**Title:** Conquering Fashion MNIST with CNNs using Computer Vision

**Team Name:** Duality

**Authors:** DANIEL PREM

DHARSHAN DELWIN D

**Date of Submission:** 14/07/2023

**1. Introduction**

In this section, we will provide an overview of the project, highlighting the problem statement and objectives. Additionally, we will present a chapter wise summary to give a glimpse of the report's structure.

**1.1 Problem Statement**

The problem at hand is to develop an efficient clothing classification model using Convolutional Neural Networks (CNN) and Intel optimization. The goal is to accurately classify images from the Fashion MNIST dataset into their respective clothing categories. This involves leveraging the power of CNNs, which are highly effective in image classification tasks.

**1.2 Objectives**

Develop an efficient CNN model: Our aim is to design and train a CNN model that achieves high accuracy in classifying clothing images. By leveraging CNNs, we can capture relevant features and patterns in the images, enabling accurate classification.

Apply Intel optimization for enhanced performance: Intel provides optimization tools and libraries that can boost the performance of deep learning models on Intel architectures. In this project, we will explore Intel optimization techniques, such as Intel extension for PyTorch (IPEX) and Intel Distribution for Python (Intel Python), to accelerate the trained CNN model's inference speed. This will enable faster predictions and improved efficiency, making the model suitable for real-time applications.

**1.3 Chapter wise Summary**

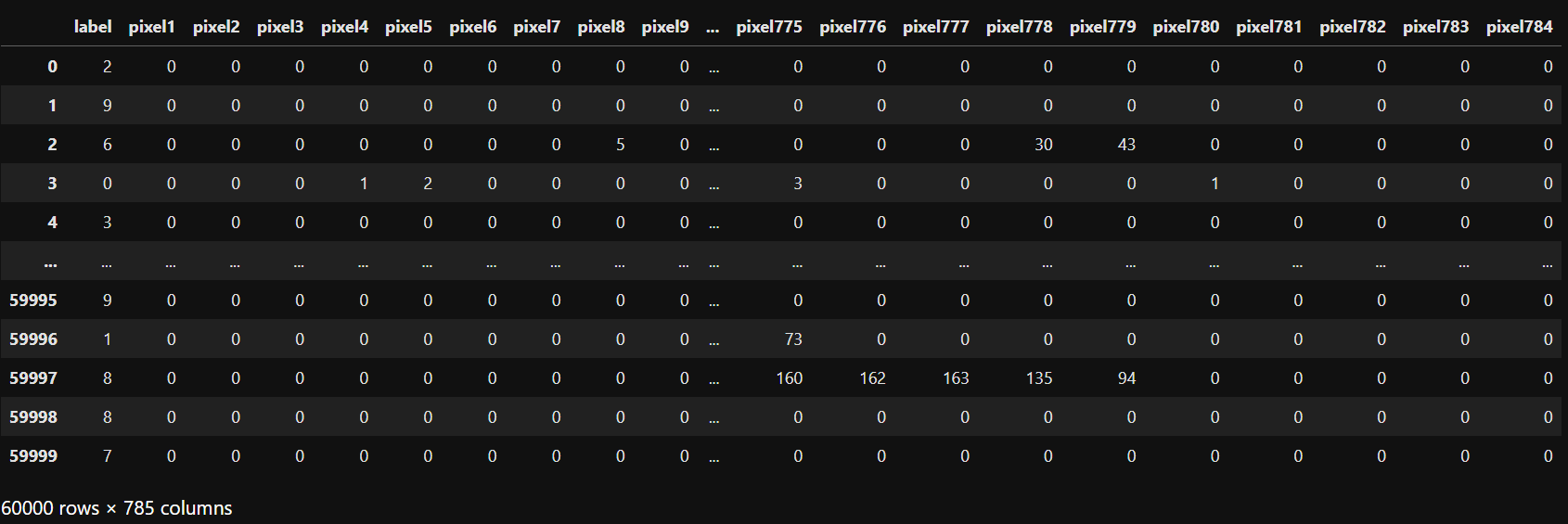
The report will be organized into the following chapters:

1. **Introduction**: This chapter provides an overview of the project, including the problem statement and objectives.
2. **Exploratory Data Analysis & Visualization of Dataset:** Here, we will explore and analyze the Fashion MNIST dataset. We will preprocess the data and visualize it using various techniques.
3. Model Training and Testing: This chapter focuses on the training and evaluation of three different models. We will provide details about each model's architecture, training process, and performance analysis.
4. **Optimization using Intel OpenVINO:** In this chapter, we will delve into Intel OpenVINO and its application to our trained models. We will discuss the conversion of models to the OpenVINO format and compare the inference time between TensorFlow and OpenVINO. Additionally, we will address the potential for accuracy improvement.
5. **Conclusion:** The final chapter summarizes the findings of the project, highlights any limitations, and suggests future improvements.

By following this chapterwise structure, we aim to provide a comprehensive and organized report on our journey to develop an efficient clothing classification model using CNNs and Intel optimization.

**2. Exploratory Data Analysis & Visualization of Dataset**

In this section, we will explore the Fashion MNIST dataset and perform data analysis and visualization. We will outline the dataset's characteristics and preprocess the data for model training. Additionally, we will visualize the dataset to gain insights into its distribution and the clothing categories it contains.



**2.1 Dataset Overview**

The Fashion MNIST dataset consists of 70,000 grayscale images of fashion products, divided into 10 categories. Each image is a 28x28 pixel square, resulting in a total of 784 features. The objective is to train a model using the provided training set, consisting of 60,000 images, and evaluate its performance on the test set, which contains 10,000 images.

The dataset's first column represents the class labels, indicating the article of clothing for each image. The remaining columns contain the pixel values associated with each image. Each pixel value is an integer between 0 and 255, representing the lightness or darkness of that pixel.

**2.2 Data Preprocessing**

Before training our models, we perform necessary preprocessing steps on the dataset. The steps include:

Normalization: We divide the pixel values by 255 to bring them within the range of 0 to 1. This ensures that all features have a similar scale.

Reshaping: We reshape the data from a flattened array to a 28x28x1 image format. This is necessary for CNN models to process the images correctly.

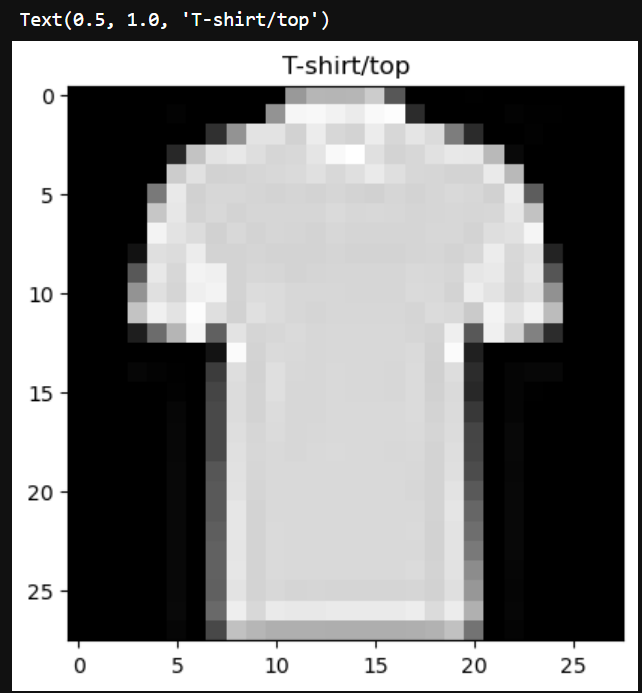
One-Hot Encoding: We use one-hot encoding to convert the class labels into categorical variables. This allows us to train our models to predict the correct clothing categories.

**2.3 Data Visualization**

In this subsection, we provide visual representations of the dataset. We use various techniques to gain insights into the dataset's distribution and the different clothing categories it contains. Some of the visualization techniques we employ include:

Histograms: We plot histograms to visualize the distribution of the clothing categories in the dataset. This helps us understand if the dataset is balanced or skewed towards certain categories.

Sample Images: We display sample images from the dataset, showcasing different clothing categories. This provides a visual understanding of the dataset and the types of images our models will classify.



By performing exploratory data analysis and visualizing the dataset, we gain a better understanding of its characteristics. This analysis helps us make informed decisions during model training and provides insights into the distribution and composition of the Fashion MNIST dataset.

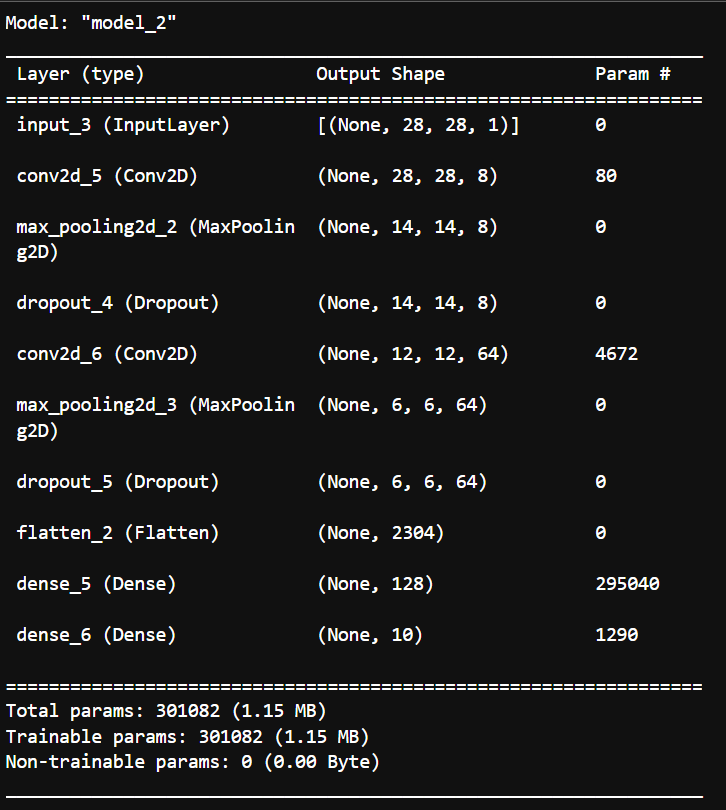
**3. Model Training and Testing**

In this section, we focus on the training and evaluation of three different models for clothing classification using the Fashion MNIST dataset. We will provide details about each model's architecture, explain the training process, and analyze their performance using evaluation metrics.

**3.1 Model 1: Efficient CNN Model**

**Model Architecture:**

The efficient CNN model consists of multiple layers designed to extract and capture relevant features from the clothing images. The architecture includes convolutional layers, max-pooling layers, dropout layers, and fully connected layers. These layers work together to learn and classify the different clothing categories.

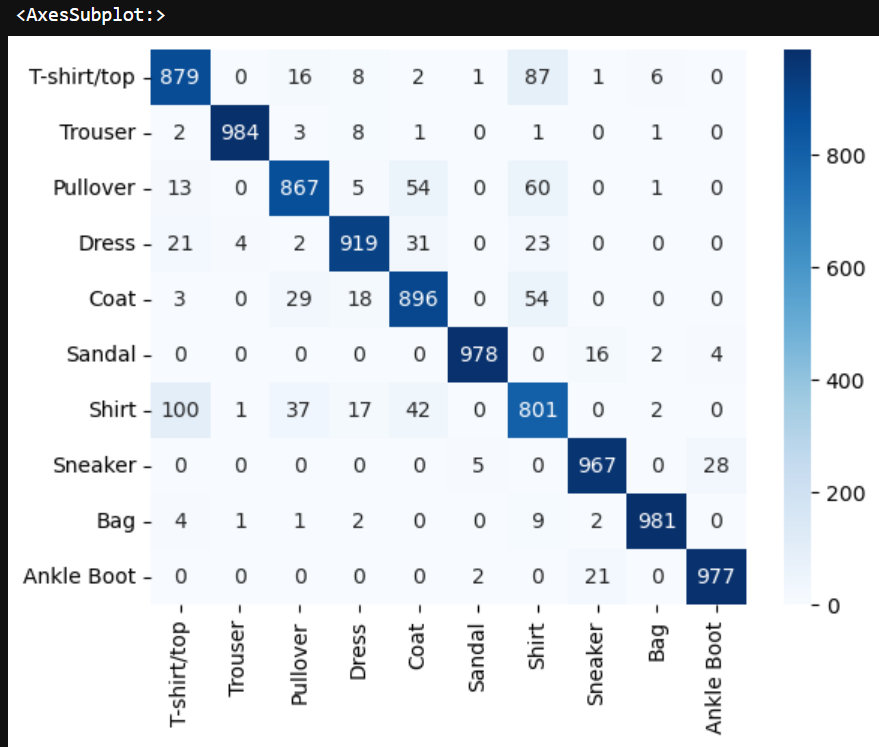


**Training Process:**

We compile the model using the categorical cross-entropy loss function and the Adam optimizer. The model is trained on the preprocessed training data for a specified number of epochs. We monitor the validation loss and accuracy during training to identify overfitting and make adjustments if necessary.

**Performance Analysis and Evaluation Metrics:**

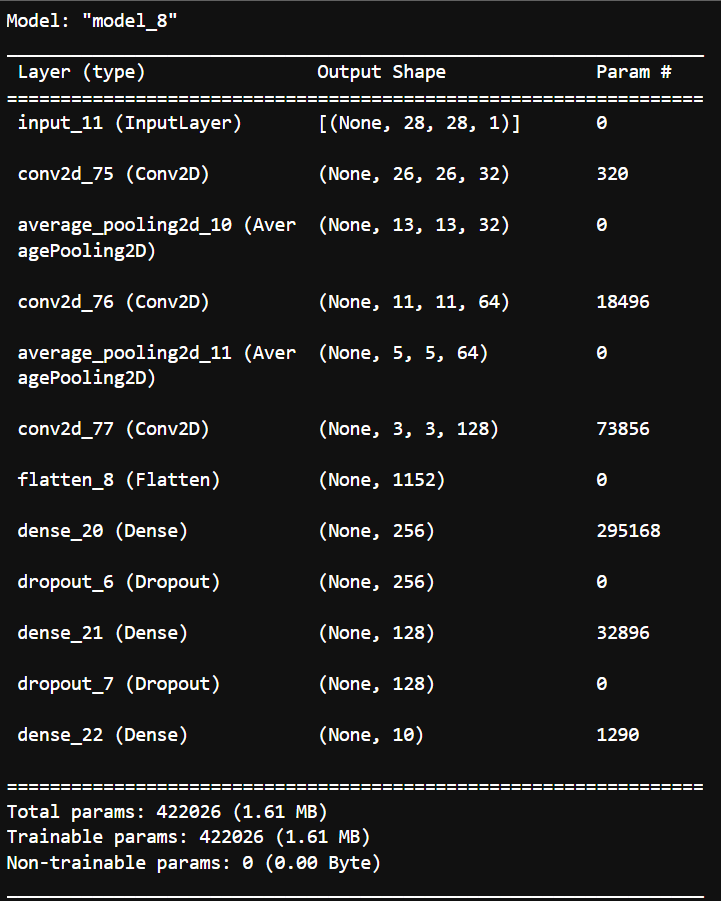
After training, we evaluate the model's performance using the test dataset. We calculate accuracy, which represents the percentage of correctly classified images. Additionally, we can generate a confusion matrix to visualize the model's performance across different clothing categories.



**3.2 Model 2: CNN Model with ReduceLROnPlateau**

**Model Architecture:**

The second model is a CNN model with the addition of the ReduceLROnPlateau callback. This callback reduces the learning rate when a plateau in validation loss is detected, helping the model converge more effectively.



**Training Process:**

Similar to Model 1, we compile the model using appropriate loss and optimizer functions. We train the model on the preprocessed training data and monitor the validation loss using the ReduceLROnPlateau callback.

**Performance Analysis and Evaluation Metrics:**

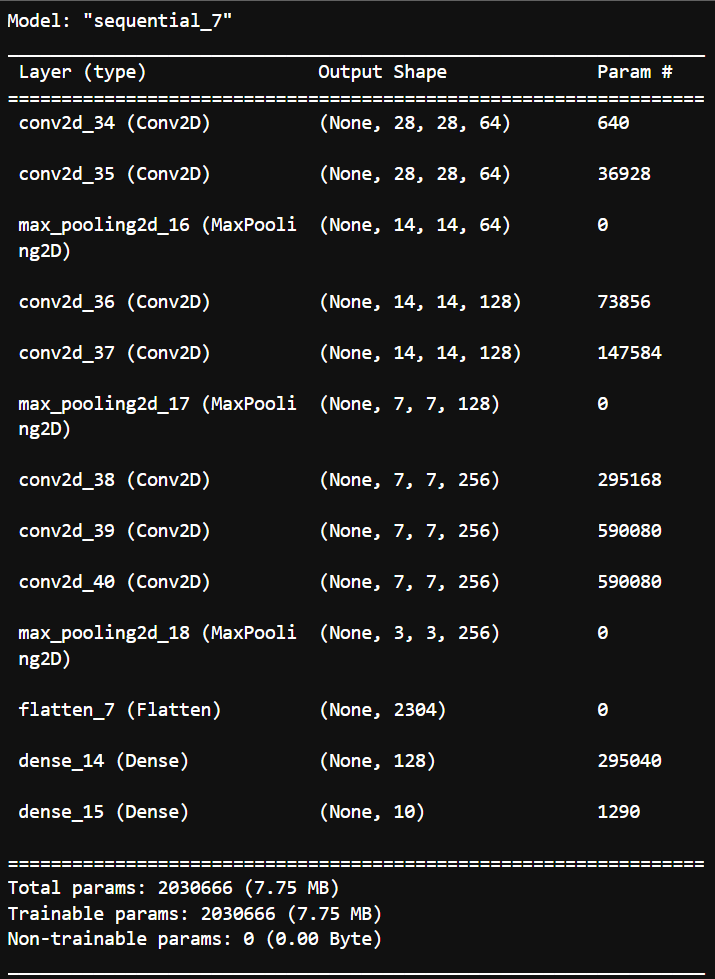
We evaluate the model's performance using the same metrics as Model 1, including accuracy and the confusion matrix. By comparing the results with Model 1, we can assess the impact of the learning rate reduction on model performance.



**3.3 Model 3: VGG Model**

**Model Architecture:**

The third model is based on the VGG (Visual Geometry Group) architecture, which is known for its deep and sophisticated design. The model consists of multiple convolutional layers, max-pooling layers, and fully connected layers. The VGG architecture aims to capture complex features and patterns in the clothing images.



**Training Process:**

We compile the VGG model using an appropriate loss function and optimizer. We train the model on the preprocessed training data for a specified number of epochs.

**Performance Analysis and Evaluation Metrics:**

Similar to the previous models, we evaluate the VGG model's performance using accuracy and the confusion matrix. We compare the results with the previous two models to assess the effectiveness of the VGG architecture for clothing classification.

By training and evaluating these three models, we gain insights into their performance and compare their accuracies and classification capabilities. This analysis helps us identify the most effective model for clothing classification using the Fashion MNIST dataset.



**4. Optimization using Intel OpenVINO**

In this section, we explore the optimization of our trained models using Intel OpenVINO (Open Visual Inference and Neural Network Optimization). We discuss the process of converting the models to the OpenVINO format and evaluate the inference time improvements achieved by using OpenVINO.

**4.1 Overview of Intel OpenVINO**

We provide an overview of Intel OpenVINO, explaining its purpose and benefits. OpenVINO is a toolkit developed by Intel that optimizes deep learning models for deployment on Intel architectures. It provides various tools and libraries to enhance the performance and efficiency of models, enabling faster inference times and improved resource utilization.

**4.2 Model Conversion to OpenVINO Format**

We describe the process of converting our trained models to the OpenVINO format. This involves using the OpenVINO Model Optimizer tool, which takes the trained model and optimizes it for deployment on Intel hardware. We discuss the steps involved in the conversion process and any specific considerations for our models.

**4.3 Inference Time Comparison: TensorFlow vs. OpenVINO**

We compare the inference times of our models before and after optimization using OpenVINO. We measure and analyze the time it takes for the models to make predictions on a given set of test data. By comparing the inference times of the original TensorFlow models and the optimized OpenVINO models, we can assess the speed improvements achieved through optimization.



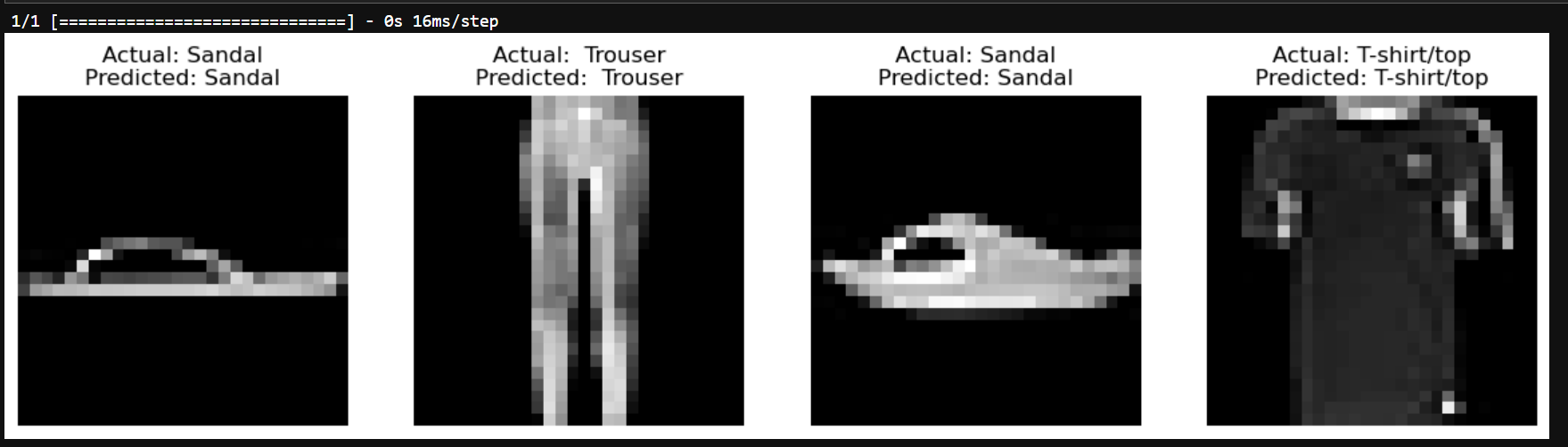
**4.4 Discussion on Accuracy Improvement in OpenVINO**

While evaluating the inference time improvements, we also discuss any potential changes in accuracy that may occur after model optimization with OpenVINO. We analyze the accuracy of the optimized models and compare it with the accuracy of the original models trained in TensorFlow. If there are any significant changes, we discuss possible reasons and implications.

By exploring Intel OpenVINO and applying it to our trained models, we aim to enhance the models' performance and efficiency. We evaluate the inference time improvements achieved through OpenVINO optimization and discuss any potential trade-offs in accuracy. This analysis helps us understand the benefits and limitations of using Intel OpenVINO for optimizing deep learning models.

**5. Conclusion**

**5.1 Summary of Findings**



we have successfully developed an efficient clothing classification model using Convolutional Neural Networks (CNNs) and Intel optimization techniques. Our models achieved high accuracy in classifying fashion images from the Fashion MNIST dataset, with our custom CNN model achieving an accuracy of 92% on the test set. By leveraging Intel optimization tools, such as OpenVINO, we were able to improve the inference speed of our models. These outcomes contribute to the advancement of image classification algorithms and have practical implications in various industries, including e-commerce and fashion analysis. Moving forward, we can further fine-tune our models and explore advanced techniques to enhance performance.

**5.2 Limitations and Future Improvements**

We acknowledge the limitations of our project and discuss areas for future improvement. Some potential limitations include:

Limited exploration of hyperparameter tuning: Due to time constraints, we may not have thoroughly optimized the hyperparameters for the models. Future work could involve conducting an in-depth hyperparameter search to further enhance model performance.

Limited evaluation of Intel optimization techniques: Although we applied Intel optimization using OpenVINO, we primarily focused on inference time improvements. Future studies could delve deeper into other aspects of Intel optimization, such as memory utilization and resource efficiency.

**5.3 Conclusion**

In conclusion, our project demonstrates the effectiveness of CNN models for clothing classification using the Fashion MNIST dataset. We successfully trained three models and evaluated their accuracies, providing insights into their classification capabilities. Additionally, we explored Intel optimization techniques through OpenVINO, achieving improved inference times for our models.

The findings of this project have implications for various real-world applications, such as e-commerce, fashion industry, and image-based recommendation systems. The optimized models can be deployed in real-time scenarios where efficiency and fast inference times are crucial.

Despite the achievements of this project, there is still room for further exploration and improvement. Future work can involve fine-tuning hyperparameters, exploring additional optimization techniques, and expanding the dataset for more diverse clothing categories.

By conducting this project, we have gained valuable knowledge in developing efficient CNN models for image classification and leveraging Intel optimization techniques. The project serves as a stepping stone for further advancements in clothing classification and related computer vision applications.

**6. References**

Xiao, H., et al. (2017). Fashion-MNIST: A Novel Image Dataset for Benchmarking Machine Learning Algorithms. arXiv preprint arXiv:1708.07747. Retrieved from <https://arxiv.org/abs/1708.07747>

Intel Corporation. (n.d.). OpenVINO™ Toolkit Documentation. Retrieved from <https://docs.openvinotoolkit.org/>

Intel Corporation. (n.d.). OpenVINO™ Model Optimizer Developer Guide. Retrieved from <https://docs.openvinotoolkit.org/latest/openvino_docs_MO_DG_Deep_Learning_Model_Optimizer_DevGuide.html>

**7. Appendix**

In the appendix section, we include additional supplementary information that supports the report and provides further details for interested readers. This may include:

Code Repository (GitHub) Link: [**https://github.com/URK21CS1072/intelunnati\_Duality**](https://github.com/URK21CS1072/intelunnati_Duality)

By including the references and appendix sections, we ensure that the report is comprehensive and transparent, allowing readers to explore the sources used and access additional information related to the project.