COMP 551 Assignment 4

Dominic Boutet, FeiYuan Zheng, Ryan Huang
McGill University

Reproducibility Summary

Scope of Reproducibility

Their work aimed to show how PCA remains relevant for dimensionality reduction for classification (1). Its speed and relatively good efficiency allows it to be a good candidate for prototyping before moving on to more efficient dimensionality reduction approaches if necessary (1).

Methodology

We used google colab to run the author's code which consisted of a bash script that called a python script. The models were written using Keras and Scikit-Learn, in the original paper they were all run on the CPU (1). We performed our tests using the GPU for the DAE because we do not believe that access to GPU acceleration is restricted enough to justify evaluating the runtime of a neural network without it.

Results

Our results show that the claims of the original paper still hold: PCA is much faster than DAEs and provide good accuracy. However, they also show that with GPU acceleration, simpler DAE models trained with bigger batch sizes and perhaps a subset of the dataset could also be viable options for prototyping. Those models are only 5 to 25 times slower than PCA but provide at least 4% more accuracy at equal number of dimensions. Our results show that the decision to chose a dimensionality reduction approach for prototyping on image classification models is less straightforward than they claimed in the original paper. It is more dependent on the number of runs required and the importance of the accuracy of the prototypes.

What was easy

The author's code was easy to use and the data was easy to collect. The paper was easy to read, the tests were easy to understand and the claims were straightforward.

What was difficult

The computational cost of their original experimental setup was very significant. It was a challenge to bring it down to a level that we could work with to perform our tests on google colab. The code was also poorly documented which made it harder to modify and handle.

Communication with original authors

We did not contact the original authors.

Submitted to ML Reproducibility Challenge 2020. Do not distribute.

1 Introduction

The curse of dimensionality is a significant problem in machine learning. There are multiple dimensionality reduction approaches each with their own strengths and weaknesses. Images are a common data type known to suffer from the curse of dimensionality. In fact, they often require the use of specific methods to extract features from the high-dimensional input in order to be used appropriately (e.g., CNN).

2 Scope of reproducibility

The objective of the paper was to verify the usefulness of PCA as dimensionality reduction approach for image input in a classification task performed by various classifiers (1). Their focus was on the K-NN classifier but they also tested logistic and quadratic discriminant analysis (1). They tested PCA against isomap, deterministic autoencoders (DAE) and variational autoencoders (VAE) (1). They performed their tests on 4 datasets, but we focused our efforts on the MNIST datasets (2; 3).

We wanted to reproduce a subset of their results: we focused on PCA and DAE applied to the MNIST datasets and limited certain components of their tests (see methodology).

We also wanted verify the following claims:

- PCA provides a faster way to reduce the dimensionality of the input to an extent that allows efficient classification, allowing it to be a good candidate for prototyping before moving on to more efficient dimensionality reduction approaches.
- Their implementation of DAE provided better results with lower dimensionality, but the computational cost was too significant.

Our main focus was on their implementation of the DAE. We wanted to investigate their architecture and design choices to make sure that their implementation was appropriate to compare with PCA. We reproduce their results with the K-NN and quadratic discriminant analysis, but perform our exploratory tests on the quadratic discriminant analysis classifier due to computational cost of their K-NN models.

3 Methodology

We used the author's code which we ran on google colab. We modified their code and added other DAE architectures to perform our tests. They did not use the GPU for their tests justifying it by arguing that access to GPUs requires funds that not everyone might have. We decided to use it regardless because we do not believe that this argument still holds given that we were able to perform our tests on the GPU without paying anything through the free version of google colab.

3.1 Model descriptions

All models were implemented in Python using Keras and scikit-learn libraries and their code was called through a bash script that initiated a main.py file with an argument parser. There are 2 components to the classification models: a dimensionality reduction and a classifier. In their paper they mostly reported the K-NN classifier which is non-parametric, but had the logistic regression which has D parameters, where D is the dimensionality of the input, and quadratic discriminant analysis which has $(C-1)\cdot((D\cdot(D+3)/2)+1)$ parameters where C is the number of classes, in their supplementary section. The dimensionality reduction algorithms that we will focus on are PCA which is non-parameteric and DAE which has a varying number of parameters depending on the architecture. Their architecture had $2\cdot(\sum_{i=0}^{2}N^2/2^{2i+1}+N/2^3\cdot D)$ parameters, where D is the number of latent dimensions and N is the number of pixels in the input. For the MNIST datasets which contain images that are black and white with 28x28 pixels, this was equal to: $2\cdot(403368+98\cdot D)$. Note that this includes the decoder, the encoder has half of those parameters.

3.2 Datasets

The MNIST datasets (digit and fashion) contain 60,000 images in their training set and 10,000 images in their testing set(2; 3). All images are black and white with 28x28 pixels shape (2; 3). There are 10 classes in the target and they all

are perfectly equally represented in both sets of the split (2; 3). The images represent handwritten digits and fashion objects respectively (2; 3). The images are relatively centered in the pixel array and of large enough size to take almost all the place in the pixel space (2; 3). The input data values range from 0 to 255 which represent the dark to white grading for the pixel representations. The data was normalized to 0-1 by dividing the input values by 255 (2; 3).

3.3 Hyperparameters

They used 5-fold cross validation for the classifiers and 90/10 split on the training loop of the DAE which was based on Adam with mean squared error loss and early stopping with patience of 10. Their base implementation of K-NN model's hyperparameter k was selected by to a random search over 60 different $k \in [1, \sqrt{n}]$ with n the size of the training set. Their DAE model was composed of a 3 layer encoder and a 3 layer decoder that mirrors the encoder's architecture. The number of units in each layer followed an exponential decay from the input size: (input_dim) / 2^{i+1} where i is the index of the layer. They used batch normalization between each layer, they used ReLU as activation function and sigmoid as output function, the latent units did not have an activation function (i.e., linear function). They reported exploring latent spaces of sizes ranging from 1 to 100. We used their hyperparameters everywhere unless specified.

3.4 Experimental setup and code

They performed 5 iterations of for each model and recorded the mean and standard deviation for the runtime of the dimensionality reduction algorithms and for the accuracy of the associated classifiers. The total computational requirements of their tests and models was significant. They performed their tests using Compute Canada resources which we do not have access to.

We decided to modify certain parameters of their tests to make them computationally feasible:

- We reproduced their findings for the MNIST datasets with PCA and DAE using K-NN exactly as they did but only for latent dimensions ranging from 1 to 10.
- We performed our tests using only 2 iterations instead of 5.
- We performed our tests using the quadratic discriminant analysis and only on the Fashion MNIST dataset.
- We used a subset of the dataset (5000 of the 60000) to perform exploratory tests over latent dimensions ranging from 1 to 10.
- We used the full dataset to reproduce our exploratory tests but with "jumps" of 2 dimensions over latent dimensions ranging from 1 to 9, and extended them to also include from 10 to 18.
- We also tried increasing the batch size: we tested their default (100) and 500.

We performed the following tests:

- Baseline for all our tests with both PCA and DAE (no changes except the ones mentioned above).
- Ablation of the batch normalization. (BNA)
- Ablation of the sigmoid output function (linear instead). (LO)
- Ablation of the sigmoid output function and input normalization. (LO+NoIN)
- Ablation of the batch normalization, sigmoid output function and input normalization. (BNA+LO+NoIN)
- Ablation of the first layer of the decoder and last layer of the decoder. (DAE 2 layers)
- Ablation of the first 2 layers of the decoder and last 2 layers of the decoder. (DAE 1 layer)

As they did, we focused on dimensionality reduction runtime and model accuracy. We performed ablation directly on the code by commenting out certain parts or by replacing a parameter input.

3.5 Computational requirements

We performed our tests on google colab using the GPU accelerator. We had a copy of their code on our drive which we accessed through the colab notebooks and called their bash script from within the notebook. The results were recorded within our google drive as csv files. The link to our notebook will provide means to replicate the bash script calls, but the code and results will be shared through the github repository for the project. To reproduce our findings through google colab, it will be necessary to download the github repository and upload it to a google drive or to find other means to connect the code to the colab notebooks as to call the bash script and store the results in an appropriate location.

The runtime varies a lot within google colab. However, to reproduce their findings with the full MNIST datasets using the K-NN classifier would require approximately 7h to 9h for PCA and 9h to 12h for the DAE. Most of the runtime is taken by the K-NN training, which means that using the GPU for the DAE will not change this significantly, but it will reduce the runtime by about 1h to 2h.

Using our settings which are significantly less computationally expensive, the runtime of experiments are more dependent on the dimensionality reduction algorithms than on the classifier. For PCA all tests take between 30s and 5min. For the DAE, the runtime varies a lot more. The exploratory tests and baseline take between 2min and 20min when using the GPU. The tests on the full dataset with "jumps" of 2 dimensions over latent dimensions ranging from 1 to 9 take between 20min and 45min and the ones ranging from 10 to 18 take between 35min and 1h20min. The variability in runtime is due to the effect of number of latent dimension, batch size, and of the architecture used on training time. To reproduce all our tests it would take approximately 70h to 90h of runtime using the GPU and considerably more without it.

4 Results

As they did in the original paper, we found that the PCA had similar accuracy as the DAE with significantly faster runtime when performed on the full dataset. This was shown both with the K-NN and QDA classifier. We found that their DAE architecture and training settings were good in the sense that ablations of certain components led to worse runtime and/or accuracy. However, some of our results show that it may be possible to modify certain components as to minimize runtime significantly while maintaining good accuracy.

4.1 Results reproducing original paper

As seen in the orignial paper, the DAE significantly outperforms the PCA with lower dimensionality but comes with a large computational cost (the GPU was not used to replicate their results).

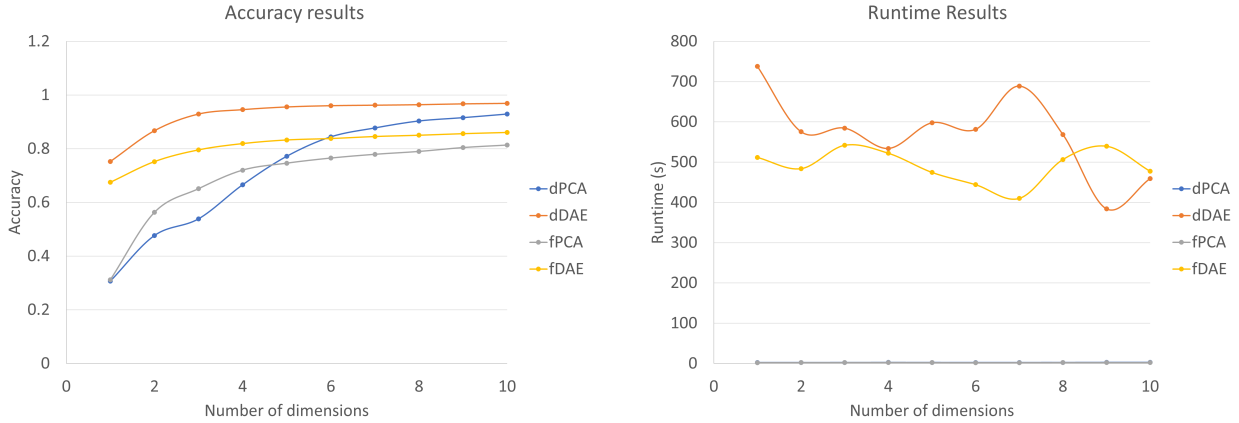

Figure 1. The accuracy (left) and runtime (right) of the PCA and DAE on the digit (d) and fashion (f) datasets.

The DAE performed well on both datasets with number of dimensions above 3, while the PCA needed more dimensions. To have an accuracy comparable to the DAE, the PCA needed at least 8 dimensions on the digit dataset and about 5 on the fashion dataset. However, the computational cost of the PCA remained constant in the 2s range regardless of the number of dimensions used, while the DAE remained above 350s in all cases.

4.2 Results from the ablation tests

4.2.1 Dummy tests

In our exploratory (Dummy) tests that were performed on 5000 data points, we found that certain components of the DAE were necessary to maintain accuracy, while others were necessary to avoid a significant increase in the runtime.

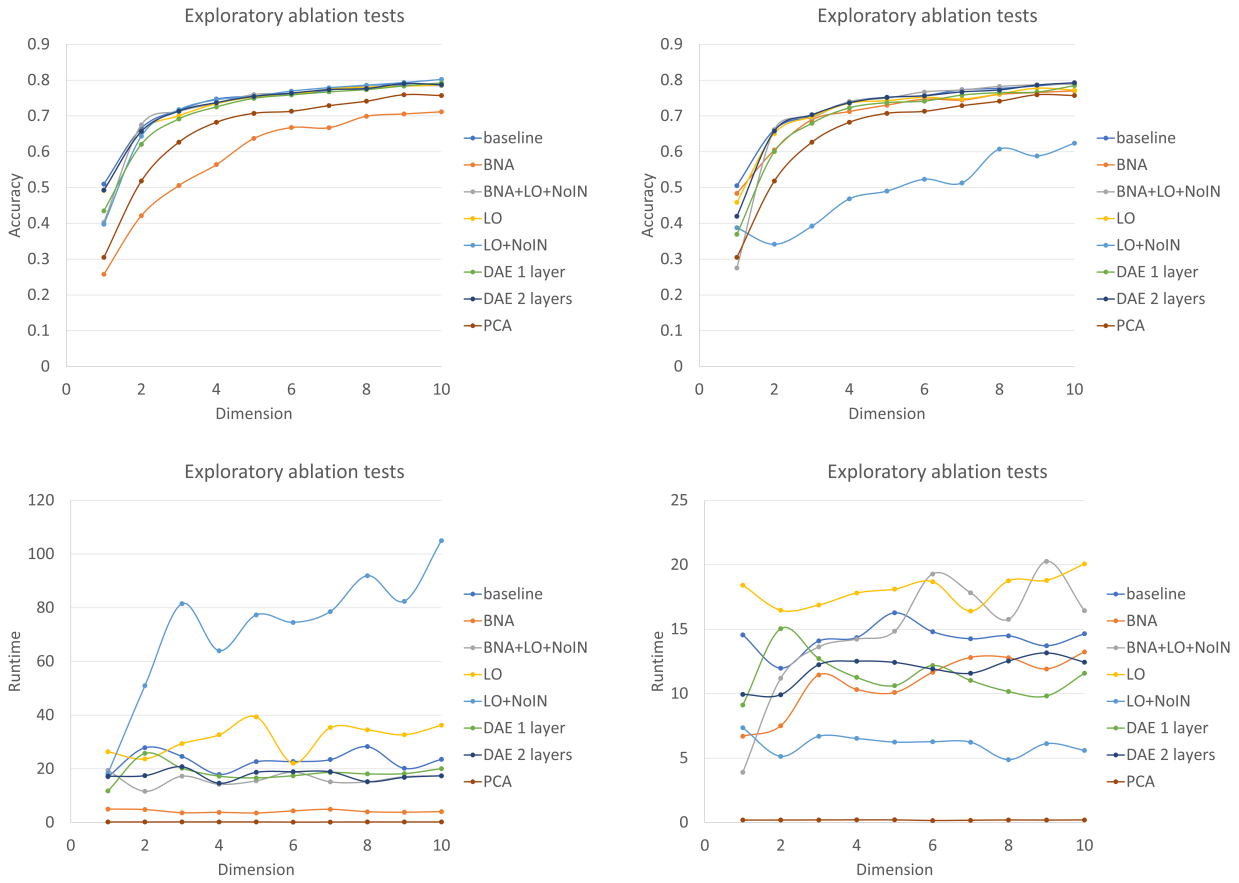

Figure 2. The accuracy (top) and runtime (bottom) for the baseline DAE and our tests.

While the accuracy for the DAE is lower in general with the "Dummy" tests due to smaller dataset, the patterns between PCA and DAE remained. The effects of the ablations seemed to be more significant on the runtime. The larger batch size did not seem to affect accuracy significantly, but did reduce the runtime by a factor of approximately 5. The combination of smaller dataset, usage of the GPU and increased batch size brought the runtime in the 10s to 20s range. The ablations seemed to have little effect on accuracy with 2 exceptions. The first exception batch normalization ablation (BNA) with the original batch size seemed to have drastically decreased accuracy, however it seemed to also have drastically reduced computational time. Given that this effect did not remain with a larger batch size (500), it is likely to be related to the scale of the learing rate. The second exception was the combination of linear output instead of sigmoid with the removal of the input normalization. With the original batch size the only effect of this ablation was to increase the runtime. However, with a larger batch size, the accuracy dropped significantly and so did the runtime. This is likely due to the input and output scale being 255 times larger in addition to the presence of the batch normalization. This specific combination of hyperparameters makes it hard for the network to find the right scale of the weights to allow such mapping. With smaller batches the early stopping is less likely to be triggered because it can adjust the gradient 5 times more frequently during each epochs compared to the batch size of 500. Overall, it appears that the ablations made the network harder to train and/or require more training iterations.

4.2.2 Full dataset tests

We expected our tests on the full dataset to mirror our tests on the "Dummy" dataset but with better accuracy and longer runtime. Even with the ablation of the different components, the accuracy of the DAE models was above the one obtained with the PCA model. In addition, we also saw the disappearance of certain aspects of the 2 exceptions mentioned earlier: the BNA and LO+NoIN did not have lower accuracy and runtime in the original batch size and the batch size of 500 respectively here. It is likely due to the increased number of gradient adjustments per epoch and the more complete dataset that allowed better learning and prevented the early stopping to be triggered as soon as it was

with the "Dummy" tests. The difference in accuracy between the PCA and the DAE was not different in the full dataset, it might appear to be, but it is only due to larger dimensionality in the plots.

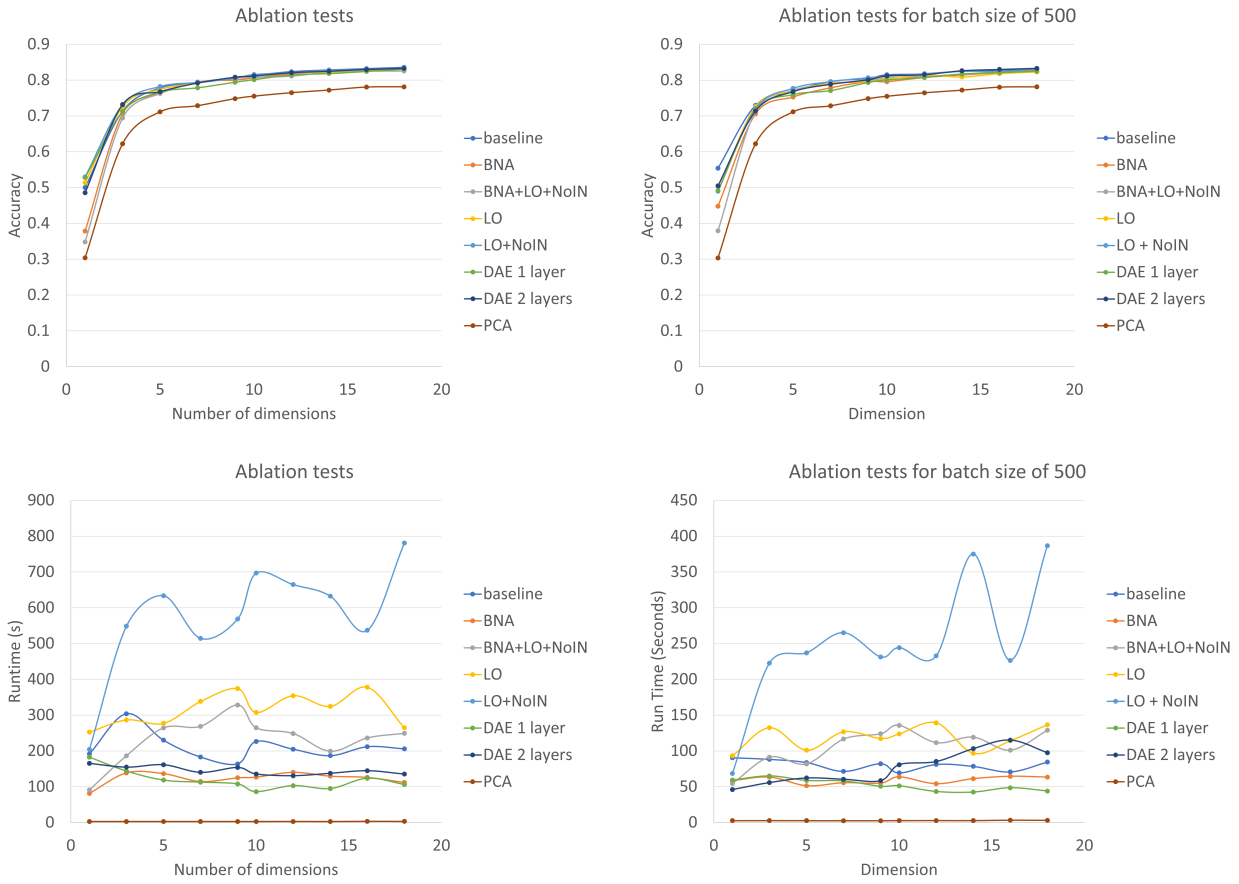

Figure 3. The accuracy (top) and runtime (bottom) for the baseline DAE and our tests.

The runtime of the DAE was still significantly larger than the PCA, but the addition of the GPU drastically reduced the runtime when compared to the runtime shown in figure 1. The batch size did not seem to affect the accuracy much here either, but it did reduce the runtime significantly bringing it down to the 50s to 100s range. Note that the DAE with only 1 layer performed almost as well as the baseline DAE and took only 50s to train on the GPU for 18 dimensions with the full dataset when the batch size was increased to 500, which is close to a decrease of 18 folds in the runtime when compared to the original paper and only 25 times more computationally expensive than the PCA. Furthermore, it has an accuracy of 82.445% versus 78.13% for the PCA.

5 Discussion

The claims from the original paper seem to hold in light of our results. PCA is much faster and does not result in significantly lowered accuracy, which makes it a good candidate for prototyping. However, our results show that it may be possible to reduce the runtime of the DAE significantly, allowing those "simpler" models to also remain good candidates for prototyping. Indeed, increasing the batch size reduced the runtime in all examples and did not affect accuracy in any way. Furthermore, either reducing the size of the dataset, which allowed the runtime to be as low as 10s, or limiting the size of the network (e.g., our DAE 1 layer), which allowed the runtime on the full dataset to be as low as 50s, could be considered as viable options when prototyping. It appears that the question one may ask before choosing between PCA and those simpler models is related to the accuracy/runtime tradeoff that remains. PCA is on average at least 4% less accurate when used with QDA than the simpler DAEs while being approximately 25x faster. Note that it may also be necessary to consider the nature of the data (how much non-linear features we expect it to contain) and the size of the dataset, but this seem to be out of the scope of the original paper and of ours. In a context

where prototyping requires a large number of runs and where the accuracy of the prototype runs is not that significant, PCA may be the more appropriate choice. However, in a context where prototyping does not require as many runs and where the accuracy may be significant for the prototyping, simpler DAEs with larger batch size and perhaps a reduced dataset may be the better option. Of course, the access to GPU acceleration is required to allow the viability of the DAEs, even the simpler ones, especially with respect to larger batch sizes (which benefit more from GPU acceleration). However, given that GPU acceleration is much more accessible nowadays (proven by our free access to it for this paper) we do not believe this to be a significant factor.

5.1 What was easy

The author's code was very easy to run for reproducibility, it was reduced to calling a bash script with the right arguments. The interpretation of the objectives, results and claims was also very easy and the tests were straightforward.

5.2 What was difficult

The main challenge was a computational one. They ran their tests on a Compute Canada server which allowed them to use a heavily optimized K-NN classifier. We had to find multiple ways to cut on the computational costs of the tests. The code was also not well documented which required a bit of time and effort to understand well enough to modify.

5.3 Communication with original authors

We did not contact the original authors.

Statement of Contributions

Dominic Boutet: Initialized the experimental setup in google colab, reproduced the original results, designed the tests, modified the author's code and wrote the colab notebook to run the tests.
Ryan Huang: Collected and analyzed the "Dummy" test data, produced the corresponding plots.
Feiyuan Zheng: Collected and analyzed the full dataset test data, produced the corresponding plots.

References

[1] Q. Fournier and D. Aloise,.

[2] Fashion MNIST source, https://github.com/zalandoresearch/fashion-mnist.

[3] LeCun, Y. Cortes, C. (2010), 'MNIST handwritten digit database'.