Comparative Analysis of Image Classification Algorithms on the Fashion-MNIST Dataset

Shuvam Chowdhury
Machine Learning & Data Science
Email: kshuvam63@gmail.com
GitHub: https://github.com/Theshuvam

***Abstract*—The Fashion-MNIST dataset is used to train and test various image classification models and compare their performance. The algorithms discussed in this report are k-Nearest Neighbors (KNN), Logistic Regression, Support Vector Machines (SVM), and Convolutional Neural Networks (CNN). Basic exploratory data analysis (EDA) addresses class imbalances, differing pixel intensities, normalization, and dimensionality reduction. Each algorithm is evaluated using accuracy, precision, recall, and F1-score. Results show that CNN best performs image classification on the Fashion-MNIST dataset with metrics of 92.07%, 0.9208, 0.9207, and 0.9197 respectively. This paper provides reasoning for algorithm selection, comparison, and reproducible experimentation.**

***Index Terms*—Fashion-MNIST, image classification, k-Nearest Neighbors, Logistic Regression, Support Vector Machine, Convolutional Neural Network**

I. Introduction

Fashion-MNIST is a dataset of 28x28 grayscale article images each with an associated label from one of 10 classes: tshirt/top, trouser, pullover, dress, coat, sandal, shirt, sneaker, bag, ankle boot. The dataset takes the same format as the MNIST dataset with 60,000 examples in the training set, and 10,000 examples in the test set. In this project, the Fashion-MNIST is further divided to include a validation set. The validation set pulls 10,000 samples from the training set, and the training set becomes 50,000 samples. An example of 25 random images from the training set is shown in Figure 1.

In this project, image classification is applied to the FashionMNIST dataset with both baseline and advanced models to ultimately compare model performance. In order, this paper reviews the machine learning algorithms K-Nearest Neighbors (KNN), Logistic Regression, and Support Vector Machines (SVM) along with the deep learning Convolutional Neural Network (CNN) algorithm. The implementation for this project is available in the project's GitHub repository here.

II. Literature Review

Image classification aims to identify features in an image and accurately assign that image to a class defined by one or more characteristics. Data used in this procedure undergoes unique pre-processing which involves segmenting each image by the normalization of its pixels. Unfortunately, flawed pictures such as those that are fragmented, noisy, or ambiguous, are inevitable. CNNs are said to manage the above factors well so are widely used in image classification.

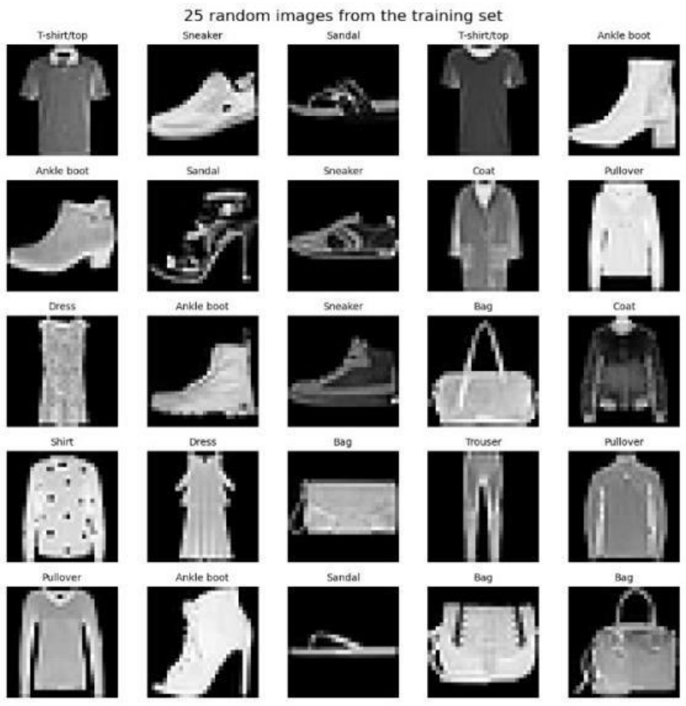

Fig. 1. 25 Random images selected from the Fashion-MNIST training Dataset

CNN extracts characteristics from images to obtain information. Therefore, features are not passed to CNN but are discovered while the network trains on a set of images. In an experiment on the CIFAR-10 dataset, when compared against VGG16, another classifier that utilizes 3x3 convolution layers and a 2x2 max pool layer, CNN resulted in a 92.22VGG16 only performed with an 89.03scaledependent pooling method of CNN improves its accuracy, but consequently also increases computational time. Computational time can be decreased with advanced architecture left undiscussed.

CNN is applied to the Fashion-MNIST dataset in our report for image classification and is discovered to perform best against other tested algorithms. The described experiment supports our conclusion that CNN is the best fit for classifying our set of images as described further along in this paper.

In a published application of CNN, the neural network is applied to Korean food images taken from an AI-Hub platform in an attempt to expand the knowledge of Korean food image classification [5]. To begin, the images underwent the aforementioned unique pre-processing for image classification to improve accuracy. The data pre-processing that will be described for use on the Korean food images dataset is like the pre- processing we executed on the Fashion-MNIST dataset.

The Korean food image dataset was divided into a training, testing, and validation set with 105,427, 30,122, and 15,061 images respectively. The images were resized to 331x331 pixels each. Then, the newly pre-processed image set was augmented to counteract overfitting. Performing data augmentation produced images with varying levels of brightness, contrast, saturation, and alignment. Features were extracted from augmented image datasets and pre-trained with deep neural networks like ResNet- 50, RestNet-101, RestNet-152, MobileNetV2, InceptionResNetV2, and NasNetLarge [5].

The pre-trained deep neural networks mentioned above were connected to a CNN used to classify objects of the images in this project. Each classification model passed through an initial training stage and a fine-tuning stage. Then, the model was evaluated under the accuracy measurement TP + TN / TP + FP + TN + FN where TP is true positive, TN is true negative, FP is false positive, and FN is false negative. The study concluded that accuracy was consistent despite the network choice for feature extraction. The classification model repeatedly incorrectly distinguished images that combined food classes or included food with similar shapes and colors [5]. The complexity of each Korean food image makes classification difficult. This study does not resolve the consistent misclassification of some Korean food classes, so could be built upon in future work.

The Fashion-MNIST dataset deals with much less complex images than the Korean food image dataset, but understanding the process of classifying a larger and more diversified dataset was useful when applying the same technique on our simpler scale.

III. Methodology

A. Data Pre-processing

The Fashion-MNIST dataset is imported into a Jupyter notebook and then divided into training and testing sets with 60,000 and 10,000 samples respectively. The training set is reduced to 50,000 samples and the extra 10,000 samples are separated for the validation set. Then, each 28x28 grayscale image taken from Fashion-MNIST is flattened as a 2D array with 784 columns. The flattened training, validation, and testing image datasets are converted into Pandas DataFrames where each row represents one image. A new column, ‘label’, is added to represent the true label of each image.

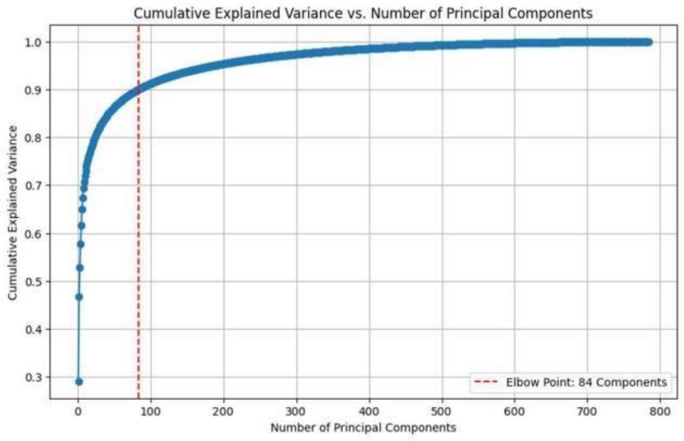

Fig. 2. Cumulative explained variance as a function of the number of PCA components. The dashed line indicates the elbow point at 84 components.

B. Exploratory Data Analysis

To understand data balance and how to proceed with EDA, class distributions are computed on the training, validation, and test sets. Class distribution should be similar between sets to ensure proper generalization. If class imbalance is present in any set, this may impact the performance of machine learning models and require resampling, class weighting, or data augmentation. The number of samples in each class for all three sets is consistent, and no class imbalance occurs. Therefore, no corrective measures are implemented

Next, pixel intensity distribution is analyzed for each class in training, validation, and test sets to gain insights into the brightness and contrast characteristics of images in each class. Pixel intensities in grayscale images range from 0 (black) to 255 (white), so analyzing this distribution can reveal whether the images are generally darker or lighter, and if there are any anomalies. The pixel intensity distributions are skewed, so range normalization is performed using the MinMaxScaler imported from sklearn. This normalization method is selected because the distribution of the pixel values is unknown and because pixel values are bounded, there are no outliers.

Principal component analysis (PCA) is performed on the normalized training data to find the number of components where cumulative explained variance surpasses 90 percent. The optimal number of components that capture 90% of the variance is illustrated by the elbow point on the graph below. The elbow point occurs at 84 components.

The data is transformed to retain only these 84 components. The DataFrames created with the results of PCA are used to train the logistic regression, KNN, and SVM algorithms, but are not used for CNN. CNNs are designed to learn their own optimal feature representations and hierarchical features from raw pixel data. PCA does not account for the spatial relationships present in image data that CNN relies on for image classification.

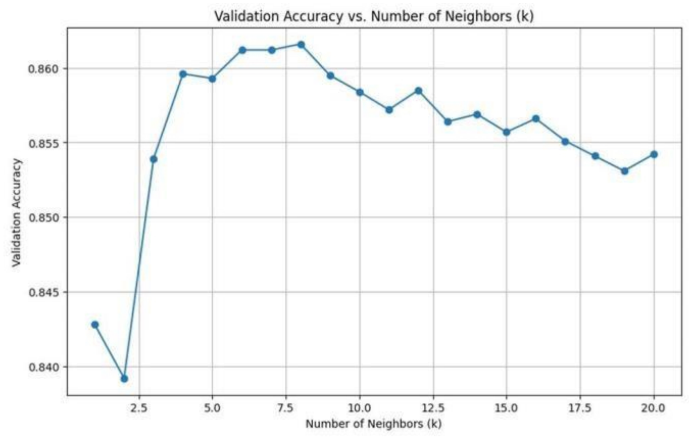

Fig. 3. Validation accuracy as a function of the number of neighbors (k) in the K-Nearest Neighbors (KNN) classifier. The plot shows the performance trend across varying k values and highlights the range where the classifier achieves its highest accuracy.

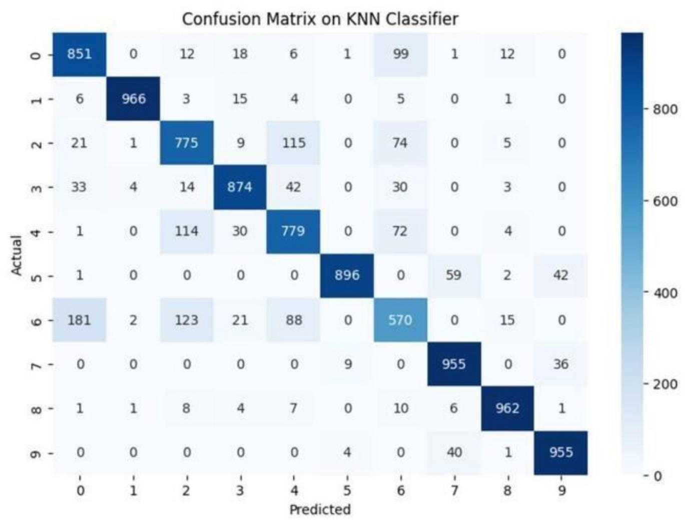

Fig. 4. Confusion matrix of the K-Nearest Neighbors (KNN) classifier on the Fashion-MNIST test set. Diagonal entries indicate correct classifications, while off-diagonal values show misclassifications across the ten clothing categories.

IV. Experimental Results

The accuracy score referred to when evaluating the following algorithms is imported from sklearn.metrics and is a measure of subset accuracy where the set of labels predicted for a sample must exactly match the corresponding set of true labels [1].

A. K-Nearest Neighbors

The goal of first employing the KNN algorithm is to achieve reasonable accuracy to justify more complex models if they outperform KNN significantly. The KNN algorithm deals with non-parametric learning. The value of k, where k defines the number of nearest training samples' class labels to a test sample, is the only hyperparameter. The Fashion-MNIST dataset does not require extensive pre-processing, and KNN works well even with basic normalization. Features like pixel intensities or PCAtransformed data can be effectively compared using distance metrics.

To tune the hyperparameter, the training data is fit to multiple KNN classifiers with varying values of k neighbors and the validation accuracy vs. number of neighbors is plotted. The maximum point on the plot, (8, 0.86), represents the optimal number of neighbors and the highest achieved accuracy.

The KNN classifier is initialized and set with 8 optimal numbers of neighbors. A Stratified K-Fold Cross Validation is defined to achieve a more reliable estimate of model performance and preserve the proportion of each class in the training and validation sets by using five splits of the data. The KNN model is fit on the training set and predictions are made on the test set. The metrics calculated for the cross-validated sets equal the metrics returned when calculated on the test set. Accuracy, precision, recall, and F1-score are all 0.86. Therefore, the model is consistent across different data splits and correctly predicts the image class 86% of the time.

Take note that KNN has the most difficult time predicting class 6 (shirt) which it most confuses with class 0 (t-shirt/top).

B. Logistic Regression

Fashion-MNIST is a relatively low-complexity dataset. Where more complex machine learning models are likely to overfit, logistic regression is a simple model that does not require higher computational power and is more capable of generalizing. Logistic regression works well for linearly separable datasets, so the goal of testing this algorithm is to check how Fashion-MNIST performs under assumptions of linearity

The logistic regression model is fit to the training data. The trained model is used to make predictions on the validation set first. Then, the model makes predictions on the test set and weighted averaging is used to compute accuracy, precision, recall, and F1 scores to handle class imbalance. The pseudocode and its output for the outlined process are below.

```
log_reg = LogisticRegression(max_iter=1000,
    random_state=0)
log_reg.fit(X_train, y_train)

y_val_pred = log_reg.predict(X_val)
# Compute classification metrics for the validation
    set

y_test_pred = log_reg.predict(X_test)
# Compute classification metrics for the test set

Output:
Validation Accuracy for Logistic Regression: 0.84
Validation Precision for Logistic Regression: 0.84
Validation Recall for Logistic Regression: 0.84
Validation F1 Score for Logistic Regression: 0.84
```

The logistic regression model performs slightly worse than the KNN model with only 84% accuracy. From the confusion

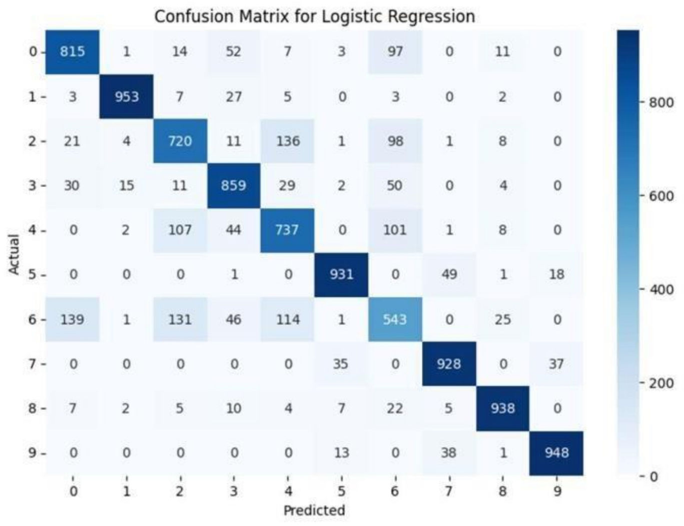

Fig. 5. Confusion matrix of the Logistic Regression classifier evaluated on the Fashion-MNIST test set. Diagonal values represent correct predictions across the ten clothing categories, while off-diagonal entries indicate misclassifications.

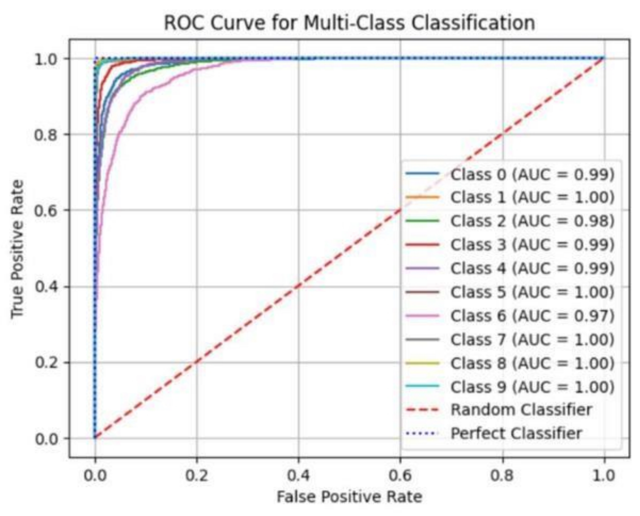

Fig. 6. ROC curves for the multi-class classification task on the Fashion-MNIST dataset. Each curve represents a single class using a one-vs-rest approach, with corresponding AUC values shown in the legend. The diagonal dashed line indicates random performance, while the dotted line shows the ideal classifier.

matrix, the model appears to have an even more difficult time identifying class 6 (shirt) than KNN. In general, this may imply the data is not well suited to linear classification.

C. Support Vector Machine

SVM is effective for small to medium-sized datasets in highdimensional spaces like Fashion-MNIST. SVM employs the kernel trick to transform the given data and find optimal decision boundaries. The kernel does not have to be linear, so SVM can capture more complex relationships between data points. SVM is utilized to compare Fashion-MNIST performance with no linear assumptions to the output of logistic regression which assumes linearity.

For hyperparameter tuning, a for loop iterates over all the combinations of two selected kernel types, radial basis function, and polynomial, and selected C values of 0.1, 1, and 10. The model is trained on the training set with each unique hyperparameter combination then accuracy is calculated by using the validation set for model prediction. The kernel, C value, and accuracy are stored in the empty list, result. Then, the highest accuracy is pulled from the results list, and the optimal kernel and C values are extracted. The pseudocode for this process is as follows.

```
result = []
kernel_options = [...]
C_values = [...]

for kernel in kernel_options:
   for C in C_values:
      svc_clf = SVC(kernel=kernel, C=C)
      svc_clf.fit(X_train, y_train)

      y_val_pred = svc_clf.predict(X_val)
      accuracy = accuracy_score(y_val, y_val_pred)

      # Append (kernel, C, accuracy) to result list
      result.append((kernel, C, accuracy))

# Identify the entry with the highest validation
    accuracy
best_result = max(result, key=lambda x: x[2])
best_kernel = best_result[0]
best_C = best_result[1]
```

The determined hyperparameters are used to initialize the SVM classifier, train the model on training data, create model predictions on testing data, and output 90accuracy. The nonlinear decision boundaries achieved by SVM are better suited to the Fashion-MNIST dataset than the linear separation performed by logistic regression.

The ROC curve illustrates the model's ability to distinguish between positive and negative classes. It measures the ranking of probabilities for the true class. Therefore, even if the model makes an incorrect prediction, but only assigns the true class a slightly lower probability, it can achieve a high area under the curve (AUC). This explains the discrepancy between accuracy and AUC. If the AUC is equal to 1, this represents a perfect fit. This plot shows the performance of the SVM classifier across each class in the testing set. Specifically, it evaluates how well the classifier distinguishes each class from the others using a one-vs-all approach. Notice all the classes have an AUC extremely close or equal to one.

D. Convolutional Neural Network

The architecture of CNN with an input and output layer and many hidden layers provides high accuracy for image classification. Each layer learns to identify different features, and because all hidden neurons in the same layer share weights and bias values, the network is tolerant to object

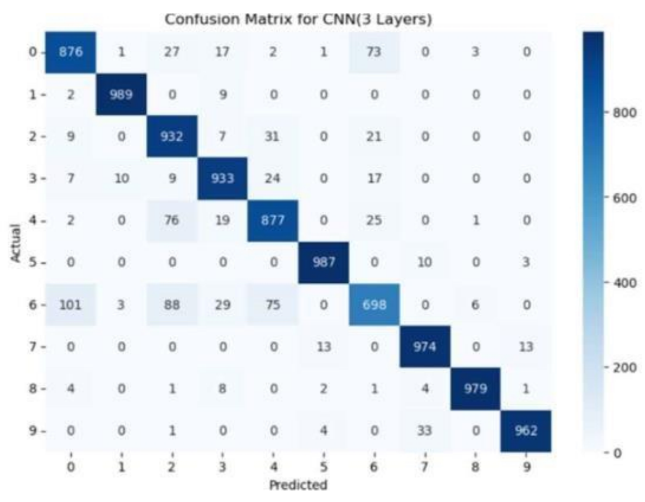

Fig. 7. Confusion matrix of the 3-layer Convolutional Neural Network (CNN) classifier evaluated on the Fashion-MNIST test set. Diagonal entries represent correct predictions, while off-diagonal values indicate misclassifications across the ten clothing categories.

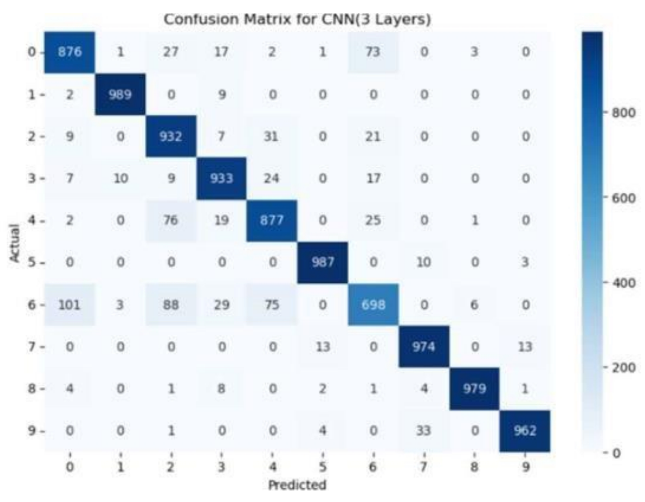

Fig. 8. Confusion matrix of the CLIP model evaluated on the Fashion-MNIST test set using the best-performing prompt. The matrix shows per-class prediction performance across all ten categories, with diagonal values indicating correct predictions and off-diagonal entries showing misclassifications.

translation within an image. CNN does not require manual feature extraction, so the data pre-processing step may be simplified for cases where CNN is used. CNN is tested to determine the accuracy of an image classification method that requires less feature engineering.

The CNN algorithm is tested with 1, 2, 3, and 4 hidden layers and begins with the raw flattened images derived from Fashion-MNIST. Each pixel is normalized and takes a value between 0 and 1. The objective function used for CNN is crossentropy loss, and the Adam optimizer is used to update the model's parameters based on the gradients computed during backpropagation.

For each CNN algorithm tested on the Fashion-MNIST dataset, the model is trained in a loop for 10 epochs. After each epoch, the accuracy and loss measurements are printed. After the training loop, the model is set to evaluation mode where the loss on the validation set is calculated. Finally, the model is assessed on test data, and the classification metrics are output. For CNN with a single layer, evaluation mode on the validation set following training is excluded. When evaluated on the test data, the CNN algorithm with 3 hidden layers performs best with accuracy, precision, recall, and F1 scores of 92.07%, 0.9208, 0.9207, and 0.9197 respectively. Since all the classification metrics for this model exceed those of the algorithms explained above, it is concluded that CNN most accurately performs image classification on the FashionMNIST dataset. This can be further confirmed by examining the confusion matrix.

This confirms CNN superiority over classical ML models.

E. Contrastive Language-Image Pretraining

Contrastive Language-Image Pretraining (CLIP) was applied to Fashion-MNIST to perform zero-shot classification. Zero-shot classification refers to the machine learning technique where a model can classify data without being trained on those classes. As a multimodal model, CLIP is capable of processing given images paired with natural language descriptions to label the data.

Three CLIP model variants were tested. Model variant "openai/clip-vit-base-patch16" performed best with 0.6803 accuracy. After choosing the proper variant, prompt testing increased the model accuracy to 0.7115 when the prompt "a centered image of a " was used. However, even with a proper prompt, the CLIP model did not produce a competitive accuracy score. The confusion matrix illustrates the model has the most difficulty classifying a shirt. This class has the most incorrect predictions across all methods described in this paper.

V. Business Insights

For retailers, being able to analyze images of clothing can enhance their business capabilities. With all its potential, image classification can aid in marketing, recommending merchandise to online shoppers through a visual search, keeping up with the latest fashion trends, providing insights for supply and demand, and generally helping improve business functionality [6].

VI. Conclusion

When compared on the Fashion-MNIST image dataset, the performance of baseline and advanced models proves the superiority of the CNN algorithm for image classification. CNN outperforms KNN, logistic regression, and SVM for every classification metric. The capabilities of CNN to best classify images may be due to its uniqueness among the four tested models. This deep learning algorithm does not require manual feature extraction, and its own optimal feature selection lends itself to producing the most accurate results. In the future, Fashion-MNIST can be a guide to image classification with more complex datasets.

References

[1] "accuracy_score," Scikit-learn. Available: https://scikit-learn.org/dev/modules/generated/sklearn.metrics.accuracy_score.html (accessed Nov. 19, 2024).

[2] A. Fulare, K. Raghavendra Prasad, P. Johri, I. S. Abdulrahman, K. B. D. Arasi, and G. Guwalani, "Convolutional Neural Network Based Image Classification and Its Comparison with VGG-16 to Measure Accuracy," in *Proc. 2023 3rd Int. Conf. Advance Computing and Innovative Technologies in Engineering (ICACITE)*, Greater Noida, India, 2023, pp. 367–370. doi: 10.1109/ICACITE57410.2023.10183128.

[3] "Fashion-MNIST," GitHub. Available: https://github.com/zalandoresearch/fashionmnist (accessed Oct. 23, 2024).

[4] G. L., "Why use SVM?," Alteryx Community. Available: https://community.alteryx.com/?category.id=external (accessed Nov. 18, 2024).

[5] M. Young, *The Technical Writer's Handbook*. Mill Valley, CA: University Science, 1989.

[6] M. Chun, H. Jeong, H. Lee, T. Yoo, and H. Jung, "Development of Korean Food Image Classification Model Using Public Food Image Dataset and Deep Learning Methods," *IEEE Access*, vol. 10, pp. 128732–128741, 2022. doi: 10.1109/ACCESS.2022.3227796.

[7] S. Kholin and A. Bitkina, "Image Classification: 6 Industries & 26 Use Cases You Can Try," Onix. Available: https://onixsystems.com/blog/imageclassification-use-cases (accessed Nov. 20, 2024).

[8] "What Is a Convolutional Neural Network?," MathWorks. Available: https://www.mathworks.com/discovery/convolutional-neural-network.html (accessed Nov. 19, 2024).

[9] "Why does Logistic Regression perform better than machine learning models in clinical prediction studies?," StackExchange. Available: https://datascience.stackexchange.com/questions/120262/why-does-logistic-regression-perform-better-than-machine-learning-models-in-clin (accessed Nov. 18, 2024).