Recognition of Online Handwritten Gurmukhi Strokes using Convolutional Neural Networks

Rishabh Budhouliya[1], Rajendra Kumar Sharma[1] and Harjeet Singh[2]
[1]*Department of Computer Science and Engineering, Thapar Institute of Engineering and Technology, Punjab, India*
[2]*Chitkara University Institute of Engineering and Technology, Chitkara University, Punjab, India*

Keywords: Convolutional Neural Networks, Data Augmentation, Stroke Warping, Gurmukhi Strokes, Online Handwritten Character Recognition.

Abstract: In this paper, we attempt to explore and experiment multiple variations of Convolutional Neural Networks on the basis of their distributions of trainable parameters between convolution and fully connected layers, so as to achieve a state-of-the-art recognition accuracy on a primary dataset which contains isolated stroke samples of Gurmukhi script characters produced by 190 native writers. Furthermore, we investigate the benefit of data augmentation with synthetically generated samples using an approach called *stroke warping* on the aforementioned dataset with three variants of a Convolutional Neural Network classifier. It has been found that this approach improves classification performance and reduces over-fitting. We extend this finding by suggesting that stroke warping helps in estimating the inherent variances induced in the original data distribution due to different writing styles and thus, increases the generalisation capacity of the classifier.

1 INTRODUCTION

Research in Online Handwritten Recognition Systems seems to have peaked since the late 1990s, considering the fact that Google′s Multilingual Online Handwritten Recognition System is able to support 22 scripts and 97 languages, which is being currently used by certain commercial products like Google Translate successfully (Keysers et al., 2017). Despite this success in Google′s system, we want to bring certain factors into light which are limited to the scope of Indo-Aryan Languages -

- Each written language has a strong variability associated with the writing style depending upon certain demographic factors like region, age and culture.
- In the case of such languages, major issues tackled by any researcher include the shape complexity of the characters, features for recognition of characters and stroke level recognition.

Motivated by these factors, our work aims to:

- Use a primary dataset called OHWR-Gurmukhi, described in section IV, to sub sample a dataset which includes 79 stroke classes with 100 samples each, referred as dataset_100. A Convolutional Neural Network has been used to classify the strokes to achieve a state-of-the-art accuracy.
- Explore and evaluate multiple CNN variants on the basis of their distribution of trainable parameters between convolution layer and fully connected layer on datasets with varying data samples per class.
- Study the effect of *stroke warping*, a technique to produce random variances within the stroke samples. The warped augmented dataset is created by applying a combination of affine transformation (rotation) and elastic distortions to images of the existing stroke samples.
- Experimentally evaluate the effect of data augmentation on classification accuracy of the classifier.

This paper is structured as follows. In section II, there is a discussion about the development of character recognition and the progression in Gurmukhi script recognition. In section III, we introduce the Gurmukhi script, covering its features and the different styles of writing the script. The generation of synthetic data is covered in section IV, where we use the technique called *stroke warping*, to augment the dataset_100. After that, we delve into our main idea in section V, an experiment which tests three classifiers to validate the benefit of data augmentation and

Budhouliya, R., Sharma, R. and Singh, H.
Recognition of Online Handwritten Gurmukhi Strokes using Convolutional Neural Networks.
DOI: 10.5220/0008960005780586
In *Proceedings of the 12th International Conference on Agents and Artificial Intelligence (ICAART 2020) - Volume 2*, pages 578-586
ISBN: 978-989-758-395-7; ISSN: 2184-433X

to achieve an optimal classification accuracy. In section VI, the framework of the experiment is laid out and results of the performed experiments are shown. We discuss the result and their consequences on the objectives of the paper in section VII and we conclude the findings in section VIII.

2 RELATED WORK

LeNet was the very first Convolutional Neural Network used for visual detection tasks including character recognition and document analysis (Jackel et al., 1995). This neural network was used to extract local geometric features from the input image in a way that preserved approximate relative locations of these features. By convolving the input image with a trainable kernel, the network was able to produce high level feature maps which were then fed to a linear classification layer. For the system they created, an overall OCR accuracy exceeding 99.00% was achieved.

Owing to the continuous academic research in the Online Handwritten Chinese Character Recognition, it has been demonstrated (Xiao et al., 2017) that methods based on CNNs can learn more discriminative features from source data, which may lead to a better end-to-end solution for Online Handwritten Recognition problems. The authors of this paper continued to design a compact CNN classifier for Online Handwritten Chinese Character Recognition using DropWeight for pruning redundant connections in a CNN architecture maintaining an accuracy of 96.88%. Handwritten Bangla Digit Recognition using a CNN with Gaussian and Gabor Filters, achieved 98.78% recognition accuracy (Alom et al., 2017).

Along with working on improving the recognition accuracy of the classifiers, researchers also worked upon writer adaptation for online handwritten recognition where they used lexemes to identify the styles present in a particular writer's sample data which resulted in the reduction of average error rate on handwritten words (Connell and Jain, 2002).

In the present paper, we demonstrate a method to capture the variability induced by different writing styles, thus enhancing the generalization accuracy of the classifier. It is pertinent here to discuss the progression in Gurmukhi script recognition. A recognizer using pre-processing algorithms (Normalization, Interpolation and Slant Correction) has been proposed to recognize loops, headline, straight line and dot features from online handwritten Gurmukhi strokes collected on a pen-tablet interface by 60 writers (Sharma et al., 2007). After this, a post processor for improving the accuracy of character recognition was built to detect and aggregate strokes using set theory to recognize characters with an accuracy of 95.60% for single character stroke sequencing (Kumar and Sharma, 2013). This work was done on a dataset of 27,231 samples categorized on the basis of the proficiency of the writers. A significant shift came after Hidden Markov Models(HMM) and Support Vector Machines(SVM) were used for classification while employing the features extracted on the basis of region and cursiveness. This experiment resulted in a 96.70% recognition rate of Gurmukhi characters (Verma and Sharma, 2016). Their experiments consisted of the methods to extract features and then classify them using SVM or HMM for classification. Our aim in this work is to use the Deep Learning concept of learning features and then performing extensive experimentation using CNNs to obtain better recognition accuracy. One of the main advantage of using a CNN is that it is able to extract features automatically and is invariant to shift and distortion (Wong et al., 2016).

3 GURMUKHI SCRIPT

Punjabi language is spoken by about 130 million people, mainly in West Punjab in Pakistan and in East Punjab in India. Indian Punjabi is written using the Gurmukhi script, which has a fairly complex system of tonal variance.
Some notable features of Gurmukhi script are :

- Gurmukhi script is cursive and written in left to right direction with top down approach.
- A horizontal line, called a "shirorekha" is found on the upper part of almost all the characters.
- Any Gurmukhi word can be divided into three sections viz. Upper Zone, Middle Zone and the Lower Zone. All the strokes are classified into one of the three zone. The upper zone consists of the region above the head line where some of the vowels reside. The middle zone is the most populated zone, consisting of consonants and some of the vowels. The lower zone contains some vowels and half characters that lie below the foot of consonants (Verma and Sharma, 2017).

3.1 Different Styles of Writing Gurmukhi Script

The critical area of research in Character recognition is to capture and detect the complex nature of any script that results in the variation of writing styles. The documented reasons for variation in handwriting

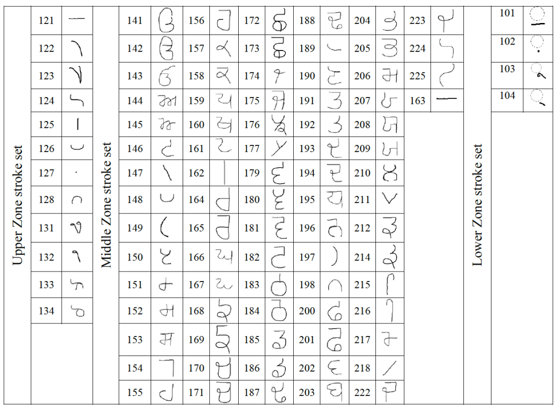

Upper Zone stroke set
Middle Zone stroke set
Lower Zone stroke set

Figure 1: Stroke classes for Gurmukhi character set.

style can be attributed to a distinct way of writing for each person, and the ways can be:

- Speed of writing
- Style of holding the pen
- Formation of a character can be influenced by the amount of strokes used to create a character. Some users use a single stroke while others may use multiple strokes to write the same thing.

3.2 Data Collection

The source population of our dataset is created by 190 writers of different age group to bring maximum variability in the population. A touch based, Tablet PC has been used as input interface. The collected data was annotated at stroke level with respective stroke classes characterized by three zones; Upper, Middle and the Lower zone as shown in Figure 1.

4 DATA AUGMENTATION

The data for this experiment was extracted from the dataset called OHWR-Gurmukhi, which contains *x*- and *y*- coordinates of the strokes captured using a tablet PC. The *x*- and *y*- coordinates from an XML file were parsed and converted into binary images using Python. We have chosen to work for the Middle Zone stroke set, particularly for the reason that it contains some of the most complex strokes in any character. Hence, dataset_100 contains 79 classes, each class representing a distinct stroke, having 100 labeled 60×60 pixel images. This dataset was used for two purposes. Firstly, it was used to create the synthetic data through the augmentation techniques. After that it was divided into a 75 images per class training set, and 25 images per class testing set. No

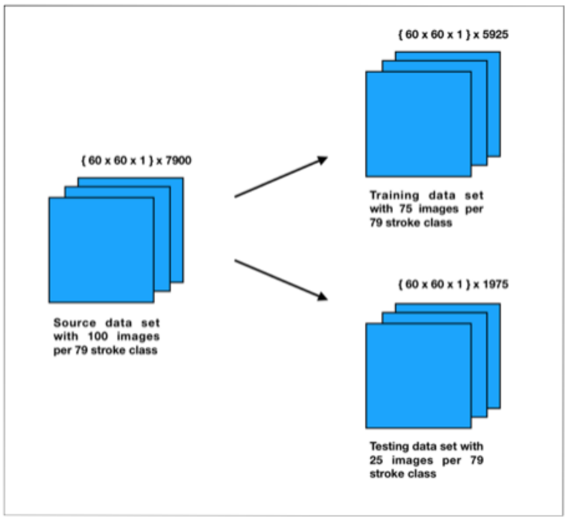

(60 x 60 x 1) x 7900
Source data set with 100 images per 79 stroke class
(60 x 60 x 1) x 5925
Training data set with 75 images per 79 stroke class
(60 x 60 x 1) x 1975
Testing data set with 25 images per 79 stroke class

Figure 2: The distribution of dataset_100 into training and test set.

validation set was created due to the small size of the dataset_100.

An important question to be answered is why does an increase in data size by augmentation of sample images increases the generalization accuracy? Also, if there is an increase, could it be attributed to the fact that stroke warping is able to capture the variability in the empirical population due to the writer's personal characteristics as discussed above. To answer these questions, we have set-up an experiment where the correlation between change in nature of the dataset due to addition of synthetic data samples and increase in recognition accuracy can be seen and verified by comparing performances of three different CNN classifiers on those different datasets.

4.1 Augmentation in Data-space

The debate on the correct usage of synthetic data for training a model was provoked at the 5th ICDAR conference (Baird, 1989), resulting in a list of conclusions:

- Usage of synthetic data was definitely beneficial as the model that is trained on the most data wins.
- Although producing synthetic data was considered a good practice, it was not considered safe. Training on a mixture of real data and synthetic data was considered the safest.
- Testing on synthetic data to claim good performance is conceptually wrong.

At present, it has been well established that data augmentation is a powerful technique to regularize neural

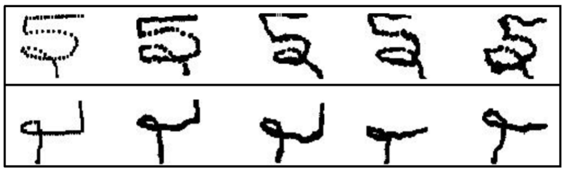


Figure 3: Stroke ID:169 and Stroke ID:223, former being the first picture position wise. Median Filter was used to smoothen the stroke images first, then stroke warping was applied, with a combination of varying α withing the range of 4-6 pixels and random rotation within the range of 5 to 30 degrees in both directions.

networks to prevent over fitting and improve performance in an imbalanced class situation. Moreover, they have shown that there is a direct correlation between the improved object detection performance and the increase in average number of training samples per class. The techniques involved in creating artificial training samples range from cropping, rotating to flipping input images. Our method of data augmentation is a combination of such simple techniques and elastic deformation, explained in the next sub-section.

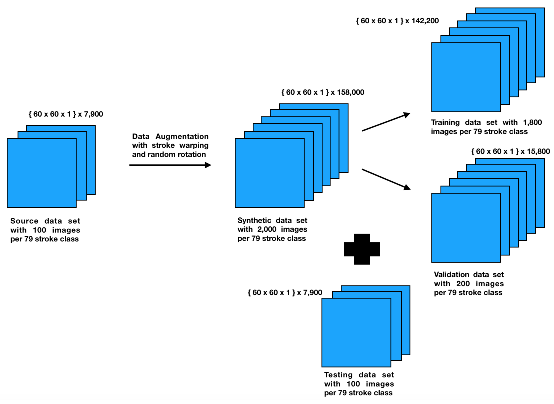
4.2 Stroke Warping

Stroke warping is the technique used during the training process in order to produce random variations in the data (Yaegar et al., 1996). This technique produces a series of characters which are consistent with stylistic variations within the writers. In the experiment, we try to find if these stylistic variations create a synthetic sample distribution which is close to the real population distribution of the 190 writers. In our experiment, the warped character stroke was created using a combination of affine transformation (rotation) and elastic deformation to the existing dataset as shown in Figure 3.

The elastic deformations on the images were created by first generating normalized random displacement field $r(x,y)$ where (x,y) is the location of a pixel such that

$$R_w = R_o + \alpha * r(x,y) \quad (1)$$

where R_w and R_o explain the location of the pixels in the warped and original images respectively. Here, α decides the magnitude of displacement. The displacement fields are convolved with a Gaussian of standard deviation of σ which is the smoothness factor of the extent of elastic deformation.


{ 60 x 60 x 1 } x 7,900
Source data set with 100 images per 79 stroke class
Data Augmentation with stroke warping and random rotation
{ 60 x 60 x 1 } x 158,000
Synthetic data set with 2,000 images per 79 stroke class
{ 60 x 60 x 1 } x 142,200
Training data set with 1,800 images per 79 stroke class
{ 60 x 60 x 1 } x 15,800
Validation data set with 200 images per 79 stroke class
{ 60 x 60 x 1 } x 7,900
Testing data set with 100 images per 79 stroke class
Figure 4: Distribution of dataset_2000 into training and validation set. Testing set was chosen to be the dataset_100.

4.3 Generation of Synthetic Data using *Stroke Warping*

The binary images of size 60×60 pixel taken as binary images from the dataset_100 were first convolved with a median filter, with kernel size 3×3 pixels to smoothen out the irregularities in the stroke images. After this, the images were subjected to a random rotational transformation varying from 5° to 30° in both clockwise and anti-clockwise direction. The images were distorted using elastic distortion, with σ as 37, and α randomly varied from 4-5 pixels with each image. This whole process was used to extrapolate 2,000 images per stroke from the 100 image per stroke dataset. Hence, the produced synthetic dataset had 1,58,000 samples in total, with 1,800 samples per stroke devoted for training data, rest 200 images per stroke for the validation set as shown in Figure 4. This dataset will be referred as dataset_2000 in this paper. It is worth mentioning that the amounts of each distortion in an image to be applied was examined by human eye to verify that they induce a natural range of variation in the dataset.

4.4 Division of Synthetic Dataset into 500, 1,000 and 1,500 Samples per Stroke

The synthetic dataset, dataset_2000, was randomly divided into smaller subsets of sizes 500, 1,000 and 1,500 images per class as shown in Figure 5. These smaller subsets will be referred as dataset_500, dataset_1000 and dataset_1500, respectively. This was done with a purpose to validate the hypothesis that classification accuracy can be increased by increasing the data samples with the help of augmentation. All the three classifier models have been trained on these

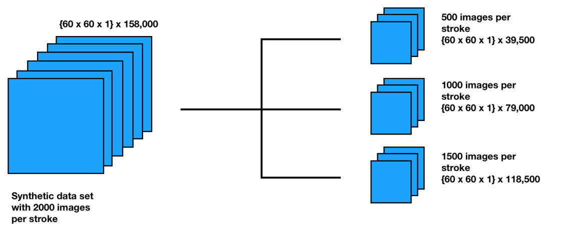

{60 x 60 x 1} x 158,000
Synthetic data set with 2000 images per stroke
500 images per stroke {60 x 60 x 1} x 39,500
1000 images per stroke {60 x 60 x 1} x 79,000
1500 images per stroke {60 x 60 x 1} x 118,500

Figure 5: Redistribution of dataset_2000 into dataset_500, dataset_1000, dataset_1500.

subsets and further tested on the original images and it has been expected that the testing accuracy increases as the size of the training dataset increases.

5 METHODOLOGY

In accordance to the initial objectives, we propose an experiment which is designed to find the optimal recognition accuracy for Gurmukhi strokes. To do so, the experiment starts with training three different CNN classifiers on five datasets (dataset_100, dataset_500, dataset_1000, dataset_1500, dataset_2000). Now, 20.00% of the input dataset is used for validation and testing is done on the dataset_100, with 7,900 images. The reason for choosing such a testing set was to ensure that our testing accuracy depicts a good picture of the capacity of the model to imitate the original distribution of dataset_100. This process has furnished three graphs, one for each model, depicting the performance of each model with varying datasets. These graphs will be used to validate the findings of the experiment.

The CNN classifier has three variants, all of them vary on the basis of their structure in terms of number of convolution layers or fully connected layers used in the model. In the next section, we introduce the reason for defining these three models and after that we explain them in detail.

5.1 Reason behind Selection of Multiple Classifiers for Experimentation

Our first step in understanding the dataset_100 was to empirically train and test it on a CNN model with 9,35,760 learnable parameters. The model is described below:

- Convolutional Layers: The model had four convolutional layers, all of them having a kernel of size 5×5 pixels with number of kernels doubling from 16 to 128 in each layer. The kernels were initialized with He normal initializer which draws samples from a truncated normal distribution (He et al., 2015). ReLU was chosen as the activation function.
- Pooling layers: Max Pooling was applied with 2×2 pixel sized filter at a sliding stride of 1.
- Fully Connected (FC) Layers: Two fully connected layers were connected to the feature extracting layers, one with 512 neurons and the next one with 128 neurons.
- Activation function for both of them was ReLU function.
- Dropout Layers: After each FC layer, a dropout layer with a dropout probability of 25.00% was placed to reduce over fitting.
- Classification Layer: A fully connected layer with 79 neurons (each representing a distinct stroke class) with Softmax activation function for finding the categorical distribution has been used.

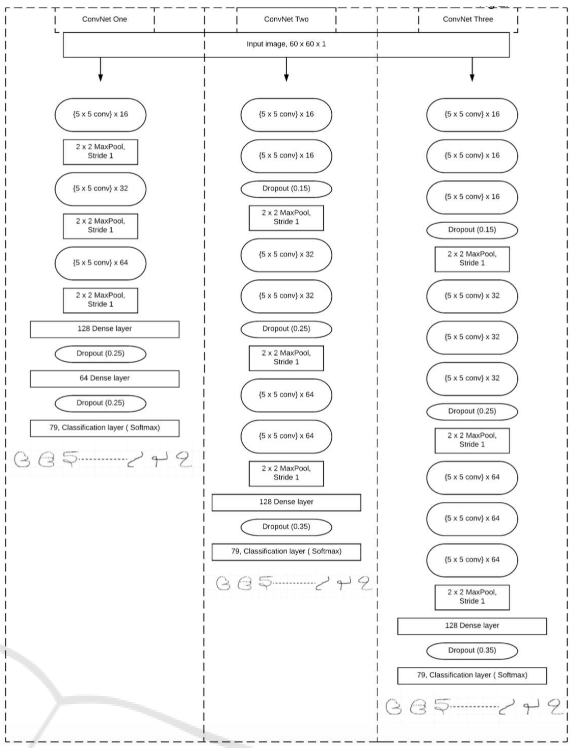


Figure 6: Classification of three CNN models on the basis of the learnable parametric distribution between convolutional layers and fully connected layers.

This model was trained with 6,320 images (80 stroke images per class) and tested on an unseen data of 1,580 images (20 stroke images per class). It is evident that this dataset is small, hence a validation set could not be created. With a batch size of 246 samples and 35 epochs in total, testing accuracy turned out to be 93.65% with a training accuracy of 99.17%. There

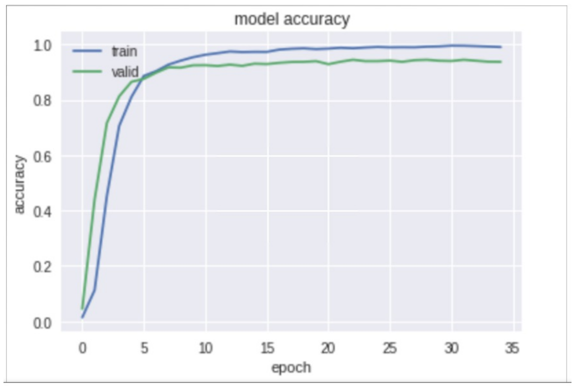


Figure 7: The gap between the training accuracy and validation accuracy depicts the high bias within the model.

is clear gap between the two accuracies, which can be seen in Figure 7, guiding us to the fact that over-fitting is deteriorating the generalization capacity of the model. After this experiment, repeated scenarios were created, and after running our data through those scenarios, we reached to the following conclusions:

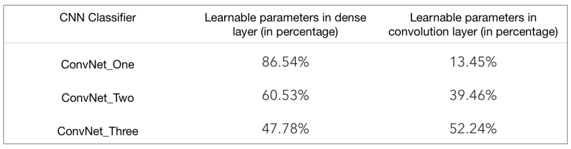
- Regularization: Among L1 regularization, L2 regularization and Dropout regularization, Dropout seems to work best to reduce over fitting. Thus for our experiment, we have chosen dropout layers only.
- Convolutional Layers vs Fully Connected Layers: Repeated experiments implied that fully connected layers may have a greater percentage of contribution towards the over-fitting in the model. The reason for such a phenomenon could be attributed to the fact that larger percentage of trainable parameters reside in these layers. For comparison, only 28.80% of the trainable parameter are contributed by the convolutional layers, while the rest of the 71.20% lies within the FC layers for this model.

These conclusions were the foundation for creating three variations of a CNN classifier. The variations have been referred as ConvNet_One, ConvNet_Two and ConvNet_Three. We assume that the distribution of trainable parameters between convolutional layer and fully connected layer, shown in Table 1 can be used as a useful hyper parameter to control the over-fitting in the model, thus finding an optimal classifier in the process.

5.2 Proposed Models: ConvNet_One, ConvNet_Two and ConvNet_Three

In Figure 6, the structure of each model is explained graphically. The fundamental difference between these models is the percentage of trainable parameters shared between convolutional layers and fully connected layers. We have stacked convolutional layers in the second and third model. We hypothesize that with each model, the effects of over-fitting should diminish due to balanced share of trainable parameters. Stacking convolutional layers could help in extracting enhanced features and we wish to test this in the experiment.

Table 1: Distribution of trainable parameters between convolutional and fully connected layer.



CNN Classifier	Learnable parameters in dense layer (in percentage)	Learnable parameters in convolution layer (in percentage)
ConvNet_One	86.54%	13.45%
ConvNet_Two	60.53%	39.46%
ConvNet_Three	47.78%	52.24%

- ConvNet_One: This is the simplest model in terms of number of layers and trainable parameters, the configuration of this model is as follows:
  - Convolutional Layers: The model has three convolutional layers, all of them having a kernel size of 5×5 pixels with number of kernels doubling from 16 to 64 for the third convolutional layer. The kernels were initialized with He normal initializer which draws samples from a truncated normal distribution. ReLU was chosen as the activation function.
  - Pooling Layers: Max Pooling was applied with 2 by 2 pixel sized filter at a sliding stride of 1.
  - Fully connected layers: Two FC layers were connected to the feature extracting layers, one with 128 neurons and the next one with 64 neurons. Activation function for both of them was ReLU function.
  - Dropout Layers: After each FC layer, a dropout layer with a dropout probability of 25.00% was placed to reduce over fitting.
  - Classification Layer: A fully connected layer with 79 neurons (each representing a distinct stroke class) with Softmax activation function for finding out the categorical distribution.
- ConvNet_Two: This model has two convolutional layers stacked together three times with dropout layer after each stack of convolutional layers. The dropout probability progress from 15.00% to 25.00% and then to 35.00% right after the FC layer. Other than this, it shares the same configuration with the first model.
- ConvNet_Three: In this network, three convolutional layers are stacked together three times with dropout layer after each stack of convolutional layers. The dropout probability progresses in the

Table 2: Configurational parameters for the experiment.

Training session	epochs	batch-size	training data set	validation data set
dataset_100	30	128	6,320	1,580
dataset_500	45	256	31,600	7,900
dataset_1000	45	256	63,200	15,800
dataset_1500	45	256	94,800	23,700
dataset_2000	45	256	1,26,400	31,600

same fashion as in ConvNet_Two. It shares the same configurational details for the rest of the parameters as ConvNet_One or ConvNet_Two.

6 EXPERIMENTS AND RESULTS

The experiment, as explained in the methodology section, is structured as follows:

- The experiment is divided into 5 training sessions, each session is associated with training on a particular dataset. Each session has the same models, with the same configuration.
- The configurational parameters for the whole experiment are given in Table 2. The optimizer for every session was chosen to be an algorithm based on adaptive moment estimation, called Adam with the default parameters as mentioned in the original paper (Kingma and Ba, 2017).
- Metrics to evaluate the performance of each model is the testing accuracy on the unseen data.
- The testing set was chosen to be dataset_100 with 7,900 images, which is the original dataset. As each model was trained in each session, testing accuracy was recorded for each dataset and we have produced a graph for each model with 'image samples per class' being the independent variable and 'testing accuracy' being the dependent variable.
- The difference between training accuracy and validation accuracy (on the validation dataset) was recorded for each observation as it is an indication of decreased over-fitting in a model.
- Among all the recorded observations, the highest testing accuracy achieved would be awarded the state of the art Gurmukhi character recognition accuracy for OHWR-Gurmukhi dataset.

This experiment was created and executed to achieve the goals laid out in the introduction section, and we discuss them in the next sections, where we present the model wise results in conformance to the structure of the experiment presented above.

Table 3: A tabular representation of the progress of training ConvNet_One.

Training session for ConvNet_One	training accuracy (in %)	validation accuracy (in %)	difference (in %)
dataset_100	96.65	94.15	2.50
dataset_500	96.19	95.39	0.80
dataset_1000	96.98	96.93	0.05
dataset_1500	97.63	97.41	0.22
dataset_2000	98.15	96.81	1.34

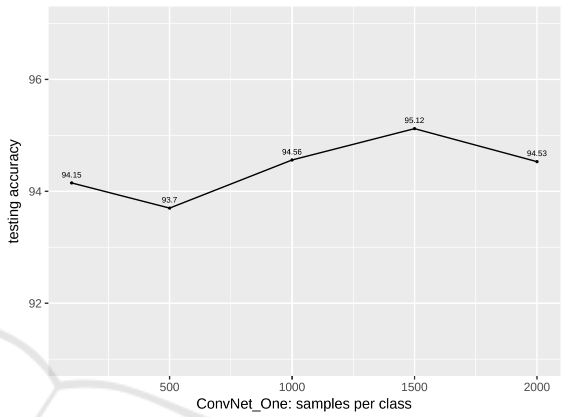


Figure 8: A graphical representation of the performance of ConvNet_One.

6.1 ConvNet_One Results

Starting with the difference between the training accuracy and validation accuracy, it is clear that overfitting decreases in ConvNet_One as the difference decreases with an increase in data size except for dataset_2000, shown in Table 3. Apart from that, a look into Figure 8 reveals that testing accuracy increases as samples per class increase but to a certain extent, as at dataset_2000, the accuracy has actually decreased from the previous observation, which could be attributed due to the increase in over-fitting, validated from the table, where the difference has increased as well.

6.2 ConvNet_Two Results

Here, by looking at Figure 9, it can be observed that performance of ConvNet_Two is better than ConvNet_One in terms of testing accuracy. Since the datasets for every model are same, this observation could be reasoned with the fact that ConvNet_Two has a greater percentage of trainable parameters in the convolution layers which in turn might have reduced the over-fitting of the model. Also, the difference between the training and validation accuracy, as shown in Table 4, has decreased in accordance to the claim that data augmentation can decrease over-fitting.

Table 4: A tabular representation of the progress of training ConvNet_Two.

Training session for ConvNet_Two	training accuracy (in %)	validation accuracy (in %)	difference (in %)
dataset_100	98.20	93.00	5.20
dataset_500	97.33	96.00	1.33
dataset_1000	97.47	96.20	1.27
dataset_1500	97.81	97.41	0.40
dataset_2000	98.91	98.79	0.12

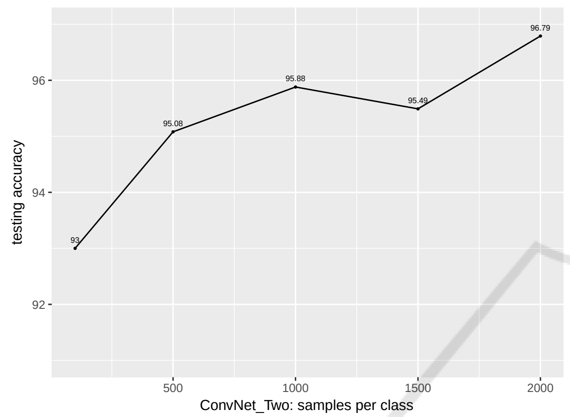


Figure 9: A graphical representation of the ConvNet_Two's performance.

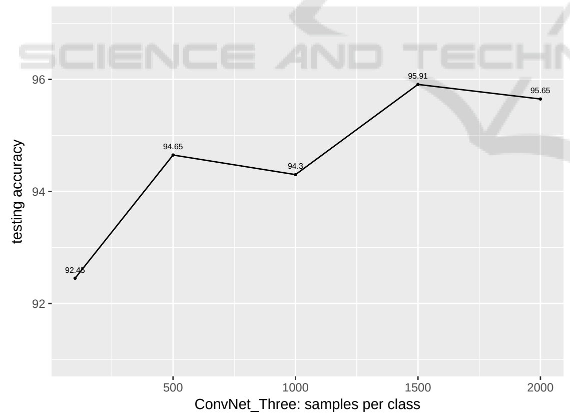


Figure 10: A graphical representation of the ConvNet_Three's performance.

6.3 ConvNet_Three Results

ConvNet_Three has the most equitable distribution of trainable parameters. Despite this, it is not able to outperform ConvNet_Two, evident in Figure 10 and Table 5. This shows that the relationship between distribution of trainable parameters and performance of a model is non linear, and the experiment conducted is inadequate to claim that a 50-50% share of trainable parameters between convolution layer and fully connected layer produces an optimal classifier.

Table 5: A tabular representation of the progress of training ConvNet_Three.

Training session for ConvNet_Three	training accuracy (in %)	validation accuracy (in %)	difference (in %)
dataset_100	97.65	92.45	5.20
dataset_500	95.42	95.15	0.27
dataset_1000	96.97	96.16	0.81
dataset_1500	96.01	96.05	0.04
dataset_2000	97.15	96.95	0.20

7 DISCUSSION

It is imperative to note that all three models showcase a common behaviour, an increase of data sample per class improves their performance. This gives us two results : First, we were able to achieve an optimal Gurmukhi character recognition accuracy of 96.79%. This is by far the highest accuracy achieved by a classifier on OHWR-Gurmukhi. Secondly, we were able to prove that usage of data augmentation is beneficial to the performance of a classifier but to a certain extent, as at dataset_2000, we saw a slight decline in performance of ConvNet_One and ConvNet_Three.

Since data augmentation was done solely through *stroke warping*, we believe that this technique is able to produce alternate character forms that are consistent with the stylistic variation within and between writers of the collected dataset as claimed earlier by the cited authors. This belief can be reinforced by our results, as the generalization accuracy of each of our model increases with the increase of synthetic samples generated through stroke warping.

8 CONCLUSION

This paper demonstrates the benefits and limitations of data augmentation on the Gurmukhi character strokes dataset, OHWR-Gurmukhi. It also attempts to find a relation between distribution of trainable parameters in a Convolutional Neural Network and the performance of a classifier, where ConvNet_Two, a model with a pair of convolution layers stacked together, is able to achieve the highest testing accuracy among the three proposed models. We also find stroke warping as an effective technique to augment the data with. We would like to perform data augmentation through Generative Adversarial Networks on OHWR-Gurmukhi. Also, there is a scope of analyzing the distribution of stylistic variations within and between writers for a particular language which would help in building more powerful classifiers in this classification field.

REFERENCES

Alom, M. Z., Sidike, P., Taha, T. M., and Asari, V. K. (2017). Handwritten Bangla Digit Recognition using Deep Learning. https://arxiv.org/abs/1705.02680/.

Baird, H. S. (1989). The State of the Art of Document Image Degradation Modeling. *Xerox Palo Alto Research Center*.

Connell, S. D. and Jain, A. K. (2002). Writer Adaptation for Online Handwriting Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(3):329–346.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. https://arxiv.org/abs/1502.01852.

Jackel, L., Battista, M., Baird, H., Ben, J., Bromley, J., Burges, C., Cosatto, E., Denker, J., Graf, H., Katseff, H., Lecun, Y., Nohl, C., Sackinger, E., Shamilian, J., Shoemaker, T., Stenard, C., Strom, I., Ting, R., Wood, T., and Zuraw, C. (1995). Neural-net applications in Character Recognition and Document Analysis. In *Neural-net applications in telecommunications*. Kluwer Academic Publishers.

Keysers, D., Deselaers, T., Rowley, H., Wang, L., and Carbune, V. (2017). Multi-Language Online Handwriting Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1180–1194.

Kingma, D. P. and Ba, J. L. (2017). ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION. https://arxiv.org/abs/1412.6980.

Kumar, R. and Sharma, R. K. (2013). An Efficient Post Processing Alogrithm For Online Handwriting Gurmukhi Character Recognition Using Set Theory. *International Journal of Pattern Recognition and Artificial Intelligence*, 27(4).

Sharma, A., Sharma, R., and Kumar, R. (2007). Online Handwritten Gurmukhi Strokes Preprocessing. In *Machine GRAPHICS & VISION*.

Verma, K. and Sharma, R. K. (2016). Comparison of HMM- and SVM-based Stroke Classifiers for Gurmukhi Script. In *The Natural Computing Applications Forum 2016*.

Verma, K. and Sharma, R. K. (2017). Recognition of Online Handwritten Gurmukhi Characters Based on Zone and Stroke identification. *Sadhana*, 42:701–712.

Wong, S. C., Gatt, A., Stamatescu, V., and McDonnell, M. D. (2016). Understanding data augmentation for classification: when to warp? https://arxiv.org/abs/1609.08764.

Xiao, X., Yang, Y., Ahmad, T., Jin, L., and Chang, T. (2017). Design of a Very Compact CNN Classifier for Online Handwritten Chinese Character Recognition Using DropWeight and Global Pooling. *14th IAPR International Conference on Document Analysis and Recognition*, pages 891–895.

Yaegar, L., Lyon, R., and Webb, B. (1996). Effective Training of a Neural Network Character Classifier for Word Recognition. In *Advances in Neural Information Processing Systems (NIPS)*.