Understanding data augmentation for classification: when to warp?

Sebastien C. Wong
Defence Science and Technology
Edinburgh
SA, Australia
Email: sebastien.wong@dsto.defence.gov.au

Adam Gatt
Australian Defence Force
Edinburgh
SA, Australia

Victor Stamatescu
and Mark D. McDonnell
Computational Learning Systems Laboratory
Information Technology and Mathematical Sciences
University of South Australia
Mawson Lakes
SA, Australia

***Abstract*—In this paper we investigate the benefit of augmenting data with synthetically created samples when training a machine learning classifier. Two approaches for creating additional training samples are *data warping*, which generates additional samples through transformations applied in the data-space, and *synthetic over-sampling*, which creates additional samples in feature-space. We experimentally evaluate the benefits of data augmentation for a convolutional backpropagation-trained neural network, a convolutional support vector machine and a convolutional extreme learning machine classifier, using the standard MNIST handwritten digit dataset. We found that while it is possible to perform generic augmentation in feature-space, if plausible transforms for the data are known then augmentation in data-space provides a greater benefit for improving performance and reducing overfitting.**

I. Introduction

The competition winning classifiers described in [1], [2], [3], [4] all utilized techniques to artificially increase the number of training examples. Previous research to systematically understand the benefits and limitations of data augmentation demonstrate that data augmentation can act as a regularizer in preventing overfitting in neural networks [5], [6] and improve performance in imbalanced class problems [7]. In this paper we empirically investigate the benefits and limitations of data augmentation on three machine learning classifiers, and attempt to answer the question of *how and when to apply data augmentation?*

To enable comparison of data augmentation for a convolutional neural network (CNN), a convolutional support vector machine (CSVM) [8], and a convolutional extreme learning machine (CELM) [9] classifier, we use the well known MNIST handwritten digit dataset [10].

II. Previous Work

This section provides a brief review of previous works that have used augmented training data to improve classifier performance.

The concept of *data warping* (for neural networks) can possibly be attributed to [11] in creating a model of character distortions that could occur in the printing (or handwriting) process, the model described deformations and defects needed to generate synthetic character examples. The term *warping* was coined by [12], in the context of producing randomly warped stroke data for handwriting character classification. The warped character stroke data was used to balance the amount of training examples for each character class in order to reduce the bias in the classifier to favor more frequently presented training examples. This approach was extended by [5] to improve the performance of a standard backpropagation-trained neural network, and achieve record performance (in 2003) of 0.4% error rate on the MNIST handwritten digit database using a convolutional neural network. The warped training data was created by applying both affine transformations (translation, shearing, rotation) and elastic distortions to images of existing character examples. The elastic distortions were generated using a scaled normalized random displacement field in image space, and are stated to correspond to uncontrolled oscillations of the hand muscles. This approach was further extended to create a simple deep neural network that was only trained on warped data [6], where new warped training data was created for each epoch of the backpropagation algorithm. This allowed for a neural network with a large number of parameters that overcame the issues of over-fitting and diminished convergence by the backpropagation algorithm during training, which achieved a state of the art (in 2010) error rate of 0.35% on the MNIST database.

The problem of class imbalance, where real-world datasets often only contain a small percentage of "interesting" or target class examples, was addressed by the use of a Synthetic Minority Over-Sampling Technique (SMOTE) [7]. In SMOTE the synthetic examples are created in feature-space from randomly selected pairs of real world feature-examples from the minority class. The advantage of *synthetic over-sampling* compared to the *data warping* approach, is that synthetic examples are created in feature-space, and thus the SMOTE algorithm is application independent.

Both the SMOTE and data warping approaches modify real world examples to create augmented data. An alternative would be to generate synthetic imagery. This approach was investigated by [13], in creating synthetic images of building roofs to augment real image data. By visualizing the data in feature-space they found that the distributions of artificially generated images and real images (that were the basis of the

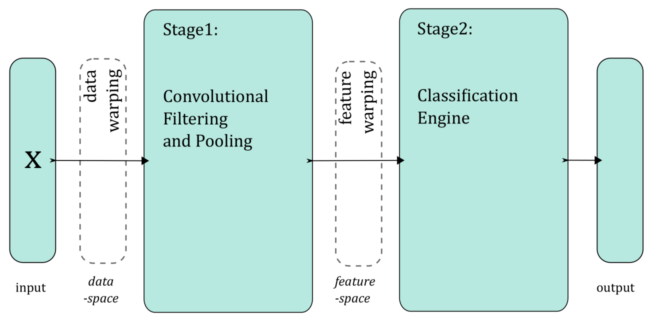

Fig. 1. Experimental architecture. Stage 1 was kept the same for all experiments, while stage 2 consisted of either a backpropagation-trained neural network, support vector machine or extreme learning machine classifier.

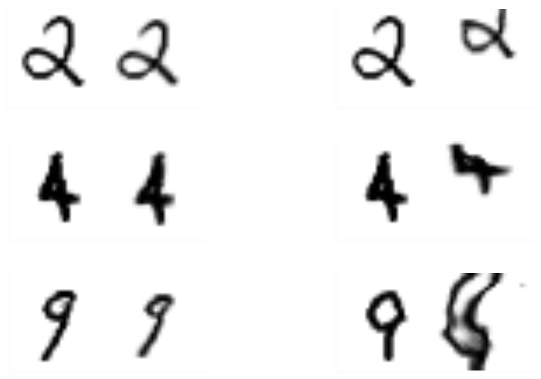

Fig. 2. Data warping using elastic deformations [5]. Original MNIST digits compared to warped digits for $\alpha = 1.2$ (left) and $\alpha = 8$ pixels (right).

synthetic examples) were displaced. The authors termed this problem the *synthetic gap*. To overcome this issue they trained a sparse auto-encoder simultaneously with real and synthetic images, such that the learnt features minimized the synthetic gap.

The issue of the difference between real and synthetic examples is discussed by [14], in using a classifier that was trained with augmented data only. The paper argues that to maximize the probability of correct classification any test sample should undergo the same warping process as the training samples.

III. Method and Data

The literature suggests that it is possible to augment training data in *data-space* or *feature-space*. To understand the benefits of each approach the experimental architecture (for each machine learning classifier) was structured into two stages, such that there was a clear distinction between data-space and feature-space, illustrated in Figure 1. Each system under evaluation had the same convolutional and pooling layer that was used to generate features for the classification engine. The weights for this convolutional and pooling layer were kept fixed to ensure that the same features were used for all experiments with the same dataset.

A. Datasets

The MNIST database [10] was used for the experiments. This database consists of a training set of 60,000 labeled 28 by 28 pixel grayscale images of handwritten digits and a test set of 10,000 labelled images. There are 10 classes, the digits 0 to 9.

An important question to be answered is how does using synthetic data augmentation compare with collecting more real data? To answer this question a set of baseline performance figures for each of the classifiers were created by reducing the amount of data available in training. The quantity of samples from each class was kept equal to remove any impact from class imbalance, which reduced the total number of available training samples to 50,000. These results can then be compared with the performance of adding an equivalent amount of augmented data. Here the augmented data was generated from a very small pool of 500 real training samples from each class.

B. Augmentation in data-space

For image data it is possible to create plausible transformations of existing samples that preserves label information, with the validation of label integrity being performed by a human observer (can a human still recognize the object). One of the significant improvements in performance of classifiers on the MNIST database was through the introduction of elastic deformations [5], in addition to the existing affine transformations, for data augmentation. The elastic deformation was performed by defining a normalized random displacement field $\boldsymbol{u}(x, y)$ that for each pixel location (x, y) in an image specifies a unit displacement vector, such that $\boldsymbol{R_w} = \boldsymbol{R_o} + \boldsymbol{\alpha u}$, where $\boldsymbol{R_w}$ and $\boldsymbol{R_o}$ describe the location of the pixels in the original and warped images respectively. The strength of the displacement in pixels is given by $\boldsymbol{\alpha}$. The smoothness of the displacement field is controlled by the parameter σ, which is the standard deviation of the Gaussian that is convolved with matrices of uniformly distributed random values that form the x and y dimensions of the displacement field $\boldsymbol{u}$.

For the MNIST data set we found that large deformations, with the displacement $\alpha \geq 8$ pixels, could occasionally result in characters that were difficult to recognize by a human observer, that is label information was not preserved. This loss of label integrity was caused by shifting a critical part of the character outside of image boundary, or by introducing a “kink” that rendered the character illegible, as illustrated in Figure 2. We empirically set $\alpha = 1.2$ pixels with $\sigma = 20$ based on performance of the (quick to train) CELM algorithm.

The samples needed for elastic warping were generated offline and the same data applied to each of the experiments.

C. Augmentation in feature-space

For some machine learning problems it may not be easy to validate that an arbitrary transformation of an existing raw data sample has indeed preserved label information. For these problems the features may have been hand crafted to extract salient information on which the classifier performs learning.

The Synthetic Minority Over-Sampling Technique (SMOTE) [7] was inspired by the use of data-warping to reduce class imbalance in the handwritten digit problem. But by being applied in feature-space it was proposed by the authors to be domain independent, and has since been used extensively in medical research where datasets with a significant minority class are common. For SMOTE a new synthetic sample is created by selecting a random point in feature-space along a line intersecting *k* randomly chosen samples of the same class.

A more recent derivative of SMOTE is the Density Based SMOTE (DBSMOTE) algorithm [15], which generates the new synthetic samples within a distance eps of the cluster centre for the class (following original notation). As the DBSMOTE algorithm generates its synthetic samples around the center of each class, it could be expected that these synthetic samples may not contribute to reducing classifier overfitting. Indeed we found that using DBSMOTE actually increased the amount of overfitting.

For the MNIST data set we used $k = 2$ random samples, for both SMOTE and DBSMOTE, and distance threshold $eps \leq 4$ for DBSMOTE.

The samples needed for SMOTE and DBSMOTE were generated online, as these algorithms are naturally embedded within the classification architecture, which is described in the next section.

IV. Experimental System Parameters

As previously stated, the two-stage architecture illustrated in Figure 1 was used for each experiment.

A. Stage-1 convolution and pooling.

For stage-1, to create a local receptive field, we adopt the method and parameters described in [9]. Conceptually, the 28×28 input image is convolved with a $W \times W$ filter, where $W = 7$ and the stride length is 1. This results in an intermediate output feature layer with dimensions $(28 - W - 1)^2$. Next LP-pooling is applied with a pool size of $q \times q$, where $q = 8$, which results in a down-sampled feature layer with dimensions 22×22. This is repeated for L filters, where $L = 96$ and corresponds to the first layer of pre-trained OverFeat filters [16].

B. Stage-2 classification.

For stage-2, we evaluated the following classification schemes: backpropagation-trained neural network, support vector machine, and extreme learning machine. The weights of the convolutional layer were held constant.

The neural network classifier was a standard multi-layer perceptron neural network with sigmoidal activation function. For simplicity of parameters, and to keep the neural architecture similar between classifiers, a single hidden layer with 1600 neurons was used. Learning was performed using backpropagation on the the hidden neurons only (and no changes were made to the convolutional layer). The number of epochs (one batch application of backpropagation on all training samples) was fixed at 2000.

The multi-class support vector machine classifier, which used the 1-vs-all L2 loss function, was based on the code and parameters settings of [8].

The extreme learning machine classifier, which used least squares regression of random weight projection neurons, was based on the code and parameter settings of [9]. For consistency with the CNN, a single hidden layer with 1600 neurons was used.

V. Experiments and Results

The first step in understanding the benefit of data augmentation was to vary the amount of real data samples available to train a classifier. Our hypothesis is that for a given number of samples (of either real or synthetic data) the benefit provided by real samples provides an upper-bound that the performance of a classifier can be improved by an equivalent number of synthetic samples. We denote this experiment as our baseline, where *training error* indicates the percentage error rate (*error %*) of the classifier when evaluated against the samples used to train the classifier, and *test error* is the *error %* on the predefined MNIST test set of 10,000 samples.

Each experiment is repeated three times to account for random variation. The primary variation in the CNN is the initialization of the weights and the order in which the training samples are presented to the classifier. The variation in the CELM output is due to the random weight projection. The CSVM implementation is deterministic, leading to no performance variation for multiple runs.

A. Baseline Results

The performance of the baseline system, illustrating the relative performance of the CNN, SVM and ELM is shown in Figure 3. Each of the classifiers have similar test *error %* for the amount of training data used. The primary observation is that as the number of real samples are increased (from 500 to 5000 per class) the gap between *training error* and *test error* decreases, which indicates a reduction in overfitting. The amount of improvement in performance varies across the classifiers. This confirms that the primary benefit of increasing the number of samples is to reduce overfitting (given that all other classifier parameters are held constant).

Another interesting observation is that the CNN benefits the most from additional training samples, in that increasing the number of samples improving both the *training error* and the *test error*. The increase in the amount of data results in more iterations of the backpropagation algorithm for the same number of epochs. The CELM varies the most in *error %* for a given amount of data, due to the random weight projection neurons in the hidden layer. The CELM also has good performance for small amounts of training data. The CSVM also exhibited similar behaviour to the CELM, with the best test *error %* for 500 samples per class, but performance did not improve as much as the other classifiers as the number of samples were increased.

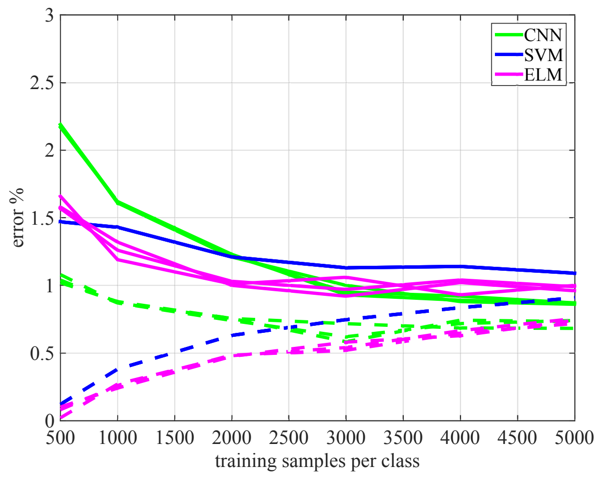

Fig. 3. Baseline performance of CNN, CSVM and CELM on MNIST using real data. The dashed lines indicate training error %, and the solid lines indicate test error %. Lower test error % indicates better performance. Reducing the difference between training and test error % is indicative of reducing overfitting.

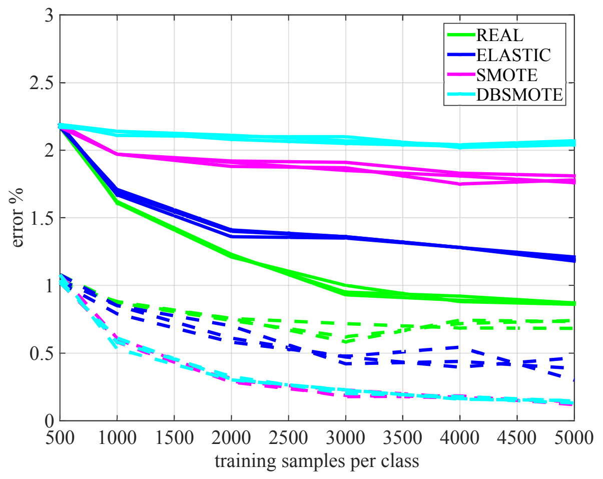

Fig. 4. CNN performance on MNIST. The dashed lines indicate training error %, and the solid lines indicate test error %.

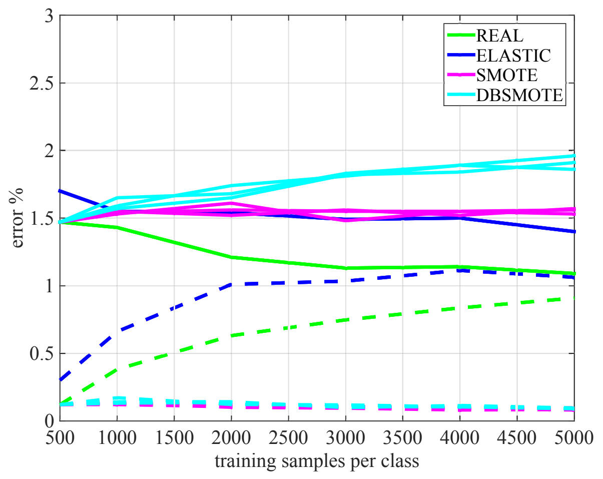

Fig. 5. CSVM performance on MNIST. The dashed lines indicate training error %, and the solid lines indicate test error %.

B. *CNN Results*

The performance of the CNN is shown in Figure 4, which illustrates how *error %* varies as the number of samples are increased. Again this shows that increasing the number of samples resulted in a performance improvement. Most notably the *test error %* decreased steadily. The CNN results were consistent, with multiple runs of the same experiment producing similar *error %*.

Augmentation in data-space using elastic deformations gave a better improvement in *error %* than augmentation in feature-space. Interestingly, while *test error %* decreased, *training error %* also decreased, and thus the gap between test and training was roughly maintained.

Augmentation in feature-space using SMOTE showed marginally promising results. Using SMOTE, the *test error %* continued to decrease as the number of samples were increased all the way to 50,000 samples.

Augmentation in feature-space using DBSMOTE resulted in a slight improvement on test *error %*. Another interesting observation is that the training error % for SMOTE and DBSMOTE followed very similar curves.

C. *Convolutional SVM Results*

The performance of the CSVM is shown in Figure 5, which illustrates how *error %* varies as the number of samples are increased. Perhaps the most interesting result is that increasing the number of synthetic samples using DBSMOTE caused the performance to degrade, i.e., the *error %* increased as more samples were used. Augmentation in feature-space using the DBSMOTE algorithm was not effective in reducing overfitting in the SVM classifier. However, this result is not surprising as the DBSMOTE promotes towards the centre of class clusters. By creating synthetic samples that are “more of the same” DBSMOTE encourages overfitting of the training data. Augmentation in data-space using the SMOTE algorithm provided little to no improvement in performance. Even augmentation in data-space using elastic warping only provided a modest improvement in *test error %*, and this only occurred at very large amounts of data augmentation. However, it should be noted that the gap between training *error %* and *test error %* did decrease steadily as the amount of augmentation was increased, thus indicating a reduction in overfitting.

D. *Convolutional ELM Results*

The performance of the CELM is shown in Figure 6. This Figure illustrates how the *training error* and *test error* varies as the number of samples are increased, which again shows that performance typically improves as the amount of synthetic samples are increased. However, this improvement is less than that given by increasing the number of real samples.

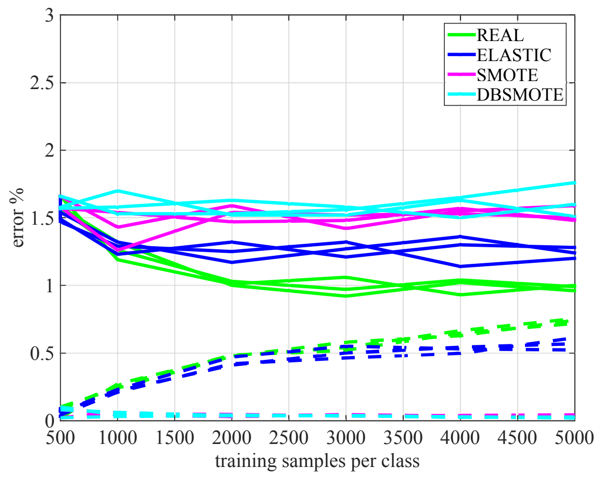

Fig. 6. CELM performance on MNIST. The dashed lines indicate training error %, and the solid lines indicate test error %.

Augmentation in data-space using elastic deformations gave the best results, which were slightly worse than having additional real training samples. However the trend is not linear, as there was a marked improvement in *error %* for 1000 training samples (500 real samples and 500 augmented samples) per class, before the gain in *error %* flatten-off once again.

Augmentation in feature-space using SMOTE seemed promising when increasing the amount of samples from 500 to 1000 samples per class. However, increasing the amount of synthetic samples further resulted in a decrease in performance. Also while *test error %* did initially decrease with the use of SMOTE, there was no corresponding reduction in training accuracy, which does occur when increasing the number of real samples. Thus, the gap between training and testing performance remained large.

Augmentation in feature-space using DBSMOTE had a slightly negative impact on *test error %*.

VI. Discussion

The experiments conducted used a classification architecture that was neatly divided into two stages: a feature generation stage and a classification stage. The purpose of this was to investigate if it is better to conduct data augmentation in *data-space* or *feature-space*?

For the experiments conducted on handwritten digits recognition, it was clearly better to perform augmentation in *data-space* using elastic distortions. We expect this to hold true for other classification tasks, if label preserving transforms are known and can be applied.

When label preserving transforms are not avaliable, the SMOTE algorithm can be used to perform data augmentation in *feature-space*, and provide some benefit to a CNN or CELM classifier. Our results suggest the DBSMOTE algorithm should not be used for data augmentation. However, a classification architecture that is neatly divided into *data-space* and *feature-space* is an artificial construct. The modern trend is to construct architectures with deep layered hierarchies of feature transformations with more complex features built upon simpler features. Nevertheless, these results should also provide insight into performing data augmentation for more modern architectures. A good overview of label preserving transformations for image data for a more modern classfication architecture is given by [17].

Another research question that we sought to answer, is how much data is enough? For most classification systems, more data is better. And more real data is better than more synthetic data. In none of the experiments did the performance of the system trained on synthetic data outperform the system trained on real data. Thus, the performance (test error %) that can be achieved by augmenting classifier training with synthetic data, is likely to be bounded by training on the equivalment amount of real data.

For the CNN, adding more synthetic data, using SMOTE and elastic warping, consistently reduced the testing error %. The experiments did not reach the limits of the amount of synthetic data that could be added before error % no longer improved.

For the CSVM adding more synthetic data, using DBSMOTE, caused classification performance to degrade.

For the CELM more synthetic data was not always better for performance. For the combination of the CELM with SMOTE there was a peak improvement provided by data augmentation at 1000 training samples per class, while increasing the number of samples further using the SMOTE technique resulted in a decrease in performance.

When comparing the three classification algorithms, the CSVM algorithm demonstrated less benefit from data-augmentation than the CELM, which showed less benefit than the CNN algorithm.

VII. Conclusion

This paper demonstrates the benefits and pitfalls of data augmentation in improving the performance of classification systems. Data augmentation can be performed in data-space or feature-space. We found that it was better to perform data augmentation in *data-space*, as long as label preserving transforms are known. The highly cited SMOTE algorithm can be used to perform data augmentation in *feature-space*. This is a more robust solution than the DBSMOTE algorithm, which can increase overfitting due to the algorithm creating new samples close to existing cluster centers. The improvement in *error %* provided by data augmentation was bounded by the equivalent amount of real-data.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[2] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.

[3] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1–9.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *arXiv preprint arXiv:1512.03385*, 2015.

[5] P. Y. Simard, D. Steinkraus, and J. C. Platt, "Best practices for convolutional neural networks applied to visual document analysis," in *2013 12th International Conference on Document Analysis and Recognition*, vol. 2. IEEE Computer Society, 2003, pp. 958–958.

[6] D. C. Ciresan, U. Meier, L. M. Gambardella, and J. Schmidhuber, "Deep, big, simple neural nets for handwritten digit recognition," *Neural computation*, vol. 22, no. 12, pp. 3207–3220, 2010.

[7] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, no. 1, pp. 321–357, 2002.

[8] A. Coates, A. Y. Ng, and H. Lee, "An analysis of single-layer networks in unsupervised feature learning," in *International conference on artificial intelligence and statistics*, 2011, pp. 215–223.

[9] M. D. McDonnell and T. Vladusich, "Enhanced image classification with a fast-learning shallow convolutional neural network," *arXiv preprint arXiv:1503.04596*, 2015.

[10] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[11] H. S. Baird, "Document image defect models," in *Structured Document Image Analysis*. Springer, 1992, pp. 546–556.

[12] L. Yaeger, R. Lyon, and B. Webb, "Effective training of a neural network character classifier for word recognition."

[13] X. Zhang, Y. Fu, A. Zang, L. Sigal, and G. Agam, "Learning classifiers from synthetic data using a multichannel autoencoder," *arXiv preprint arXiv:1503.03163*, 2015.

[14] I. Sato, H. Nishimura, and K. Yokoi, "Apac: Augmented pattern classification with neural networks," *arXiv preprint arXiv:1505.03229*, 2015.

[15] C. Bunkhumpornpat, K. Sinapiromsaran, and C. Lursinsap, "Dbsmote: density-based synthetic minority over-sampling technique," *Applied Intelligence*, vol. 36, no. 3, pp. 664–684, 2012.

[16] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "Overfeat: Integrated recognition, localization and detection using convolutional networks," *arXiv preprint arXiv:1312.6229*, 2013.

[17] A. G. Howard, "Some improvements on deep convolutional neural network based image classification," *arXiv preprint arXiv:1312.5402*, 2013.