Abstract :
In this project, we have built a fashion apparel recognition using the Convolutional Neural Network (CNN) model. To train the CNN model, we have used the Fashion MNIST dataset. After successful training, the CNN model can predict the name of the class given apparel item belongs to. This is a multiclass classification problem in which there are 10 apparel classes the items will be classified.
The fashion training set consists of 70,000 images divided into 60,000 training and 10,000 testing samples. Dataset sample consists of 28x28 grayscale images, associated with a label from 10 classes.
So the end goal is to train and test the model using Convolution neural network.
This report focuses on the performance of several classifiers on the Fashion-MNIST dataset. Fashion-MNIST is a more difficult version of the classic MNIST benchmark image dataset. This dataset consists of 10 classes of 28x28 grayscale images of fashion items. The data is normalized and principal component analysis and Fisher's Linear Discriminant are applied. MPP cases 1, 2, and 3, k-nearest neighbors, and decision trees are evaluated on the dataset. Additionally, 3-layer backpropagation neural networks and
a convolutional neural network (CNN) are also tested. Performance for these classifiers is compared using the data's built-in train/test split and using 10-fold cross validation. Additionally, k-means and winner-takes-all clustering techniques are investigated for visualizing and reproducing the clusters in the data. The CNN classifier achieves the best result of 92.9%.

Objectives :
The prime objective of this article is to implement a CNN to perform image classification on the famous fashion MNIST dataset. In this, we will be implementing our own CNN architecture. The process will be divided into three steps: data analysis, model training, and prediction.
The other objective is to identify (predict) different fashion products from the given images using various best possible Deep Learning Models (Algorithms) and compare their results (performance measures/scores) to arrive at the best ML model.

Introduction :
The Fashion-MNIST clothing classification problem is a new standard dataset used in computer vision and deep learning.
Although the dataset is relatively simple, it can be used as the basis for learning and practicing how to develop, evaluate, and use deep convolutional neural networks for image classification from scratch. This includes how to develop a robust test harness for estimating the performance of the model, how to explore improvements to the model, and how to save the model and later load it to make predictions on new data.
Sight is the sense which humans rely on the most. It's used in all facets of life from driving to recognizing objects and faces. While this is an intuitive task to a human, image recognition is much more difficult for computers. This is due to the high dimensionality of images, the large number of classes of objects, and the potential variation within classes of things. Computer vision has applications in many areas. It can be used in transportation, such as self-driving cars. Another field which can see a big, life-changing impact is medical imaging. In radiology, computer vision can potentially be used to detect cancers from tissue scans. Since computer vision is very important application of pattern recognition, we decided to investigate it in this project using the Fashion-MNIST dataset.

Theory :

Deep learning is a subfield of machine learning related to artificial neural networks. The word deep means bigger neural networks with a lot of hidden units. Deep learning's CNN's have proved to be the state-of-the-art technique for image recognition tasks. Keras is a deep learning library in Python which provides an interface for creating an artificial neural network. It is an open-sourced program. It is built on top of Tensorflow.

In CNNs, the nodes in the hidden layers don't always share their output with every node in the next layer (known as convolutional layers). Deep learning allows machines to identify and extract features from images. This means they can learn the features to look for in images by analysing lots of pictures.

CNN is a subclass of an artificial neural network(ANN) which is mostly used for image-related applications. The input for a CNN is an image, and there are different operations performed on that image to extract its important features of it and then decide the weights to give the correct output. These features are learned using filters. Filters help to detect certain image properties such as horizontal lines, vertical lines, edges, corners, etc. As we go deep into the network, the network learns to defect complex features such as objects, face, background, foreground, etc.

Methodology:

The Fashion-MNIST dataset was developed as a response to the wide use of the MNIST dataset, that has been effectively "solved" given the use of modern convolutional neural networks.Fashion-MNIST was proposed to be a replacement for MNIST, and although it has not been solved, it is possible to routinely achieve error rates of 10% or less. Like MNIST, it can be a useful starting point for developing and practicing a methodology for solving image classification using convolutional neural networks.

Instead of reviewing the literature on well-performing models on the dataset, we can develop a new model from scratch.The dataset already has a well-defined train and test dataset that we can use.

In order to estimate the performance of a model for a given training run, we can further split the training set into a train and validation dataset. Performance on the train and validation dataset over each run can then be plotted to provide learning curves and insight into how well a model is learning the problem.The Keras API supports this by specifying the "validation_data" argument to the model.fit() function when training the model, that will, in turn, return an object that describes model performance for the chosen loss and metrics on each training epoch. In order to estimate the performance of a model on the problem in general, we can use k-fold cross-validation, perhaps 5-fold cross-validation. This will give some account of the model's variance with both respect to differences in the training and test datasets and the stochastic nature of the learning algorithm. The performance of a model can be taken as the mean performance across k-folds, given with the standard deviation, that could be used to estimate a confidence interval if desired.We can use the KFold class from the scikit-learn API to implement the k-fold cross-validation evaluation of a given neural network model. There are many ways to achieve this, although we can choose a flexible approach where the KFold is only used to specify the row indexes used for each split.We will hold back the actual test dataset and use it as an evaluation of our final model.

Code :

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import tensorflow as tf
import keras

(X_train, y_train), (X_test, y_test)=tf.keras.datasets.fashion_mnist.load_data()
X_train.shape,y_train.shape, "**********" , X_test.shape,y_test.shape

X_train[0]

y_train[0]

class_labels = [  "T-
shirt/top", "Trouser", "Pullover", "Dress", "Coat", "Sandal", "Shirt", "Sneaker",  "Bag",  "Ankle
boot"]

class_labels

plt.imshow(X_train[0],cmap='Greys')

plt.figure(figsize=(16,16))

j=1
for  i in np.random.randint(0,1000,25):
  plt.subplot(5,5,j);j+=1
  plt.imshow(X_train[i],cmap='Greys')
  plt.axis('off')
  plt.title('{} / {}'.format(class_labels[y_train[i]],y_train[i]))

X_train.ndim

X_train = np.expand_dims(X_train,-1)

X_train.ndim

X_test=np.expand_dims(X_test,-1)

X_train = X_train/255

X_test= X_test/255

from sklearn.model_selection import  train_test_split
```

```
X_train,X_Validation,y_train,y_Validation=train_test_split(X_train,y_train,test_size=0.2,rando
m_state=2020)

X_train.shape,X_Validation.shape,y_train.shape,y_Validation.shape

model=keras.models.Sequential([
                keras.layers.Conv2D(filters=32,kernel_size=3,strides=(1,1),padding='valid',activ
ation='relu',input_shape=[28,28,1]),
                keras.layers.MaxPooling2D(pool_size=(2,2)),
                keras.layers.Flatten(),
                keras.layers.Dense(units=128,activation='relu'),
                keras.layers.Dense(units=10,activation='softmax')
])

model.summary()

model.summary()

model.fit(X_train,y_train,epochs=10,batch_size=512,verbose=1,validation_data=(X_Validation,
y_Validation))

y_pred = model.predict(X_test)
y_pred.round(2)

y_test

model.evaluate(X_test, y_test)

plt.figure(figsize=(16,16))

j=1
for i in np.random.randint(0, 1000,25):
  plt.subplot(5,5, j); j+=1
  plt.imshow(X_test[i].reshape(28,28), cmap = 'Greys')
  plt.title('Actual = {} / {} \nPredicted = {} / {}'.format(class_labels[y_test[i]], y_test[i], class_la
bels[np.argmax(y_pred[i])],np.argmax(y_pred[i])))
  plt.axis('off')

plt.figure(figsize=(16,30))

j=1
for i in np.random.randint(0, 1000,60):
  plt.subplot(10,6, j); j+=1
  plt.imshow(X_test[i].reshape(28,28), cmap = 'Greys')
```

```
  plt.title('Actual = {} / {} \nPredicted = {} / {}'.format(class_labels[y_test[i]], y_test[i], class_la
bels[np.argmax(y_pred[i])],np.argmax(y_pred[i])))
  plt.axis('off')

"""## Confusion Matrix"""
from sklearn.metrics import confusion_matrix
plt.figure(figsize=(16,9))
y_pred_labels = [ np.argmax(label) for label in y_pred ]
cm = confusion_matrix(y_test, y_pred_labels)
sns.heatmap(cm, annot=True, fmt='d',xticklabels=class_labels, yticklabels=class_labels)

from sklearn.metrics import classification_report
cr= classification_report(y_test, y_pred_labels, target_names=class_labels)
print(cr)

"""# Save Model"""
model.save('fashion_mnist_cnn_model.h5')
#Building CNN model
cnn_model2 = keras.models.Sequential([
                    keras.layers.Conv2D(filters=32, kernel_size=3, strides=(1,1), padding='valid',acti
vation= 'relu', input_shape=[28,28,1]),
                    keras.layers.MaxPooling2D(pool_size=(2,2)),
                    keras.layers.Conv2D(filters=64, kernel_size=3, strides=(2,2), padding='same', act
ivation='relu'),
                    keras.layers.MaxPooling2D(pool_size=(2,2)),
                    keras.layers.Flatten(),
                    keras.layers.Dense(units=128, activation='relu'),
                    keras.layers.Dropout(0.25),
                    keras.layers.Dense(units=256, activation='relu'),
                    keras.layers.Dropout(0.25),
                    keras.layers.Dense(units=128, activation='relu'),
                    keras.layers.Dense(units=10, activation='softmax')
                    ])

# complie the model
cnn_model2.compile(optimizer='adam', loss= 'sparse_categorical_crossentropy', metrics=['accur
acy'])

#Train the Model
cnn_model2.fit(X_train, y_train, epochs=20, batch_size=512, verbose=1, validation_data=(X_V
alidation, y_Validation))

cnn_model2.save('fashion_mnist_cnn_model2.h5')

"""######## very complex model"""
```

```
#Building CNN model
cnn_model3 = keras.models.Sequential([
                    keras.layers.Conv2D(filters=64, kernel_size=3, strides=(1,1), padding='valid',acti
vation= 'relu', input_shape=[28,28,1]),
                    keras.layers.MaxPooling2D(pool_size=(2,2)),
                    keras.layers.Conv2D(filters=128, kernel_size=3, strides=(2,2), padding='same', a
ctivation='relu'),
                    keras.layers.MaxPooling2D(pool_size=(2,2)),
                    keras.layers.Conv2D(filters=64, kernel_size=3, strides=(2,2), padding='same', act
ivation='relu'),
                    keras.layers.MaxPooling2D(pool_size=(2,2)),
                    keras.layers.Flatten(),
                    keras.layers.Dense(units=128, activation='relu'),
                    keras.layers.Dropout(0.25),
                    keras.layers.Dense(units=256, activation='relu'),
                    keras.layers.Dropout(0.5),
                    keras.layers.Dense(units=256, activation='relu'),
                    keras.layers.Dropout(0.25),
                    keras.layers.Dense(units=128, activation='relu'),
                    keras.layers.Dropout(0.10),
                    keras.layers.Dense(units=10, activation='softmax')
                    ])

# complie the model
cnn_model3.compile(optimizer='adam', loss= 'sparse_categorical_crossentropy', metrics=['accur
acy'])

#Train the Model
cnn_model3.fit(X_train, y_train, epochs=50, batch_size=512, verbose=1, validation_data=(X_V
alidation, y_Validation))

cnn_model3.save('fashion_mnist_cnn_model3.h5')

cnn_model3.evaluate(X_test, y_test)
```

Project link :
https://colab.research.google.com/drive/1j9m9EWh7SWm4Rm7nNc81M8gB0AuLxEcB?usp=sharing

Conclusion :
We have successfully implemented the project of Fashion MNIST Data Classification using deep learning.