how to use them to their fullest. In addition, augmenting synthetic data using generative models is a topic we came across which we would like to further consider in this type of task.

Thank you for giving us the privilege to learn and experience these subjects' hands on.

Appendix 1

Full page example of the Excel export error analysis tool



img	true label	predicted	loss
	viii	vi	10.44044
	x	iv	6.17576
	vii	i	6.15833
	iv	v	6.02863
	vi	ix	5.99575
	vii	iii	5.94442
	x	viii	5.71199
	ii	ix	4.888738
	vi	v	4.87061
	ix	vi	4.8359
	ix	vi	4.61879
	ii	ix	4.60812
	iii	i	4.29827
	x	ix	4.26044
	x	ii	4.23671
	viii	vii	4.102639
	v	i	4.1006
	ix	vi	3.83714