Data augmentation analysis for image classification

Gergely Dániel Németh[1], Hanliang Rao[1]

***Abstract*—Data augmentation is an important part of the preprocessing steps. When it comes to image classification, the publicly available datasets tend to be rather unbalanced. The importance of data augmentation is clear but the effect of it is rather uncertain. This experiment aims to analyse the different data augmentation methods (rotate, shear, elastic distortion) and their effect on the performance of different Machine Learning models (LR, SVM, KNN) on different image classification datasets (MNIST, Fashion-MNIST, CIFAR-10).**

***Index Terms*—data augmentation, image classification**

I. Introduction

Image classification is a widely used Machine Learning application. The basic concept is to choose the most probable predefined type for the upcoming images, e.g. realise if there is a dog in the image. There are several large image datasets with millions of images (Open Images, ImageNet) and a wide scale of state-of-the-art image classification methods (VGG, ResNet) openly available in many Machine Learning related APIs.

However, these image classification datasets are usually unbalanced. There are relatively few classes with a lot of samples and a long tail of smaller classes. Fig. 1 shows the characteristics of a dataset with approximately 9 million images, the Open Images Dataset [1]. Imagine a model trained on an unbalanced dataset, for example with one hundred thousand dogs and only a hundred wolfs. It is likely to incorrectly classify the new wolf images because according to the training data, it is more probable to be a dog.

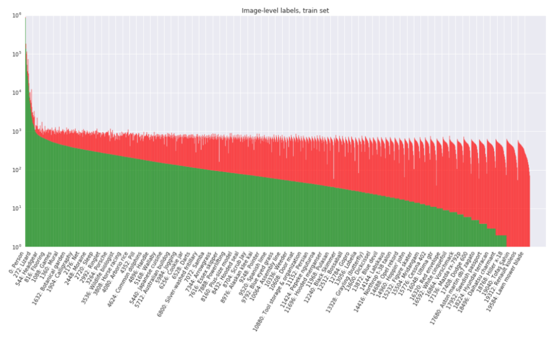

Fig. 1. Open Images class distribution statistics. The green represents the number of positive samples in each of the classes. Note that the y-axis is on a log scale. Image available at the Open Images Description page. Accessed: 29/10/2018, https://storage.googleapis.com/openimages/web/factsfigures.html

Data augmentation is a popular method to solve this problem. The concept is to create new images from the original so it keeps the essence of it. For example, a wolf is a wolf even if the image is mirrored vertically. Choosing the right data augmentation methods is important. Mirroring a wolf horizontally generates a wrong image, it is unlikely to take pictures of wolfs in that angle.

In this project, the authors investigate the effect of some popular image augmentation methods on the performance of some image classification algorithms.

II. Datasets

In the previous section, we described that there is a variable distribution of data in image datasets. However, in this experiment, we selected balanced datasets and generated unbalanced data by eliminating images. The datasets are available with separate training and testing sets but we used the OpenML database to fetch all the data in one and made our own train-test split.

These datasets are relatively balanced, however, we reduced the number of samples in the more wealthy classes to the amount in the smallest class, therefore, we created a dataset with exactly the same number of samples in each class.

We made an unbalanced dataset by reducing the samples in one class to a given proportion while keeping all the other classes in the same size. Data augmentation recreates images in the selected class to restore the original balanced dataset. Table I shows the number of samples in the classes as available in the dataset, after balancing, after splitting test data, after reducing the samples in the selected class and after restoring the balanced dataset. The actual training set here contains 50510 images while the original training set contains 60000 images.

TABLE I
Preprocessing sample size in the MNIST dataset. Test rate: 0.2, selected class: 4. Reducing proportion: 0.2

Label	Original	Balanced	Train	Reduced	Augmented
0	6903	6313	5051	5051	5051
1	7877	6313	5051	5051	5051
2	6990	6313	5051	5051	5051
3	7141	6313	5051	5051	5051
4	6824	6313	5051	**1010**	5051
5	**6313**	6313	5051	5051	5051
6	6876	6313	5051	5051	5051
7	7293	6313	5051	5051	5051
8	6825	6313	5051	5051	5051
9	6958	6313	5051	5051	5051

All 3 datasets contain around 60-70000 images in 10 classes. Each image is a relatively small square picture.

1) MNIST: The MNIST database contains images of 28x28 pixel hand-written digits, therefore the classes are 0, 1 ... 9[2]. The digits are located in the middle 20x20 pixel of the images.

[1]MSc students at the University of Manchester. {gergely.nemeth, hanliang.rao}@postgrad.manchester.ac.uk

2) CIFAR-10: The CIFAR-10 dataset contains images of 32x32 pixel objects[3]. The classes of the dataset are animals (dog, cat, horse, ...) and vehicles (automobile, airplane, ship, ...). Note that these are RGB images while the other 2 sets contain grayscale images.

3) Fashion-MNIST: This dataset is an MNIST inspired collection of fashion-related images [4]. The images are 28x28 pixel grayscale pictures of clothes like T-shirt, sandal, coat.

III. Data augmentation methods

There are several papers using large variety of data augmentation methods [5], [6], [7]. Some datasets are already enlarged with data augmentation. In this experiment, we collected some of the most popular methods to analyse. Some of the methods are quite simple and commonly used (e.g. mirroring), therefore it is hard to detect who introduced their usage in data augmentation. However, there are more complex ones with a clear origin, like elastic distortion [5].

A. Augmentation method examples

1) Mirroring and rotating: The simplest method is to mirror the image vertically or horizontally. Another simple method is to rotate the image with 90, 180 or 270 degrees. Using these methods keeps the size of a square image. However, this generates only 5 new images from every sample and many of those are not useful (remember the upside-down wolf from the introduction).

2) Scaling and crop: Selecting a part of the image. Removing one pixel from every edge of the image is still almost the same image. If the object is in the middle of the image, we can select and crop a smaller part of it containing the object. However, if we want to recreate the original image size we have to scale back the reduced image.

3) Rotating and crop: Rotating the image with a random degree (not 90, 180, 270) opens up the possibilities but has the same issue as the previous method: we have to scale and crop the image in order to reproduce the original image size.

4) Shearing and crop: Shearing tilts the image along x-, or y-axis. The function is very similar to the rotating thus the issue is the same: we have to scale and crop back to the original size.

5) Vignette: This is a different approach to the problem. While the previous methods used the same transformation in each pixel, here the idea is to reduce the brightness of the pixels far from the centre of the image, adding a circular effect to the image.

6) Elastic distortion: This is a relatively more complex method. The previous methods used a defined method to transform pixels. Here we define $\Delta x, \Delta y$ where $\Delta x(x, y) = rand(-1, 1)$. Every pixel (x, y) gets a value to shift in x-, y-axis. Before the transformation, we convolve these $Deltax$, Δy with a Gaussian of standard deviation σ. This creates an elastic effect using the right σ value [5].

B. Augmentation methods in the experiment

We summarized 6 methods here but in this experiment, we focus on only 3. We cannot use the first one, because none of the new images are real digits in the MNIST dataset. We do not use the scaling because it changes the centre of the images. Two of our datasets are centred, therefore the new images would not fit in the scheme. Neither is the vignette method applicable because both MNIST and Fashion-MNIST have a black background, therefore, the vignette does not do a significant difference; nor does it work if we invert the colours. If the images are black digits on a white background, and the new images have black edges, the models might optimize on this new feature.

Therefore the methods we use are *rotating*, *shearing*, *distortion* and *all* where we used all three methods on the images. We used the implemented methods of the Python library, Augmentor [8]. Fig. 2 illustrates the use of the library.

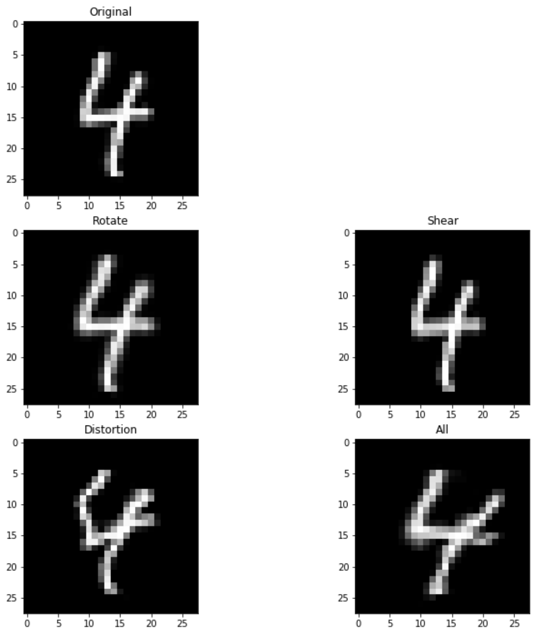

Fig. 2. Data augmentation types used in the experiment. Augmentor allows users to set maximum values and choose a value randomly between the limits. Max. rotation: 7^o, max. shear: 15^o, distortion grid: 5x5, magnitude: 2.

IV. Classifier models

Image classification is a widely investigated machine learning problem. The concept is to determine the class of the new images according to the previously labelled (classified) images, the training data. There are a large number of classification models. When one implements a new machine learning model, it is important to compare it with other state-of-the-art models, regarding the performance of different datasets, the model size and the training speed. Here, we used models described in the corresponding University course.

We used the previously implemented models from Scikit-learn [9]. Scikit-learn is a machine learning library for the Python programming language. Fig. 3 summarises the different machine learning models implemented in the library.

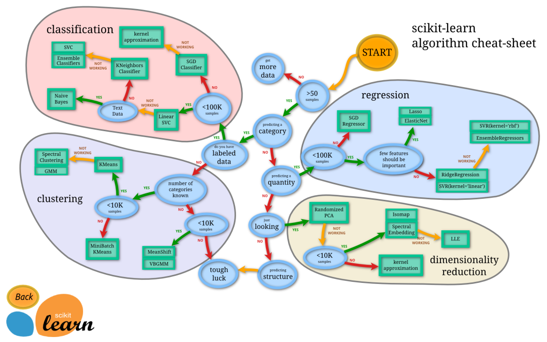

Fig. 3. Scikit-learn machine learning estimators. Different estimators are better suited for different types of data and different problems. Description from the Scikit-learn documentation. Accessed: 01/11/2018. http://scikit-learn.org/stable/tutorial/machine_learning_map/index.html

A. Logistic Regression (LR)

The first classifier is the logistic regression. This is a widely used linear model for classification using sigmoid function:

$$f(x) = \frac{1}{1 + e^{-(w^T x - t)}}$$

The loss function is the cross-entropy:

$$L(f(x), y) = -\{y \log f(x) + (1 - y) \log(1 - f(x))\}$$

and the learning algorithm is the Stochastic Gradient Descent. We used the Scikit-learn algorithm `SGDClassifier(loss="log")`.

B. Support Vector Machine (SVM)

The second classifier is the Support Vector Machine (SVM) [10]. The SVMs maximise the margin, using the hinge loss function

$$L_{hinge} = \max\{0, 1 - y_i f(x_i)\}.$$

Rather than using the `SVC` implementation in Scikit-learn, we used the `LinearSVC` which implements only the *linear* kernel function but scales better on more data. Therefore the kernel function in our experiment was:

$$K(x_i, x') = (1 + x_i^T x').$$

C. k-Nearest Neighbour Classification(KNN)

The final classification method we included in this experiment is the k-Nearest Neighbour Classifier (KNN). This method aims to find the most similar training samples for every new image to classify according to those neighbours' classes rather than using error functions to learn the nature of the data. We used 5 neighbours in our experiment.

V. Experiments

In the following experiments, we trained the LR and SVM models five times and measured the average and the standard deviation to improve the reliability of our results.

We measured the accuracy of the models on the selected class' test samples and the other 9 classes' test samples separately.

A. Comparing ML models and augmentation methods

We trained all 3 models on the balanced, the unbalanced, and the 4 different augmentation models. We used 0.2 proportion for generating unbalanced data here. We used the same model parameters for all training with all the 3 datasets and did not use hyperparameter tuning for the different tasks, therefore, our accuracy might fall below the best results available in other scientific experiments.

The unbalanced dataset reducing proportion is 0.2 in this case. We used the class label 4 for the MNIST dataset, label 8 - Bag for the Fashion-MNIST dataset and 5 - dog for the CIFAR-10. Table II summarises the accuracy of the different models trained in different datasets. MNIST contains $28 \cdot 28$ features while CIFAR-10 contains $32 \cdot 32 \cdot 3$. The training time for CIFAR-10 is higher than the MNIST or the FASHION-MNIST, therefore after seeing the low accuracy results for CIFAR-10, we focused on the different experiments rather than continuing to measure the other 3 augmentation for CIFAR-10.

To distinguish between accuracy on the selected class and other classes, we plotted the results in a graph. The x-axis represents the accuracy of the other 9 classes, meanwhile, the y-axis corresponds to the accuracy on the selected class (the class we used in the augmentation). Therefore in these graphs, the ideal model is in the top right corner. The bottom right corner represents a model with zero success on the selected class but a high accuracy on everything else. Thus, the worst model is near the bottom left corner of these graphs.

Fig. 4 shows the average performance of the Linear Regression models. As we can see, the unbalanced dataset performed the best on the other classes, however, this produced the worst accuracy regarding the selected class. On the other hand, the original dataset performed the worst on the other classes, but the best in the selected class. The 4 augmented datasets fall between these two pilots. Interestingly, the augmentation methods performed close on the other classes. Therefore, we can conclude that the shear and the distortion methods were more successful data augmentations than the rotate method and using all three of them.

In Fig. 5 we can see the performance of all 3 different methods. In general, we can say that the KNN was the best model regarding the task. However, here our aim is to compare different augmentation techniques not the models. Regarding the different augmentation methods, we can conclude that almost all of them caused results that are more like the original one, therefore augmentation achieve its goals. The performance of the augmentation using all three methods with SVM is an unexpected result. This is the only one that performed worse than the unbalanced dataset in both accuracies. Overall, we can say that using all three augmentation methods resulted in a lower level of accuracy than any single one.

B. Augmentation using different proportions of the original images

In this section, we focused on the unbalanced dataset. The goal here was to measure the critical number of samples in the class in order to reproduce the original data using data

TABLE II
ACCURACY OF THE DIFFERENT MODELS TRAINED IN DIFFERENT DATASETS USING DIFFERENT AUGMENTATION METHODS. SELECTED CLASS: MNIST: '4', FASHION-MNIST: 8 - BAG, CIFAR-10: 5 - DOG.

		MNIST			Fashion-MNIST			CIFAR-10		
		LR	SVM	KNN	LR	SVM	KNN	LR	SVM	KNN
Original	Other classes	0.8612	0.8537	0.9695	0.7733	0.7654	0.8473	0.2556	0.2869	0.3557
	Selected class	0.8731	0.8929	0.9683	0.9037	0.9023	0.9543	0.0150	0.0242	0.2300
Unbalanced	Other classes	0.8777	0.8640	0.9707	0.7832	0.7726	0.8476	0.2157	0.3046	0.3661
	Selected class	0.7775	0.7708	0.8613	0.8627	0.7633	0.8957	0.0800	0.0050	0.0250
Rotate	Other classes	0.8703	0.8533	0.9676	0.8012	0.7789	0.8472	0.2311	0.2474	0.3499
	Selected class	0.7989	0.9255	0.9588	0.8877	0.8204	0.9479	0.4908	0.2358	0.1983
Shear	Other classes	0.8690	0.8535	0.9679	0.7644	0.7560	0.8471			
	Selected class	0.8623	0.8477	0.9517	0.8686	0.8957	0.9429			
Distortion	Other classes	0.8695	0.8514	0.9690	0.7754	0.7896	0.8470			
	Selected class	0.8412	0.8843	0.9279	0.9369	0.8377	0.9464			
All augmentation	Other classes	0.8672	0.8633	0.9691	0.7907	0.7606	0.8471			
	Selected class	0.7889	0.6456	0.9160	0.8466	0.8600	0.9357			

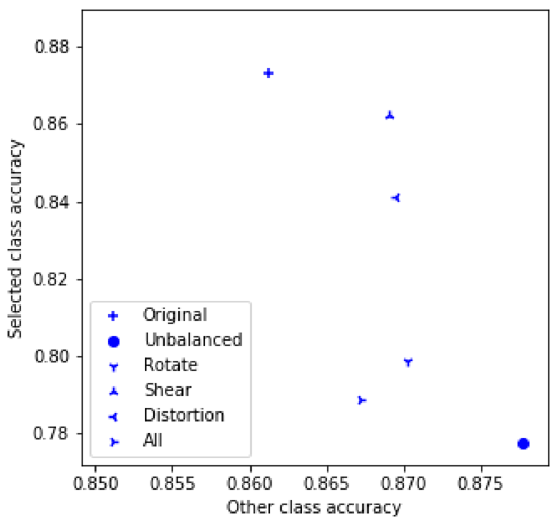

Fig. 4. LR model's accuracy on the MNIST dataset, proportion: 0.2, selected class: '4'

augmentation. We used the *shear* augmentation method on the MNIST dataset with the 4 class in this experiment.

In this experiment, we trained the models in different proportions of the original MNIST dataset. With 5051 samples in one class, the 0.001 proportion value means that the selected class in the unbalanced dataset contains only 5 images. This means that we created 5046 new images from the original 5. As we can see on Fig 6, data augmentation helps even in this case, however, the accuracy remains low. Taking a closer look at the next category, we can see that the model achieves a high increase in accuracy for 0.005 and 0.01. From 0.05 the augmented data's model performs on the same level as the original data. We can also conclude that in case of a relatively balanced dataset, using augmentation causes a negative effect.

C. Augmentation performance on different classes

Data augmentation heavily depends on the nature of the data. As we discussed above, there are a lot of cases when the augmentation method generates invalid data. In other cases, the newly generated data does not provide useful information. For example, rotating a circle-like 0 is not effective. Therefore in this experiment, we aim to measure the effect of the data augmentation on the different classes in the MNIST dataset. Fig. 7 shows the results using 0.05 proportion.

VI. DISCUSSION

Both Fig. 6 and Fig. 7 shows that if the model trained on the unbalanced data performs differently from the original then the training on augmented data makes the model more like the original one. However, if the unbalanced data's model performs relatively close to the original data's, the augmentation does not help the model to perform better. Therefore we can conclude that when one talks about the effect of using data augmentation, he should say that it performs more like a model trained on balanced data rather than saying that it performs better. In most cases, data augmentation is applied in a task because the model performed worse in the specific class, therefore, shifting the performance of that class more closely to the others causes a better overall performance hence it usually comes with a small decrease in the other classes' performance.

We also showed that data augmentation can be applied to different image classification problems, the effect of the augmentation is similar in most cases. The characteristics of the data augmentation do not change regarding the different classes.

Fig 6 shows, that data augmentation can help even the smallest sample sizes. Using the right augmentation can improve the accuracy of the model on the under-represented class samples making it 4 times more accurate. However, to achieve the balanced data's performance, the sample size of the unbalanced class needs to be at least 5% of the average. Using data augmentation on relatively balanced data (the unbalanced class is 50% of the average) changes the accuracies of the model into a less balanced-like performance.

The effect of the different augmentation methods can be different but the character of the change of the performance is the same. However, using too much augmentation can harm the model (Fig. 5 SVM All augmentation).

While the overall effect of the augmentation is the same for different models, each model has its best performing augmentation method and the performance of the models differs with

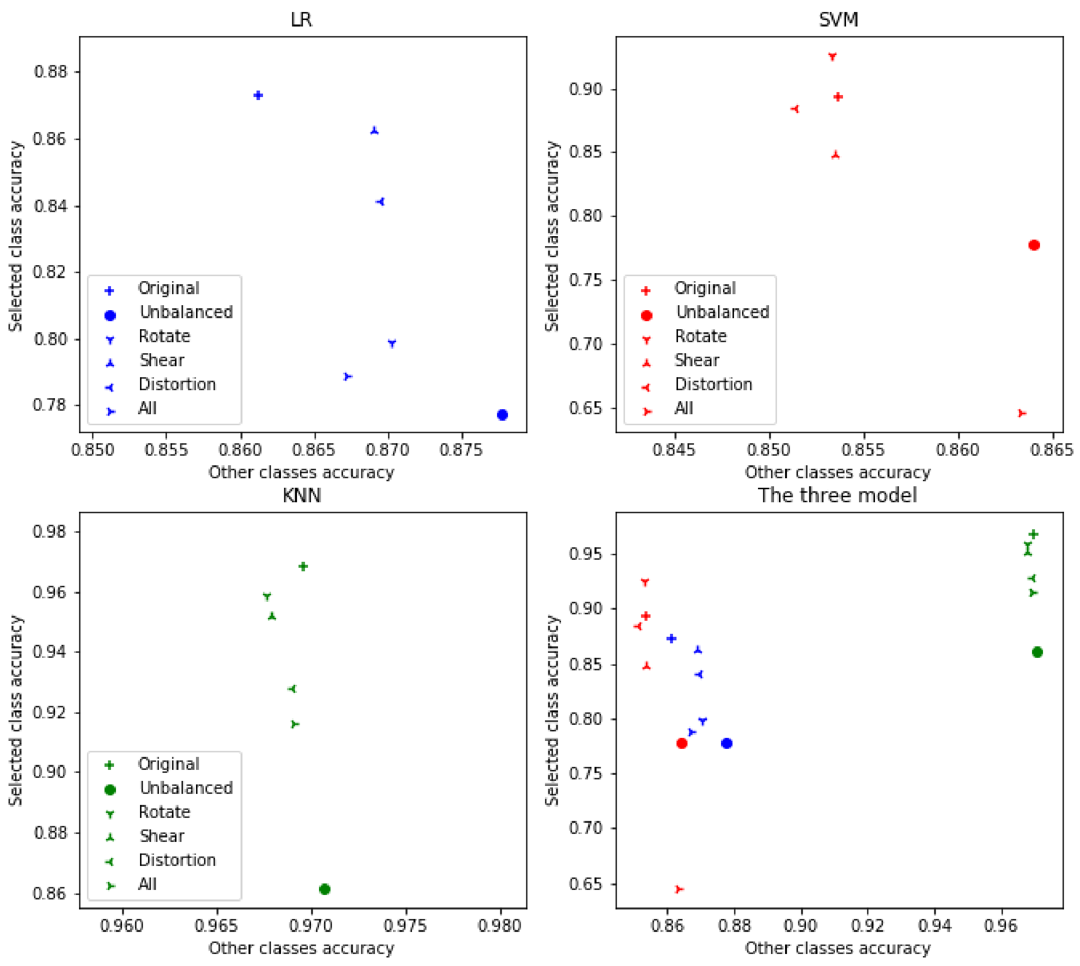

Fig. 5. The three models' accuracy on the MNIST dataset, proportion: 0.2, selected class: '4' All three models are run with default parameters. Linear Regression (LR, blue): `SGDClassifier(loss"log"`, Support Vector Machine (SVM, red): `LinearSVC()`, k-Nearest Neighbour (KNN, green): `KNeighborsClassifier(n_neighbors=5)`

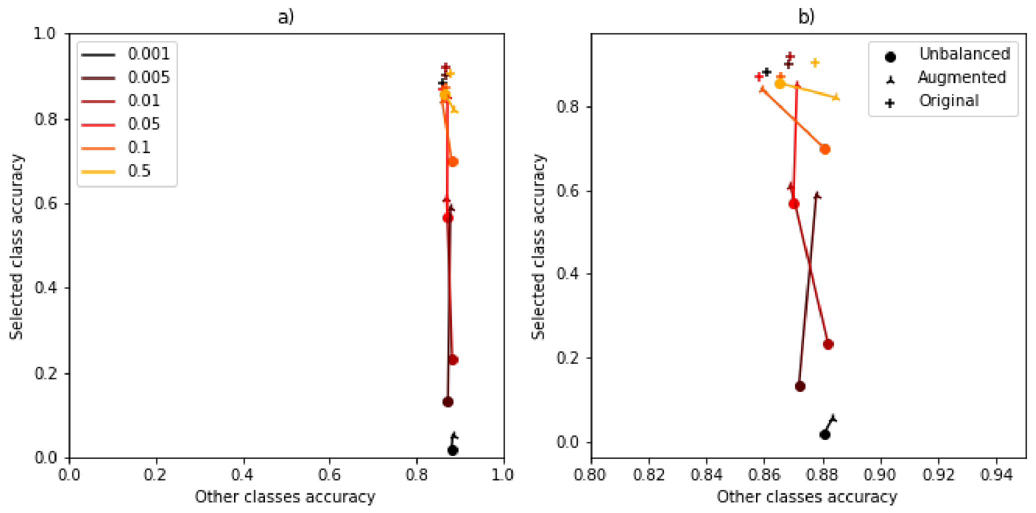

Fig. 6. Training LR models using different proportion of the selected class('4'). a) same-scale axis, b) x-axis limited to $(0.8, 0.95)$ Point and line colours represent the different proportion size, point markers represent the different datasets.

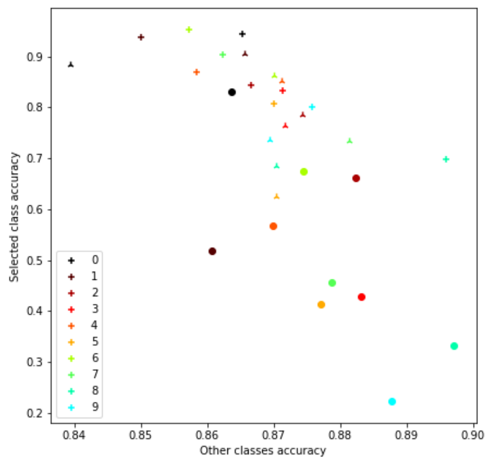

Fig. 7. Different classes used as the selected class in the MNIST dataset. Proportion: 0.05, augmentation method: shear

the different methods. The best performing method for the Support Vector Machine and the k-Nearest Neighbours model was the rotation while it was the worst (single) augmentation for the Logistic Regression. Shearing was a relatively good augmentation method for every model.

VII. Conclusion

The aim of the present research was to examine the effect of different data augmentation methods on different Machine Learning models. One of the significant findings to emerge from this study is that data augmentation does not make the model better, it rather makes the performance of the model more like the balanced data's model. The study also shows that balanced-like performance can be restored from only a small proportion of the original sample size, but it has limitations.

Although the current study is based on small datasets and relatively easy models, the findings suggest that using data augmentation with different Machine Learning models has the same impact on its performance. Source code for the experiments is available on GitHub[1].

References

[1] I. Krasin, T. Duerig, N. Alldrin, V. Ferrari, S. Abu-El-Haija, A. Kuznetsova, H. Rom, J. Uijlings, S. Popov, S. Kamali, M. Malloci, J. Pont-Tuset, A. Veit, S. Belongie, V. Gomes, A. Gupta, C. Sun, G. Chechik, D. Cai, Z. Feng, D. Narayanan, and K. Murphy, "Openimages: A public dataset for large-scale multi-label and multi-class image classification." *Dataset available from https://storage.googleapis.com/openimages/web/index.html*, 2017.

[2] 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.

[3] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Citeseer, Tech. Rep., 2009.

[4] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

[1]https://github.com/negedng/data-augmentation-analysis

[5] P. Y. Simard, D. Steinkraus, and J. C. Platt, "Best practices for convolutional neural networks applied to visual document analysis," in *null*. IEEE, 2003, p. 958.

[6] R. Wu, S. Yan, Y. Shan, Q. Dang, and G. Sun, "Deep image: Scaling up image recognition," *arXiv preprint arXiv:1501.02876*, 2015.

[7] 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.

[8] M. D. Bloice, C. Stocker, and A. Holzinger, "Augmentor: an image augmentation library for machine learning," *arXiv preprint arXiv:1708.04680*, 2017.

[9] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[10] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the fifth annual workshop on Computational learning theory*. ACM, 1992, pp. 144–152.