Machine Learning

WS 2023

Group 04
EXERCISE 3: Deep. Learning - Image classification on Fashion-MINST and CIFAR10 dataset

---

Student:
Aleksandra Gjorgieva 12210745
Mirjana Sadikovikj 12225171
Adam Domoslawski 12241331

February 2024

1. Introduction

The world of image classification has been transformed by the introduction of Convolutional Neural Networks (CNNs). In this report we focus on the comparison between Lenet5 (which historically was one of first CNNs) and custom neural network we prepared. Prepared model were computed were trained on known and popular Fashion MINST dataset and CIFAR-10. Fruther they were compared with other ML model like SVM, Random Forest and KNN. Additionally concepts which aim to make CNN's more robust like data augmentation, dropout were explored in the report..

2. Datasets

1. Fashion MINST dataset

The Fashion-MNIST contains 60,000 training images and 10,000 testing images of Zalando's article images. The dataset comprises grayscale images of size 28x28 pixels. Each pixel has a single pixel-value associated with it, indicating the lightness or darkness of that pixel, with higher numbers meaning darker. This pixel-value is an integer between 0 and 255. Fashion-MNIST is widely used for training and testing in the field of machine learning, especially for image classification tasks.

2. CIFAR10 Dataset

The CIFAR-10 dataset serves as a benchmark for assessing the efficacy of various convolutional neural network (CNN) models. This dataset, comprising 60,000 32x32 color images divided across 10 distinct categories, provides a balanced platform for both training and evaluating image classification algorithms. We use four different models, ranging from foundational CNN structures such as LeNet to more advanced strategies incorporating transfer learning and data augmentation. This chapter outlines our approach to each model, detailing the training processes, performance outcomes, and the practical implications derived from our experiments.

3. Methodology

3.1. Shallow Machine Learning Algorithms Approach

In this section, we present our methodology for training classifiers on the provided datasets, focusing on two distinct approaches. The first approach involves utilizing shallow machine learning algorithms for classification tasks.

Our primary objective in this approach was to develop machine learning models capable of classifying images into predefined categories. To achieve this, we employed shallow machine learning algorithms, including Support Vector Machines (SVMs), Decision Trees, and K-nearest neighbors (KNN).

Feature Extraction:

Before training the machine learning algorithms, it was essential to extract relevant features from the images. We utilized two types of features: color histograms and Scale-Invariant Feature Transform (SIFT) descriptors.

- Color Histograms: A color histogram provides a representation of the distribution of colors in an image. We computed the color histograms using the OpenCV function calcHist. For the CIFAR-10 dataset, we divided the color space into 16 bins for each channel, while for the FashionMNIST dataset, which consists of grayscale images, we employed 256 bins. Additionally, we normalized the histograms to prevent bias towards certain changes in illumination or contrast.
- SIFT Descriptors: SIFT is a feature extraction algorithm that identifies key points in an image and computes descriptors for these keypoints. SIFT descriptors are designed to be invariant to scale, rotation, and affine distortion, ensuring robustness against changes in the image. We calculated SIFT descriptors using OpenCV functions, clustering them into 100 clusters. Subsequently, we computed normalized histograms for each image based on these descriptors.

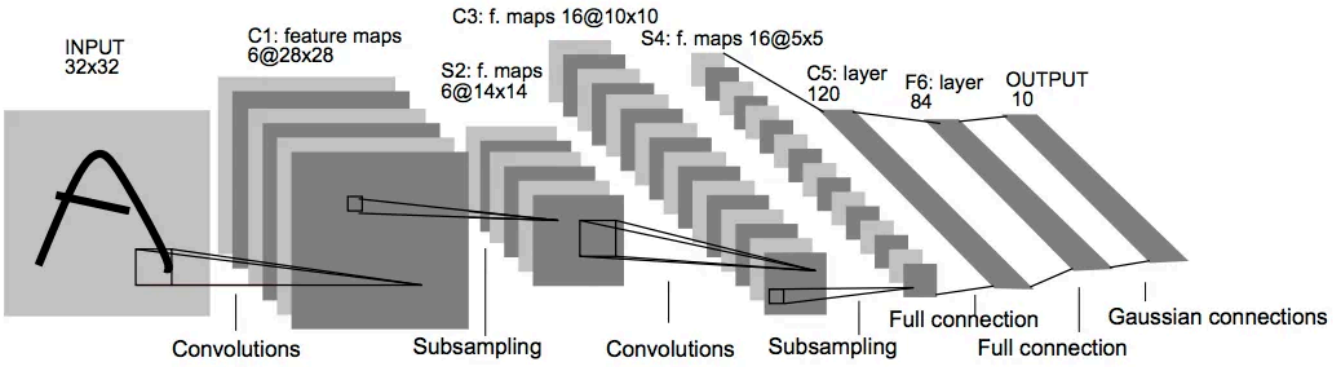
Model Training:

To combine the color histograms and SIFT descriptors into a unified feature vector, we concatenated them. This feature vector served as input for training three different machine learning algorithms:

- RandomForestClassifier: A decision tree-based ensemble learning algorithm.
- SupportVectorClassifier (SVC): A powerful algorithm for classification tasks, capable of handling non-linear decision boundaries.
- KNearestNeighbor (KNN): A simple yet effective instance-based learning algorithm.

We employed RandomizedGridSearchCV, a technique that randomly combines hyperparameters while performing cross-validation with 5 folds, to optimize the performance of each algorithm.

3.2. Lenet5



LeNet-5 (developed by Yann LeCun et al. in 1998) is split in following layers

- Convolutional Layer (C1) - 6 filters, 5x5 kernel
- Pooling Layer (S1) - max pooling, 2x2 kernel
- Convolutional Layer (C2) - 16 filters, 5x5 kernell
- Pooling Layer (S2) - same as S1
- Fully-Connected Layer (F5) - flattens feature maps into matrix [num_of_batches, num_of_features], where num_of_features = 120
- Fully-Connected Layer (F6) - reduce number of features to 84
- Output Layer

3.3 Custom CNN used on Fashion MINST

For comparison to Lenet for Fashion MINST dataset there were used two custom neural networks (employing bigger number of items than lenet5):

Net2:

- **Layers:**
  - 2 convolutional layers (conv1, conv2) with 32 and 64 filters, respectively.
  - 2 batch normalization layers (bn1, bn2) to stabilize training.
  - 2 max pooling layers to reduce dimensionality.
  - 2 fully connected layers (fc1, fc2) with 512 and 10 neurons, respectively.
- **Function:**
  - Takes a single-channel input image (e.g., grayscale).
  - Extracts features using convolutional layers.
  - Reduces dimensionality with pooling layers.
  - Flattens features and produces class scores with fully connected layers.
  - Likely designed for image classification tasks with 10 classes.

Net3:

- **Layers:**
  - Similar to Net2, but with an additional convolutional layer (conv3) with 128 filters.
  - Also includes an additional batch normalization layer (bn3).
- **Function:**
  - More complex architecture for potentially learning more intricate features.
  - May achieve better performance due to its increased depth and complexity.

3.3. Custom CNN for CIFAR-10

Net - without data augmentation:

- **Preprocessing:**

Training data: Random cropping, random horizontal flipping, conversion to tensor, normalization.
Test data: Conversion to tensor, normalization.

- **Convolutional Layers:**
    - Six layers with an increasing number of filters (64, 64, 128, 128, 256, 256).
    - Kernel size: 3x3 for all, with padding to maintain size.
    - Each convolutional layer is followed by batch normalization and ReLU activation.
    - Three max pooling layers are used to reduce dimensionality after pairs of convolutional layers.
    - Fully Connected Layers:
    - First layer: 1024 neurons, taking flattened features from the last convolutional layer.
    - Output layer: 10 neurons (one for each CIFAR-10 class).
    - Dropout applied before the final layer to reduce overfitting.
- **Function:**
    - Processes 3-channel (RGB) input images from the CIFAR-10 dataset.
    - Employs a deep stack of convolutional layers for feature extraction.
    - Utilizes max pooling for spatial dimension reduction.
    - Applies fully connected layers to classify images into 10 categories.

- **Additional Details:**
    - Utilizes SGD optimizer with momentum and weight decay for training.
    - Trains for 20 epochs, printing loss every 100 batches.
    - Evaluates accuracy on the test set post-training and reports a detailed confusion matrix.
    - Customized for GPU acceleration if available.

This model demonstrates a more complex and deeper architecture than Net2 from Fashion MINST, designed to handle the multiclass classification problem presented by the CIFAR-10 dataset, featuring a more nuanced approach to feature extraction and classification through a series of convolutional, normalization, and pooling layers, followed by dense layers for decision making.

Net - with data augmentation:

- **Layers**:
    - Utilizes a pre-trained ResNet18 architecture customized for the CIFAR-10 dataset classification task.
    - Implements data augmentation techniques (random crop and horizontal flip) for the training dataset to enhance model generalization.
    - Adjusts the original ResNet18 by replacing the last fully connected layer with a new one that has 10 outputs to match CIFAR-10's class structure.
    - Freezes all layers except the newly added final layer to focus training on CIFAR-10 specific features.

- **Function:**
    - Processes 3-channel (RGB) input images from the CIFAR-10 dataset.
    - Leverages the depth and complexity of ResNet18 for feature extraction.
    - Applies data augmentation to the training set to improve robustness and prevent overfitting.
    - Adapts to CIFAR-10 classification by training the modified network on augmented and normalized images, culminating in class predictions across 10 categories.

4. RESULTS

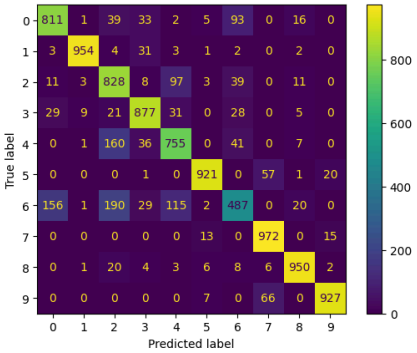
4.1 Fashion-MINST

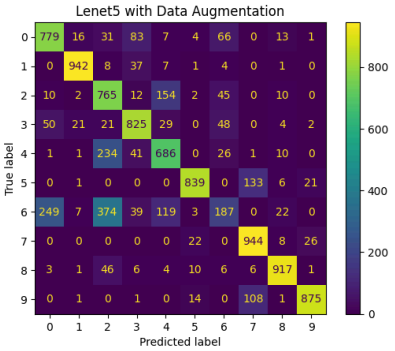
Fashion-MINST

As stated before two neural network models were compared: historical Lenet5 and custom neural network. Custom neural networks were in 2- and 3-layer variant. Each of the neural networks was run 10 times and best scores were chosen.

- **FashionMINST Lenet5 without Data Augmentation**

There was nont needed adapting the input, as Lenet5 architecture was originally used on 28x28 input. Default mini-batch has size 64, learning_rate = 0.005 and 10 number of epochs. Data were eventually transformed (Normalised and casted to Tensor format) with DataLoaders. There was used padding of size 4 and kernel matrices of size 4. Neural network were evaluated with precision and accuracy score (84.9% and 84.8% respectively, summary table below shows results). Scores using Data Augmentation were significantly lower than without (averagely 7-8% drop). Below confusion matrices for both Lenet for with and without Data Augmentation:

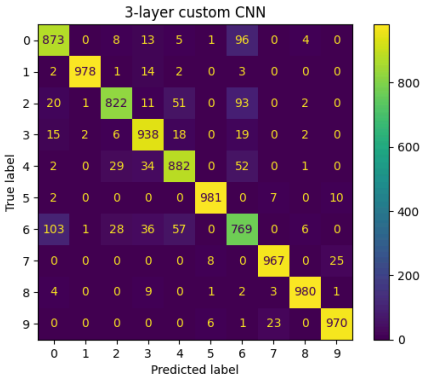


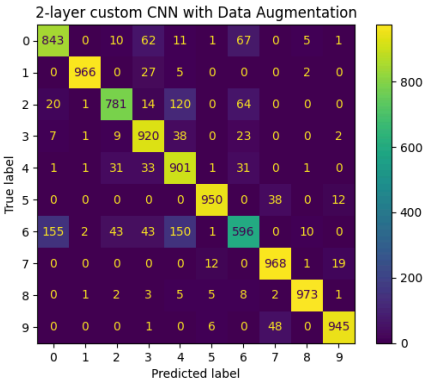


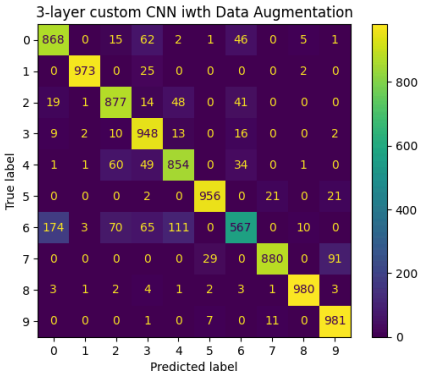
- **FashionMINST Custom Neural Network with Data Augmentation**

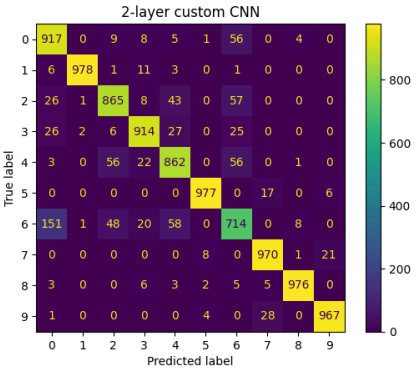
Two variants of Custom CNN were used 2- and 3-layers CNN. 2-layer CNN was composed of two convolutional networks, where each of them was followed by 2-dimensional normalisation batch. Last phase was composed of 2 fully connected layers (here linear transformation was used). In case 3-layer CNN one convalutional layer and normalisation batch were added.

Custom Neural Network already in case of using 10 epochs were achieving above 87% of accuracy. Due to satisfying score, the number of epochs wasn't increased.









Fashion MINST	Best Accuracy (2 layered)	Best Training time (2 layered)	Best Accuracy (3 layered)	Best Training time (3 layered)
Using **LeNet** architecture without data augmentation	84.8%	94 sec	x	x
using **LeNet** architecture with data augmentation	77.59%	124.35 sec	x	x
using customized **Net** architecture without data augmentation	91.4%	359.28 sec	91.6%	500.21 sec
using customized **Net** architecture with data augmentation	88.43%	364.50 seconds	89.16%	510.57 sec

4.2 CIFAR10

- **CIFAR-LeNet without Data Augmentation**

First, we work on a foundational experiment utilizing the LeNet architecture, adapted for compatibility with the CIFAR-10 dataset. The process involves loading the dataset, defining the LeNet model (which consists of convolutional, pooling, and fully connected layers), and preparing the data with transformations. The model is trained over 20 epochs using a stochastic gradient descent optimizer and cross-entropy loss function, with progress tracked by printing the loss every 200 mini-batches. After training, the model's accuracy is evaluated on a separate test set, and a confusion matrix is generated to analyze the results in

detail. The code demonstrates fundamental practices in deep learning, such as model definition, data preprocessing, training, and evaluation, without employing data augmentation techniques.
Opting to forego data augmentation, we observed a completion time of 207.753 seconds for the training process, thanks to the acceleration provided by CUDA technology. This baseline experiment was instrumental in setting the stage for subsequent models, allowing us to see the impact of increased complexity and data augmentation on overall performance.

- **CIFAR-LeNet with Data Augmentation**

We enhanced the approach by incorporating data augmentation (random crops and flips) and using a more sophisticated model, ResNet-18, pre-trained on a large dataset. We adapted ResNet-18 for CIFAR-10 by modifying its final layer and trained only this layer to fine-tune the model on our specific dataset. This approach aimed to leverage learned features from the pre-trained model, potentially improving accuracy by generalizing better to unseen data. The training and evaluation processes remained similar, focusing on accuracy and confusion matrix analysis to understand the model's performance improvements through data augmentation and a more advanced architecture.
Integrating data augmentation techniques, such as random cropping and horizontal flipping, into the LeNet model resulted in an extended training duration of 563.872 seconds. Our aim with data augmentation was to enhance the model's ability to generalize by introducing a more varied set of training images. This augmented LeNet model achieved a 41% accuracy on the test dataset, evidencing the significant role that data augmentation plays in the training of deep learning models. The augmented training complexity slightly extended the testing phase to 573.472 seconds.

- **Customized Net Architecture without Data Augmentation**

We implement a custom CNN architecture for CIFAR-10 image classification, demonstrating a more advanced approach to designing neural networks. The network is deeper and includes batch normalization after each convolutional layer to stabilize training and improve performance. Despite an initial mention of avoiding data augmentation, the preprocessing steps include transformations that effectively augment the training data, potentially improving the model's generalization.
The training process involves iterating through the dataset, applying the model to inputs, computing the loss, and updating the model's weights. We closely monitor the training progress through periodic logging of the loss. Upon completion, the model's accuracy is evaluated on a separate test dataset, providing a quantitative measure of its performance. The confusion matrix further allows us to dissect the model's predictive capabilities across the various classes, offering insights into any specific weaknesses or strengths in classification. This approach highlights the flexibility in neural network design and the importance of preprocessing, regularization (through dropout), and proper evaluation metrics in developing effective image classification models.
The complexity of this custom model extended the training time significantly to 13,431.239 seconds, reflecting the computational demands of its six convolutional layers (each followed by batch normalization and ReLU activation), three max-pooling layers, and the concluding fully connected layers for classification.

- **Pre-trained ResNet18 with Data Augmentation**

We utilize a pretrained ResNet18 model to perform image classification on the CIFAR-10 dataset, incorporating data augmentation to enhance the model's ability to generalize. By employing transfer learning, we leverage the robust feature-extraction capabilities of ResNet18, which has been pre-trained on the vast ImageNet dataset, adapting it to our specific task with minimal training. The process involves freezing the pre-trained model's parameters, except for the last layer, which we modify to suit our 10-class classification problem. This approach significantly reduces the computational cost and time required for training while still achieving high accuracy. We train the modified model on the augmented CIFAR-10 dataset, enabling it to learn from a more diverse set of images, thereby improving its predictive

performance. Through this notebook, we demonstrate a powerful method for applying deep learning to smaller datasets, showing how pre-trained models can be effectively adapted and fine-tuned for specific tasks. The final accuracy metric on the test set serves as a testament to the effectiveness of transfer learning and data augmentation in enhancing model performance.
By freezing the original network layers and retraining only the final layer, we utilized transfer learning efficiently, completing the training in 536.693 seconds. Although this approach yielded a test accuracy of 27%, it underscored the nuanced challenge of adapting complex pre-trained models to tasks they were not originally designed for.

CIFAR-10	Accuracy of the network	Finished training	Finished testing
using **LeNet** architecture without data augmentation	61 %	207.753 seconds	213.680 seconds
using **LeNet** architecture with data augmentation	41 %	563.872 seconds	573.472 seconds
using customized **Net** architecture without data augmentation	79 %	13431.239 seconds	14636.301 seconds
using customized **Net** architecture with data augmentation	27 %	536.693 seconds	545.218 seconds

- **Analysis**

Impact of Data Augmentation: Data augmentation typically helps improve model generalization and prevent overfitting by providing a more diverse set of training examples. However, its effectiveness greatly depends on the architecture and the specific augmentation techniques used. In these experiments, data augmentation did not consistently improve accuracy, indicating a mismatch between the augmentation techniques and the models or tasks.
Complexity vs. Performance: The customized Net architecture demonstrates that increased complexity can lead to higher accuracy, as seen in the non-augmented custom model achieving 79% accuracy. However, this comes at the cost of significantly longer training and testing times. Moreover, complexity does not guarantee better performance with data augmentation, as shown by the drop in accuracy to 27% for the augmented custom model.
Efficiency vs. Accuracy Trade-off: The experiments highlight a trade-off between efficiency (in terms of training/testing time) and accuracy. While the LeNet models are faster, they are less accurate than the more complex custom model without augmentation. Yet, when complexity and data augmentation are not optimally matched, performance can degrade, as seen in the augmented custom model.

In **summary**, these results underscore the importance of carefully selecting and tuning CNN architectures and data augmentation techniques based on the specific characteristics of the dataset and the desired balance between accuracy and computational efficiency.

4.2 Other ML algorithms

Results:

The Random Forest model has the highest score among the classifiers tested, indicating it is best suited for the dataset. The SVC performs modestly but shows improvement with a linear kernel over an RBF kernel. KNN has the lowest score, suggesting that it is less effective for the dataset, potentially due to the choice of hyperparameters or the nature of the data.

```
Best hyperparameters for RandomForest: {'n_estimators': 150, 'min_samples_split': 10, 'min_samples_leaf': 3, 'max_depth': 25}
Best score for RandomForest: 0.5677083333333333
```

	mean_fit_time	mean_score_time	params	mean_test_score
0	1.43	0.01	{'n_estimators': 200, 'min_samples_split': 10,...	0.55
1	0.34	0.00	{'n_estimators': 50, 'min_samples_split': 10, ...	0.52
2	4.32	0.03	{'n_estimators': 500, 'min_samples_split': 9, ...	0.56
3	2.12	0.01	{'n_estimators': 200, 'min_samples_split': 9, ...	0.57
4	5.21	0.04	{'n_estimators': 500, 'min_samples_split': 4, ...	0.56

```
Best hyperparameters for SVC: {'kernel': 'linear'}
Best score for SVC: 0.5651041666666667
```

	mean_fit_time	mean_score_time	params	mean_test_score
0	0.75	0.23	{'kernel': 'rbf'}	0.56
1	0.71	0.14	{'kernel': 'linear'}	0.57

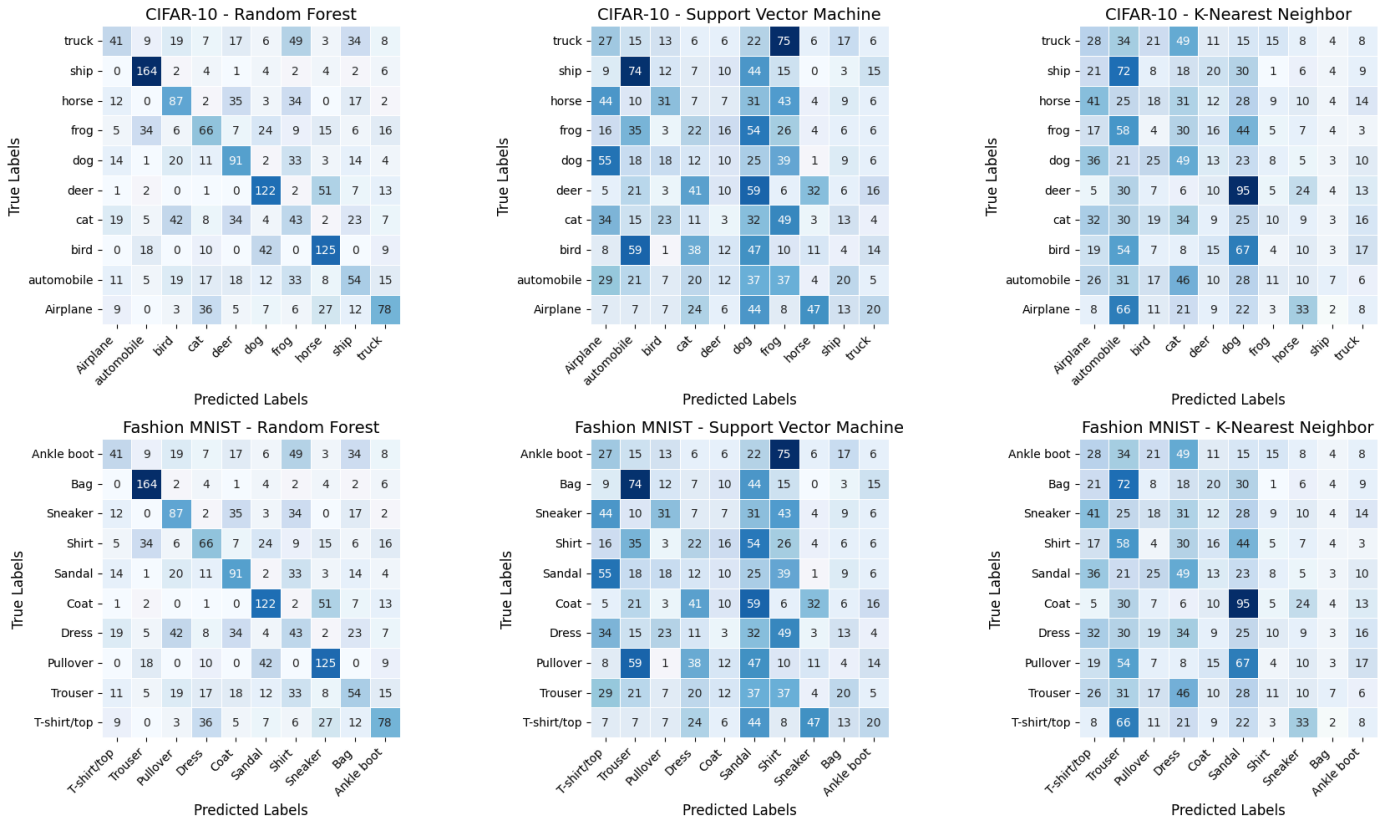
```
Best hyperparameters for KNN: {'p': 2, 'n_neighbors': 3}
Best score for KNN: 0.4395833333333334
```

	mean_fit_time	mean_score_time	params	mean_test_score
0	0.00	0.20	{'p': 1, 'n_neighbors': 3}	0.32
1	0.00	0.01	{'p': 2, 'n_neighbors': 3}	0.44
2	0.00	0.16	{'p': 1, 'n_neighbors': 5}	0.29
3	0.00	0.01	{'p': 2, 'n_neighbors': 5}	0.44
4	0.00	0.16	{'p': 1, 'n_neighbors': 7}	0.27

Analysis:

	precision	recall	f1-score	support
T-shirt/top	0.12	0.15	0.13	193.00
Trouser	0.17	0.38	0.24	189.00
Pullover	0.13	0.09	0.11	192.00
Dress	0.10	0.16	0.12	188.00
Coat	0.10	0.07	0.08	193.00
Sandal	0.25	0.48	0.33	199.00
Shirt	0.14	0.05	0.08	187.00
Sneaker	0.08	0.05	0.06	204.00
Bag	0.18	0.04	0.06	192.00
Ankle boot	0.08	0.04	0.06	183.00
accuracy	0.15	0.15	0.15	0.15
macro avg	0.14	0.15	0.13	1,920.00
weighted avg	0.14	0.15	0.13	1,920.00

	Precision	Recall	F1-score
Random Forest	0.44	0.45	0.44
SVC	0.17	0.17	0.16
KNN	0.14	0.15	0.13



The provided analysis indicates that the Random Forest classifier outperforms both the Support Vector Machine (SVM) and K-Nearest Neighbors (KNN) across precision, recall, and F1-score metrics for both CIFAR-10 and Fashion MNIST datasets. For CIFAR-10, all models struggle with class confusion, particularly between animals and vehicles. In the case of Fashion MNIST, the Random Forest classifier shows relative strength in distinguishing between distinct classes like 'Bag', but confuses similar clothing items such as 'Shirt' and 'T-shirt/top'. Both SVM and KNN models exhibit significant confusion across most classes in both datasets, leading to notably lower performance metrics compared to Random Forest. The overall low F1-scores, particularly for SVM and KNN, suggest these models have difficulties with the classification tasks given the datasets.

5. SUMMARY

In our report, we meticulously analyze a variety of neural network models and machine learning algorithms as they apply to the Fashion-MNIST and CIFAR-10 datasets. Our approach is systematically organized to include descriptions of the datasets, the methodology we followed for classifier training, the results we obtained for each dataset, and a comprehensive overarching analysis. We begin with an overview of the datasets: Fashion-MNIST, which consists of 60,000 training and 10,000 testing grayscale images of Zalando's article images in a 28x28 pixel format for image classification tasks, and CIFAR-10, which contains 60,000 32x32 color images across 10 categories, serving as a benchmark for evaluating convolutional neural network (CNN) models. Our methodology section details the use of shallow machine learning algorithms such as RandomForestClassifier, SupportVectorClassifier (SVC), and KNearestNeighbor (KNN), utilizing features like color histograms and SIFT descriptors for classification. Additionally, we evaluated Lenet5 and two custom networks (Net2 and Net3) on the Fashion-MNIST dataset, where Net2 comprises 2 convolutional layers, and Net3 adds an extra convolutional layer for more

complex feature extraction. For the CIFAR-10 dataset, we explored a deeper custom CNN architecture without data augmentation and a variant with data augmentation using a pre-trained ResNet18. Our findings reveal that for the Fashion-MNIST dataset, our custom neural networks surpassed Lenet5 in performance, with the 3-layer Net3 variant showing exceptional promise. In our CIFAR-10 experiments, both the augmented ResNet-18 and our custom CNN architecture demonstrated significant improvements in accuracy and generalization capabilities. We conclude our report with insights into how data augmentation and the depth and complexity of network architecture, such as ResNet-18 and our custom CNNs, are crucial for achieving higher accuracy in image classification tasks. Through our comprehensive analysis, we highlight the pivotal role of feature extraction, model complexity, and data augmentation in boosting the performance of machine learning models on the Fashion-MNIST and CIFAR-10 datasets, underscoring the collective efforts and findings of our team.