IMAGE CLASSIFICATION-FASHION MNIST

SRUJANA GOLCONDA **JAYAVANI AKKIRAJU**

sgolcond@buffalo.edu **jayavani@buffalo.edu**

50442205 **50442863**

Abstract

As one of the most prominent fields of today, machine learning and deep learning are rapidly improving many aspects of our lives. Classification of images is at the core of computer vision, and it plays a pioneering role in many other fields as well. As the online fashion market continues to grow, clothing sales companies can use algorithms to identify garments and gain an understanding of potential customers' profiles, focus on sales targeting specific niches, create campaigns tailored to the customer's needs, and enhance the customer experience. Fashion MNIST dataset contains thousands of images and labels that help researchers to find the best classification algorithms to achieve the required goal. The objective of this project is to compare the results of some of the state-of-the-art algorithms and find the best algorithms for image classification.

1 Introduction:

The Fashion-MNIST dataset is a collection of Zalando's article images. It contains 60,000 training examples and 10,000 test examples. There are 10 classes represented by 28x28 grayscale images. For benchmarking machine learning algorithms, Zalando intends to replace MNIST with Fashion-MNIST. In both training and testing, the size and structure of the images are the same.

1.1 License

The MIT License (MIT) Copyright © [2017] Zalando SE, https://tech.zalando.com. Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions: The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software. THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

1.2 Dataset Description

Each image has 28 pixels in height and 28 pixels in width, totaling 784 pixels. Using pixel values, each pixel is assigned a value indicating its lightness or darkness, with higher numbers indicating a darker pixel. Pixel values range from 0 to 255. There are 785 columns in the training and test data sets. Clothing articles are represented in the first column by class labels. Other columns contain the associated image's pixel values. Sample images of the Fashion MNIST.

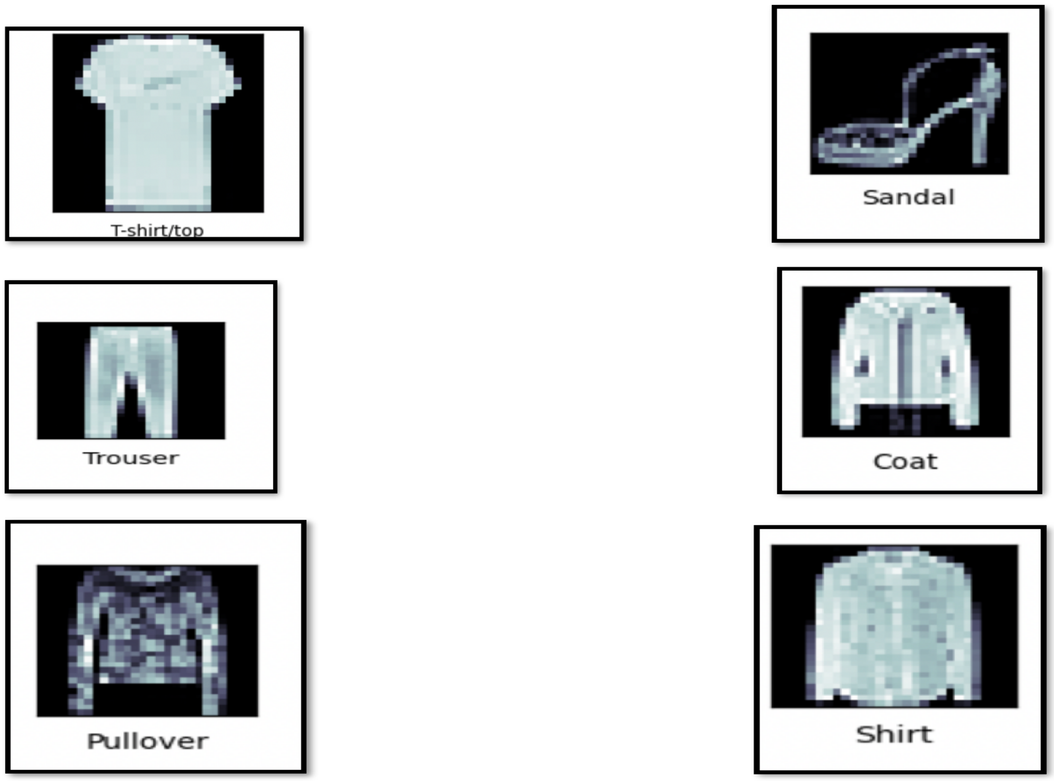


Figure 1. Sample images of Fashion MNIST

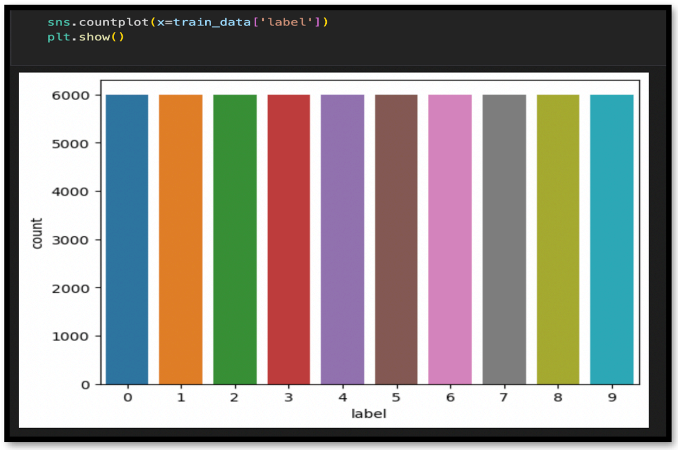
2. Data Preprocessing

We initially split the training and test data by their labels and predictors. We check for the null values. As there are no null values in the data we proceed with the data as is. Reshaping the data to 28*28 matrix of pixels. Normalization Pixel values range from 0 to 256, apart from 0, the range is 255. By dividing all the values by 255, we get a range from 0 to 1. A class vector (integers) is converted to a binary class matrix. The labels of train and test data are converted to a binary class matrix. The label range is 0-9 as there are 10 labels in the given data. Once the preprocessing is done, we proceed with the model fitting on the train data.

3 Data Visualization

3.1 Count plot

Count plot to show the counts of various images in the training data



```
sns.countplot(x=train_data['label'])
plt.show()
```

Figure 4. Count plot of all images in x_train

4 State-of-the-art Algorithms

4.1 Pca and Knn

Steps for PCA

Initially, we imported train and test data. Calculating variance from standard deviation along the column of the train data of predictors. Reshaping train and test data using variance. Converting train and test label data into binary class. Fitting the PCA model on to the train data to reduce the dimensionality of the data. Retrieve cumulative sum.Fitting KNN along with PCA in the end.

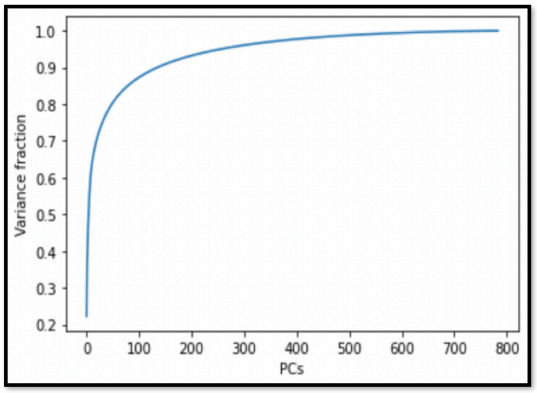


Figure 7. Plot comparing variance ratio and principal components

KNN

After building the model with PCA KNN with 5 nearest neighbors and 50 components the accuracy on test data was found to be 83.19%

4.2 Neural Networks

In neural networks, a computer learns to perform some tasks by analyzing training examples. When a neural network is being trained, all its weights and thresholds are initially set to random values. A training dataset is fed into the bottom layer - the input layer - and passes through successive layers, being multiplied and added in complex ways, until it eventually arrives, transformed, at the output layer. Weights and thresholds are continuously adjusted during training until training data with the same labels consistently produce similar results.

Model 1

First Dense Layer: 64 neurons/nodes

Second Dense Layer (Output): returns a logit array with length of 10.

As there are 10 labels (0-9) output is 10 here in the second dense layer.

Activation: relu

Optimizer: Adam

Loss function: Sparse Categorical Crossentropy

Epochs = 15

Test Accuracy: 87.8%

4.3 Convolution Neural Networks

Base Model

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_3 (Conv2D)            (None, 26, 26, 32)        320
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 24, 24, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 12, 12, 64)        0
_________________________________________________________________
batch_normalization_3 (Batch (None, 12, 12, 64)        256
_________________________________________________________________
dropout_2 (Dropout)          (None, 12, 12, 64)        0
_________________________________________________________________
```

```
batch_normalization_4 (Batch (None, 12, 12, 64)       256
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 10, 10, 128)      73856
_________________________________________________________________
dropout_3 (Dropout)          (None, 10, 10, 128)      0
_________________________________________________________________
flatten_1 (Flatten)          (None, 12800)            0
_________________________________________________________________
dense_2 (Dense)              (None, 256)              3277056
_________________________________________________________________
dense_3 (Dense)              (None, 10)               2570
=================================================================
Total params: 3,372,810
Trainable params: 3,372,554
Non-trainable params: 256
```

Accuracy – 90.73%

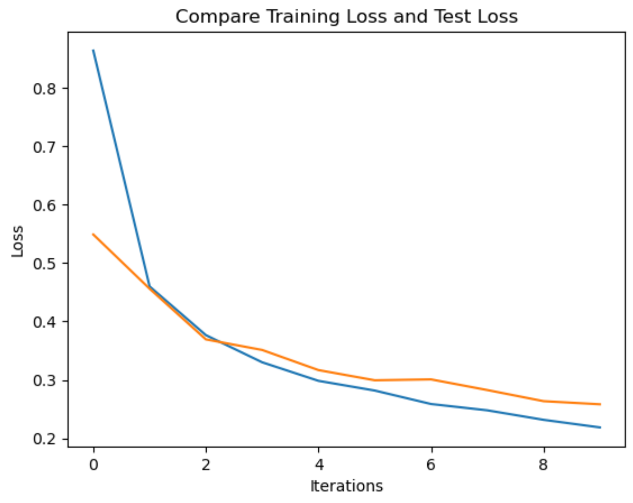


Figure 8. Graph comparing training loss and testing loss

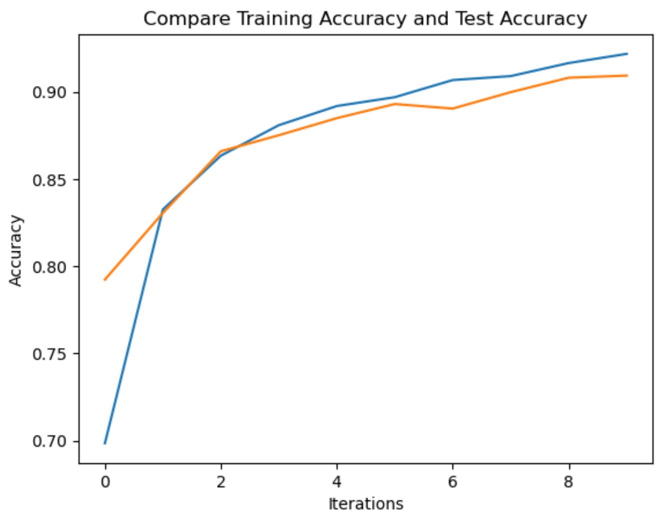


Figure 8. Graph comparing training accuracy and testing accuracy

Hyper Tuning of CNN Model

Activation	Loss	optimizer	dropout	Test Accuracy
Relu,Softmax	Categorial Crossentropy	adam	0.04,0.02	54.26% early stopping
relu,softmax	Categorial Crossentropy	adam	0.25	86.9%
Relu,softmax	Categorial Crossentropy	adam	0.01	91.89%
relu,relu,softmax	Categorial Crossentropy	nadam	0.05	90.79%
Relu,relu,softmax	Categorial Crossentropy	adadelta	0.02,0.04	48.72%

After hyper tuning the cnn model, it is observed that the model with 0.01% dropout rate gave the highest accuracy on test data.

4.4 AlexNet

Eight layers make up the architecture: five convolutional layers and three fully connected layers. The AlexNet model is extremely powerful and capable of achieving high accuracy on challenging datasets. AlexNet's performance will drastically suffer if any of the convolutional layers are removed. AlexNet is a leading architecture for any object-detection task.

The model uses Optimizer ‘adam’ and activation as ‘relu’. The dropout applied is 0.5 and the poolsize is 3 for MaxPooling. The accuracy on test data that was observed: 80.31%.

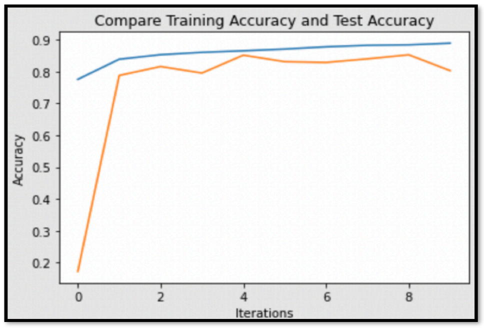


Figure 9. Training vs Test Accuracy of AlexNet Model

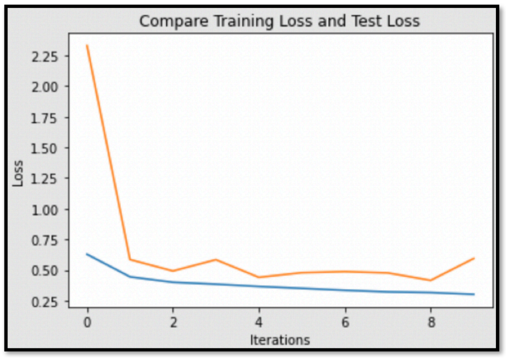


Figure 10. Training vs Test Loss of AlexNet model

4.5 LeNet

On a high level, LeNet (LeNet-5) is composed of two components: A convolutional encoder with two convolutional layers and a dense block with three fully connected layers. The units in every convolutional block are a convolutional layer, a sigmoid activation function, and an average pooling operation. A 5×5 kernel and a sigmoid activation function are used by each convolutional layer.

LeNet Model:

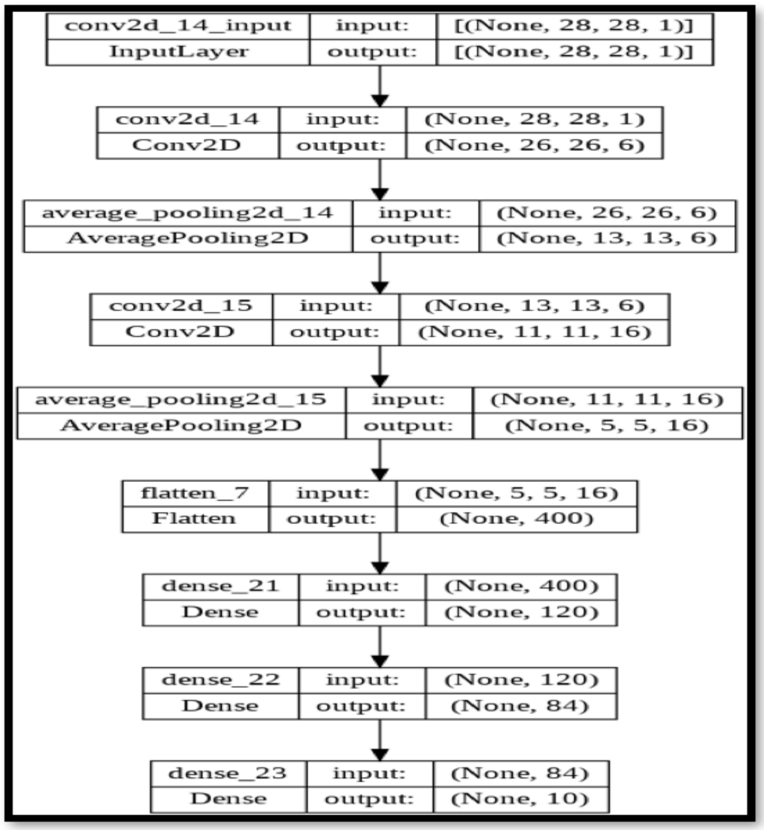

conv2d_14_input | input: | [(None, 28, 28, 1)]
InputLayer | output: | [(None, 28, 28, 1)]
conv2d_14 | input: | (None, 28, 28, 1)
Conv2D | output: | (None, 26, 26, 6)
average_pooling2d_14 | input: | (None, 26, 26, 6)
AveragePooling2D | output: | (None, 13, 13, 6)
conv2d_15 | input: | (None, 13, 13, 6)
Conv2D | output: | (None, 11, 11, 16)
average_pooling2d_15 | input: | (None, 11, 11, 16)
AveragePooling2D | output: | (None, 5, 5, 16)
flatten_7 | input: | (None, 5, 5, 16)
Flatten | output: | (None, 400)
dense_21 | input: | (None, 400)
Dense | output: | (None, 120)
dense_22 | input: | (None, 120)
Dense | output: | (None, 84)
dense_23 | input: | (None, 84)
Dense | output: | (None, 10)

Figure 11. LeNet Architecture

Model: Activation = 'relu' Activation = 'softmax' for Dense Output Layer Optimizer = 'adam' loss='categorical_crossentropy' Epoch = 10 Batch size = 1000 Accuracy: The accuracy of test data is 74.82%.

4.6 VGG 16

VGG-16(Visual Geometry Group) is a convolutional neural network that is 16 layers deep. VGG16 would be able to classify images of different types. It is a huge network.

Model:Activation = 'relu'Activation = 'softmax' for Dense Output layer

Optimizer = 'adam'

loss = 'categorical_crossentropy'

Epochs = 10

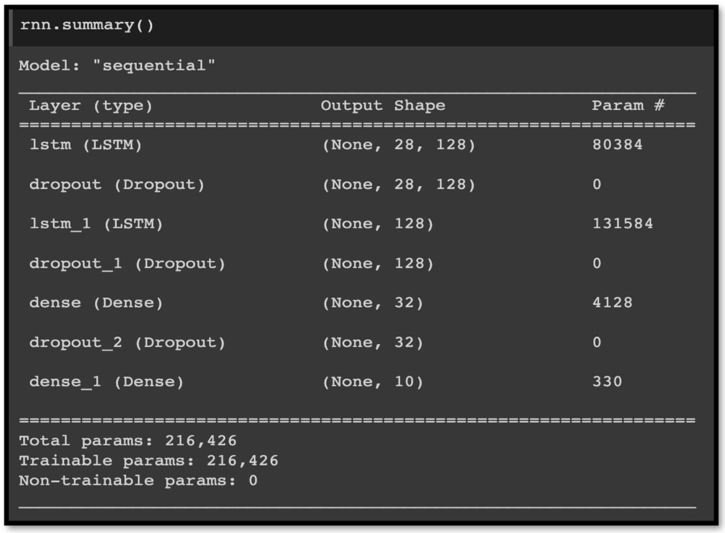
Batch_size = 256

min_lr = 0.00001

A test accuracy of 91.69% was obtained using VGG-16 as the model on the dataset.

4.7 RNN

A recurrent neural network (RNN) is an artificial neural network using sequential data or time series data. Recurrent neural networks learn from training data. As they can use information from prior inputs to influence the current input and output, they are distinguished by their "memory." Recurrent neural networks produce outputs based on previous elements within the sequence, unlike traditional deep neural networks, which assume inputs and outputs are independent of each other.



```
rnn.summary()

Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 28, 128)           80384

 dropout (Dropout)           (None, 28, 128)           0

 lstm_1 (LSTM)               (None, 128)               131584

 dropout_1 (Dropout)         (None, 128)               0

 dense (Dense)               (None, 32)                4128

 dropout_2 (Dropout)         (None, 32)                0

 dense_1 (Dense)             (None, 10)                330

=================================================================
Total params: 216,426
Trainable params: 216,426
Non-trainable params: 0
_________________________________________________________________
```

Layers:

LSTM (128); Activation = 'relu'

Dropout (0.2)

LSTM (128); Activation = 'relu'

Dropout (0.2)

Dense (32); Activation = 'relu'

Dropout (0.2)

Dense (10);

Activation = 'softmax'

Optimizer = 'Adam'

lr=1e-3, decay=1e-5

Test Accuracy – 90.07%

5. Comparison of the accuracy of the models

Model	Test Accuracy
CNN	91.89
AlexNet	80.31
LeNet	78.05
PCA + KNN	83.19
Neural Network	89.66
VGG16	91.69
RNN	90.07

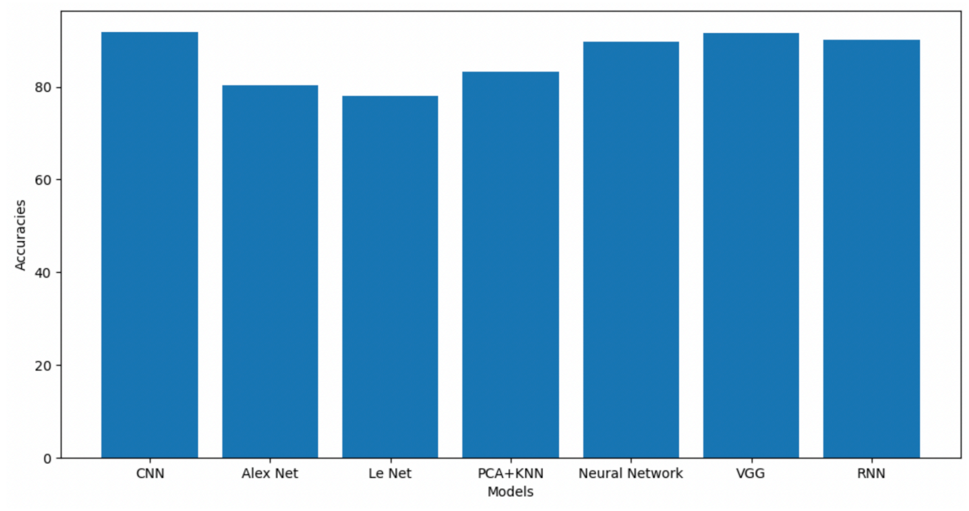


Figure 9. Graph comparing accuracies of all the models

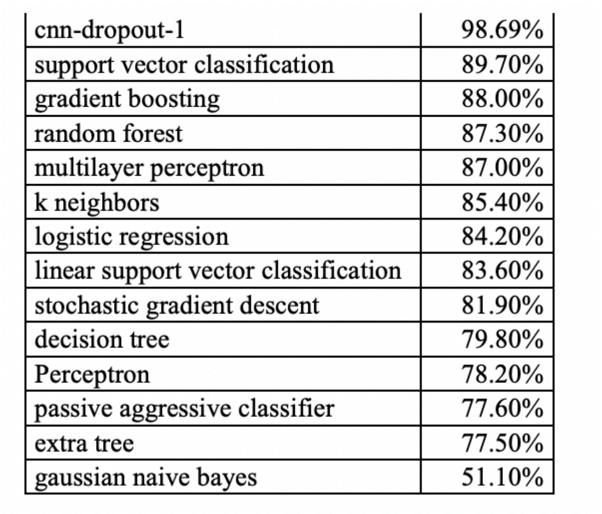
It is observed that CNN is the best model for the Fashion MNIST dataset.

6. Comparison with existing research papers

1. Since we are using existing dataset, the accuracy of our models can be compared with the models run in

[Kayed, Mohammed & Anter, Ahmed & Mohamed, Hadeer. (2020). Classification of Garments from Fashion MNIST Dataset Using CNN LeNet-5 Architecture. 238-243.10.1109/ ITCE48509.2020.9047776

Research Paper 1]



cnn-dropout-1	98.69%
support vector classification	89.70%
gradient boosting	88.00%
random forest	87.30%
multilayer perceptron	87.00%
k neighbors	85.40%
logistic regression	84.20%
linear support vector classification	83.60%
stochastic gradient descent	81.90%
decision tree	79.80%
Perceptron	78.20%
passive aggressive classifier	77.60%
extra tree	77.50%
gaussian naive bayes	51.10%

It is observed that the models which have applied have given better accuracy than the above models, with CNN being the best model.

2. If we compare with the LeNet 1

[Lecun Y, Bottou L, Bengio Y, et al. Gradient-based learning applied to document recognition [J]. Proceedings of the IEEE, 1998, 86(11):2278-2324.]. The accuracy in the paper is 75.3% and we were able to get an accuracy of 78.05% for our LeNet model.

3. If we compare with the VGG 11 model as in

[Smirnov E, Timoshenko D, Andrianov S, et al. Comparison of Regularization Methods for ImageNet Classification with Deep Convolutional Neural Networks [J]. AASRI Procedia, 2014:89-94.] got an accuracy of 90.5%

Our model which is VGG 16 was able to get an accuracy of 91.69%

7. Conclusion

By observing the obtained results, we can deduce that classification of fashion mnist images by convolution neural networks can be more accurate when compared to other conventional algorithms in our case. This is based on the test accuracies. The training accuracies can account to nearly 95%, which states that the data fitting is good enough.

The Jupyter Notebook with all the executions are present at https://drive.google.com/file/d/1HgiI0H1FAXxa1Glq_hMkjLlFf7aZJttW/view?usp=share_link

References:

https://keras.io/guides/sequential_model/

https://numpy.org/

https://pandas.pydata.org/docs/

https://matplotlib.org/stable/tutorials/introductory/pyplot.html

https://seaborn.pydata.org/

https://keras.io/api/datasets/fashion_mnist/

https://www.tensorflow.org/tutorials/images/cnn

https://en.wikipedia.org/wiki/AlexNet

https://www.tensorflow.org/api_docs/python/tf/keras/utils/plot_model

https://www.tensorflow.org/api_docs/python/tf/keras/utils/to_categorical

https://www.tensorflow.org/

https://www.tensorflow.org/api_docs/python/tf/keras/layers

https://www.tensorflow.org/tutorials/images/cnn

https://www.tensorflow.org/tutorials/keras/classification https://aws.amazon.com/what-is/neural-network/

https://www.ibm.com/topics/knn

https://towardsdatascience.com/step-by-step-vgg16-implementation-in-keras-for-beginners-a833c686ae6c

https://www.kaggle.com/code/guilhermesdas/fashion-mnist-vgg16

https://royalsocietypublishing.org/doi/10.1098/rsta.2015.0202#:~:text=Principal%20component%20analysis%20(PCA)%20is,same%20time%20minimizing%20information%20loss.

https://github.com/zalandoresearch/fashion-mnist#benchmark

https://notebook.community/bkimmig/bkimmig.github.io/blog/fashion-mnist-embedding/index

https://www.kaggle.com/code/siumisandu/fashion-mnist-cnn-rnn

https://stackoverflow.com/questions/66785014/how-to-plot-the-accuracy-and-and-loss-from-this-keras-cnn-model

https://builtin.com/data-science/recurrent-neural-networks-and-lstm