Introduction:

Zalando's Fashion MNIST dataset - https://www.kaggle.com/datasets/zalando-research/fashionmnist - is a set of 70,000 grayscale images, inspired from the original MNIST dataset. The problem statement for us was to build a convolutional neural network using computer vision to classify these images accurately.

Approach:

To classify the items in the Fashion MNIST dataset, we followed a step-by-step approach, in which we started off by preprocessing data, creating a model with numerous layers and then further optimizing it using other functions, enabling us to achieve a result of almost 95%. Here's the step by step approach:

1. Data Loading and Exploration: The first step was to load the Fashion MNIST dataset using the TensorFlow library. The dataset was divided into training and testing sets. By exploring the shape of the data, we ensured that it was loaded correctly.

2. Data Preprocessing: The images were preprocessed by converting the class labels into categorical format and scaling the pixel values between 0 and 1. This step is essential for training the neural network effectively.

3. Convolutional Neural Network (CNN): A Convolutional Neural Network (CNN) was constructed using the Sequential API from Keras. The architecture consisted of multiple convolutional layers with ReLU activation, batch normalization, max pooling, and dropout layers. These layers help extract and learn features from the input images.

4. Model Training: Initially, the model was trained for 10 epochs using the training data. Data augmentation techniques, such as rotation, shifting, and flipping, were applied using the ImageDataGenerator. These techniques help increase the model's ability to generalize to unseen data.

5. Model Evaluation and Improvement: After the initial training, the model's performance was evaluated using the validation set. To improve the model's accuracy, various techniques were applied:

- Hyperparameter Tuning: Different hyperparameters, such as the number of convolutional layers, filter sizes, pooling sizes, and dropout rates, were experimented with to find the optimal configuration. This involved tweaking the architecture of the CNN and observing its effect on the model's accuracy.

- **ReduceLROnPlateau Callback:** A ReduceLROnPlateau callback was used to reduce the learning rate if the validation loss did not improve for a certain number of epochs. This helped the model converge better during training.

- **Early Stopping:** An EarlyStopping callback was implemented to stop training if the validation loss did not improve for a certain number of epochs. This prevented overfitting and saved computational resources.

6. Model Evaluation and Reporting: The final model's performance was assessed on the test set using classification metrics such as precision, recall, and F1-score. The classification report provided insights into the model's accuracy for each class. The training and validation accuracy over epochs were visualized to analyze the model's learning progress.

Outcomes:

Throughout the process, different approaches were tried and evaluated based on their impact on the model's accuracy. The reasoning behind each approach was to improve the model's ability to generalize and classify the clothing items correctly. By experimenting with hyperparameters and implementing callbacks, the model's performance was optimized.

We were able to get a training accuracy of 99.6% on 48,000 images, over 50 epochs, validation accuracy of 94.67% over 12,000 images and a testing accuracy of 94.37% over 10,000 images.

The final outcome was a well-performing CNN model showcasing its effectiveness in classifying fashion items accurately.