A Comparative Study of Overfitting Strategies in Multilayer Perceptrons

Rabbani Nur Kumoro
(21/472599/PA/20310)
Department of Computer Science and Electronics, Universitas Gadjah Mada
Building C, 4th Floor North
Yogyakarta, Indonesia
rabbani.nur.kumoro@mail.ugm.ac.id

***Abstract* — Deep learning has seen extensive applications across various domains, encompassing search engines, big data, natural language processing, computer vision and the automated speech recognition. This study conducts a comprehensive exploration of overfitting mitigation strategies within the context of Multilayer Perceptrons (MLPs) using the Fashion MNIST dataset. The study investigates the effectiveness of methods such as undersampling, feature selection, data transformation, dropout layers, regularization, and early stopping. Notably, the results highlight the significance of regularization as the most effective method for mitigating overfitting and enhancing model generalization. This finding holds promise for practitioners aiming to optimize neural network models in diverse application domains.**

Keywords* — *Artificial Neural Networks, Deep Learning, MNIST, Multilayer Perceptron, Overfitting

I. Introduction

In the field of deep learning, significant progress has been made in both theoretical developments and practical implementations. Typically, the objective is to minimize both training error and generalization error in neural networks. The challenges posed by overfitting during neural network training have a substantial impact on the model's ability to generalize effectively **[1]**. A crucial determinant contributing to overfitting in deep learning is the model's complexity, which depends on the number of hyperparameters it incorporates.

The most prevalent application of neural networks is the Multilayer Perceptron (MLP), an architecture that organizes perceptrons into layers, as illustrated in **Figure 1**. These neural networks undergo training using backpropagation, a technique that employs gradient descent to minimize errors in the network's output by adjusting weight values **[1]**.

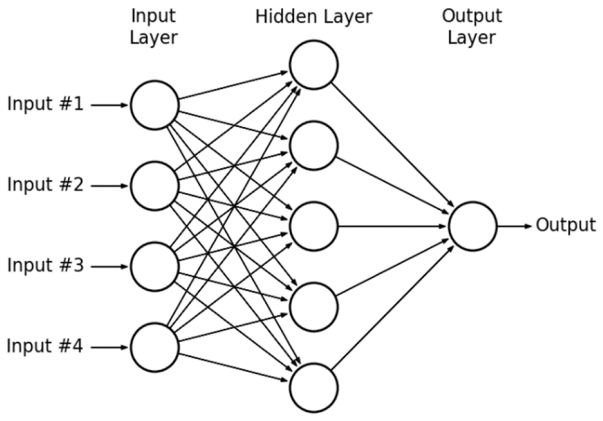


Figure 1. Multilayer Perceptron Architecture

Essentially, the MLP acquires an understanding of a function through the assimilation of data. When presented with a set of features and a target variable, it gains the capacity to grasp a nonlinear function. MLP demonstrates its proficiency by iteratively refining the input at each layer. Its strength lies in its ability to amplify crucial aspects of the input that are vital for tasks such as image classification, while simultaneously diminishing irrelevant details. This characteristic proves immensely advantageous in tasks like image classification, where the intricate process of object identification can be deconstructed into more manageable subtasks, such as detecting curves and edges. An important aspect and advantage of MLP is that this decomposition is autonomously acquired through the training process.

Overfitting remains a significant challenge in deep learning, impacting a model's ability to generalize effectively **[2]**. It occurs when a model becomes excessively complex, often as a result of an abundance of hyperparameters, and starts to memorize noise and idiosyncrasies in the training data rather than capturing the underlying patterns. This phenomenon hinders the model's capacity to extract meaningful information from new or unseen data, undermining its real-world applicability. Mitigating overfitting is a crucial concern in deep learning, particularly in widely employed neural network architectures like the MLP.

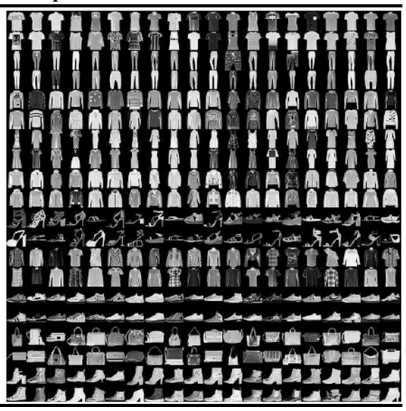
This study aims to identify optimal techniques for mitigating the risk of overfitting in artificial neural networks, with a specific focus on MLP deployed in the challenging domain of image classification, particularly using the widely recognized Fashion Modified National Institute of Standards and Technology (MNIST) dataset **[2]**. The primary objective is to conduct a comparative analysis to elucidate how various strategies for combating overfitting might influence the performance of MLP. Through the systematic testing and adjustment of methods such as undersampling, feature selection, data transformation, dropout layers, regularization, and early stopping, the aim is to ascertain the most effective approaches for countering overfitting in terms of accuracy and loss reduction.

II. Dataset

Fashion MNIST is a dataset curated from product images found on Zalando's website **[2]**. It comprises a total of 70,000 grayscale images, which are further divided into 60,000 samples for training and 10,000 samples for testing. These images were originally of dimensions 51x37 but have been resized to a common format of 28x28 pixels. Unlike the traditional digit MNIST dataset, Fashion MNIST consists of 10 distinct classes, representing various types of

clothing, bags, and shoes. **Table 1** illustrates the class labels and provides examples from the Fashion MNIST dataset. The decision to utilize Fashion MNIST instead of the more widely used MNIST dataset is based on its novelty, offering a fresh and less explored dataset for experimentation. Furthermore, Fashion MNIST provides a versatile benchmark for evaluating diverse algorithms and classifiers.

Table 1. Fashion MNIST Dataset Images Labels and Description

Label	Description	Examples
0	T-Shirt/Top	
1	Trouser	
2	Pullover	
3	Dress	
4	Coat	
5	Sandals	
6	Shirt	
7	Sneaker	
8	Bag	
9	Ankle boots	

The structured composition and clear objectives of this dataset render it a superb option for delving into and assessing a wide array of neural network models. Fashion MNIST is particularly noteworthy for its suitability, characterized by its straightforwardness and widespread recognition within the field of deep learning.

III. Methodology

The methodology employed in this study comprises two primary phases: pre-processing and modeling. In the pre-processing phase, specific data enhancements are applied. These include the conversion of raw labels into one-hot vectors and the normalization of pixel values. These enhancements aim to improve the efficiency of the model's training process. In the subsequent modeling phase, an MLP architecture is constructed. Various methods to overcome overfit, such as undersampling, feature selection, data transformation, dropout layers, regularization, and early stopping. The primary goal of this modeling phase is to perform clothing classification using the Fashion MNIST dataset while exploring how these overfitting strategies impact the model's performance.

Figure 2 outlines the procedural steps undertaken in this research, commencing with the acquisition of the Fashion MNIST dataset. Following this, the dataset undergoes preprocessing steps and is subsequently partitioned into distinct training and testing subsets. These subsets serve as the fundamental building blocks for training and evaluating multiple MLP models. After training, these models are then utilized for classification purposes, effectively discerning the associated fashion labels.

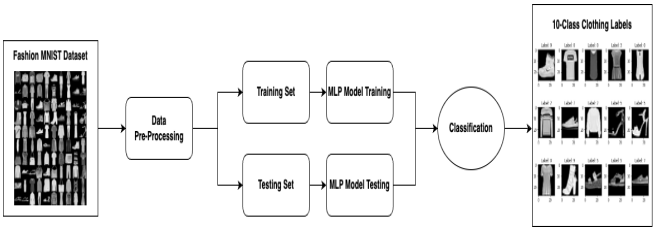


Figure 2. Methodology Flowchart Diagram

A. Data Pre-Processing

In this pre-processing method, several essential steps are taken to prepare the data illustrated in **Figure 3** to be compatible with the MLP model. The initial step involves reshaping the input data, which consists of 28 x 28 images, into a single 784-dimensional vector. This transformation is necessary because MLPs require single vectors as inputs rather than 2D images. However, it's important to note that this reshaping results in some information loss, as the spatial arrangement of pixels in the original images is not preserved **[3]**.

Additionally, the preprocessing involves normalizing the pixel values. In their original form, the pixel values range from 0 to 255, which can lead to issues in training the model effectively. To address this, the pixel values are rescaled to fall within the [0-1] range. This normalization step ensures that the neural network can work with consistent and scaled inputs, facilitating convergence during training **[3]**.

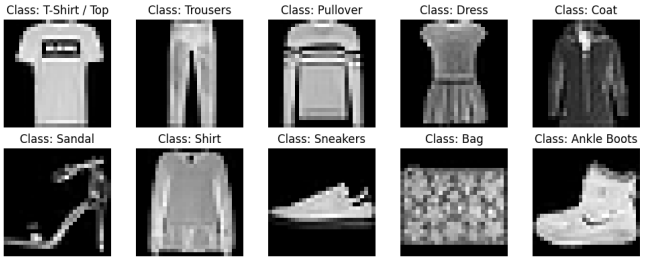


Figure 3. Fashion MNIST Labels

Furthermore, the preprocessing method includes the application of one-hot encoding to the target labels. The original labels are represented as numerical values (0, 1, 2, etc.), which might mislead the MLP into interpreting the relationships between these values incorrectly. One-hot encoding transforms these numerical labels into categorical representations, where each category is uniquely identified. This ensures that the model treats each class independently and avoids misconceptions about the relative significance of the encoded values **[3]**.

B. Modeling

In this study, an in-depth examination of the MLP model is conducted, employing a consistent architecture characterized by two hidden layers, each comprising 512 nodes. The model's illustration can be seen in **Figure 4**.

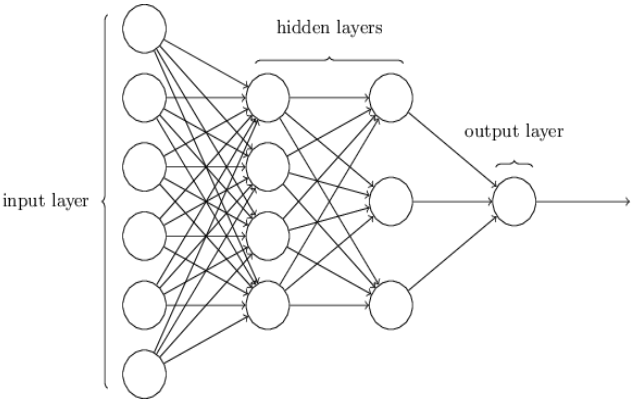


Figure 4. Default MLP Architecture with 2 Hidden Layers

The model's configurations, representing the default setup, incorporate essential elements to optimize performance. Specifically, the Rectified Linear Unit (ReLU)

activation function is applied in both input and hidden layers, introducing crucial non-linearity to enhance the network's capacity for capturing intricate data patterns. Additionally, the models utilize the softmax activation function in their output layers, a standard choice for proficiently handling multi-class classification tasks. This activation function ensures that the model's predictions are transformed into valid probability distributions, facilitating the interpretation of class probabilities for each input instance **[4]**.

Furthermore, to ensure a solid starting point for training, the models employ Xavier's initialization method, initializing weights with a learning rate of 0.001. To effectively address multi-class classification, particularly categorizing various fashion items in the dataset, the chosen loss function is Categorical Cross Entropy. To optimize the model's overall performance, the Adam optimizer is employed **[4]**.

Table 2 presents the hyperparameters under scrutiny in this study. These hyperparameters constitute the default framework for comprehensive analysis in the experimentation, shaping the MLP model's performance in clothing classification.

Table 2. Default Model Hyperparameter Configurations

Hyperparameters	Value
Hidden Layer	2
Number of Nodes near Input Layer	512
Number of Nodes near Output Layer	512
Learning Rate	0.001
Weight Initialization	Xavier
Loss Function	Categorical-Cross Entropy
Optimizer	Adam

C. Overfitting Strategies

Overfitting poses a common challenge in neural network training, and several strategies have been devised to mitigate it effectively. Undersampling tackles class imbalance, feature selection enhances model efficiency, and data transformation stabilizes training. Dropout layers and regularization techniques promote model robustness, while early stopping prevents overfitting by monitoring training progress. These strategies provide a versatile toolkit for combating overfitting and improving model generalization.

The first method employed is Random Undersampling, as depicted in **Figure 5**. This technique is used to create a more balanced distribution of classes by randomly removing instances from the majority class **[5]**. This approach involves the random removal of instances from the majority class, effectively reducing its dominance and helping to mitigate the challenges posed by class imbalance. While random undersampling proves valuable in datasets where there's an unequal class distribution but a sufficient number of minority class examples to build a meaningful model, it's not without its limitations. One key drawback is the indiscriminate removal of instances from the majority class, potentially discarding valuable or significant data points that could be crucial in defining a robust decision boundary. Since this process is random, it does not differentiate between informative and less informative majority class instances. Thus, the decision to employ random undersampling should be made judiciously, considering the trade-offs between achieving class balance and preserving potentially useful data for model learning and generalization **[5]**.

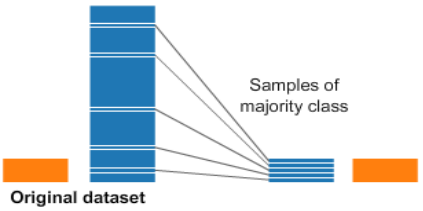


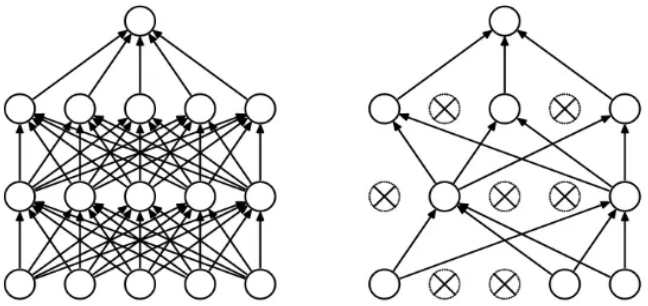
Figure 5. Undersampling Process

The second method, Feature Selection, is employed to enhance the efficiency and effectiveness of the model by focusing on the most informative attributes. In this implementation, the SelectKBest method is utilized, selecting the top k features based on their highest scores. By employing the chi-squared scoring function, it becomes particularly suitable for classification data, aiding in the identification of key features. Feature selection holds paramount importance when dealing with large datasets, as it allows for the elimination of less relevant data components, consequently reducing training time and storage requirements. The advantages of feature selection include a reduction in overfitting, potential accuracy improvement, and faster training times. However, it's essential to note that feature selection can introduce certain challenges, such as increased computational time and potential overfitting risks when the dataset is limited. These challenges can be mitigated by caching selected features and optimizing the selection process to achieve a balance between model efficiency and accuracy **[6]**.

The third method, Data Transformation, involves the application of a log transformation to the input data, a common preprocessing technique used to enhance the suitability of data for deep learning models. Log transformation offers several significant advantages such as it reduces data skewness, creating a more symmetric and normal-like distribution that benefits algorithms assuming data normality. Additionally, it improves linearity between dependent and independent variables, reducing heteroscedasticity and enhancing model interpretability. Log transformation also mitigates the impact of outliers and extreme values, promoting model stability and generalization. Moreover, it simplifies the analysis and modeling of multiplicative relationships by converting them into additive forms. However, it's crucial to note that log transformation can only be applied to positive values, requiring special handling for datasets containing zero or negative values **[6]**.

The fourth method, Dropout Layers, is a crucial technique employed in neural networks to combat overfitting. The term "dropout" refers to dropping out the nodes (input and hidden layer) in a neural network, as seen

in **Figure 6**. All the forward and backward connections with a dropped node are temporarily removed, thus creating a new network architecture out of the parent network. The nodes are dropped by a dropout probability of p. This probabilistic approach effectively prevents co-adaptation among nodes, as in every iteration, the presence of a unit is highly unreliable. By randomly dropping a few units (nodes), it forces the layers to take more or less responsibility for input, enhancing the network's generalization capabilities. However, it's essential to consider potential trade-offs, including increased training time due to the need to train multiple networks and the complexity of tuning hyperparameters, such as the dropout probability and learning rate **[7]**.



(a) (b)

Figure 6. (a) Default MLP; (b) MLP with Dropout

In contrast, the fifth method, Regularization, aims to reduce generalization error by modifying the learning algorithm during training. Specifically, L2 Parameter Regularization is utilized in this case, often referred to as Ridge Regression as it can be seen through **Figure 7**. Regularization adds a penalty term to the loss function as model complexity increases, discouraging overfitting. L2 regularization shrinks parameter values, effectively reducing model complexity and enhancing its ability to generalize. Notably, L2 regularization is less sensitive to outliers and better handles multicollinearity compared to L1 regularization **[7]**. However, it retains all features in the model, lacking automatic feature selection capabilities.

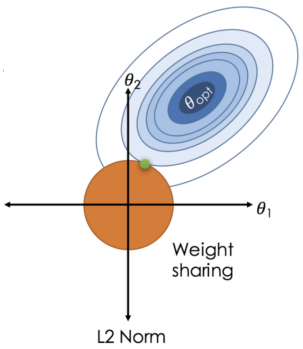


Figure 7. L2 Regularization

The final method employed is Early Stopping. Training neural networks involves deciding the appropriate number of training epochs, a task that can be challenging. Too many epochs can lead to overfitting, while too few may result in an underfit model. Early stopping is a technique that addresses this dilemma by allowing to set a large number of training epochs and halt training once the model's performance ceases to improve on a holdout validation dataset, as shown in **Figure 8**. However, it's important to note that early stopping's effectiveness can be influenced by factors such as the choice of the validation set, the selection of the criterion, and the setting of the patience parameter or threshold **[8]**.

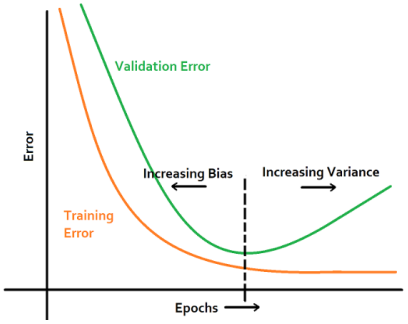


Figure 8. Early Stopping

IV. Results and Analysis

This section provides a comprehensive analysis of the performance of each overfitting strategy applied to the MLP model. The experiments were conducted meticulously, considering all relevant hyperparameters. A consistent epoch setting of 25 was maintained across all experiments to ensure fair comparisons. The section begins with an introduction to the key validation and evaluation parameters employed in the experiments. Subsequently, the accuracy and loss functions are graphically presented for validation purposes. The evaluation phase utilizes performance metrics, primarily accuracy, to assess the model's effectiveness in accurately classifying clothing labels. These evaluations offer a detailed insight into the impact and efficacy of the various overfitting strategies in enhancing the model's generalization ability.

A. Default Model

The initial preset model, denoted as Default Model follows the configuration outlined in **Table 2**. This model features 2 hidden layers, each comprising 512 nodes in proximity to both the input and output layers.

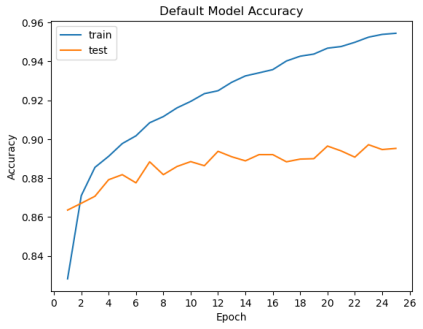


Figure 9. Default Model Accuracy

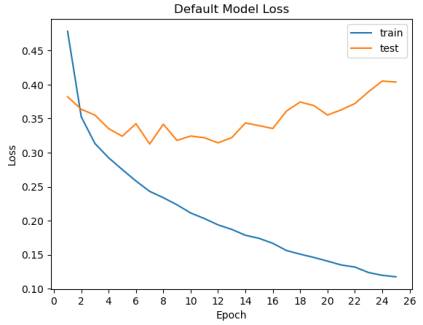


Figure 10. Default Model Loss

Within the Default Model, several crucial insights come to light. While the model achieves a Test Accuracy of 0.895 and a Test Loss of 0.403, it becomes evident that these seemingly impressive results were not so impressive due to overfitting, as shown in **Figure 9** and **Figure 10**. The model's performance is illustrated graphically, revealing that high accuracy does not necessarily equate to a good model, in fact, it indicates a propensity to overfit. This overfitting hinders the model's ability to generalize effectively when presented with new data.

B. *Model with Undersampling*

The first model employs the Undersampling method, a strategy aimed at addressing class imbalance within the dataset. In this approach, instances from the majority class, which are overrepresented, are intentionally reduced to create a more balanced class distribution. By doing so, the model is exposed to a more equitable representation of both classes, helping it to learn and generalize effectively without being biased toward the majority class.

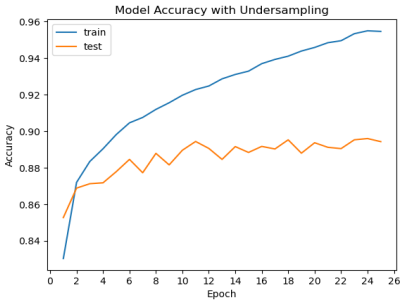


Figure 11. Undersampling Model Accuracy

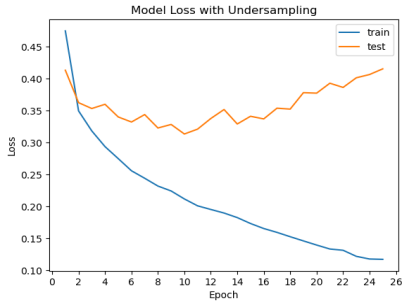


Figure 12. Undersampling Model Loss

The analysis of **Figure 11** and **Figure 12** reveals that the Random Undersampling method does not effectively mitigate overfitting for this dataset. Despite achieving relatively high Test Accuracy and low Test Loss scores of 0.894 and 0.415, respectively, these models still struggle to generalize to new, unseen data. The underlying issue lies in the random nature of the Undersampling process, which indiscriminately removes instances from the majority class without distinguishing between informative and less informative samples.

C. *Model with Feature Selection*

The Feature Selection method, utilized in the second model, focuses on identifying and selecting the most informative features from the dataset. By choosing the top k features based on their highest scores, this approach aims to retain the most relevant attributes while discarding less important ones. In this specific implementation, the chi-squared scoring function is employed, which is well-suited for classification data.

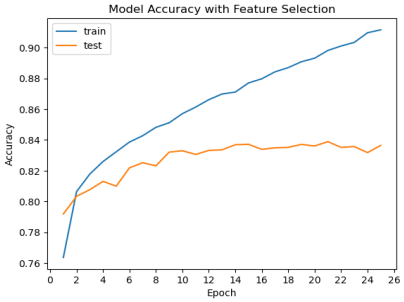


Figure 13. Feature Selection Model Accuracy

Figure 14. Feature Selection Model Loss

Despite achieving a Test Accuracy of 0.836 and a Test Loss of 0.529 shown through the graph of **Figure 13** and **Figure 14**, the model incorporating feature selection still exhibits overfitting tendencies. This outcome can be attributed to the challenge of overfitting when dealing with a limited number of observations. In situations where the dataset lacks sufficient examples to effectively capture the complexity of the underlying patterns, overfitting can persist even when feature selection is applied. This highlights the importance of considering dataset size and complexity when implementing feature selection as an overfitting strategy, as it may not always yield the desired generalization benefits when dealing with smaller datasets.

D. *Model with Data Transformation*

The third model introduces Data Transformation as a strategy to mitigate overfitting. This technique applies a log transformation to the input data, as the transformation enhances the linearity between dependent and independent variables.

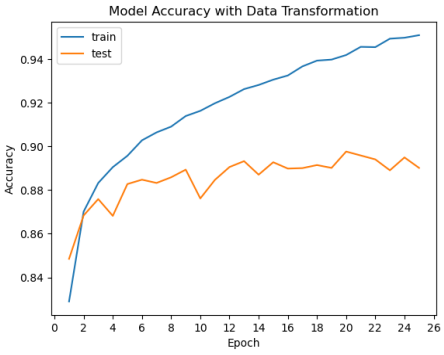


Figure 15. Data Transformation Accuracy

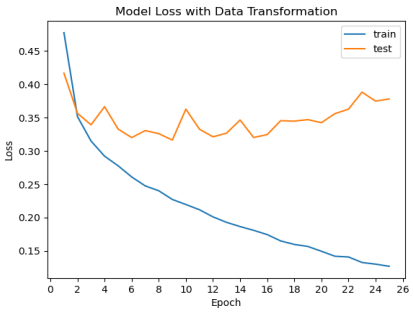


Figure 16. Data Transformation Loss

Despite the efforts, the third model employing Data Transformation, with the best performance metrics of 0.890 Test Accuracy and 0.377 Test Loss depicted in **Figure 15** and **Figure 16**, still exhibits overfitting tendencies. This overfitting may be attributed to various factors, including the intrinsic complexity of the dataset or the presence of noise that cannot be entirely eliminated by data transformation. Therefore, while data transformation can be beneficial in reducing skewness, enhancing linearity, and stabilizing the model, it may not be sufficient to fully address overfitting in certain cases. Additional strategies may need to be considered to achieve a more robust and generalized model.

E. Model with Dropout Layers

The fourth model introduces the concept of Dropout Layers, a technique designed to combat overfitting in neural networks. Dropout layers function by randomly excluding a certain proportion of nodes (units) during each training iteration. This stochastic process compels individual nodes to become more robust and less reliant on others for making predictions.

Consequently, dropout prevents co-adaptation among nodes, enhancing the model's ability to generalize to new, unseen data. By taking a probabilistic approach and intermittently dropping units, dropout layers promote a more balanced distribution of responsibilities across network layers, contributing to improved model performance and reduced overfitting. This technique is particularly effective when dealing with complex datasets where overfitting is a concern **[9]**.

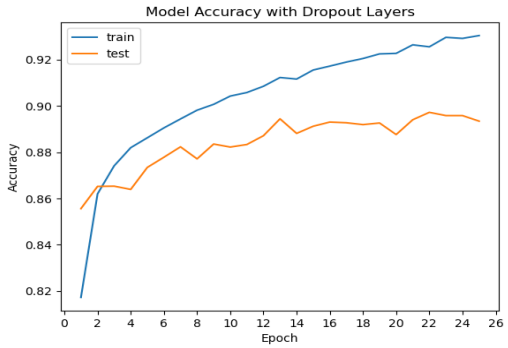


Figure 16. Dropout Layers Accuracy

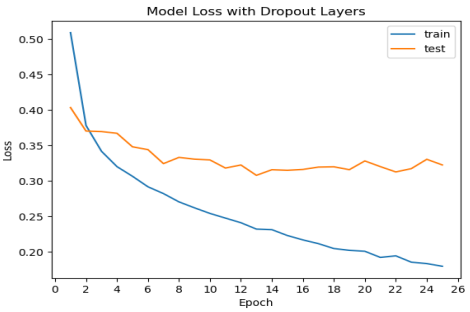


Figure 17. Dropout Layers Loss

While effective to some extent, Dropout Layers weren't sufficient to completely mitigate overfitting in this model. Despite achieving a high Test Accuracy and relatively low Test Loss of 0.893 and 0.322 respectively, the presence of overfitting is evident from the graphical representations in **Figure 16** and **Figure 17**. In this case, the model may have excessively relied on certain nodes, leading to the overfitting problem.

F. Model with Regularization

The fifth model incorporates L2 regularization, which is a technique to prevent overfitting by adding a penalty term to the loss function during training. This penalty encourages the model's weights to remain small, reducing the complexity of the model. L2 regularization, also known as Ridge regularization, is particularly useful because it addresses the risk of overfitting while maintaining the interpretability of the model. By adding this regularization term to the loss function, it discourages the model from assigning excessive importance to any single feature or node during training. This makes the model less prone to fitting noise and irrelevant details in the training data, thereby improving its ability to generalize to new, unseen data. L2 regularization is known for its robustness against outliers and its ability to handle multicollinearity, making it a valuable tool in preventing overfitting in neural networks.

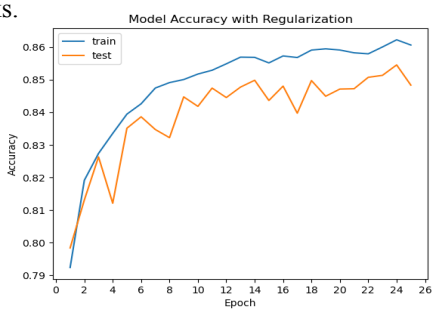


Figure 18. Regularization Accuracy

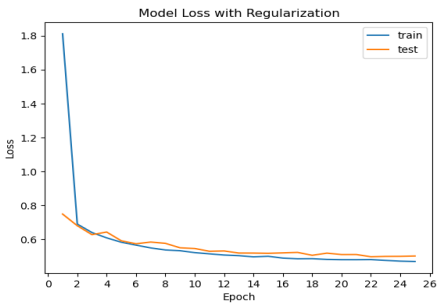


Figure 19. Regularization Loss

Among the various strategies evaluated, the fifth model, employing L2 regularization, stood out as the most effective approach. While it didn't attain the highest Test Accuracy, achieving a score of 0.848, and had a Test Loss of 0.502, as illustrated in **Figure 18** and **Figure 19**, it excelled in its ability to mitigate overfitting. L2 regularization's primary advantage lies in its capacity to prevent the model from excessively fitting the training data, thus ensuring better generalization to new, unseen data by learning robust and meaningful patterns. Consequently, the regularization model proved to be the most reliable solution for addressing overfitting in this context, striking a balance between accuracy and generalization **[9]**.

G. *Model with Early Stopping*

The final method, Early Stopping, introduced a different approach by terminating the training process after just 12 epochs, well before reaching the intended maximum of 25 epochs. Despite achieving a Test Accuracy of 0.888 and a Test Loss of 0.319, as depicted in **Figure 20** and **Figure 21**, this method exhibited a slight tendency toward overfitting, which could be attributed to its premature stopping. While it effectively prevented the model from continuing to fit the training data excessively, it may have stopped before the model had fully converged, resulting in a minor drop in generalization performance **[10]**.

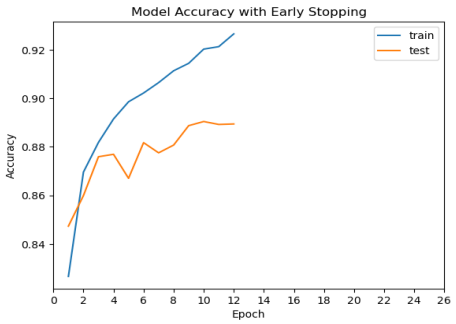


Figure 20. Early Stopping Accuracy

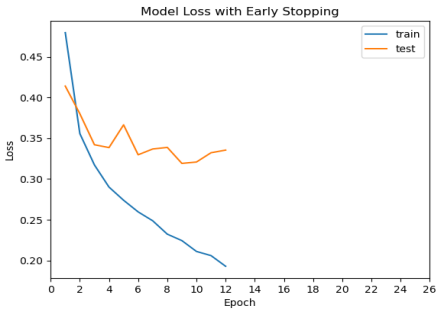


Figure 21. Early Stopping Loss

V. Conclusions

In conclusion, the MLP model's performance in classifying fashion clothing images has been thoroughly evaluated through an extensive experiment involving six different strategies. While one model achieved a slightly higher accuracy of approximately 89% by utilizing Undersampling, it was accompanied by overfitting issues, making it less desirable. In contrast, the model employing L2 Regularization, with an accuracy of approximately 84% and a test graph that is close to the training graph, emerged as the most effective approach. Notably, it successfully mitigated overfitting, ensuring robust generalization to unseen data. The Regularization model demonstrated a balance between accuracy and generalization, making it a reliable choice for image classification tasks.

In pursuit of further improving MLP accuracy in fashion image classification, future works can explore combining various overfitting mitigation strategies like Regularization, Dropout Layers, and Data Transformation. Investigating advanced neural network architectures, like state-of-the-art algorithms such as the Convolutional Neural Network (CNN) **[8]** for image data, and leveraging transfer learning techniques could enhance model performance. Increasing dataset diversity and size, along with data augmentation, may push MLP's accuracy beyond 90%. These directions offer promising avenues for future research in image classification tasks **[10]**.

Acknowledgment

I would like to express my sincerest gratitude to Mr. Wahyono, S. Kom., Ph.D., for the invaluable guidance and advice provided throughout the Deep Learning course this semester. Thank you for imparting your extensive knowledge and expertise in the field, which has been instrumental in shaping my ideas and expanding my understanding of the subject matter. Furthermore, I understand that the study may not have been as comprehensive as some of the other research in the field, which utilizes more advanced hyperparameters and techniques to overcome overfitting. Nonetheless, I am immensely proud of what I was able to create and the potential practical applications it holds.

References

[1] H. Li et al., "Research on overfitting of Deep Learning," 2019 15th International Conference on Computational Intelligence and Security (CIS), 2019. doi:10.1109/cis.2019.00025

[2] D. C. Cireşan, U. Meier, L. M. Gambardella, and J. Schmidhuber, "Deep big multilayer perceptrons for digit recognition," Lecture Notes in Computer Science, pp. 581–598, 2012. doi:10.1007/978-3-642-35289-8_31

[3] E. Xhaferra, E. Cina, and L. Toti, "Classification of Standard Fashion Mnist dataset using Deep Learning based CNN algorithms," 2022 International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT), 2022. doi:10.1109/ismsit56059.2022.9932737

[4] W. Di, "A comparative research on clothing images classification based on neural network models," 2020 IEEE 2nd International Conference on Civil Aviation Safety and Information Technology (ICCASIT, Weihai, China, 2020, pp. 495-499, doi: 10.1109/ICCASIT50869.2020.9368530.

[5] K.V. Greeshma and V. Gripsy, "Image Classification using HOG and LBP Feature Descriptors with SVM and CNN", International Journal of Engineering Research & Technology (IJERT), 2020.

[6] S. Shen, "Image Classification of Fashion-MNIST Dataset Using Long Short-Term Memory Networks", Research School of Computer Science, Australian National University, 2018.

[7] D. K. Mohanty, P. Das Gupta, R. Dey and S. Pattnaik, "Modified Convolutional Neural Network for Fashion Classification," 2022 International Conference on Advancements in Smart, Secure and Intelligent Computing (ASSIC), Bhubaneswar, India, 2022, pp. 1-9, doi: 10.1109/ASSIC55218.2022.10088358.

[8] S. K. Noon, M. Amjad, M. A. Qureshi and A. Mannan, "Overfitting Mitigation Analysis in Deep Learning Models for Plant Leaf Disease Recognition," 2020 IEEE 23rd International Multitopic Conference (INMIC), Bahawalpur, Pakistan, 2020, pp. 1-5, doi: 10.1109/INMIC50486.2020.9318044.

[9] Y. Liu, J. A. Starzyk and Z. Zhu, "Optimized Approximation Algorithm in Neural Networks Without Overfitting," in IEEE Transactions on Neural Networks, vol. 19, no. 6, pp. 983-995, June 2008, doi: 10.1109/TNN.2007.915114.

[10] F.-I. Chou, Y.-K. Tsai, Y.-M. Chen, J.-T. Tsai, and C.-C. Kuo, "Optimizing parameters of multi-layer convolutional neural network by modeling and Optimization Method," IEEE Access, vol. 7, pp. 68316–68330, 2019. doi:10.1109/access.2019.2918563.

APPENDIX

Dataset & Source Code: http://tiny.cc/GoogleColab-MLP-Overfit