

Intel® Unnati Industrial Training - Summer 2023

Problem Statement –

**Conquering Fashion MNIST Dataset by
CNN using Computer Vision**

Submitted by –

Team Name - Byte Cruncher

Team Member Name – Vivek Kumar Singh

Branch- B.Tech CSE with AI & ML (Section -2)

Semester- IV

Admission No. – 21SCSE1180163

College Mentor – Mr. Gopal Chandra Jana

Industry Mentor – Mohan Nikam

Date of Submission : 09.06.2023

Abstract –Fashion MNIST is a popular benchmark dataset widely used in the field of computer vision for evaluating machine learning algorithms and deep neural networks. This report presents a comprehensive study on the classification of the Fashion MNIST dataset using Convolutional Neural Networks (CNNs). Recently, deep learning has been used extensively in a wide range of domains. A class of deep neural networks that give the most rigorous effects in solving real-world problems is a Convolutional Neural Network (CNN). Fashion businesses have used CNN on their e-commerce to solve many problems such as clothes recognition, clothes search and recommendation. A core step for all of these implementations is image classification. However, clothes classification is a challenge task as clothes have many properties, and the depth of clothes categorization is highly complicated. This complicated depth makes different classes to have very similar features, and so the classification problem becomes very hard. We have proposed a medium level CNN architecture using tensorflow and keras with two convolutional layers and two hidden layers which is giving an accuracy of approx. 92% on the training dataset and approx.. 90% accuracy on the testing dataset.

Keywords

Deep neural network; Fashion MNIST; Image classification; Convolutional Neural Network(CNN); Keras; Tensorflow ; softmax activation function; relu activation function.

Introduction

Deep Learning has been employed extensively and has shown excellent results in a variety of fields over the past few years, including computer vision, large data, automatic speech recognition, and natural language processing. CNN is a typical deep neural network architecture. CNN is a multi-layer perceptron neural network that is trained using the neural network back-propagation technique that extracts properties from the input data. From a huge amount of input (pictures), CNN can learn intricate, high-dimensional, non-linear mappings. CNN also provides a fantastic categorization average for photographs. The primary benefits of CNN are that it extracts the prominent features that never change and that it is invariant to shifting, scaling, and distortions of input data (pictures). Labels and classes are distorted in a similar way, but CNN does not suffer from these flaws.

Fashion classification, in which labels that describe the type of clothing are assigned to the photographs, is one of the most difficult multi-class classification problems. The diversity of the clothing attributes and the extensive level of clothing categorization both contribute to the challenge of this multi-class fashion classification problem. Different labels and classes have comparable characteristics as a result of this intricate depth.

Materials And Methods:

(1) Description of dataset

The Fashion MNIST dataset consists of 70,000 grayscale pictures of apparel that are broken down into 10 categories, as indicated in Table 1: Sneaker, Trouser/Jean, Handbag, Dress, Coat/Blazer, Slipper/Sandal, Shirt, T-shirt/top, Pullover, and Ankle Boot. There are 784 features total for each image, which is a 28x28 pixel square. A model must be trained using the provided training set, which consists of 60,000 photos, and its performance must be assessed using the test set, which consists of 10,000 images. The test photographs should be able to be classified by the model into one of the 10 fashion categories.

Each image has dimensions of 28 pixels in height and 28 pixels in width, for a total of 784 pixels. Each pixel has a single pixel value that describes its lightness or darkness, with greater values signifying darker pixels. This pixel value is an integer between 0 and 255. There are 785 columns in both the training and test data sets. The class labels in the first column, which stand in for the item of clothing, are seen above. The corresponding image's pixel values are located in the remaining columns.

Class label	Class index	
Sneaker	0	
Trouser/Jean	1	
Handbag	2	
Dress	3	
Coat/Blazer	4	
Slipper/Sandal	5	
Shirt	6	
T-shirt/top	7	
Pullover	8	
Ankle boot	9	

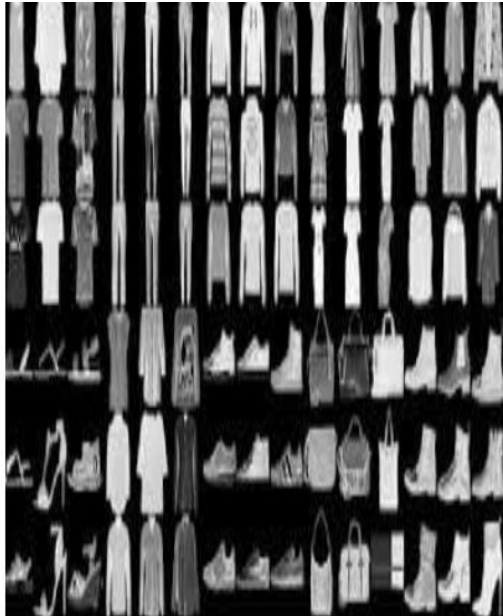


Table 1: Fashion-MNIST Dataset

(2) Dataset preprocessing

I. Loading the Dataset:

The first step in preprocessing the Fashion MNIST dataset is to load the dataset into memory. The dataset is available in the form of separate training and testing sets, typically stored in a suitable file format such as CSV or binary files. We load the dataset using a suitable library or framework such as NumPy or Pandas, ensuring that the data is correctly read and organized for further processing.

II. Data Cleaning:

Before proceeding with any further preprocessing steps, it is crucial to check the dataset for any missing or corrupted data. This step helps in identifying and handling any anomalies that might affect the subsequent analysis. Fortunately, the Fashion MNIST dataset is relatively clean and well-maintained, and thus, no specific data cleaning steps are required.

III. Normalization:

Normalization is a standard preprocessing step that aims to scale the input features to a common range. This step is particularly important in computer vision tasks, as it helps improve the convergence of optimization algorithms. For the Fashion MNIST dataset, normalization is performed by dividing the pixel values of each image by 255. This process ensures that all pixel values are within the range $[0, 1]$, making them suitable for further processing.

IV. Reshaping:

The Fashion MNIST dataset comes in the form of grayscale images with a resolution of 28x28 pixels. However, many machine learning algorithms and frameworks expect input data to be in a specific shape, such as a flattened vector or a specific tensor shape. Therefore, we reshape each image in the dataset to a 1D vector of size 784 (28x28). This transformation allows the images to be compatible with a wide range of machine learning models and algorithms.

V. Label Encoding:

The labels in the Fashion MNIST dataset are represented as integers ranging from 0 to 9, indicating the corresponding fashion category. To make the labels compatible with

machine learning algorithms, it is necessary to perform label encoding. This step converts the integer labels into a suitable format, such as one-hot encoding or ordinal encoding, depending on the specific requirements of the algorithm being used.

VI. Train-Validation Split:

To evaluate the performance of machine learning models accurately, it is common practice to split the dataset into separate training and validation sets. The training set is used to train the model, while the validation set helps in assessing the model's performance and tuning hyperparameters. Typically, a split of around 80% training and 20% validation is used for the Fashion MNIST dataset. After splitting the training dataset, we get 48000 images for training and 12000 images for validation.

Proposed approach with block diagram

CNN:

Convolutional neural networks are a subset of deep feed-forward artificial neural networks that are mostly employed in the processing of images for purposes including segmentation, classification, and other functions.

Convolutional, pooling, and fully-connected layers are the three types of layers it has. Convolution, in the context of CNNs, refers to the process of applying a set of filters or kernels to the input image. Each filter detects specific patterns or features, such as edges or textures, by convolving it across the image. This results in the extraction of relevant features and the creation of feature maps that highlight the presence of these patterns in different regions of the image.

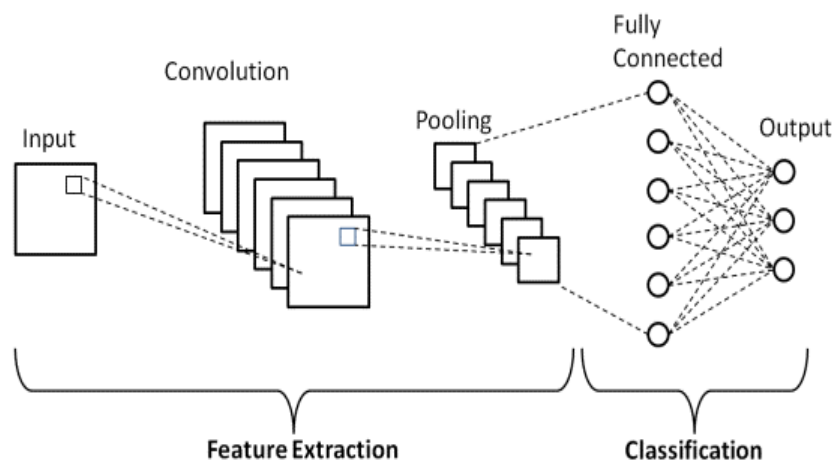


Figure 1: CNN Architecture

Below is the architecture of my final CNN model. It has 2 convolutional layers, 2 max. Pool layers, 2 dropout layers, 2 fully connected (dense) layer and 1 output (softmax) layer. Apart from these, it also has a flatten layer whose purpose is just to ‘flatten’ the output, i.e. convert a 2-D output to a 1-D output (which is then fed to the dense layer).

CONV layers compute the output of neurons that are connected to local regions in the input, each computing a dot product between their weights and a small region to which they are connected in the input volume. We used 32 filters in the first layers of all architectures. As a result, we have a volume of [26x26x32] in the output. The activation function RELU is used in all of the conv layers.

MAX POOL layers perform a down sampling operation along spatial dimensions (width, height), resulting in a volume of [13x13x32] from the first layer (Max1).

The fully-connected (FC) layers will calculate class scores. The RELU activation function is used in the two FC layers, while the SoftMax activation function is used in the last layer.

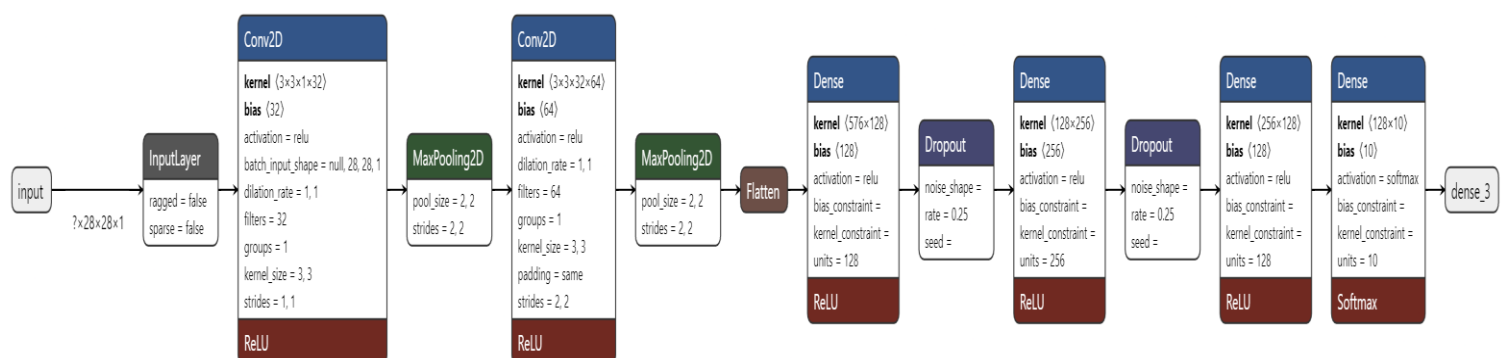


Figure 2: Final CNN Model Architecture

Tensorflow

You may design extremely adaptable CNN networks for computer vision problems using the open source TensorFlow framework.

Data from deep learning is represented via tensors. They are multidimensional arrays that are used to store different dataset dimensions. A feature is the term used to describe each dimension. As an illustration, a 3-dimensional tensor is used to depict a cube that stores data across X, Y, and Z accesses. Tensors can

contain very high dimensionality, with hundreds of dimensions of characteristics that are frequently utilised in deep learning applications.

Keras

A robust and user-friendly open-source Python library for creating and analysing deep learning models is called Keras. With only a few lines of code, you can define and train neural network models using this component of the TensorFlow library. Setting up your environment, installing Keras and Tensorflow, importing libraries and modules, loading picture data from MNIST, and preprocessing input data for Keras are all necessary steps in the construction of a CNN using Keras. Pixels in photos are used by convolutions to recognise and categorise them.

Result and discussion

Cross validation (validation dataset = 10% of training dataset) and hyperparameter tuning were done on the training dataset itself during the training phase.

The results of training and cross validation (validation) for a CNN model that I first tested out without using any regularisation techniques are shown in the table below.

The results below demonstrate that deep learning models have a considerable tendency to overfit (perform better on training datasets than on validation/test datasets).

Model	Model Architecture	Training Accuracy	Validation Accuracy	Test Accuracy
CNN	1 Convolutional layer, 1 max pool layer, 1 fully connected layer, 1 output layer	96.12	90.46	91.07

Table 1 : Results of CNN model without applying Regularization

The model is overfitted if training accuracy is 96% and test accuracy is 91%.

I used the regularisation approach known as “Dropout” to address the model overfitting issue, and I also added a few more max. pool layers. In the section under “CNN Model Architecture,” the architectural diagram for this CNN model is displayed above. The outcomes for the same are listed below.

Model	Model Architecture	Training Accuracy	Validation Accuracy	Test Accuracy
CNN	2 Convolutional layer, 2 max pool layer, 2 fully connected layer, 2 dropout layer, 1 output layer, Optimizer- adam, Loss- Sparse Categorical Crossentropy	92.13	90.44	90.01

Table 2 : Results of the CNN model with regularization (dropout) applied

A training accuracy value of 92% and test accuracy of 90% confirms that model is performing fine and there is no overfitting. Thus, this is our final CNN model.

Below are the plots for ‘Accuracy’ and ‘Loss’ for training and validation(test) phases for this final CNN model.

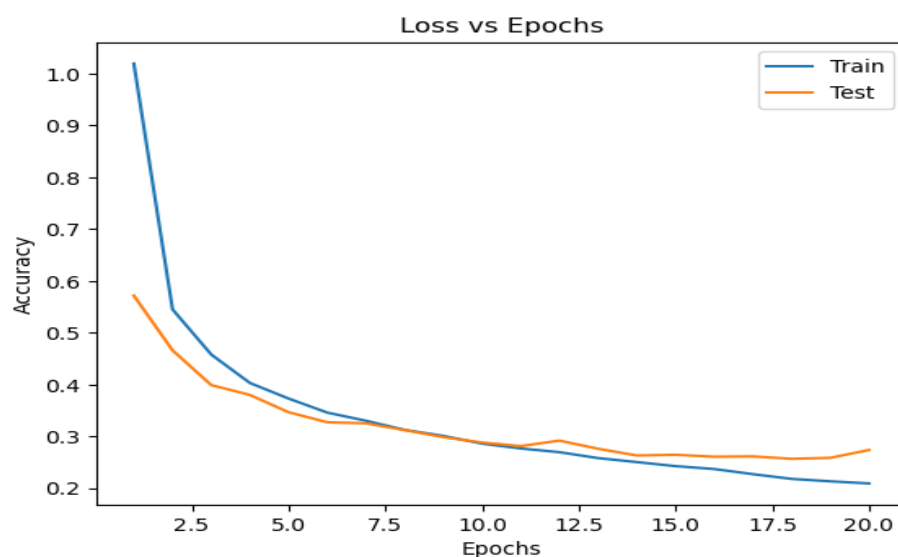


Figure 3: ‘Accuracy’ plot for CNN Model for training and validation(test) dataset

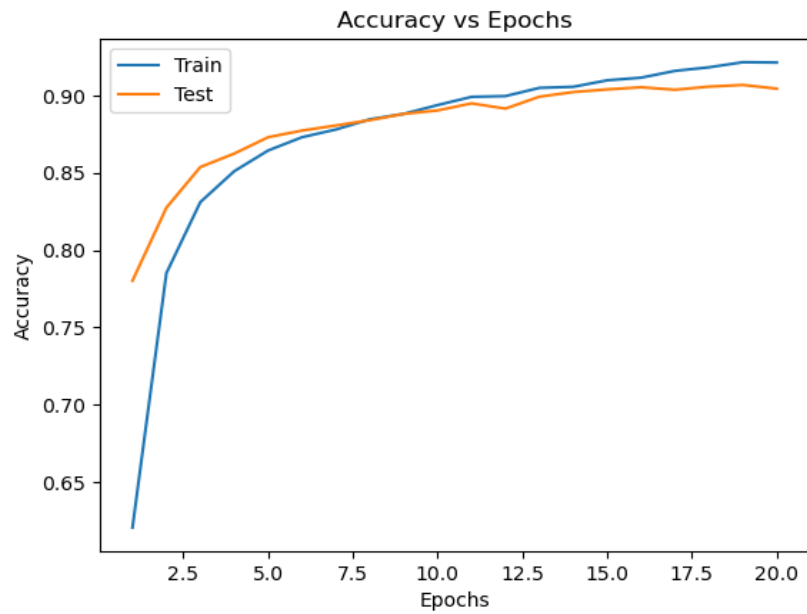


Figure 4: ‘Loss’ plot for CNN Model for training and validation(test) dataset

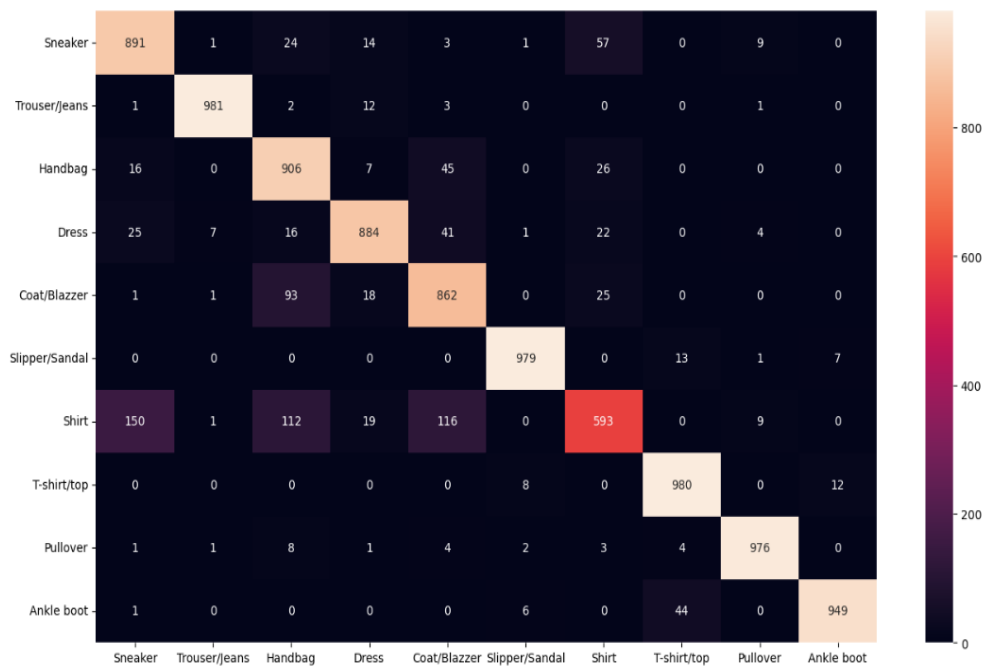


Figure 5: The model confusion matrix

	precision	recall	f1-score	support
Sneaker	0.85	0.85	0.85	1000
Trouser/Jeans	0.99	0.98	0.98	1000
Handbag	0.80	0.90	0.85	1000
Dress	0.89	0.91	0.90	1000
Coat/Blazzer	0.80	0.88	0.84	1000
Slipper/Sandal	0.98	0.97	0.97	1000
Shirt	0.81	0.61	0.69	1000
T-shirt/top	0.96	0.96	0.96	1000
Pullover	0.97	0.98	0.97	1000
Ankle boot	0.97	0.97	0.97	1000
accuracy			0.90	10000
macro avg	0.90	0.90	0.90	10000
weighted avg	0.90	0.90	0.90	10000

Table 2: Classification Report

Conclusion and future scope

CNN image recognition is widely used in the fashion industry for autonomous clothing labelling, retrieval, and categorization. Deep learning approaches have become more popular in recent years.

In this paper, we apply keras(Tensorflow) architecture on the Fashion MNIST dataset. Keras gives a higher performance (an accuracy over 92% was obtained) as compared to other existing models. We plan to conduct a comprehensive comparison between different CNN architectures (such as VGG16 and LeNet-5) on other clothes datasets (such as Image Net). We also plan to apply keras and other CNN architectures on a dataset of real clothes images collected by our self for the evaluation purposes.

References

- [1] <https://doi.org/10.1109/DTS52014.2021.9497988>
- [2] <https://doi.org/10.1016/j.eswa.2021.114805>
- [3] <https://doi.org/10.1109/ASET.2018.8379825>
- [4] <https://doi.org/10.3390/s22239544>
- [5] <https://doi.org/10.1109/ISPA54004.2022.9786364>
- [6] <https://doi.org/10.1109/ICCCIS56430.2022.10037748>
- [7] <https://dl.acm.org/doi/abs/10.1145/3520304.3533949>

Links to Solution

Code/Results Github link – https://github.com/svivek2002/intelunnati_ByteCruncher.git

Model link - <https://jupyter.oneapi.devcloud.intel.com/user/u194786/lab/workspaces/auto-s/tree/Model.ipynb>