Fashion MNIST Classification Through a Combination of Machine Learning Techniques

Mehek Niwas

Machine Learning, MTH 4224, Spring 2024
Project 2

Abstract

Often used as a more challenging alternative to the MNIST dataset of handwritten digits, this project focuses on classifying Fashion MNIST into 1 of 10 classes of fashion items. While the dataset is favored by classification with convolutional neural networks, it presents an opportunity to improve models with other techniques in machine learning that require much lower computational resources than neural networks. In this project, an unsupervised technique called principal component analysis (PCA) was implemented to strategically reduce the dimensionality of the input image. In combination with PCA, quantitative descriptive analysis (QDA), a supervised machine learning method, was utilized to classify the images of reduced dimensionality. However, one of the main problems with PCA is the tendency of data to loose interpretability once information as been extracted. Since this project focuses on reducing the dimensionality of images, feature visualization was performed to allow a visual image of the result from the PCA reduction. To improve model performance, tuning experiments were conducted on PCA parameters such as explained variance proportions and comparisons of algorithm variations such as Incremental and Sparse PCA. After optimizing model performance for PCA, parameter tuning for QDA was conducted. Results indicate that many of the model variations for PCA and QDA have a subtle or non-substantial effect on performance for the Fashion MNIST dataset. The experimental data further suggests that the explained variance proportion has an influential effect on performance, as well as inconsistent runtime trends for some parameters. An accuracy of 81% was achieved with the finalized best performing model by the end of the project.

1 Introduction

With the use of different machine learning techniques, a prediction can be made on anything that can be vectorized. Since computers already store images as a matrix of pixel values, computer vision has become popular in the field. However, these images are often high dimensional and contain thousands of values, making it computationally expensive to train machine learning models to classify images.

Introduced in 2017 by Xiao et. al [9], Fashion MNIST is a dataset introduced by that has these qualities of high dimensionality and provides a challenging benchmark dataset when testing new algorithms and models. The grayscale images of 10 classes of fashion items are scaled down to 28x28 pixels, resulting in a feature space of 5 x 10Î884. This prompts the use of dimensionality reduction techniques to allow for faster computation by models. The usage of convolutional neural network (CNN) is often favored to classify the Fashion MNIST dataset, with Bhatnagar et. al [1] having achieved an accuracy of 92.54% using a two-layer CNN, and a 93.3% accuracy achieved with a more advanced CNN model by Seo & Shin [7].

However, Fashion MNIST can be used to evaluate the effectiveness of other machine learning techniques as well, such as qualitative data analysis (QDA). QDA method estimates mean vectors, covariance matrices, and prior probabilities to calculate a “discriminant function” used to classify samples. As described by Wu et. al [8], this method starts to fail when met with high dimensional data due to the sheer number of unknown parameters to estimate, a trait of QDA recognized by a multitude of studies [6]. This weakness in the QDA algorithm highlights the necessity of reducing the dimensionality of data. A particular unsupervised technique, called principal component analysis (PCA), is often used as a preprocessing step to transform data into smaller dimensions. First seen in literature by Pearson [5] and Hotelling [3], the goal of PCA is to reduce the number of dimensions without losing necessary information. It organizes information into principal components, which are calculated by computing covariance matrices and the eigenvalues and eigenvectors [4]. Principal components represent the directions of the data that explain a maximum amount of variance, allowing the percent of explained variance to be modified by a researcher. There are also many other variations of PCA, such as Kernel PCA to better represent non-linear trends, Incremental PCA to handle large datasets with batches, and Sparse PCA [10] to calculate more interpretable transforms of the data.

PCA can be used as a way to reduce dimensionality, feeding the transformed data into a QDA for classification. In this project, PCA and QDA algorithms were optimized to achieve the highest accuracy possible on the Fashion MNIST dataset.

2 Methods

2.1 Dataset & Preparation

The Fashion MNIST dataset was loaded through Keras, previously pre-processed and re-sized to 28x28 pixels. It contains 70,000 images labeled into 1 of 10 classes of fashion items: t-shirt/top, trouser, pullover, dress, coat, sandals, shirt, sneaker, bag, and ankle boot. The original dataset contains a 85% training split of 60,000 images, and 15% testing split of 10,000 images.

The original training and test splits were provided in the data loading function in Keras. The model functions for PCA and QDA were imported from sklearn. Since PCA requires all data for dimensionality reduction, the training and test splits were concatenated and flattened. This step was performed running the benchmark QDA model as a form of preprocessing for the PCA. This was especially necessary for the PCA algorithm to make accurate calculations.

After PCA dimensionality reduction, the data was randomly split into a training, validation, and testing split of 60% (42,000 images), 20% (14,000 images), and 20% (14,000) images respectively using the train_test_split function from scikit-learn.

Table 1 Fashion MNIST Inputs and Outputs

	Rows	Cols
X = Input [Image]	70,000	28x28x1
Y = Output [Fashion Classification]	70,000	1

2.2 Approach to Experimentation

Experimentation with both PCA and QDA was set-up by focusing on a singular hyperparameter or variable at a time. The previous best model was used as a constant in the subsequent experiments, for a total of 6 experiments. First, a benchmark model was established using only the QDA algorithm. Then, in Experiment 1, a model was evaluated using a baseline PCA algorithm in combination. Since performance improved, the majority of the subsequent experiments were dedicated to optimizing the PCA algorithm for the Fashion MNIST dataset. After tuning parameters for the PCA algorithm, the QDA algorithm was optimized to achieve highest accuracy, in Experiment 6.

Criteria and methods to determine the “best” model are described in the Section 2.3. Experiment 2 compares different explained variance ratios of 0.65, 0.75, 0.80, 0.90, and 0.95. Since the input images were flattened to a size of 784, this meant that each image had 784 components for PCA to convert into a set of principal components. Explained variance per component and cumulative explained variance was plotted (Fig. 1) to determine the explained variance ratios to test in this experiment.

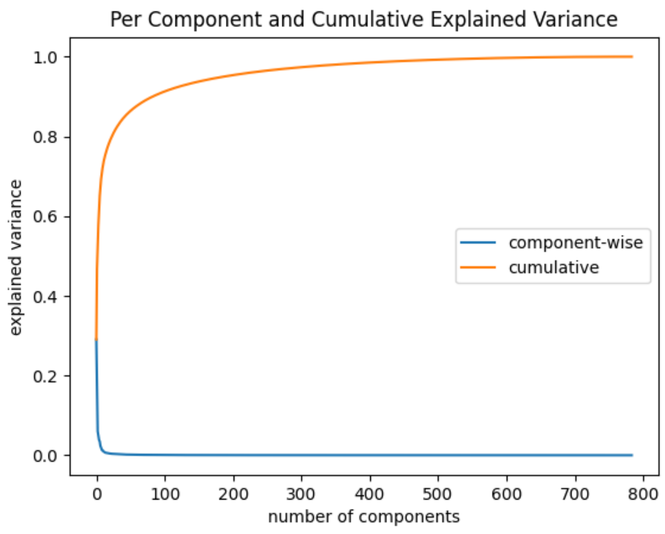

Fig. 1 Shows explained variance per component and cumulative explained variance for PCA

Experiment 3 introduces Incremental PCA with QDA and compares it to models with PCA. Experiment 4 compares different batch sizes of 100, 150, and 200 for Incremental PCA with the previous best performing PCA model. In Experiment 5, models with Sparse PCA and Kernel PCA were also compared with the previous best performing model. After determining the best model of experiments 1-5, the PCA parameter was finalized and used in Experiment 6 to optimize QDA by testing different regularization parameters of 0.005, 0.02, 0.05, 0.07, and 0.09.

To allow for accurate comparisons, all PCA models in Experiments 1-5 were run in conjunction with the benchmark QDA classifier. The QDA classifier was kept constant in order to accurately compare PCA parameters within experiments. Once PCA parameters were tuned, QDA was optimized in Experiment 6 by comparing the performance of models with different regularization parameters. The PCA parameters were kept constant in this last experiment of the project (Experiment 6). The planned experiment and model architecture progression of the project is shown in Fig. 2.

2.3 Evaluation & Selection of Models

As the goal was to achieve a high accuracy, overall accuracy and overfitting of the model was recorded. Overall runtime, PCA runtime, QDA runtime, and F1 scores were used as additional metrics to evaluate the models with similar overall accu-

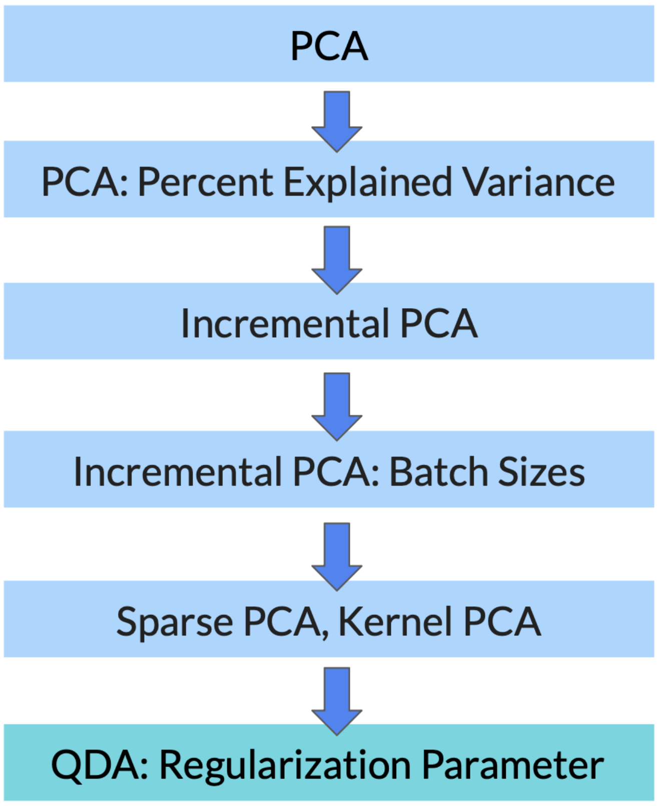
PCA
PCA: Percent Explained Variance
Incremental PCA
Incremental PCA: Batch Sizes
Sparse PCA, Kernel PCA
QDA: Regularization Parameter

Fig. 2 Planned Experiment and Model Architecture Progression

racies. A confusion matrix of both the training and validation predictions was plotted for every model of every experiment.

2.4 Feature Visualization

Feature visualization was also important in this project to observe the effect of the PCA dimensionality reduction on the original image. For each model utilizing PCA, an inverse transform of the resulting image was performed to reorganize the new data as a 28x28 image that could be viewed by the researcher. Although an inverse_transform function was supplemented in sklearn for the models, there was not an inverse_transform function for the Sparse PCA variation. As a result, this had to be done manually for the Sparse PCA variation. With reference to calculations by Chehade & Shi [2], the inverse transform was performed by computing the matrix multiplication of the reduced data output of Sparse PCA with its respective components. Nonetheless, it was difficult to verify whether the manual inverse transform had been applied correctly.

When comparing the original and transformed images, the pixel values were mapped to a color map in matplotlib instead of gray scale. Virdis, the default color map, was used for the images in this project to be able to more easily identify differences and variation in pixel value. This is more difficult to identify when the images are in grayscale.

3 Results

Results of the project indicated that some variations of parameters in PCA and QDA have more subtle effects on accuracy, overfitting, and runtime than others.

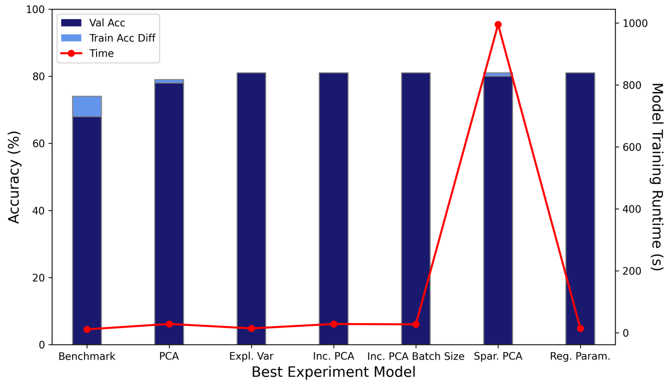

Fig. 3 Best Experiment Models vs. Accuracy & Runtime. Includes Benchmark Model results

Overall plot of best models from each experiment versus accuracy and runtime (Fig. 3) shows how runtime fluctuates, and varies from model to model. The plot also indicates the estimated overfitting of each model through the highlighting difference between the accuracy achieved on the validation and training data splits. The trends suggest that overfitting is not heavily affected by the different model variations in a majority of the experiments. Since many of the model variations in a majority of experiments had similar accuracies, selection of the best model was based on some of the additional metrics discussed in Section 2.3. In experiments 2-5, the best model was determined by accuracy and runtime. However, in experiment 6, the F1 scores of individual classes were also considered to select the best model.

3.1 PCA Parameter Tuning

Experiments 1-5 optimized PCA parameters and compared PCA types to improve model performance. Some experiments did not improve model performance, so the parameter tested for those experiments was not implemented in the final model architecture. The best performing model from each of these experiments is highlighted in red (see Tables 4, 5, 6) valuation. The best performing model for experiments for parameters tested that were implemented in the final model architecture are highlighted in green (see Tables 2, 3).

Table 2 Experiment 1 - Benchmark vs. PCA Results

Model	Train Acc.	Val Acc.	QDA Runtime
Benchmark	74%	68%	10.7 s
Exp. 1: PCA	79%	78%	28.1 s

In experiment 1, the model with PCA preprocessing was compared with the benchmark model of only a QDA classifier. Since the PCA model performed much better, it was selected as the best performing model and used in the next experiment. Table 2 shows that the runtime of the QDA classifier decreased after implementing PCA as a preprocessing step to reduce dimensionality.

Table 3 Experiment 2 - PCA Explained Variance Results

Model	Train Acc.	Val Acc.	PCA Runtime
expl var = 0.65	72%	72%	20.1 s
expl var = 0.75	77%	76%	25.2 s
expl var = 0.80	81%	80%	19.6 s
expl var = 0.90	81%	81%	13.6 s
expl var = 0.95	81%	80%	21.0 s

Model variations in experiment 2 had the largest difference in performance of all 6 experiments. The table of results indicates that explained variance has an influence on model accuracy. However, it is difficult to describe a trend of the effect of explained variance on runtime from Table 3. The best performing model of this experiment had a explained variance proportion of 0.90, meaning that the dimensions were reduced from 784 components to 84 components with the PCA algorithm.

Table 4 Experiment 3 - PCA vs. Incremental PCA

Model	Train Acc.	Val Acc.	PCA Runtime
PCA	81%	81%	13.6 s
Incremental PCA	81%	81%	27.6 s

Experiment 3 compared an Incremental PCA model with the best performing PCA model from the previous experiment. The best explained variance portion of 0.90 from the previous experiment was kept as a constant in both models. The main difference in the algorithms between Incremental PCA and PCA is that Incremental PCA computes based on batches of components – designed for larger datasets. Usage of batches may sometimes introduce additional noise, so Incremental PCA had the potential to have an interesting effect on the performance of the model. However, as shown in Table 4, the results of experiment 3 indicate that Incremental PCA did not have a different effect on model performance and unnecessarily increased runtime. Since a batch size of 200 was used for the Incremental PCA, an additional experiment (Experiment 4) was conducted on different batch sizes to investigate if a difference in batch size could influence performance.

Table 5 Experiment 4 - Incremental PCA Batch Size

Model	Train Acc.	Val Acc.	PCA Runtime
batch size = 100	81%	81%	51.66 s
batch size = 150	81%	81%	26.51 s
batch size = 200	81%	81%	27.63 s

The results of experiment 4 from Table 5 indicated that batch sizes of incremental PCA had no effect on model performance. While a batch size of 150 was determined to be the best performing model of the experiment due to the shortest runtime, Incremental PCA was not implemented in the next experiment or the final model architecture.

Table 6 Experiment 5 - PCA Variations - Sparse & Kernel

Model	Train Acc.	Val Acc.	PCA Runtime
Sparse	81%	80%	994.64 s
Kernel	n/a	n/a	out of memory

Sparse PCA and Kernel PCA were implemented in models for experiment 5. Both continued the use of 0.90 explained variance proportions. The results Sparse PCA seemed to have a negative effect on model performance and had the highest runtime of the entire project of 16 minutes (Table 6). Kernel PCA was also implemented, however, it could not be utilized due to the excessive amount of computer memory it required. This was expected since Kernel PCA is generally more suited towards smaller datasets. While it is possible to use Kernel PCA through batches, it was not explored further in this project. Due to the lack of improvement from experiments 3, 4, and 5, the normal PCA with 0.90 explained variance was determined to be used in the last experiment. This finalized the PCA tuning for the project.

3.1.1 Experiment Feature Visualization After applying PCA to the original images, an inverse transform of the reduced dimensions was performed and re-sized for feature visualization. This process, detailed in Section 2.4, was repeated for each model in each PCA parameter tuning experiment. Although the transformed images seemed very similar for the models within experiments, some differences can be seen when comparing between experiments. In Fig. 4, the bottom two images resulting from the best performing PCA model in Experiment 1 (explained variance = 0.90) and best performing Incremental PCA model in Experiment 3 (batch size = 150) seem

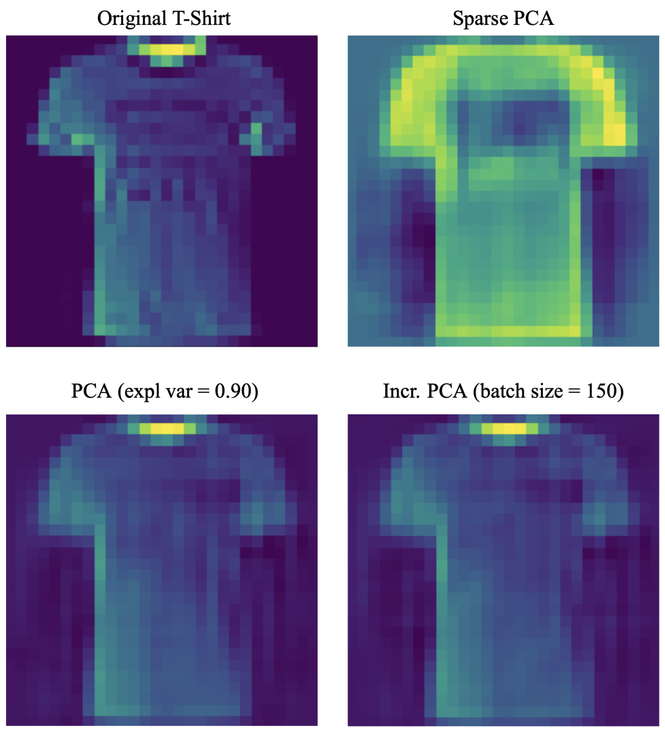

Fig. 4 Original and Transformed PCA images from Experiment 2, 4, and 5. (Note: True images and transformations are grayscale. See Section 2.4 for methodology)

almost identical. When looking at the images closely, a discrepancy can be identified in the middle section, and middle right side of the T-shirt. This shows that even though the results of both models were the same, the behavior of the transformations are slightly different. Additionally, all three transformed images for PCA, Incremental PCA, and Sparse PCA show some form of blurring of the original T-shirt's logo or design in the middle section. Something else interesting is the gradient variation in the background of the transformed T-shirt images, not present in the original image. In the image for Sparse PCA, the the values seem to be inverted, which is interesting considering the classification accuracy resulting from Sparse PCA was very similar to the other models. While the method of using PCA for dimensionality reduction decreases the interpretability of the model, feature visualization with images can help show some insight into the model's behavior.

3.2 QDA Parameter Tuning

The last experiment, Experiment 6, was conducted to test different regularization parameters for QDA to improve model performance.

The PCA tuning was concluded from previous experiments. The best PCA model (0.90 explained variance) was kept as a constant for Experiment 6. In this experiment, parameter tuning for regularization in the QDA algorithm was performed.

Table 7 Experiment 6 - QDA Regularization Parameter

Model (Reg.)	T Acc.	V Acc.	F1	QDA Runtime
reg = 0.005	81%	80%	0.48	0.42 s
reg = 0.02	81%	80%	0.50	0.32 s
reg = 0.05	81%	81%	0.52	0.29 s
reg = 0.07	81%	81%	0.53	0.30 s
reg = 0.09	81%	81%	0.53	0.43 s

Shown in Table 7, training and validation accuracy was almost identical among all regularization parameters. To determine the best performing model, the F1 scores of the individual classes were considered. Most of the classes had nearly identical F1 scores for all of the model variations, except for the Shirt class which was also the class that models usually performed the worst in across all models of the project. Models with regularizer parameters of 0.07 and 0.09 achieved the highest Shirt class F1 score of 0.53. Since the 0.07 model had a shorter QDA runtime than the 0.09 model, it was chosen as the best performing model. 0.07 regularization parameter was used for the final model architecture.

After Experiment 6, the architecture of the best performing model overall was finalized. The finalized model was evaluated on the test data set to provide a final estimate on the generalization ability of the model.

4 Discussion & Conclusion

The finalized architecture was achieved by evaluating a different parameter of PCA or QDA in each experiment (Fig. 5). The best performing model was determined in each experiment and used for the finalized model architecture if it showed an improvement on the previous experiment model. A QDA only benchmark model was run in the beginning of the project, achieving a validation accuracy of 68% with a training runtime of around 11 seconds. The best model from the end of the project achieved the highest validation accuracy of 81% and highest Shirt class F1 score of 0.53, with a training runtime of approximately 14 seconds. The finalized model architecture consisted of: PCA preprocessing, 0.90 explained variance to reduce 784 components to 84, and a QDA regularization parameter of 0.07. This model was able to achieve a final accuracy of 81% on the testing data split, which indicates the estimated accuracy that should be expected if data of the same format is used to classify fashion images into 1 of 10 classes. The individual class F1 scores achieved on the testing data split was 0.79 for T-shirt, 0.95 for Trouser, 0.77 for Pullover, 0.84 for Dress, 0.76 for Coat, 0.77 for Sandal, 0.52 for Shirt, 0.79 for Sneaker, 0.94 for Bag, and 0.90 for Ankle Boot.

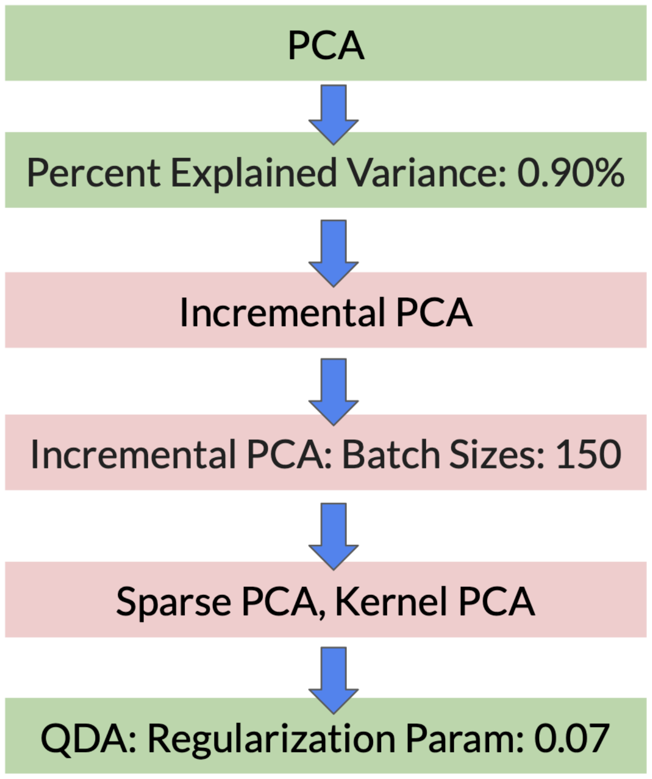
PCA
Percent Explained Variance: 0.90%
Incremental PCA
Incremental PCA: Batch Sizes: 150
Sparse PCA, Kernel PCA
QDA: Regularization Param: 0.07

Fig. 5 Finalized Model Architecture Progression through Experiments

Some model and experimentation improvements that can be made in the future include the introduction of more variations of PCA, combining PCA with different machine learning techniques such as kNN, LDA, or neural networks. Additionally, it could be a point of interest to use these methods on a more challenging dataset with higher resolution to observe greater differences in model variations.

Overall, the project was able to demonstrate how model performance changes through accuracy, overfitting, runtime, and individual class accuracy from optimization of components in PCA and QDA. Certain variations like batch size in Incremental PCA, and Sparse PCA did not indicate a substantial difference in model performance, while other variations such as in QDA regularization parameter indicated more subtle differences.

5 Dataset Availability Statement

The dataset used in this project can be directly accessed trhough Keras, and is publicly available on Kaggle. Link: kaggle.com/datasets/zalando-research/fashionmnist.

References

[1] Bhatnagar, S., Ghosal, D., & Kolekar, M. H. (2017). Classification of fashion article images using convolutional neural networks, *2017 Fourth International Conference on Image Information Processing (ICIIP)*. https://doi.org/10.1109/ICIIP.2017.8313740

[2] Chehade, A., & Shi, Z. (2019). The sparse reverse of principal component analysis for fast low-rank matrix completion. *Arxiv: Computer Science, Machine Learning*. https://doi.org/arXiv:1910.02155v1

[3] Hotelling, H. (1933). Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, *24*, 417–441. https://doi.org/10.1037/h0071325

[4] Jolliffe, I., & Cadima, J. (2016). ”principal component analysis: A review and recent developments”. *Adaptive data analysis: theory and applications*, *374*. https://doi.org/10.1098/rsta.2015.0202

[5] Pearson, K. (1901). On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, *2*, 559–572. https://doi.org/10.1080/14786440109462720

[6] Qin, Y. (2018). A review of quadratic discriminant analysis for high-dimensional data. *WIREs Computational Statistics*, *10*. https://doi.org/10.1002/wics.1434

[7] Seo, Y., & Shin, K.-s. (2019). Hierarchical convolutional neural networks for fashion image classification. *Expert Systems with Applications*, *116*, 328–339. https://doi.org/10.3390/s22239544

[8] Wu, Y., Qin, Y., & Zhu, M. (2019). Quadratic discriminant analysis for high-dimensional data. *Statistica Sinica*, *29*, 939–960. https://doi.org/10.5705/SS.202016.0034

[9] Xiao, H., Rasul, K., & Roland, V. (2017). Fashion-mnist: A novel image dataset for benchmarking machine learning algorithms. *Arxiv: Computer Science, Machine Learning*. https://doi.org/arXiv:1708.07747v2

[10] Zou, H., Hastie, T., & Tibshirani, R. (2004). ”sparse principal component analysis”. *Journal of Computational and Graphical Statistics*, *15*, 265–286. https://doi.org/10.1198/106186006X113430