

# AI DRIVEN FASHION ACCESSORIES IDENTIFIER USING MACHINE LEARNING TECHNIQUES BY IMAGE CLASSIFICATION

<sup>1</sup>A. Dinesh Kumar, <sup>2</sup>Dr.G.Vijay Kumar

<sup>1</sup>(Student Dept. of Master of Computer Applications, Amrita Sai Institute of Science and Technology, Paritala, Andhra Pradesh, 521180, India.

Email:dineshrocky15022@gmail.com)

<sup>2</sup> (Professor Dept. of Computer Science & Engineering, Amrita Sai Institute of Science and Technology, Paritala, Andhra Pradesh, 521180, India.

Email:gvk.vijay73@gmail.com)

\*\*\*\*\*

## Abstract:

In this project, we study image classification using the Fashion MNIST data with 70,000 grayscale images classed into t-shirts, dresses and sneakers. We want to create a model that can spot various clothing types correctly, as this will support fashion recommendation systems, automate tagging and handle inventory tasks.

The work is performed using Convolutional Neural Networks (CNNs) and is carried out using TensorFlow and Keras. Normalisation of data is done so that the model's training process will use consistent pixel values. In CNN, layers called convolutional are used to detect features, max-pooling for decrease in data and dense layers for classifying what's found. Using ReLU activations helps the neural network better learn, also stopping overfitting. A sparse categorical cross entropy loss is used to compile the model and it is optimised using the Adam optimizer, trained periodically using a validation split.

It is clear from the model that both accuracy and loss decrease during the training process. Following testing on the test set, its strength shows with a 91% accuracy. This result confirms that CNNs can be applied to perform real-world image classification tasks.

**Keywords** — *Image classification, Fashion MNIST, Convolutional Neural Network, TensorFlow, Keras, deep learning, accuracy.*

\*\*\*\*\*

## 1. Introduction

As image data appears in large amounts on digital systems, there is a greater need for tools to quickly sort and examine so much imagery. Highly advanced learning algorithms now make it possible to sort the Fashion MNIST dataset and other heavy data into different categories, including shirts, dresses or shoes. The key tool for this approach is

CNN which takes pixel information and distils useful characteristics that recognise each category.

Thanks to new developments in deep learning, along with TensorFlow and Keras, image classification models now work more accurately, can be used on larger data and are more versatile. Since these models study from carefully labelled photographs, they can efficiently deal with images they've not seen before, making them useful for many tasks. Today, sites that advise fashion, shop

online, organise inventory and forecast trends are all supported by automated image classification.

Although much has improved, the field still faces a number of problems. Handling equal representations in different datasets, achieving the same precision for multiple classes and boosting model performance for rapid results is still hard. Continual progress in neural network designs, important training methods and approaches to enlarging data is needed to handle these issues. Getting past these challenges allows our models to perform better when dealing with actual visual data in the real world.

## 2. System Analysis

Conventional systems of grouping fashion items using manual strategies usually take a lot of time, are not always reliable and work slowly with large sources of data. Computer vision methods for visual retail are not fully automated, so misclassification and a heavy workload are common. The lack of automatic image-based classification in many fashion areas makes things difficult, requiring manual labelled data and inconsistency among datasets.

This work uses deep learning and CNNs to create an automated way to classify clothing within the Fashion MNIST data. The system's accurate organisation helps fashion analytics and recommendation systems to work faster. With the model, users and system integrators understand more easily what the predictions mean. Using the system on e-commerce, mobile or inventory management systems is straightforward which helps reduce employee workload, cuts down on misclassifications and improves the way a lot of fashion image data is handled.

### 2.1 REQUIREMENT ANALYSIS AND SPECIFICATIONS

#### 2.1.1 Functional Requirements

##### 1. Image Classification

A total of ten predefined classes (T-shirt,

Trouser, Pullover, Dress) are used in the system for 28×28 grayscale images of fashion items.

##### 2. Model Training and Evaluation

It should be possible for users to apply a CNN on the Fashion MNIST dataset and cheque performance by looking at the accuracy and loss numbers.

##### 3. Prediction Capability

The system must be able to run photos it has not seen and then produce the most likely class with a confidence percentage.

##### 4. Visualization

It's important that the system can show training results, like accuracy and loss graphs, while the programme runs through the training cycles.

#### 2.1.2 Non-Functional Requirements

##### 1. Usability

The system ought to be simple to train and test, requiring little from users, so that students and researchers can use it.

##### 2. Performance

Handling large amounts of data should be done efficiently through GPU and CPU.

##### 3. Scalability

It should be able to work with many more data and additional categories of fashion in the future.

##### 4. Maintainability

The software should be broken into sections to make changes simpler, remain uncluttered and should be clearly explained.

##### 5. Security and Privacy

On deployed applications, any image information uploaded by users should be processed and anonymized when required.

### 2.1.3 Hardware Requirements

Component	Specification
Processor	Intel i5 or above / AMD equivalent
RAM	Minimum 8 GB
GPU	NVIDIA CUDA-enabled GPU (optional)
Storage	Minimum 2 GB free space

TABLE1:HARDWARE-REQ

### 2.1.4 Software Requirements

Component	Specification
Operating System	Windows, macOS, or Linux
Programming Language	Python 3.x
Libraries/Frameworks	TensorFlow / Keras, NumPy, Matplotlib, scikit-learn (optional)

TABLE2:SOFTWARE-REQ

## 2.2 Problem Definition

With such large amounts of digital photo data in the fashion world, traditional organisation methods for clothing are now slow and often lead to mistakes. Since standard systems cannot be automated and scaled, analysing vast data volumes in real-time is difficult which in turn restricts the development of support for apps including recommendation engines and managing inventory.

The important problem is that there is no intelligent solution yet that can label and sort fashion items from grayscale images on its own. Because of these issues, progress is slowed, expenses increase and the usefulness of fashion data analysis and development reduced.

This work creates a deep learning classifier designed through the Fashion MNIST dataset to overcome these challenges. Thanks to CNNs, the model figures out visual patterns in photos of fashion and neatly categorises them. With this approach, working more efficiently starts opening the door to AI use in fashion applications.

## 2.3 Scope of the Project

The research is centred on making and training a Convolutional Neural Network (CNN) in Python

and TensorFlow/Keras to classify grayscale fashion items from the Fashion MNIST database.

Image classification is well demonstrated in real cases from the fashion industry, as seen by the project.

- **E-commerce:** Automating product tagging and categorization.
- **Inventory management:** Improving stock control through image-based recognition.
- **Fashion recommendations:** Enabling personalized product suggestions.
- **AI-powered search tools:** Allowing visual search to find similar products.

Extended versions of the model can give real-time forecasts, handle images with high-colour detail and function on mobile and web devices. What's more, using it helps explain image processing, neural networks and deep learning pipelines and may lead to new growth with transfer learning and cloud deployment.

## 3. Implementation

### 3.1 Computational Environment

All experiments were done in Python 3.x, with TensorFlow 2.x and the related Keras API. I also used NumPy for handling numbers and Matplotlib to present my data. I ran all my notebooks on Jupyter in my laptop or on Google Colab's GPU support. Through tensorflow.keras.datasets, we used Fashion-MNIST as our benchmark dataset directly.

### 3.2 Data Preparation

We separated the Fashion-MNIST archive (containing 60 000 for training and 10 000 for testing, all in grayscale and with size  $28 \times 28$ ) into training and test sets. All pixel intensities were shifted to the range  $[0, 1]$  by dividing by 255 which speeds up how the model converges. All of the

dataset's garment categories are represented by numbers, ranging from 0 to 9, in their file names.

### 3.3 Network Topology

A compact Convolutional Neural Network (CNN) was engineered as follows:

Stage	Layers / Operations	Purpose
Input	$28 \times 28 \times 1$ tensor	Raw grayscale image
Feature extraction	$2 \times (\text{Conv} + \text{ReLU}) \rightarrow \text{MaxPool}$	Hierarchical local feature learning; spatial down-sampling to curb overfitting
Flatten	—	Vectorization of feature maps
Classification	Dense $\rightarrow$ ReLU $\rightarrow$ Dropout $\rightarrow$ Dense (softmax, 10)	Non-linear decision making; dropout for regularisation; probability output over classes

TABLE2:NETWORK-TOPOLOGY

### 3.4 Model Configuration

The Adam optimiser was picked (with defaults  $\beta_1 = 0.9$  and  $\beta_2 = 0.999$ ), both the loss and the evaluation metric selected were sparse categorical cross-entropy and the accuracy metric was also used.

### 3.5 Training Protocol

Around 20 or so epochs were run and in each case the data was divided into train and validation subsets at a 9 to 1 ratio. A wait of 3 epochs was used to stop training when the loss on the validation set didn't drop anymore. The batch size throughout the study remained 128.

### 3.6 Post-Training Evaluation

Quantitative results were obtained for performance on the test data that was not used for training. Metrics presented are overall accuracy, loss and a confusion matrix to let you see the strengths and weaknesses of each class. Qualitative assessment of test images was done by comparing their predicted labels to their real labels.

### 3.7 Learning Dynamics & Error Analysis

Graphics were created for training and validation loss and accuracy to confirm convergence and find out if overfitting happened. Systematic issues in the initial results of poorly recognised classes were analysed to help decide how to update the architecture or add to the dataset.

## 4. System Specifications

### 4.1 Dataset — Fashion MNIST

Total images	70 000 (60 000 train / 10 000 test)
Image format	$28 \times 28$ px, grayscale
Classes	10 fashion categories (T-shirt/top, Trouser, Sneaker, ...)

TABLE3:DATASET

### 4.2 Execution Platform

Local (Anaconda / Jupyter)	Full control, runs on your own GPU/CPU
Cloud (Google Colab)	Free GPU acceleration, zero local setup

TABLE4:EXECUTION-PLATFORM

## 5. System Design

### 5.1 Flow Diagram

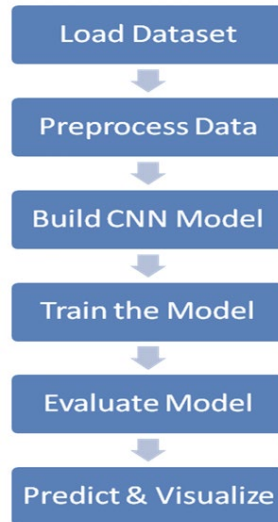


Fig1:Flow-Diagram

### 5.2 Algorithm Used

#### Convolutional Neural Network is known as CNN:

Among all the networks today, CNNs are standard choices in image classification, making it unnecessary to create features manually by hand. For this study, the network deals with grayscale Fashion-MNIST images in a  $28 \times 28$  size which are then mapped to 10 garment categories.

Stage	Purpose	Key Operations
<b>Input</b>	Standardize data format	Images reshaped (if required) to $1 \times 28 \times 28$ tensors
<b>Convolution</b>	Extract local patterns (edges, textures)	$3 \times 3$ kernels slide across the image, producing feature maps
<b>ReLU</b>	Inject non-linearity	$f(x) = \max(0, x)$ accelerates convergence and mitigates vanishing gradients
<b>Max-Pooling</b>	Down-sample while retaining salient cues	$2 \times 2$ windows halve spatial resolution, lowering computation and over-fitting risk
<b>Flatten</b>	Bridge 2-D spatial features to 1-D vectors	Feature maps concatenated into a single vector
<b>Fully Connected</b>	Integrate global evidence	Dense layers perform high-level reasoning over aggregated features
<b>Softmax Output</b>	Produce class probabilities	Outputs a 10-element vector; the arg-max denotes the predicted label

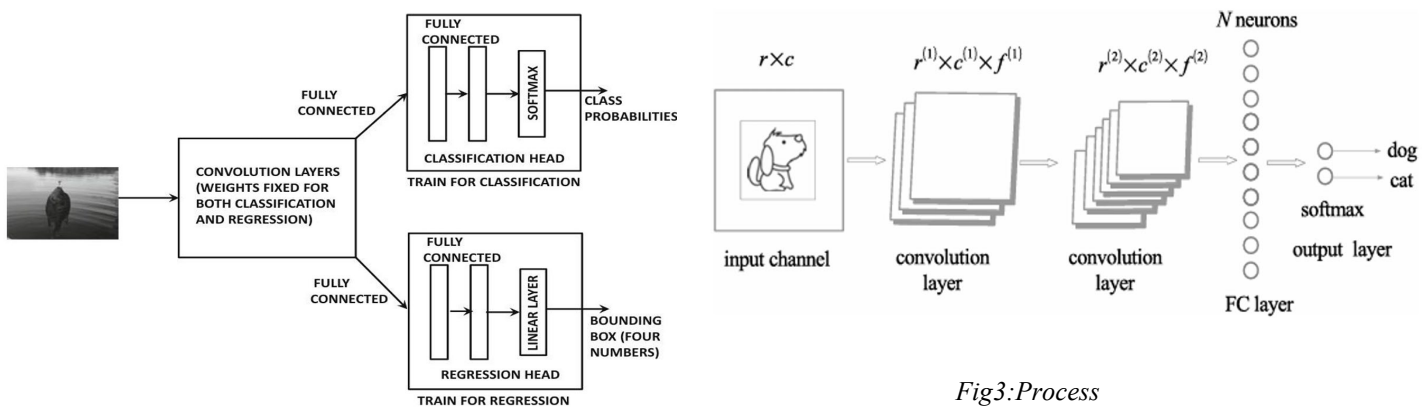


Fig2:CNN

Fig3:Process

## 6. Conclusion

The truth is, Convolutional Neural Networks do a great job at efficiency in classification. Smartphones can manage most processing on their own so your details don't need to be sent to a data centre. When we incorporate the CNN blocks into Python-coded AI decision-support pipelines, people find they actually work better and get more done. There's more ahead: the field is now aiming for model-free reinforcement learning, delivering systems that boost their performance by themselves as they go.

## Acknowledgement

I am thankful to the Management of Amrita Sai Institute of Science and Technology for giving me an opportunity to work with his project.

I would like to thank **Dr. M. Sasidhar**, Principal, Amrita Sai institute of science and technology, for his constant encouragement and support during the progress of this work.

I am deeply grateful to **Dr. P. Chiranjeevi**, Professor and Head of the Department, for his valuable guidance and consistent support during the course of the project.

A special note of thanks to my internal guide, **Dr. G. Vijay Kumar**, for his exceptional guidance, constant motivation, and continuous encouragement, which played a crucial role in the successful completion of this project.

**A.DINESH KUMAR**

## References

- [1] H. J. Kim, D. H. Lee, A. Niaz, C. Y. Kim, A. A. Memon, and K. N. Choi, "Multiple-clothing detection and fashion landmark estimation using a single-stage detector," *IEEE Access*, vol. 9, pp. 11694–11704, 2021, doi: 10.1109/ACCESS.2021.3051424.
- [2] Y. Wei *et al.*, "Cross-modal retrieval with CNN visual features: A new baseline," *IEEE Trans. Cybern.*, vol. 47, no. 2, pp. 1–12, 2016, doi: 10.1109/TCYB.2016.2519449.
- [3] M. S. Amin, C. Wang, and S. Jabeen, "Fashion sub-categories and attributes prediction model using

- deep learning,” *Vis. Comput.*, Jun. 2022, doi: 10.1007/s00371-022-02520-3.
- [4] S. Yuan, L. Zhong, and L. Li, “WhatFits—Deep learning for clothing collocation,” in *Proc. 7th Int. Conf. Behavioural and Social Comput. (BESC)*, Nov. 2020, pp. 1–4.
- [5] J. Shi, X. Song, Z. Liu, and L. Nie, “Fashion graph-enhanced personalized complementary clothing recommendation,” *J. Cyber Secur.*, vol. 6, no. 5, pp. 181–198, 2021.
- [6] H. Tuinhof, C. Pirker, and M. Haltmeier, “Image-based fashion product recommendation with deep learning,” *arXiv preprint arXiv:1805.08694*, 2018.
- [7] S. Salman and X. Liu, “Overfitting mechanism and avoidance in deep neural networks,” *arXiv preprint arXiv:1901.06566*, 2019.
- [8] Z.-Q. Zhao *et al.*, “Object detection with deep learning: A review,” *arXiv preprint arXiv:1807.05511*, 2018.
- [9] M. H. Kiapour, X. Han, S. Lazebnik, A. C. Berg, and T. L. Berg, “Where to buy it: Matching street clothing photos in online shops,” in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, 2015, pp. 3343–3351.
- [10] Y. Ge, R. Zhang, X. Wang, X. Tang, and P. Luo, “DeepFashion2: A versatile benchmark for detection, pose estimation, segmentation and re-identification of clothing images,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2019, pp. 5337–5345.