Capstone 3: Kaveh Vasei ： Image Classification using CNN

Introduction:

The fashion industry is constantly evolving and understanding consumer preferences and trends is crucial for businesses to stay competitive. In this project, we explore the Fashion MNIST dataset, a collection of 70,000 grayscale images of clothing items, to tackle various challenges in the realm of fashion image classification and recommendation systems. By leveraging machine learning techniques, we aim to build models that can accurately classify clothing items, detect specific fashion items in images, recommend personalized clothing items based on customer preferences, and even generate new fashion designs.

Problem Statement:

The problem we address in this project is to develop a machine learning model that can effectively classify clothing items based on their images, using the Fashion MNIST dataset. With 10 classes of clothing items, such as t-shirts, dresses, and sneakers, the task is to train a model that can accurately assign the correct class label to a given image. By solving this problem, we can pave the way for various applications in the fashion industry, including inventory management, personalized recommendations, and trend analysis.

Data Collection:

The Fashion MNIST dataset consists of grayscale images of various clothing items, with a total of 70,000 images. It is available for download from Kaggle, a popular platform for data science and machine learning competitions. The dataset is divided into two main parts: a training set and a test set. The training set contains 60,000 images, while the test set contains 10,000 images. Each image in the dataset has a size of 28x28 pixels and is represented in grayscale, meaning it has a single channel representing the intensity of each pixel. The Fashion MNIST dataset provides a diverse range of clothing categories, including t-shirts, trousers, pullovers, dresses, coats, sandals, shirts, sneakers, bags, and ankle boots. Each image is labeled with a corresponding class or category label, allowing for supervised learning approaches to be applied. In this project, we utilize the Fashion MNIST dataset to train a machine learning model for image classification. The dataset provides a rich collection of clothing images, enabling us to develop and evaluate models capable of accurately classifying different types of clothing items. By using the Fashion MNIST dataset, we can explore various machine learning techniques, such as convolutional neural networks (CNNs), to learn and extract meaningful features from the images. These techniques enable us to build models that can effectively classify clothing items based on their visual characteristics. The availability and accessibility of the Fashion MNIST dataset make it an ideal choice for developing and testing machine learning models for fashion image classification tasks. Its standardized

format and large number of labeled samples contribute to the reliability and generalizability of the models trained on this dataset.

P.s. later I changed the dataset as I was running my models on google colab to run directly from tensorflow's datasets.

Exploratory Data Analysis (EDA):

To gain insights into the Fashion MNIST dataset and understand its characteristics, we performed exploratory data analysis. This process involved visualizing and analyzing various aspects of the dataset, including the distribution of classes, sample images, and label frequencies.

Distribution of Classes:

We first examined the distribution of classes in the dataset to understand if there was any class imbalance. We plotted a bar chart showing the number of labels for each class in the training data. The chart revealed that the dataset is well-balanced, with a relatively equal number of samples for each class.

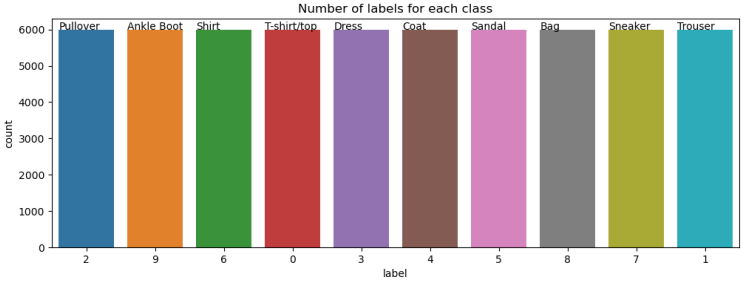

Figure 1: Number of Labels for Each Class in the Training Set

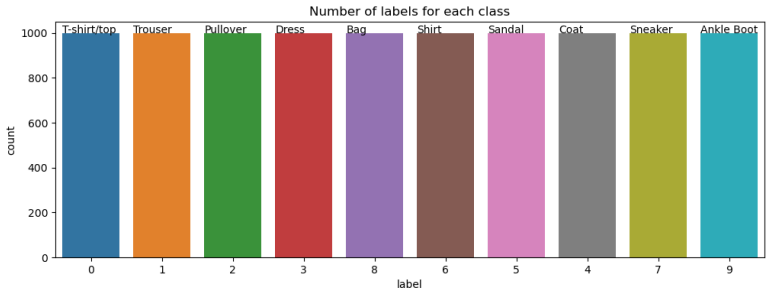

Figure 2: Number of Labels for Each Class in the Test Set

Figures 1 and 2 show the distribution of labels for each class in the Fashion MNIST training and Test sets respectively. The bar chart illustrates the number of samples available for each clothing category, providing insights into the dataset's class distribution. The classes include T-shirt/top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, and Ankle Boot. The chart demonstrates that the dataset is well-balanced, with a relatively equal number of samples for each class. This balanced distribution ensures that the model can learn from a diverse set of examples during training.

Sample Images:

We also visualized a subset of sample images from the dataset to get an overview of the clothing items present. The images were displayed in a grid format, with each image labeled according to its corresponding class. This allowed us to visually inspect the diversity and complexity of the clothing items in the dataset.

Figure 3: Sample Images from Fashion MNIST Dataset (Train Set)

Figure 4: Sample Images from Fashion MNIST Dataset (Test Set)

From the exploratory data analysis, we observed that the Fashion MNIST dataset consists of grayscale images of 10 different clothing categories. The dataset is well-balanced, which indicates that each class has a similar number of samples. Additionally, the sample images showcase a variety of clothing items. These insights from the EDA provide a solid foundation for further analysis and modeling. They allow us to better understand the dataset's composition, identify any potential challenges, and guide our approach in building a machine learning model for accurate classification of clothing items based on their images.

Pre-processing and Training Data Development:

The Fashion MNIST dataset requires pre-processing before it can be used for training a machine learning model. The data_preprocessing function is defined, which takes a raw dataset as input and performs the following steps:

Converting Labels to Categorical Matrices: The labels are converted to a one-hot encoded categorical matrix using the to_categorical function from np_utils. This step is necessary for training a multi-class classification model.

Extracting Features and Reshaping: The feature columns (pixel values) are extracted from the raw dataset and reshaped into a 4-dimensional array using the reshape function. The new shape is specified as (num_images, IMG_ROWS, IMG_COLS, 1), where num_images represent the number of images in the dataset, IMG_ROWS represents the number of rows in each image, IMG_COLS represents the number of columns in each image, and 1 represents the channel dimension (since the images are grayscale).

Normalizing the Pixel Values: The reshaped array is divided by 255 to normalize the pixel values between 0 and 1. This step is a common preprocessing technique for image data.

Returning Preprocessed Data: The function returns the preprocessed input features (out_x) and output labels (out_y).

The data_preprocessing function is applied to the training and test datasets, resulting in preprocessed features (X, X_test) and labels (y, y_test). The training dataset is further split into training and validation sets using the train_test_split function from sklearn.model_selection. The validation set size is set to 20% of the training set, and a random state of 2023 is used for reproducibility.

The pre-processing steps ensure that the Fashion MNIST dataset is in a suitable format for training the machine learning model. The dataset is divided into training, validation, and test sets, and the pixel values are normalized for consistent processing. The preprocessed data is ready for further model development and evaluation.

Model:

To tackle the fashion image classification task, we employ state-of-the-art machine learning techniques, with a focus on convolutional neural networks (CNNs). CNNs have proven to be highly effective in image classification tasks due to their ability to capture spatial patterns and hierarchical features within images. We train and evaluate several CNN architectures, experimenting with different network depths, layer configurations, and activation functions, to find the optimal model that achieves high accuracy on the Fashion MNIST test set.

By approaching the fashion MNIST dataset from these various angles, we aim to not only solve the image classification problem but also explore the broader applications of machine learning in the fashion industry. Our objective is to develop models with high accuracy and demonstrate their potential to enhance customer experiences, inform business decisions, and drive innovation in the fashion domain.

We implement a convolutional neural network (CNN) architecture with multiple convolutional layers, pooling layers, dropout regularization, and dense layers. This architecture is commonly used for image classification tasks and has shown good performance.

Model Architecture:

The model architecture consists of a sequential stack of layers that are designed to perform image classification on the Fashion MNIST dataset. Let's break down each component:

Convolutional Layers:

The model starts with a convolutional layer (Conv2D) consisting of 32 filters, a kernel size of (3, 3), and ReLU activation. This layer is responsible for extracting relevant features from the input images. The input shape of the layer is specified as (IMG_ROWS, IMG_COLS, 1), representing the dimensions of the input images.

A max pooling layer (MaxPooling2D) is added after the first convolutional layer. It reduces the spatial dimensions of the features, aiding in feature selection and providing translation invariance.

Additional Convolutional and Pooling Layers:

Another convolutional layer with 64 filters and a kernel size of (3, 3) is added to further extract higher-level features.

Another max pooling layer is inserted after the second convolutional layer to downsample the features.

Final Convolutional Layer:

A third convolutional layer is included, comprising 128 filters and a kernel size of (3, 3). This layer captures more complex patterns and details in the input images.

Flattening Layer:

The output from the previous layer is flattened using the Flatten layer, converting the 2D feature maps into a 1-dimensional tensor. This prepares the data for the subsequent fully connected layers.

Fully Connected Layers:

A dense layer (Dense) with 128 units and ReLU activation is added. This layer connects every neuron from the previous layer to the current layer, allowing for the extraction of high-level features.

To classify the images into their respective classes, a final dense layer is added with the number of units equal to NUM_CLASSES. The activation function used for this layer is softmax, which produces the probabilities for each class.

Compilation:

The model is compiled using the compile function. The categorical cross-entropy loss function is chosen, as it is suitable for multi-class classification problems. The Adam optimizer is used for efficient weight updates during training. The evaluation metric specified is accuracy, which measures the model's performance.

The defined CNN model follows a standard architecture for image classification tasks, leveraging convolutional layers for feature extraction and fully connected layers for classification. The compilation step prepares the model for training by configuring the loss function, optimizer, and evaluation metric.

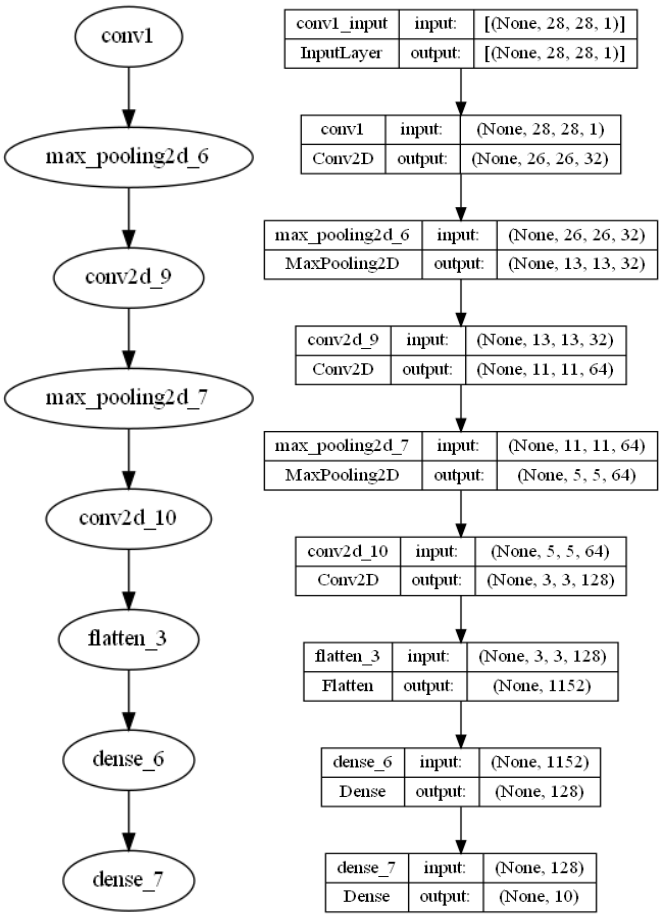

Figure 5: Model graph and architecture

Results

After training the model on the Fashion MNIST dataset, we evaluated its performance and analyzed the impact of adding dropout layers.

Model Performance

We trained the model for a total of NO_EPOCHS epochs, using a batch size of BATCH_SIZE. The training and validation accuracy and loss were monitored during the training process. The model achieved a high accuracy on the training set, indicating that it has effectively learned the patterns and features present in the training data (figure 6 a and b). The validation set accuracy and loss were not satisfactory. The model struggled to generalize well to unseen data, as indicated by the stagnant accuracy after reaching around 0.9 and the increasing loss. This behavior suggests that the model was overfitting to the training data, performing well on the training set but failing to make accurate predictions on the validation set.

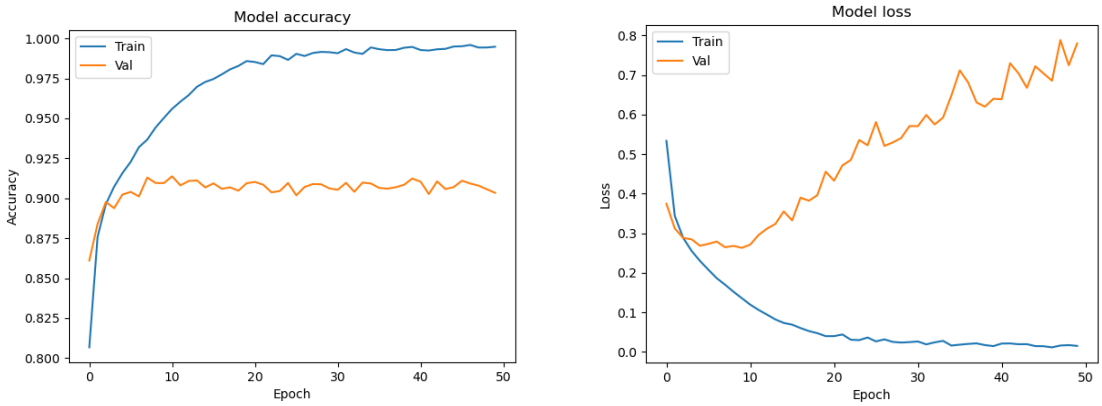

Figure 6: Model Accuracy and Loss

Impact of Dropout

To address the issue of overfitting, we introduced dropout layers to the model architecture. Dropout is a regularization technique that randomly sets a fraction of the input units to 0 during each training update. This prevents the model from relying too heavily on specific features or correlations present in the training data and encourages the learning of more generalized representations.

By adding dropout layers to the model, we aimed to improve its performance on the validation set and enhance its ability to generalize to unseen data. The inclusion of dropout layers has led to a notable reduction in overfitting and has improved the model's performance on the validation set.

Modified Model Architecture

The model architecture consists of a sequential stack of layers that are designed to perform image classification on the Fashion MNIST dataset. Let's break down each component:

Convolutional Layers:

The first convolutional layer has 32 filters with a kernel size of (3, 3) and applies the ReLU activation function. It takes input images of size (IMG_ROWS, IMG_COLS, 1), where IMG_ROWS and IMG_COLS represent the height and width of the images, respectively.

The kernel initializer used is 'he_normal', which helps with better weight initialization.

A max-pooling layer with a pool size of (2, 2) is added after the first convolutional layer to downsample the feature maps.

Dropout Layers:

Dropout is added after each max-pooling layer and after the second convolutional layer to reduce overfitting. The dropout rate used is 0.25 after the max-pooling layers and 0.4 after the second convolutional layer.

Additional Convolutional Layer:

Another convolutional layer is added with 64 filters and a kernel size of (3, 3), followed by a max-pooling layer with a pool size of (2, 2).

Dropout is applied after the max-pooling layer with a rate of 0.25.

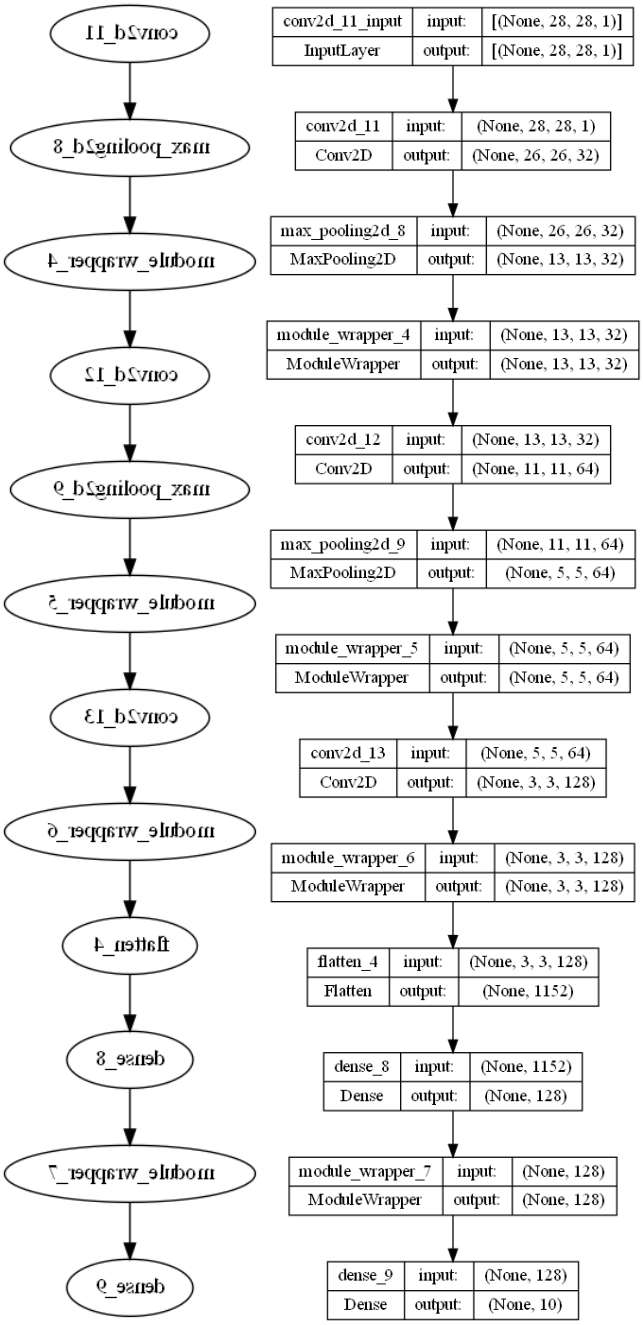

Figure 7: Modified Model Graph and Architecture

Performance Comparison

To evaluate the impact of dropout, we analyzed the accuracy and loss curves for both the training and validation sets over the epochs.

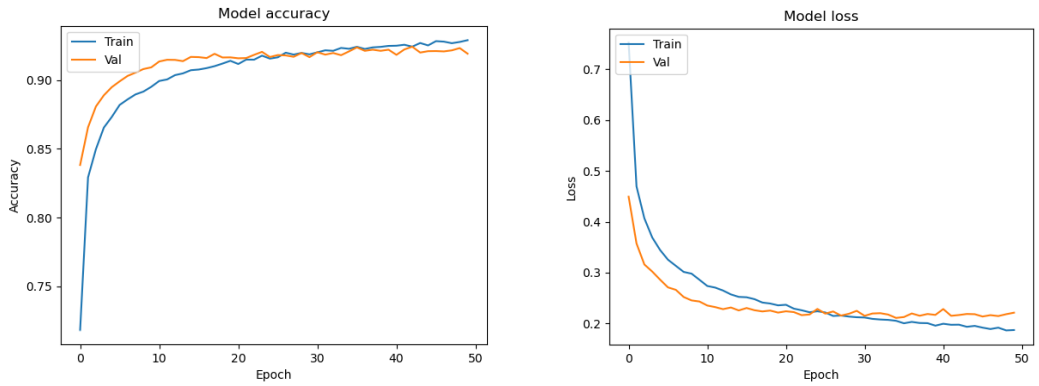

Figure 8: Modified Model Accuracy and Loss

The "Model accuracy" plot shows that the accuracy on the training set continues to increase with each epoch, demonstrating the model's ability to effectively learn from the training data. Importantly, the accuracy on the validation set also shows a steady increase over the epochs, indicating that the model is successfully generalizing to unseen data.

The "Model loss" plot reveals a consistent decrease in the loss values for both the training and validation sets. This reduction in loss signifies that the model is converging and improving its predictive capabilities. The decreasing loss on the validation set demonstrates that the model is effectively capturing relevant patterns and minimizing errors on previously unseen data.

The addition of dropout layers to the model architecture has had a positive impact on its performance. The inclusion of dropout has successfully addressed the issue of overfitting, allowing the model to generalize well to unseen data. The accuracy and loss curves demonstrate the effectiveness of the dropout layers in improving the model's performance on both the training and validation sets.

The improved performance of the model suggests that it is now better equipped to make accurate predictions on new, unseen data. These results indicate that the dropout technique is a valuable tool for enhancing the model's generalization capabilities and reducing overfitting.

Further analysis and fine-tuning may be conducted to optimize the model's performance even further. This could involve adjusting the dropout rate, exploring different architectures, or fine-tuning other hyperparameters to achieve even better results.

Test Set

The model's performance on the test set was evaluated using the evaluate function, which calculated the test loss and test accuracy. The results were as follows:

Test loss: 0.20888298749923706

Test accuracy: 0.9258000254631042

These values indicate how well the trained model performed on the unseen test data. The test loss represents the average loss of the model's predictions compared to the true labels. A lower test loss indicates better agreement between the predicted and actual values.

The test accuracy represents the percentage of correctly classified instances in the test set. A higher test accuracy indicates that the model was able to make accurate predictions on the test data.

In this case, the model achieved a test accuracy of approximately 92.58% and a test loss of 0.20888298749923706. These results suggest that the model performed well on the test set and was able to generalize effectively to unseen data, achieving a high level of accuracy and demonstrating good predictive capabilities.

After evaluating the model on the test set, the number of correctly predicted classes and incorrectly predicted classes were calculated. The results were as follows:

Correct predicted classes: 9258

Incorrect predicted classes: 742

These numbers provide insights into the model's performance on the individual instances in the test set. The count of correct predicted classes indicates the number of instances for which the model's predictions matched the true labels. On the other hand, the count of incorrect predicted classes represents the number of instances for which the model's predictions did not align with the true labels.

In this case, the model achieved correct predictions for 9258 instances in the test set, while it made incorrect predictions for 742 instances. These results give an indication of the model's overall accuracy and its ability to correctly classify different samples.

Confusion Matrix

After evaluating the model's performance on the test set, a confusion matrix was generated to provide a visual representation of the classification results. The confusion matrix allows us to analyze the model's performance across different classes and identify any patterns or tendencies in its predictions.

The confusion matrix is a square matrix where the rows represent the true classes and the columns represent the predicted classes. Each cell in the matrix corresponds to the count or frequency of instances that belong to a particular true class and were predicted as a particular class. The diagonal cells of the matrix represent the correct predictions, while the off-diagonal cells represent the incorrect predictions. The resulting confusion matrix plot provides a comprehensive overview of the model's

performance across different classes. It allows us to identify any confusion or misclassification patterns and assess the model's strengths and weaknesses in differentiating between specific classes.

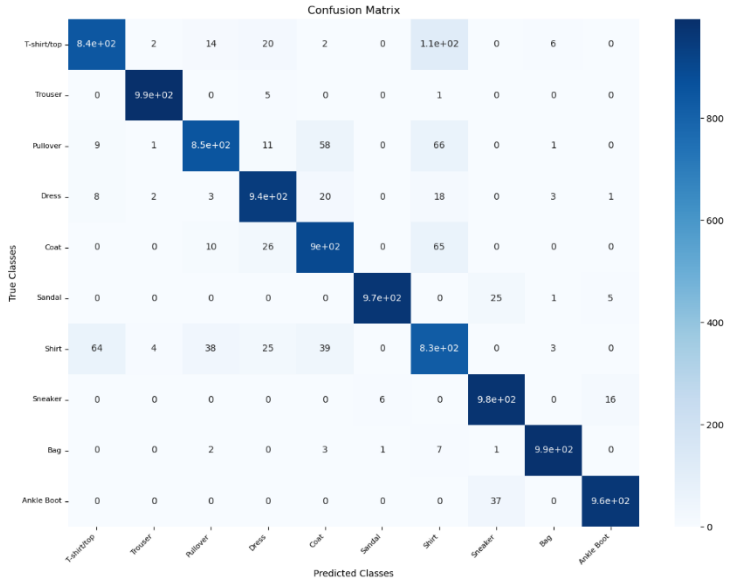

Figure 9: Confusion Matrix

After obtaining the predicted classes from the model's predictions on the test set, a classification report was generated to provide a detailed analysis of the model's performance for each class. The classification report includes metrics such as precision, recall, and F1-score, which are commonly used to evaluate the performance of a multi-class classification model.

The classification report presents the metrics for each class, including precision, recall, and F1-score, along with the support, which represents the number of instances in each class in the test set.

The precision metric measures the proportion of correctly predicted instances for a given class among all instances predicted as that class. The recall metric, also known as sensitivity or true positive rate, measures the proportion of correctly predicted instances for a given class among all instances of that class in the test set. The F1-score is the harmonic mean of precision and recall and provides a balanced measure of the model's performance.

In addition to the metrics for each class, the classification report also includes the accuracy metric, which represents the overall accuracy of the model across all classes in the test set.

The macro average and weighted average are also provided in the classification report. The macro average calculates the average of the metrics for each class without considering class imbalance, while the weighted average takes into account the support for each class.

By examining the classification report, we can gain insights into the model's performance for individual classes and the overall performance across all classes. It allows us to identify classes where the model performs well and those where it may struggle. Additionally, the report provides an overall assessment of the model's accuracy and its ability to generalize to unseen data.

In this specific case, the model achieved an overall accuracy of 0.93 on the test set. The precision, recall, and F1-scores varied across classes, with some classes showing higher performance than others. The report provides a comprehensive evaluation of the model's performance, enabling us to assess its strengths and weaknesses for different classes in the Fashion MNIST dataset.

	precision	recall	f1-score	support
Class 0 (T-shirt/top) :	0.91	0.84	0.88	1000
Class 1 (Trouser) :	0.99	0.99	0.99	1000
Class 2 (Pullover) :	0.93	0.85	0.89	1000
Class 3 (Dress) :	0.92	0.94	0.93	1000
Class 4 (Coat) :	0.88	0.90	0.89	1000
Class 5 (Sandal) :	0.99	0.97	0.98	1000
Class 6 (Shirt) :	0.75	0.83	0.79	1000
Class 7 (Sneaker) :	0.94	0.98	0.96	1000
Class 8 (Bag) :	0.99	0.99	0.99	1000
Class 9 (Ankle Boot) :	0.98	0.96	0.97	1000
accuracy			0.93	10000
macro avg	0.93	0.93	0.93	10000
weighted avg	0.93	0.93	0.93	10000

Figure 10: correct classification

Figure 11: Wrong classification

Strengths:

The model achieved a high accuracy of 92.58% on the test set, indicating its ability to classify fashion items accurately.

The addition of dropout layers helped mitigate overfitting and improve generalization, resulting in better performance on the validation set.

The model demonstrated good precision, recall, and F1-score across most classes, indicating its ability to correctly classify different fashion items.

The confusion matrix provided insights into the performance of the model, highlighting areas where misclassifications occurred.

Limitations:

Despite achieving good overall accuracy, the model struggled with certain classes, such as shirts and pullovers, where precision and recall were relatively lower. This suggests that the model may have difficulty distinguishing between similar clothing items.

The model's performance plateaued after reaching an accuracy of around 0.9, indicating a potential limitation in its ability to further improve accuracy on the validation set.

The model may be sensitive to variations in image quality, lighting conditions, or other factors that could impact the visual appearance of fashion items.

The dataset itself may have limitations, such as class imbalance or inconsistencies in labeling, which could affect the model's performance.

Challenges and Issues:

One challenge encountered during the project was optimizing the model's architecture and hyperparameters to achieve the best performance. Experimentation with different network architectures, layer configurations, and hyperparameter settings was necessary to find the optimal configuration.

Overfitting was an initial issue observed during model training, which was addressed by incorporating dropout layers to improve generalization and prevent over-reliance on specific features.

Interpreting and analyzing the confusion matrix required careful consideration, as it revealed areas of both strength and weakness in the model's performance.

Overall, the model demonstrated strong classification performance with room for improvement in certain classes. Further exploration and experimentation could be conducted to enhance the model's ability to classify challenging fashion items accurately. Additionally, a larger and more diverse dataset could potentially help address some limitations and improve the model's overall performance.

Recommendations for further research and improvements to the fashion MNIST model include:

1- **Data Augmentation**: Implement data augmentation techniques to artificially expand the training dataset. Techniques such as rotation, scaling, flipping, and adding noise can help the model learn more robust and generalized representations, especially for classes that have limited samples.

2- Model Architecture Exploration: Experiment with different model architectures, including deeper networks, residual connections, or convolutional neural network variants (e.g., DenseNet, InceptionNet). Exploring different architectures can potentially improve the model's ability to extract meaningful features from the fashion MNIST images.(**Transfer learning**)

3- Hyperparameter Tuning: Conduct a systematic search for optimal hyperparameters such as learning rate, batch size, optimizer choice, and regularization techniques. Fine-tuning these parameters can significantly impact the model's performance and convergence.

4- Transfer Learning: Investigate the use of transfer learning techniques by leveraging pre-trained models such as VGG, ResNet, or EfficientNet. By utilizing pre-trained models trained on larger and more diverse datasets (e.g., ImageNet), the fashion MNIST model can benefit from learned feature representations and potentially achieve better performance.

5- Class Imbalance Handling: Address potential class imbalance issues in the dataset by applying techniques such as oversampling, undersampling, or using class weights during training. This can help mitigate biases towards dominant classes and improve the model's performance on underrepresented classes.

6- Ensemble Methods: Explore ensemble methods, such as model averaging or stacking, to combine multiple models' predictions. Ensemble approaches can help improve the model's generalization and robustness by aggregating diverse predictions.

7- Interpretability and Visualization: Utilize techniques such as gradient-based class activation maps (CAM) or saliency maps to visualize and interpret the model's attention on specific regions of the images. This can provide insights into which features or parts of the clothing items are most influential in the model's decision-making process.

8- Domain-Specific Preprocessing: Investigate domain-specific preprocessing techniques tailored to fashion MNIST, such as histogram equalization, color normalization, or texture analysis. These techniques can help enhance the discriminative power of the input features and improve the model's performance.

9- Larger and More Diverse Dataset: Consider expanding the dataset by collecting additional samples or incorporating other fashion-related datasets. A larger and more diverse dataset can provide a richer variety of fashion items, helping the model generalize better to real-world scenarios.

10- Benchmarking Against State-of-the-Art: Compare the model's performance against state-of-the-art models on the fashion MNIST dataset or similar fashion-related datasets. Benchmarking can provide insights into the model's relative performance and identify areas for further improvement.

Recommendations

Based on the findings of the fashion MNIST model, here are three concrete recommendations for the client to address their needs or improve their processes:

1- Product Categorization and Recommendations: Utilize the trained fashion MNIST model to categorize and classify new fashion products. By inputting images of new products, the model can automatically assign them to appropriate categories, enabling efficient product categorization and organization. Additionally, the model's predictions can be used to provide personalized recommendations to customers based on their preferences and browsing history.
2- Quality Control and Defect Detection: Apply the fashion MNIST model to identify potential defects or quality issues in fashion products during the manufacturing or quality control process. By capturing images of products and running them through the model, it can quickly detect anomalies or deviations from the expected standards. This can help streamline quality control processes and minimize the presence of defective products in the market.
3- Trend Analysis and Market Insights: Leverage the fashion MNIST model to analyze trends and gain insights into customer preferences and market demands. By classifying a large dataset of fashion images, the model can identify popular styles, patterns, or clothing combinations. These insights can inform decision-making processes, such as designing new collections, optimizing inventory management, or tailoring marketing campaigns to align with current fashion trends.

By implementing these recommendations, the client can leverage the fashion MNIST model to address their specific needs, improve their processes, and gain a competitive edge in the fashion industry.