Using Synthetic Data Safely in Classification

Jean Nonnemaker, Henry S. Baird

Computer Science & Engineering Dept
Lehigh University
19 Memorial Dr West
Bethlehem, PA 18017 USA

E-mail: `jen2@lehigh.edu`

ABSTRACT

When is it safe to use synthetic data in supervised classification? Trainable classifier technologies require large representative training sets consisting of samples labeled with their true class. Acquiring such training sets is difficult and costly. One way to alleviate this problem is to enlarge training sets by generating artificial, synthetic samples. Of course this immediately raises many questions, perhaps the first being "Why should we trust artificially generated data to be an accurate representative of the real distributions?" Other questions include "When will training on synthetic data work as well as - or better than training on real data ?".

We distinguish between sample space (the set of real samples), parameter space (all samples that can be generated synthetically), and finally, feature space (the set of samples in terms of finite numerical values). In this paper, we discuss a series of experiments, in which we produced synthetic data in parameter space, that is, by convex interpolation among the generating parameters for samples and showed we could amplify real data to produce a classifier that is as accurate as a classifier trained on real data. Specifically, we have explored the feasibility of varying the generating parameters for Knuth's Metafont system to see if previously unseen fonts could also be recognized.

Keywords: interpolation parameter

1. INTRODUCTION

It has been widely accepted in pattern recognition research that the classifier trained on the most data wins [1], [2], [3]. Of course, putting this strategy into practice can be troublesome, since large training sets are expensive or impossible to obtain, and may not be representative - or sets may be imbalanced, where one class is represented by too few samples, and others have too many. This is in the context of supervised classification in which classifiers are designed fully automatically by reading in files of labeled training samples so that the classifier can learn from example patterns.

Good results in pattern recognition have been achieved by the use of supervised classifiers such as nearest neighbors algorithms; however these results require large amounts of training data together with carefully labeled g*round truth* (the true classes of each sample) which can be even more expensive to provide than the data itself. Often the acquisition of such data becomes a problem, as much time and labor must be spent to locate existing training data, or alternatively, to classify new training data properly.

We believe that one way out of this impasse is to *amplify* the training data: that is to increase it artificially by generating more. We call such generated data *synthetic* in contrast to data collected in the field, *real* data. We conducted experiments to explore the validity and uses of such synthetically generated data.

We made the assumption, as is done classically in Bayesian decision theory, that we had a classifier technology which could be trained on a training set and tested on a separate set. Typically the training samples arise in a natural *sample space.* In our domain, the sample space was the set of all real images.

Usually the first step in crafting a classifier is to choose numerical features (say d of them) that can be algorithmically extracted so each sample is represented by a set of features: that is, as a point in d-dimensional space. In this sense, data also live in *feature space.*

However, since we know how images are generated, then the parameters that control the generation process are yet another way of describing the data – so synthetically generated data can be said to live in *parameter space* also.

It may be informative to give an example of a method of generating artificial data which 'lives' in each space. For example, in sample space, one might generate synthetic data by taking several real images, cutting them up, pasting them together in a random fashion, and so, making a collage of them.

To generate data synthetically in feature space, one might extract features for several samples, giving several points in feature space, then interpolate between the points to generate a new point. An example of feature space interpolation might be recording the width and height of characters to be used as features and interpolating numeric values between them.

Samples that are generated in parameter space can be synthesized by varying the generating parameters. In the case of document images, these generating parameters are among the following; type, size, image degradations such as blur, threshold, additive noise, etc. and layout dimensions such as gutter width, line spacing. We can say that pairs of real images span ranges in parameter space and thus allow the generation of synthetic training data densely within that range. Figure 1 depicts the use of parameters to create new samples of an image by acting in parameter space.

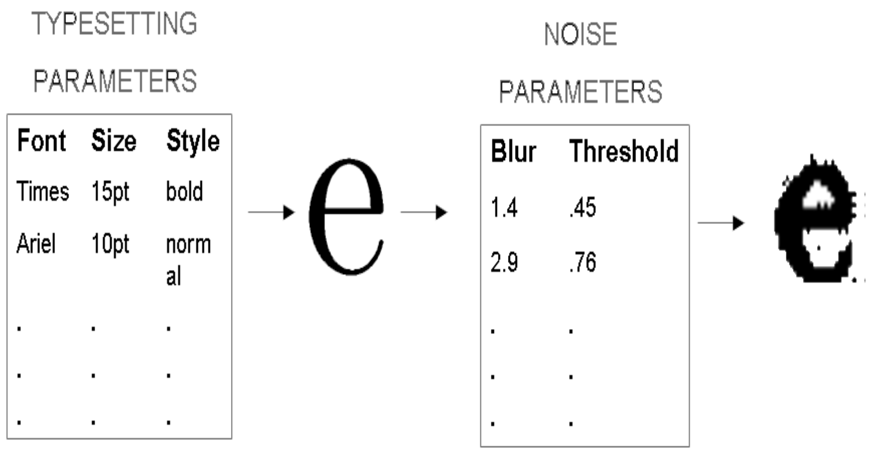

Figure 1. Creating New Samples in Parameter Space

We explored the relationship between parameter space, sample space and feature space which can be thought of as follows:

$$\text{parameter space} \rightarrow \text{sample space} \rightarrow \text{feature space}$$

2. PREVIOUS WORK

We are not the first to be interested in interpolated data, however, to the best of our knowledge we are the first to be interested in generating data by interpolating in parameter space.

In an effort to determine if performance of a recognition system is affected by the size and quality of the training data, Varga, T. and Bunke, H. [4] , [3] examined under what circumstances a larger training set improves the accuracy of handwriting recognition systems. They show how synthetic generation of new training samples can be achieved, e.g. through perturbation of, or interpolation between the original samples. They tested the use of continuous nonlinear functions that control a class of familiar geometrical transformations applied on an existing handwritten text line.

Chawla et al. [5] propose that undersampling of the majority (normal) is a good means of increasing the sensitivity of a classifier to the minority class. The authors use a combination of over-sampling the minority class and under-sampling the majority class to achieve better classifier performance.

Ho T. K. and Baird H. S. [1] present three studies that rely on synthetic data generated pseudo-randomly in accordance with an explicit stochastic model of document image degradations.

Sun J. et al. [6] generate synthetic degraded character images based on a simplified video degradation model, common in facial recognition systems using a new feature extraction method based on a dual eigenspace decomposition which is used along with a degradation model

3. METAFONT INTERPOLATION

For our experiments, We generated training samples in parameter space and tested if a set of images generated from interpolated typeface styles performed as well as a same-size set of images generated from original metafont type styles on previously seen data. This tested the safety of our algorithm. Next we tested if our set of interpolated training images performed better on previously unseen data. This tested the effectiveness of our algorithm.

A Metafont [7] program was created to interpolate between two or more fonts from the Computer Modern Roman families of fonts. We will discuss the case of a two-font interpolation. First, the parameter values were listed for each font and differences between the same parameter in each font were calculated. Since we were creating nine interpolations, the difference was divided by 10 and one tenth of this value was added to or subtracted from the first font until we arrived at the second font. Boolean values were either True or False. One example of a Boolean parameter would be the Serifs parameter (Serifs = True or Serifs = False). If both fonts had True, the interpolations were also True. If one was False and one was True, the first half of the interpolations was set to False, while the second half were set to True and vice versa. A Metafont program was used to create the fonts and once they were created, LaTeX was used to create an image of the letters c and e (or i and j) as a PNG file.

4. TEST SET CREATION

The IDMGEN [8] program created by Henry Baird as an image defect model generator was used to generate the images of a letter in a selected font. The program was given an image of an isolated character in PNG format, from which it generates a series of the same character in text-line format in an ascii file. The text characters are pseudo-randomly distorted using a quantitative model of the printing and imaging process.

A parameter for seed (-S) was input at the start of each letter generation run to set the pseudo-random number generator and produce an image of a character. Input parameters were used to generate scalar random variables with defined distributions which were applied to the generated images. Three of the parameters we used to introduce variation in the images were blurring, a value which represents the point-spread (or impulse response) function of the combined printing and imaging process; threshold, a parameter which models binarization as a test on each pixel; and sensitivity, a parameter which randomizes each pixel's photo-receptor sensitivity in two stages. The interested reader may find the details of these parameters in the IDMGEN documentation referenced above.

The end result was a series of images with variations in blur, brightness and intensity, some very readable, and some greatly distorted. Each of the experiments within a series of fonts used different values of these parameters to control distortion. Samples of images generated by our process are shown in figure 2.

5. OUR INTERPOLATIONS

Interpolations between CMR (Computer Modern Roman) and CMFF (Computer Modern Funny Font) were constructed by smoothly interpolating the parameters used in creating both CMR and CMFF. Figure 3 shows an example showing the letter e in CMR, nine interpolations, and finally CMFF.

We also tested a similar interpolation between CMR and CMSS (Computer Modern Sans Serif) on the letters c and e as well as on the letters i and j. Lastly we tested a three-way interpolation between CMR, CMFF,

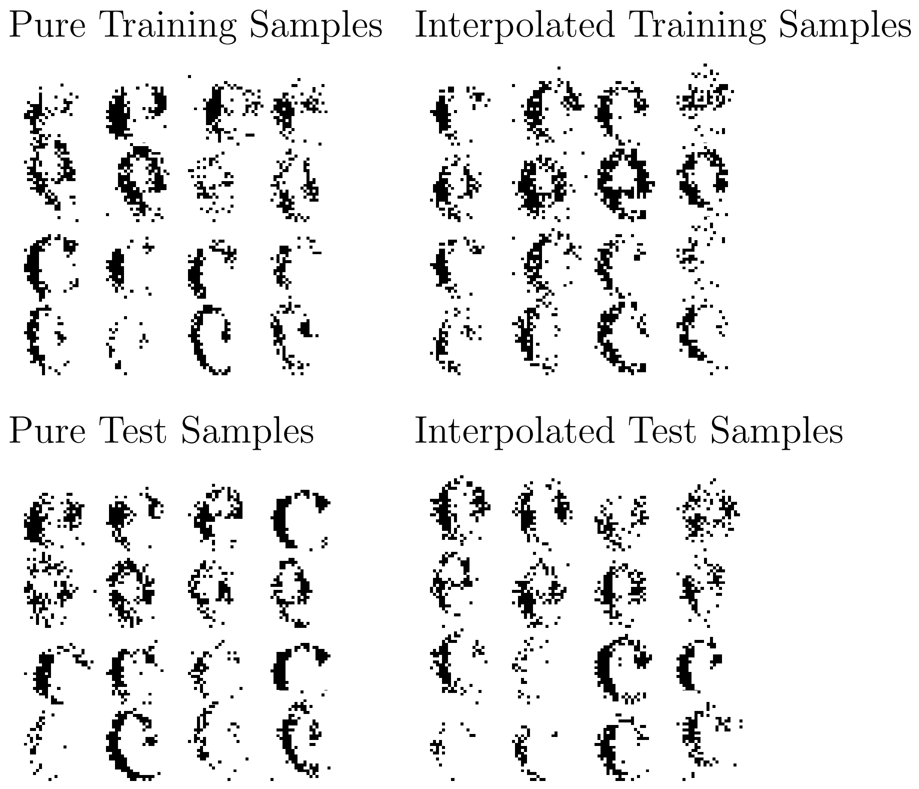

Figure 2. Samples of Generated Images

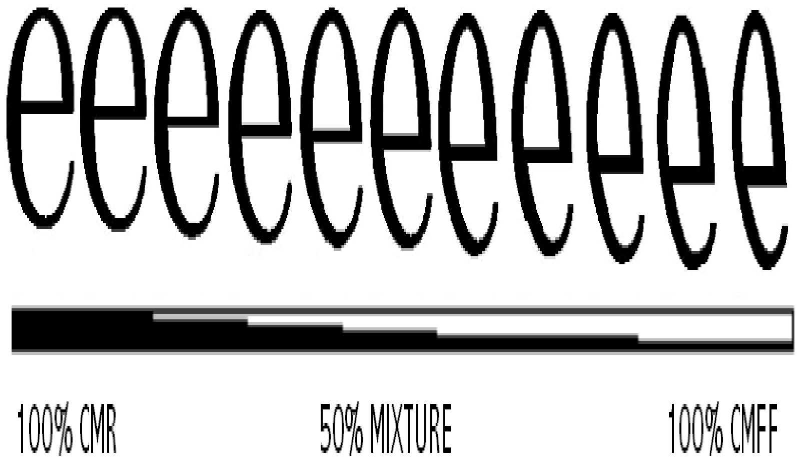

Figure 3. Example of CMR-CMFF Interpolation

and CMSSI (Computer Modern Sans Serif Italic) fonts on the letters c and e. An example of the three-way interpolation is shown in figure 4.

Each set of experiments had tests involving differing degrees of blur, noise and variance. Each test compared the performance of pure versus mixed (interpolated) training sets on both pure and mixed test sets. Some of the tests in each set took test data from within the entire range of interpolated samples, while some took all the interpolated data from the midpoint between the two fonts.

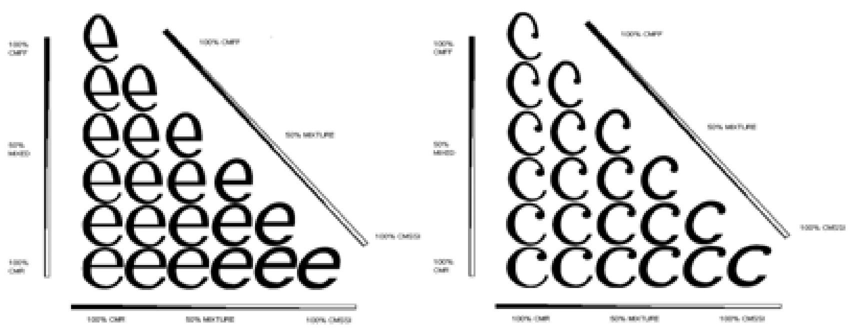

Figure 4. Example of Three-Way Interpolation

6. EXPERIMENTAL DESIGN

Seven experiments were performed on CMR-CMSS e and c interpolations followed by CMR-CMFF e and c, CMR-CMSS i and j, and 3-way CMR-CMFF-CMSSI.

We wanted to test whether an interpolated training set is safe and effective by performing a series of tests. In describing our experiments the first typefaces will be referred to as *pure*. These are well known, standard typefaces created from original Knuth's Metafont type styles which are widely used. Existing classifiers have been trained on them. The next set of typefaces we call *interpolated*. These typefaces have been created by interpolating between the parameters used to create the pure typefaces. They may never have been used but are legible and could be used. First we tested whether a classifier trained on a set of images generated from interpolated typeface styles performed as well as a classifier trained on a same-size set of images generated from pure type styles when tested on pure test samples. This tested the safety of our algorithm. Next we wanted to see whether our classifier trained on the set of interpolated training images performed better than the classifier trained on pure styles when tested on interpolated samples. This tested the efficacy of our algorithm.

The experiments are described below.

6.1 Pure training and test sets

Eight hundred samples of the letter c and e were generated using the idmgen program. For the first group of experiments, the ideal prototype of each of the training samples was a machine print type form of these letters in Knuth's CMR (Computer Modern Roman) typeface. (800 samples total)
Additionally, eight hundred samples of the letter c and e were generated using the idmgen program and a machine print type form of the letter e and the letter c in Knuth's CMSS (Computer Modern Sans Serif) typeface as the ideal prototype (800 samples total). This provided a training set of 1600 samples, equally divided between CMR and CMSS. We call this training set A.

A kNN Classifier was trained on the CMR and CMSS training sets. The classifier was then tested on a test set consisting of 200 CMR samples and 200 CMSS samples, similarly generated, which we call test set A. Rates of accuracy for the 400 test samples were recorded.

6.2 Interpolated training set:

Next, 1600 mostly interpolated samples of the letter c and the letter e were generated using the idmgen program. The ideal prototype of each of the ten sets was as follows; a machine print type form of the letter e or the letter c in Knuth's CMR (Computer Modern Roman), CMSS or one of our interpolated typefaces. This was called training set B.

The classifier was trained on the interpolated samples (training set B) and then tested the pure test samples (test set A). This tested if the classifier which has been trained on interpolated data performed equally well on samples in previously known fonts as a classifier trained on pure data.

6.3 Interpolated test set:

A test set consisting of 400 mostly interpolated samples of the letter c and the letter e were generated using the idmgen program using CMR and CMSS typefaces and nine interpolations between them. This became Test Set B.

We then trained a classifier on the known font type samples (Training Set A) and tested it on the interpolated samples (Test Set B) to test how well it performed when it was tested on previously unseen fonts.

Finally, the classifier trained on the interpolated samples (Training Set B) was tested on the interpolated test set (Test Set B) to see how well the classifier performed when it had been trained on interpolated data and tested on previously unseen data.

Two hypotheses were proposed and tested using the χ^2 statistic

6.4 Hypothesis 1:

AB is trained on only pure data and tested on mostly interpolated data. BB is trained and tested on interpolated data. We expect that BB will perform significantly better than AB. Therefore our null hypothesis is that AB will perform better than BB. This part of the experiment speaks to the performance or strength of our algorithm.

The areas of interest in this part of our experiment are the differences between AB and BB. If the null hypothesis is rejected then our classifier trained on mostly interpolated data has performed better than the classifier trained on only pure data and we can say that our interpolated classifier is better at classifying interpolated data.

6.5 Hypothesis 2:

AA is trained and tested on pure data. BA is trained on mostly interpolated data and tested on pure data. We would hope that BA would not perform significantly worse than AA. If this is the case, we have shown that in our experiment, training on interpolated data does not “hurt” the classifier in the identification of pure data. In this way we are testing if our algorithm is “safe”. Our null hypothesis is that BA and AA have the same accuracy and we would hope that the null hypothesis holds.

The areas of interest in this part of the experiment is the difference between AA and BA. If our null hypothesis is not rejected then we can say that training on interpolated data does not appear to “hurt” the classifier in the identification of pure data. A graphical representation of our experimental design is shown in figure 5.

7. RESULTS

7.1 CMR-CMSS tests

Our first set of experiments was performed on interpolations between the Computer Modern Roman (CMR), a serifed font, and the Computer Modern Sans Serif (CMSS), a sans-serif font, both from the Computer Modern (CM) family of fonts. These fonts were fairly similar to one other and results of testing on a set of pure test samples proved that the classifier trained on interpolated training samples performed as well as the one trained on pure samples. Table 1 summarizes the results of this set of experiments.

The classifier trained on the interpolated samples performed better than the one trained on pure samples in the two instances when test samples were taken from the full range of interpolated samples, were greatly blurred, and had little variance from the midpoint of the blurring parameter. The interpolated classifier also performed better when the test samples were all taken from the midpoint between CMR and CMSS and were only slightly blurred with little variance.

In the first two tests, the image quality was either normal or only slightly blurred. Both classifiers were able to recognize a large fraction of the characters and both performed equally well on the interpolated samples,

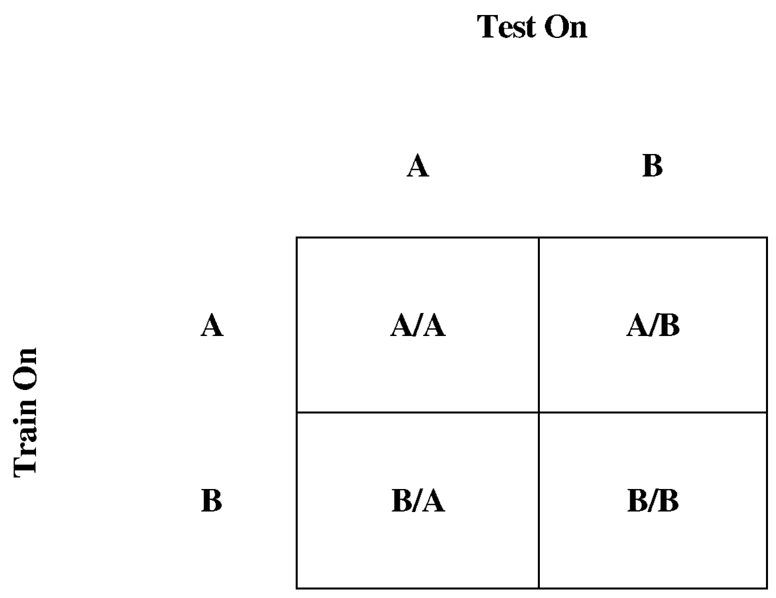

Figure 5. Design of Experiment: A = previously known data (CMR and CMSS fonts) B = previously unseen data (interpolated fonts)

which were taken from the full range of interpolated images. As blurring and variance increased in tests three through five, both classifiers started to perform badly. Only when the variance was decreased did the interpolated classifier pull ahead.

For the last two tests the interpolated samples were taken entirely from the midpoint between the two fonts, and thus were equally dissimilar from each font. When the test samples were greatly blurred, neither classifier was able to recognize many of the images, however when the images became less blurred, the interpolated classifier did better.

Table 1. CMR-CMSS e and c Results

		Hypothesis 1				Hy pothesis 2			
		Errors		Statistic		Errors		Statistic	
IMAGE QUALITY	TEST SET	AB	BB	χ^2	Rej	AA	BA	χ^2	Rej
normal	full	0	1	1.00	no	0	0	0.00	no
slightly blurred	full	0	0	0.00	no	0	1	1.00	no
greatly blurred, high variance	full	62	72	.88	no	74	69	.20	no
greatly blurred, some variance	full	60	42	3.62	no	44	43	.20	no
greatly blurred, little variance	full	64	39	6.90	yes	41	45	.20	no
greatly blurred, little variance	mid	79	71	.50	no	52	40	1.76	no
slightly blurred, little variance	mid	23	10	5.20	yes	6	7	1.40	no

7.2 CMR-CMFF tests

We next thought it would be interesting to test our classifier on two fonts which were less similar. While they were once again taken from the Computer Modern family, the fonts chosen were much different from each other. We chose Computer Modern Roman (CMR) and Computer Modern Funny Font (CMFF) for this set of experiments. The same seven experiments were performed as before. Table 2 presents a summary of the results.

Once again, both classifiers performed equally when tested on the pure (CMR and CMFF) test sets. However, this time, the classifier trained on the interpolated data performed better when tested on the slightly blurred test sets (both full range and midpoint), as well as the greatly blurred-little variance test set (both ranges).

Table 2. CMR-CMFF e and c Results

		Hypothesis 1				Hy pothesis 2			
		Errors		Statistic		Errors		Statistic	
Image Quality	Range	AB	BB	χ^2	Rej	AA	BA	χ^2	Rej
normal	full	23	4	10.71	yes	0	0	0.00	no
slightly blurred	full	18	3	7.93	yes	0	2	0.00	no
greatly blurred, high variance	full	102	106	.80	no	91	83	.46	no
greatly blurred, some variance	full	84	83	.10	no	65	73	.54	no
greatly blurred, little variance	full	81	57	5.20	yes	58	64	.32	no
greatly blurred, little variance	mid	74	56	2.96	no	70	53	2.74	no
slightly blurred, little variance	mid	49	16	17.97	yes	8	10	.22	no

7.3 CMF-CMFF i and j tests

The next set of tests was performed on the two fonts, CMR and CMFF with the letters i and j. Once again, both classifiers performed equally when tested on the pure test sets. In every case, the interpolated classifier was at least as good as the pure classifier. However, when tested on the interpolated test images, there was also no difference between the recognition of the images. We think that there was either enough difference between the i's and j's that the classifiers were able to correctly identify them, or perhaps the i's and j's were similar enough in the fonts chosen that the interpolated samples did not provide enough variety.

Table 3. CMR-CMFF i and j Results

		Hypothesis 1				Hy pothesis 2			
		Errors		Statistic		Errors		Statistic	
Image Quality	Range	AB	BB	χ^2	Rej	AA	BA	χ^2	Rej
normal	full	0	0	0.00	no	0	1	1.00	no
slightly blurred	full	2	3	2.00	no	1	1	.33	no
greatly blurred, high variance	full	71	65	.3	no	83	96	1.20	no
greatly blurred, some variance	full	72	64	.54	no	94	93	.10	no
greatly blurred, little variance	full	58	53	.26	no	85	87	.20	no
greatly blurred, little variance	mid	59	44	2.49	no	86	97	.85	no
slightly blurred, little variance	mid	10	10	0.0	no	29	37	1.40	no

7.4 CMR-CMFF-CMSSI tests

The last set of test was even more interesting and challenging. In these experiments we interpolated among three fonts, Computer Modern Roman (CMR), Computer Modern Funny Font (CMFF) and a completely different one, Computer Modern Sans Serif Italics (CMSSI). While these fonts are all from the Computer Modern family, they vary as to serifs, slant, thickness and many other characteristics. Table 4 shows the results of this set of experiments.

We found once again that both classifiers performed equally when tested on the pure samples. That is to say, there was no loss of accuracy when the classifier trained on interpolated samples was tested on the pure test set.

Interestingly enough, the classifier trained on the interpolated samples performed better when tested on the interpolated samples in five out of the seven cases. It performed better when tested on the full range of samples in every case except the one in which the images were greatly blurred with high variance. Both classifiers did badly on this one. The interpolated classifier, however, performed better when the images were greatly blurred and taken from the midpoint range only. Both classifiers performed equally well when tested on the slightly blurred images taken from the midpoint.

It is noteworthy that this last result differs from that of the CMR-CMFF test on the slightly blurred midpoint test samples. In that test, the classifier trained on the pure images could not recognize the midpoint samples as well as the one trained on the interpolated images. Why was the CMR-CMFF-CMSSI classifier able to recognize the interpolated midpoint images better than the CMR-CMFF one? Could it be that the addition of the third font, even though it is a pure font, makes the classifier better able to recognize a previously unseen font? And if that is so, would the addition of many more fonts, both pure and interpolated make it even better?

Table 4. CMR-CMFF-CMSSI e and c Results

		Hypothesis 1				Hy pothesis 2			
		Errors		Statistic		Errors		Statistic	
Image Quality	Range	AB	BB	χ^2	Rej	AA	BA	χ^2	Rej
normal	full	42	3	31.57	yes	0	0	0.00	no
slightly blurred	full	18	0	14.54	yes	0	0	0.00	no
greatly blurred, high variance	full	111	111	0.0	no	91	92	.10	no
greatly blurred, some variance	full	109	48	29.34	yes	64	72	0.54	no
greatly blurred, little variance	full	86	55	8.21	yes	54	70	2.42	no
greatly blurred, little variance	mid	57	22	17.20	yes	67	52	2.19	no
slightly blurred, little variance	mid	9	4	.72	no	7	15	2.96	no

8. CONCLUSIONS

We have found that training on interpolated data is for the most part safe, that is to say never produced more errors, when tested on the pure samples. Furthermore, the classifier trained on interpolated data often but not always improved (about one third of the time) classification when tested on previously unseen interpolated samples.

I refer the reader to table 5 for a concise graphical summary of our results.

In our systematic family of tests, we have demonstrated that the use of interpolated data in the training sets has never worsened the results, and has frequently improved the results. The improvement is greater when the fonts being interpolated are most different from each other and when the images are greatly blurred with little variance. The three-way interpolation tests showed the most number of significant improvements for the interpolated training sets. Note that the results shown are for the two easily confused pairs of characters e/c and i/j.

Our research has brought up many interesting ideas. No one in the world has yet used typeface interpolations to generate synthetic data and we are pleased to be the first and hope there is a useful application for our research results. We would offer up the following ideas as food for thought.

- Legibility is convex in parameter space, that is to say that any font interpolated between two legible fonts is still legible.
- Legibility is convex both in typographic space and in image quality space.

Table 5. Overall Results

CHARS	FONT STYLES	IMAGE QUALITY	TEST SET	SAFE?	BETTER?
e and c	CMR-CMSS	normal	full range	yes	–
		slightly blurred	full range	yes	–
		greatly blurred, high variance	full range	yes	–
		greatly blurred, some variance	full range	yes	–
		greatly blurred, little variance	full range	yes	yes
		greatly blurred, little variance	midpoint	yes	–
		slightly blurred, little variance	midpoint	yes	yes
e and c	CMR-CMFF	normal	full range	yes	–
		slightly blurred	full range	yes	yes
		greatly blurred, high variance	full range	yes	–
		greatly blurred, some variance	full range	yes	–
		greatly blurred, little variance	full range	yes	yes
		greatly blurred, little variance	midpoint	yes	yes
		slightly blurred, little variance	midpoint	yes	yes
e and c	CMR-CMFF-CMSSI	normal	full range	yes	yes
		slightly blurred	full range	yes	yes
		greatly blurred, high variance	full range	yes	–
		greatly blurred, some variance	full range	yes	yes
		greatly blurred, little variance	full range	yes	yes
		greatly blurred, little variance	midpoint	yes	yes
		slightly blurred, little variance	midpoint	yes	–
i and j	CMRCMFF	normal	full range	yes	–
		slightly blurred	full range	yes	–
		greatly blurred, high variance	full range	yes	–
		greatly blurred, some variance	full range	yes	–
		greatly blurred, little variance	full range	yes	–
		greatly blurred, little variance	midpoint	yes	–
		slightly blurred, little variance	midpoint	yes	–

- Conversely, can we say that any interpolated font between two illegible fonts is illegible? We have not proved that, but it is interesting to consider.

Based on our results we would offer the reader this advice — engineers who wish to build classifiers to test all legible typefaces can use synthetically generated data. This will not harm, and occasionally will improve the classifier.

REFERENCES

[1] T. K Ho and H. S. Baird. Large-scale simulation studies in image pattern recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(10):1067–1079, 1997.

[2] Simard, P., Le Cun, Y., Denker, J., and Victorri, B. Transformation invariance in pattern recognition - tangent distance and tangent propagation. In G. B. Orr Miller and K-R, editors, *Neural Networks: Tricks of the Trade*, volume Chapter 12. Springer, 1998.

[3] T. Varga and Bunke H. Comparing natural and synthetic training data for off-line cursive handwriting recognition. In IEEE, editor, *9th International Workshop on Frontiers in Handwriting Recognition*, 2004.

[4] T. Varga and Bunke H. Effects of training set expansion in handwriting recognition using synthetic data. In *11th Conf. of the International Graphonomics Society*, pages 200–203, Scottsdale, Arizona, USA, 2003.

[5] N. et al Chawla. Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, 2002.

[6] J. et al Sun. Low resolution character recognition by dual eigenspace and synthetic degraded patterns. In ACM, editor, *HDP '04*, pages 15–22, Washington, DC, USA, 2004. ACM.

[7] Knuth, D. *Computer Modern Type Faces*. Adedison Wesley Publishing Company, 1986.

[8] Baird, H. Document Image Defect Models. In H. S. Baird, H. Bunke, K. Yamamoto, editors, *Structured Document Image Analysis*, Springer-Verlag, 1992.