Data Augmentation: A performance review on different models

Karim Hasbini

Concordia University

Montreal, Canada

karim.hasbini@mail.concordia.ca

40053498

*Abstract***—*****Image (pre)processing is routinely conducted in many different software fields, particularly machine learning and deep learning. Three convolutional neural networks (CNNs) are created and trained on three different datasets: the plain Modified National Institute of Standards and Technology (MNIST) dataset composed of 70,000 gray-scale images of handwritten digits, a modified version of the dataset where several linear and non-linear image processing techniques such as shearing, rotation, flips have been applied to each image (transformed), and a hybrid dataset with an equal ratio of images from the previous two datasets (hybrid) with the same sample size of 70,000. We examine the required architecture necessary to achieve an acceptable accuracy score (defined as over 80%) on the testing subset of each dataset, as well as the accuracy score of each trained model on the different datasets. We hypothesize that the model trained on the unmodified dataset (plain model) would have the simplest architecture, i.e. the most shallow, while the model trained purely on the transformed dataset would have an architecture that is deeper but with an acceptable accuracy score on the plain dataset. The hybrid model would have a testing accuracy score that is lower on the hybrid dataset than the other models on their respective testing datasets, but would outperform the other models on the testing datasets which the model has not been trained with, i.e. it is more generalized than the other models which would be more specialized. We found that partially transforming the data before the training phase resulted in a hybrid model with an additive performance on the other datasets.***

I. Introduction

Image processing is an essential step of autonomous machine perception. It includes data augmentation, which is a technique that enhances both the size and the quality of a given dataset. It enhances model performance by serving as a preventative measure against overfitting, which is a common issue that occurs when a model excels at classifying or predicting on data that was included in the training set, but fails to do so on data that it was not trained on. However, enhancing generalizability, is not its sole value. Because deep learning models rely on a great deal of data to effectively train and obtaining labeled data is an expensive process, data augmentation satisfies another very important need[1].

Luckily, there are a variety of large databases that may be accessed by the public for machine learning endeavors. One such database, the Modified National Institute of Standards and Technology (MNIST) database is a sizable and popular database of 28 by 28 pixel grayscale images of handwritten digits, consisting of 60,000 training images, and 10,000 testing images. It is widely used for training and testing models among machine learning practitioners. In fact, many different Convolutional Neural Networks (CNNs) have been trained on the MNIST dataset, with the highest recorded accuracy rate ever achieved for a deep CNN on the original, un-augmented dataset being 98.4% [2]. Utilizing this dataset, this paper seeks to explore the effect on model performance of data augmentation methods as a regularization step for deep learning image classification while applying some of the different image processing principles and algorithms learned throughout the semester.

II. Methodology

The plain dataset is composed of 70,000 28 by 28 pixel images having only one channel. Initially, the images are loaded into memory as two dimensional numpy arrays (matrices) with pixel intensity values ranging from 0 (black) to 255 (white). As a preprocessing step, the matrices are standardized using the sklearn library's preprocessing module's scaling function[5]. We chose the Tensorflow library[6] as the source for the data as well as the library to build the CNNs and to transform the images. The dataset was split into 60,000 samples for training and 10,000 samples for testing. In order to create the transformed copy, we selected a set of image transformations and applied a random subset of the transformations throughout the set. Each transformation was selected randomly within a predefined range which can be seen in Table 1. The rotation angle range was chosen to be below 90 degrees purposely, otherwise it would lead to a complete loss in distinguishability (effectively introducing an extraneous variable) between the digits '9' and '6', which may lead to a decrease in accuracy for the hybrid and transformed models. For transformations that affect the shape of the image, borders have been filled with a constant equal to 0. The same training and testing sample sizes were used to split the transformed data. To create the hybrid data, a non-overlapping section of each set was taken with equal cardinality, as can be seen in (1).

$$\begin{gathered} P \subset plain \\ T \subset transformed \\ hybrid = \{ P \cup T \vee P \cap T = 0 \wedge |P| = |T| \} \end{gathered} \tag{1}$$

TABLE I. TRANSFORMATIONS AND THEIR RANGES APPLIED TO CREATE THE TRANSFORMED DATASET

Transformation	Range
Rotation	0–60 degree angle
Width shift	Up to 1/3 of the width
Height shift	Up to 1/3 of the width
Brightness shift	A factor of 0.2 to 1
Shear	Up to 30 degrees
Horizontal flip	N/A
Vertical flip	N/A

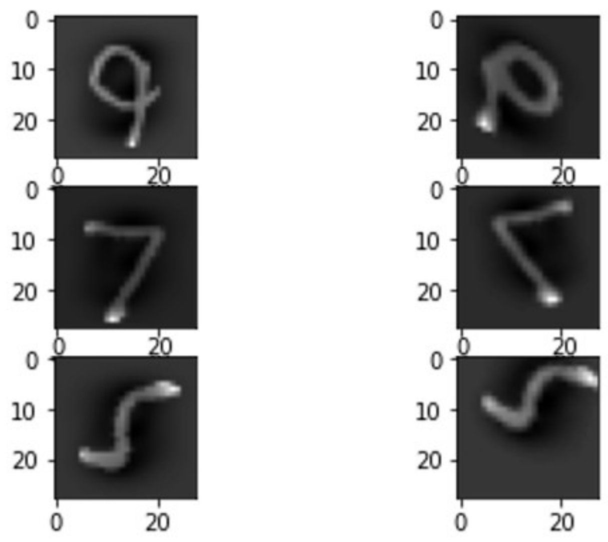


Fig. 1. Plain samples (left) and their transformed counterparts within the transformed dataset (right)

After concatenating the two subsets, we shuffled the data to avoid any biases. Samples of the plain and transformation dataset can be seen in Figure 1.

We designed the architecture of each CNN without referring to any of the previously created models. We devised general rules and went through a process of trial and error to obtain a satisfactory accuracy score:

- The number of filters for hidden layers ought to be a multiple of the image dimensions.
- ReLu is the activation function for any 2 dimensional convolution layer.
- Softmax is the activation function for any internal dense layer.
- A final max pooling layer, linearization layer and dense layer must be applied.
- The final dense layer should have a number of filters equal to or a multiple of 10 (the number of distinct classes).
- The kernel size ought to be 3, but can be increased for wider layers.
- Padding should be minimal but enough to avoid decreasing the input dimensions.

Following these guidelines, a satisfactory model for the plain dataset was constructed (Figure 2). The transformed model was designed by taking the plain model's architecture and adding additional layers (Figure 3). A hidden max pooling layer was added to decrease training time without compromising accuracy. The hybrid model was derived from the transformed model, with some deviations according to the previous guidelines (Figure 4) to better tune the hyperparameters.

Each model was then tested on the unseen data from the other two datasets to allow us to gain a better understanding on the generalizability and the specificity of each trained model. The code was run on colab.research.google.com with GPU runtime. Due to Google's limitation on GPU resources, eventually Google restricted access to its GPU runtime, which prevented us from training more models with different architectures, as training time became a significant bottleneck.

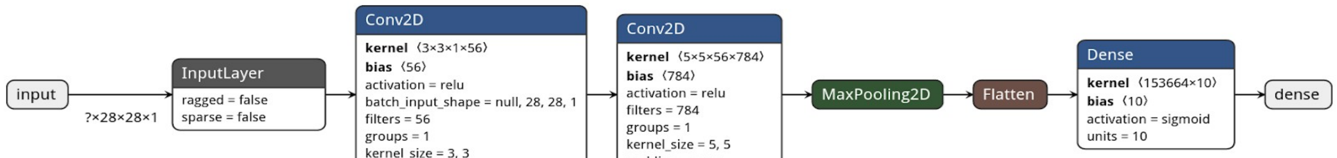
III. RESULTS

We chose the accuracy score as an overall measure of a model's performance on the image classification tasks based on the invariant properties of this metric, making it a suitable benchmark score for overall classification[4]. The accuracy is simply the number of correct predictions divided by total predictions.

TABLE II. TESTING ACCURACY SCORE MATRIX FOR EACH MODEL ON EACH DATASET

Models	Datasets		
	Plain	Transformed	Hybrid
Plain	98.75%	14.22%	56.65%
Transformed	20.70%	82.94%	52.55%
Hybrid	98.68%	80.35%	88.77%

As demonstrated in Table 2, each model achieved an accuracy score of over 80% on its respective testing data. Confusion matrices for the predictions of each model on its respective testing dataset were also generated (Figure 5). Models that were tested on transformed data, particularly the transformed model, had some trouble distinguishing between digits '2' and '5' and between digits '6', and '9'. This is further evidenced in the confusion matrix plots for the rows representing the cases that were labeled with these digits versus the model's predictions.



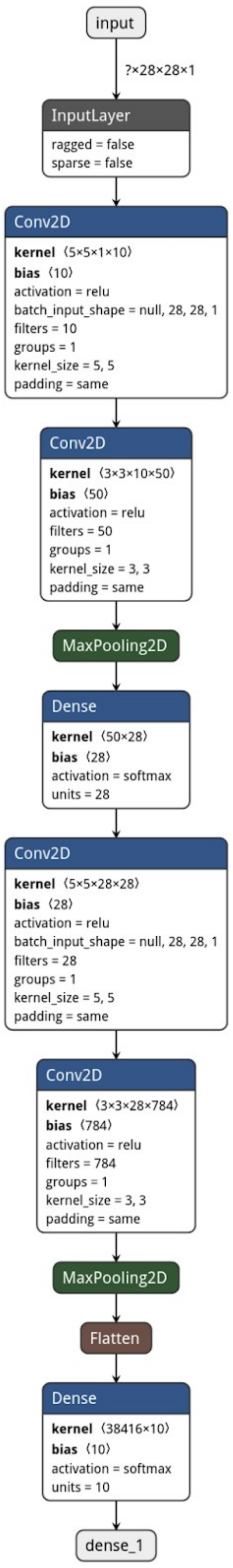


Fig. 3. CNN architecture for the modified model

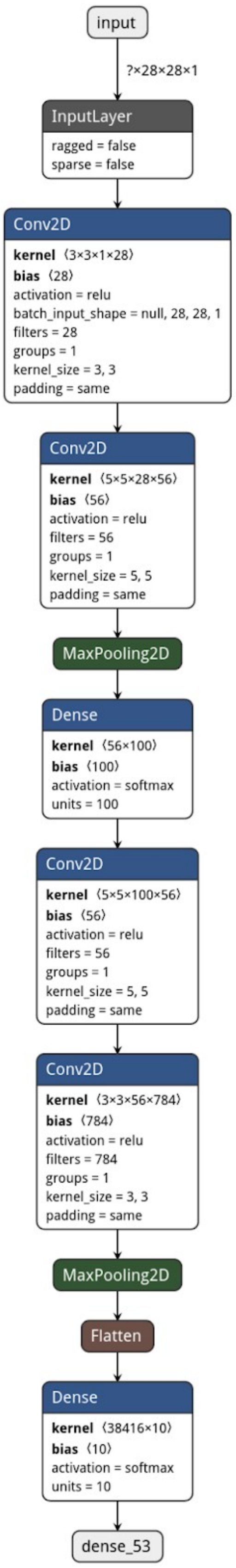


Fig. 4. CNN architecture for the hybrid model

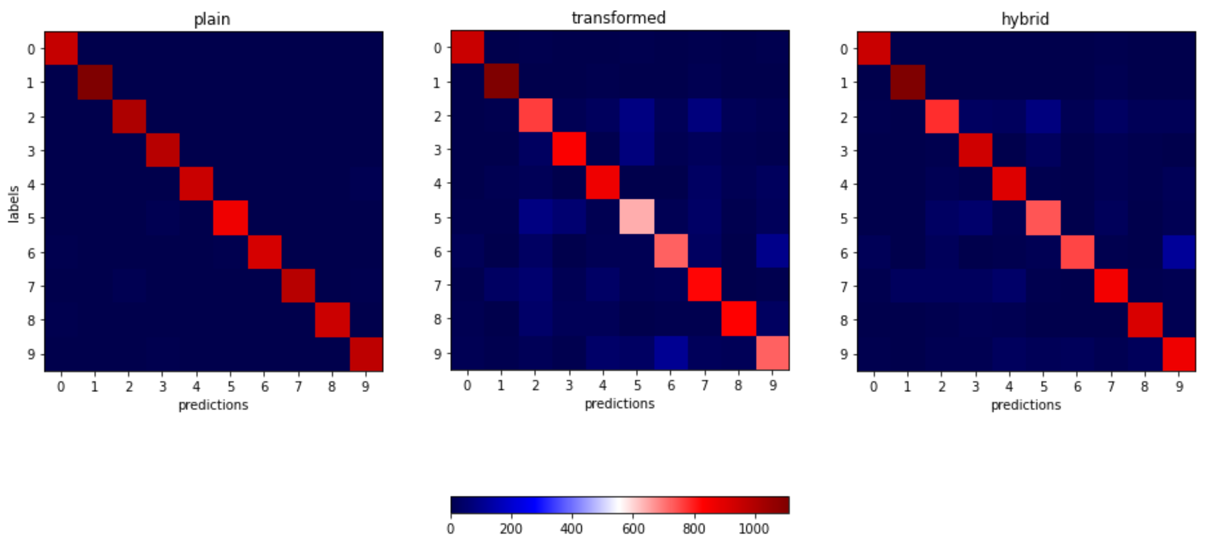


Fig. 4. Confusion matrices for the plain model, the transformed model and the hybrid model on their respective datasets

We tried visualizing the filters of the hidden layers, however due to the high number of dimensions, this step was abandoned as the visualizations were found to be too abstract for any meaningful interpretation.

IV. Discussion

In this study, we built three different image classification CNNs on three different datasets. The models were built such that the knowledge gained while structuring, training, and testing one was used as a starting point for the next. More specifically, the transformed model was built based on the architecture of the successful plain model, and the hybrid model was built based on the architecture of the transformed model. By calculating the accuracy scores, we were able to assess the overall performance of a model's prediction algorithm. A 98.4% accuracy rate was previously achieved by a simple CNN with no data preprocessing with a similar architecture [2]. By preprocessing our dataset and applying some transformations on a subset of the data, we were able to create a model that is as accurate on the plain dataset while making it more robust, as was shown in the Results section.

We can consider the transformations applied to be significant because our plain model received a very low accuracy score when tested on the augmented dataset. A score of 14.6% for an image classification algorithm that is trained on a dataset with 10 classes indicates predictions that are near-arbitrary. Indeed, even by visual inspection of a small sample of the transformed images, we noticed some samples were hard to guess by the examiner.

Counter-intuitively, the transformed model performed poorly on the plain dataset even though there is less noise and alterations to the images. This is a possible indication of over-fitting for the transformed model. Given the low accuracy scores for the plain and transformed models on each other's dataset, an accuracy score close to 50% for each on the hybrid dataset is unsurprising, as each model was able to accurately guess the subset which corresponded to its own testing set.

Interestingly, the hybrid model performed just as well on the plain and transformed datasets while achieving an 88.8% score on the hybrid data. This indicates that the hybrid model is more resilient to different noises and inconsistencies found within images which can be especially important for image datasets that are taken by human actors.

A significant difference between the transformed model's confusion matrix and the confusion matrix of the hybrid model is the shade of red on the (5,5) cell. This likely indicates that the rotation transformation may have been extreme.

Higher accuracy scores have been achieved on the MNIST dataset, requiring a committee of CNNs or by using other types of neural networks[3] and/or elastic distortion. However, this paper highlights the importance of augmenting a dataset in order to produce image classifiers that are able to handle a wider range of diversity in data. We can conclude that our hypothesis is partially correct—the noisier data required a deeper CNN to achieve a satisfactory accuracy score. This may be explained by the image having less contrast, as well as different orientations, which would require filters for more unique features for each class. On the other hand, we predicted that the hybrid model would perform poorer on its respective dataset than the other models would for their respective datasets. This is shown not to be the case, as the model performed with a score of 88.8% on its own dataset.

By partially augmenting a dataset, we demonstrated that it is possible to achieve better model performance without necessarily adding more layers to a CNN, which may affect training times.

References

[1] C. Lei, B. Hu, D. Wang, S. Zhang, and Z. Chen, "A Preliminary Study on Data Augmentation of Deep Learning for Image Classification," Proceedings of the 11th Asia-Pacific Symposium on Internetware, 2019.

[2] P. Y. Simard, D. Steinkraus, and J. C. Platt, "Best practices for convolutional neural networks applied to visual document analysis," Seventh International Conference on Document Analysis and Recognition, 2003. Proceedings., Aug. 2003.

[3] B. B. Романюк, "Training Data Expansion and Boosting of Convolutional Neural Networks for Reducing the MNIST Dataset Error Rate," Research Bulletin of the National Technical

University of Ukraine "Kyiv Polytechnic Institute", no. 6, pp. 29–34, 2016.

[4] M. Sokolova and G. Lapalme, “A systematic analysis of performance measures for classification tasks,” Information Processing & Management, vol. 45, no. 4, pp. 427–437, 2009.

[5] “sklearn.preprocessing.scale¶,” scikit. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.scale.html#sklearn.preprocessing.scale. [Accessed: 25-Apr-2021].

[6] “Module: tf : TensorFlow Core v2.4.1,” TensorFlow. [Online]. Available: https://www.tensorflow.org/api_docs/python/tf. [Accessed: 25-Apr-2021].