INDIAN JOURNAL OF SCIENCE AND TECHNOLOGY

RESEARCH ARTICLE

Implementation of CNN and ANN for Fashion-MNIST-Dataset using Different Optimizers

Sumera[1]*, R Sirisha[2], Nadia Anjum[3], K Vaidehi[4]

1 Assistant Professor, Department of Computer Engineering, Stanley College of Engineering and Technology for Women, India
2 Assistant Professor, Department of Artificial Intelligence & Data Science, Stanley College of Engineering and Technology for Women, India
3 Assistant Professor, Department of Artificial Intelligence & Data Science, Stanley College of Engineering and Technology for Women, India
4 Professor, Department of Computer Engineering, Stanley College of Engineering and Technology for Women, India

OPEN ACCESS

Received: 09-09-2022
Accepted: 14-11-2022
Published: 21-12-2022

Citation: Sumera , Sirisha R, Anjum N, Vaidehi K (2022) Implementation of CNN and ANN for Fashion-MNIST-Dataset using Different Optimizers. Indian Journal of Science and Technology 15(47): 2639-2645. https://doi.org/10.17485/IJST/v15i47.1821

* **Corresponding author**.

mohdd.ubaidd@gmail.com

Funding: None

Competing Interests: None

Published By Indian Society for Education and Environment (iSee)

ISSN
Print: 0974-6846
Electronic: 0974-5645

Abstract

Objectives: The paper presents application of convolution neural network and artificial neural network for image classification problem for clothing dataset along with their performance comparison against different optimizers. The major objective of this paper is to perform image classification on fashion-mnist clothing dataset images. **Methods:** The methods used here are, the traditional ANN and CNN. Here image classification is performed on Fashion-mnist, clothing dataset using CNN and ANN with different optimizers. The performance of the working of ANN and CNN in classifying images from fashion-mnist dataset is compared against different optimizers namely stochastic gradient Descent, Adagrad, RMS prop and Adam optimizer. **Findings:** The study found that CNN worked better than ANN yielding training accuracy of 95%, 93% and testing accuracy of 91%, 89% when used with Adam and RmsProp respectively. **Novelty:** The novelty of this work is to present a comparative study of image classification using CNN, ANN using different optimizers, since not many studies or research articles showed the performance comparison of traditional and convolution neural networks in image classification along with different optimizers. Since the real-world scenarios of today require enormous data to be processed, CNN can fit well to diversify applications since they highly reduce the number of parameters to be trained that speeds up the training process. Moreover, to be specific on image classification problems they require the best and most prominent features to be detected and uncovered; this can be achieved using CNN since it has the concept of convolution using filters at its Core. Hence, CNN is highly recommended for such image classification applications than the traditional artificial-neural-networks because of the aforementioned reasons.

Keywords: ANN (Artificial NeuralNetworks); CNN (ConvolutionNeural Networks); Optimization; SGD (Stochastic Gradient Descent); RmsProp (Root mean Square propagation); Adam (Adaptive moment estimation); AdaGrad (Adaptive Gradient)

1 Introduction

Over the past few years neural networks have supported diverse tasks in computer vision, medical-diagnosis etc. Neural networks are designed to handle variety in the input data such that it can classify that variety in a generic way. Artificial neural networks do not have a concept of filters, pooling unlike CNN. The number of parameters that are supposed to be trained and altered in back propagation to reduce the cost function are very large in number. This goes beyond the memory of our normal system as it slows down training the model. Secondly, training too much of neurons and more number of parameters also means overfitting, thus it can also affect the performance of our model. Another benefit of CNN is that they can capture or are able to learn relevant features from an image at different levels (since we use filters) similar to human brain/ intelligence, a concept called feature learning. These details are explained by implementing image classification on a clothing dataset, Fashion-mnist. In [(1)], presents a proposed system for the classification of images using the power of CNN. The dataset used was CIFAR-10 with Matconvnet used as a platform to implement CNN for classification. Here the batch size for training the model was chosen as 60, numerous epochs around 300, learning rate of 0.0001 in order to attain an accuracy of 93%. The current work tried to classify the images considering different optimizers in which Adam produced highest accuracy. In [(2)], refers a system in which animal species-fauna dataset is considered for image classification using CNN by using the concepts of transfer learning VGG16, ReLu obtaining an accuracy of about 91% on training data. In [(3)], presents a work wherein authors considered plant digital images to predict pigments of photosynthesis to show the performance of different optimizers on different CNN architectures. Among all the considered optimizers, Adam worked the best, producing the least Mean-squared-error. In [(4)], introduces a work in which the author did a comparative study on different optimizers used to reduce the overall loss in CNN for 4 different datasets namely- MNIST, kaggle-flower, CIFAR–10 and labeled faces. It was observed that different optimizers outperformed for different datasets. In [(5)], the author considered MNIST and CIFAR datasets to solve the image recognition problem using various optimizers wherein the NAG, Adadelta outperformed the others. In [(6)], proposed here performed CNN classification using Lenet-5 architecture that resulted an overall accuracy of 98%. In [(7)], outlines the study on classifying X_ray-images using a machine learning-approach named SVM using various image enhancement methods In [(8)], presented a similar work in which the image classification was performed on Fashion-Mnist dataset. A good work in which they showed how the use of filters, optimizers, activation functions can affect the correctness of the results. In [(9)], presents a similar work on Fashion-Mnist dataset in classifying images with different architectures of CNN considering different numbers of layers, filter size, hyper parameters etc.

Since there only a few studies done on how image classification is performed by traditional ANN and how today's CNN outperforms it, this proposed study aims at showing these two kinds of Neural Networks. The study proposed here performs image classification on Fashion-MNIST dataset using ANN, CNN against different optimizers namely SGD, Adagrad, RmsProp and Adam.

2 Methodology

2.1 Proposed system

Images from the Fashion-mnist dataset are taken in which 60,000 are considered as training samples and 10,000 as test samples. An artificial Neural Network model is created wherein input layer is formed by flattening images of size 28*28*1 into 1-D vector followed by making up the other layers of ANN, ReLu as activation function and Softmax in the last layer is used since we are performing multiclass classification. Similarly, CNN is designed with two convolution layers of 32 and 64 filters respectively, ReLu and Softmax activation functions are used and different optimizers are considered to check the performance. Dataset considered, architecture along with trainable parameters are presented clearly in the next sections accordingly.

2.1.1 Dataset

The fashion MNIST dataset of clothing article images is the most easily available and a convenient way to consider and work with. Considering this as a base dataset for training/building the models for predictions (using ANN and CNN in this study), makes it much easier to implement and understand the diversified concepts of classification and prediction algorithms. Fashion MNIST dataset consists of 60000 training set examples and a test set of 10000 examples where each sample is a 28 x 28 grayscale image associated with a label of 10 classes. There are in total 785 columns the very first column being the class label to represent the article of clothing. In the study proposed here, two architectures are built for the image classification one using ANN and the other using CNN and their performance is assessed against different optimizers for image classification.

2.1.2 Artificial-Neural-Network

In the work proposed here, ANN built is a simple sequential model consisting of an input layer and three dense layers in which the last dense layer is the output layer. For the neurons to make up the input layer the input images (of size 28 x 28 x 1), are flattened into one dimensional vectors. The second i.e; first dense or hidden layer is made up of 3,000 neurons and the activation function used in this layer is Relu to introduce non linearity in the model. The third layer comprises 1,000 neurons with Relu as its activation function considered here. The final or output layer consists of a neurons, along with the softmax activation function also called as categorical cross entropy used for multiclass classification problems in the output layer.

2.1.3 Convolution-Neural-Network

In contrast to the ANN, CNN here is a sequential model consisting of a stack of layers where a lot of computation is done at each layer to figure out the most prominent features as we move deep into the network. The input images of size 28 x 28 x 1 are convolved by applying the 32 filters/ Kernel of size 3 x3. The activation function used here is relu. The max pooling operation is done by considering a kernel size of 2 x 2. This forms the first convolution layer. The second convolution layer consists of 64 filters of size 3 X 3 with relu as the activation function and a maxpool of 2 x 2. After having two consecutive convolutional layers, the next Layer is the first fully connected layer with 64 neurons. The feature map from the previous convolutional layers is flattened and connected to the 64 neurons here. The activation function used here is relu.The output layer is simply a dense network of 10 neurons with softmax as the activation function used. These models are compiled and trained for classification against different optimizers like SGD, Adagrad, RMS-prop and Adam to reduce the overall loss and make the models better at their predictions.

2.2 Discussions on optimizers used in this study

2.2.1 Stochastic gradient descent

A graphic representation of the normal Gradient Descent is presented in Figure 1, wherein it shows how the overall cost of the model increases/decreases upon altering weights. SGD is similar to a mini batch gradient Descent but here the batch is equal to 1. Instead of dividing the data set in batches, here we pass one data point at once, train the model update the weights then again pass another data point and so on until entire data set is passed. It typically reaches convergence much faster since it updates/manipulates fewer data (weights) at a single time than a normal gradient descent. SGD will not follow a normal path to the local minima but will be converging with some noise (zigzagness observed in the graph) due to these frequent updates.) (since we are considering one training example, the cost will fluctuate/spin around the training examples and will not necessarily decrease but eventually this cost decreasing may lead to a local minima, not exactly but fluctuating around). The general equation of gradient descent for weight updations is given by the formula-

$W = W - \alpha^* (\partial \text{cost}) / \partial W)$ (here ∂ cost)$/\partial W$ is the change in the cost/error with respect to the weights (old) $B = B - \alpha^* (\partial \cos f_0 \cdot t) / \partial B$

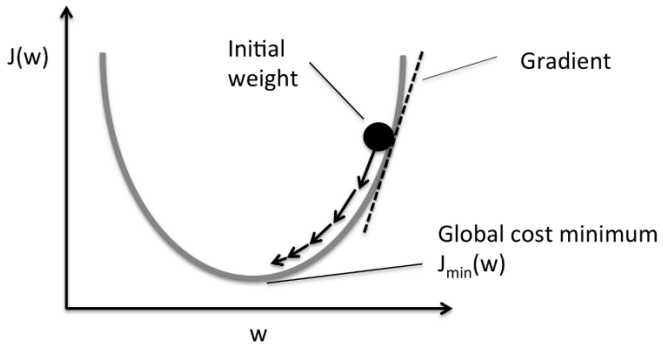

Fig 1. Convergence of Gradient Descent

Since there is a lot of noise in SGD while converging, smoothening happens using exponentially weighted moving averages. As we move forward in time, we keep on encountering new data points. In Exponential weighted moving average. The average is calculated at each step. As we encounter new points, it is calculated in such a way that we give higher weightage to the newer points, while the lower weightage to the older points. The exponentially weighted moving average is calculated with the help of this equation Vt=β*V(t-1) +(1-β)*θt with average at any time stamp T Vt is calculated by multiplying this beta hyper parameter with the previous average and 1 - beta with the current data point. So will calculate different v's at different timestamps based on the above concept. Final plot will be that graph..Red line that acts as an approximate average.

Hyper parameter beta value is also considered between 0 and 1. Mostly considered as 0.9. At any time stamp t, Vt= (1-β) *[θt+$\beta^{1*}\theta$t-1+$\beta^{2*}\theta$t-2+$\beta^{3*}\theta$t-3+$\cdots$+t-1]*θJ+β^t*θ0]. We will use this to implement momentum, We know that the weight updation in gradient descent is given by this equation.W=W-α*(∂cost)/∂W, B=B-α*(∂cost)/∂B (here ∂cost)/∂W is the change in the cost/error with respect to the weights(old). Instead of this del cost/delW and delcost/delB, we will replace it with vdw and vdb. W=W-α*vdW, b=b-α*vdb, where vdW=β*vdWprev+ (1-β) dW (here dW is (∂cost)/∂W), vdb=β*vdbprev+ (1-β) db.Vdw and Vdb be are nothing but the exponentially moving weighted average. Now as we are taking the exponentially weighted moving average of these points, the average of these points in the vertical direction will be approximately close to zero only while the average in the horizontal direction will be higher (consider the counter plot for the corresponding 3-D graph for 2 parameters (weights), say W and b in the vertical and horizontal directions respectively) shown in Fig.6. Thus, the net result will be mostly in the horizontal direction while very little in the vertical direction thus in this way Momentum will increase or speed up the training of our model.

2.2.2 Rms prop

Rms prop is an optimization algorithm which speed-ups the training of our model. Faster than SGD with momentum seen above. (Doubles-since we are using square in the equation here). We know that the weight updation in the gradient descent is given by these two equations;

W=W-α*(∂cost)/∂W, B=B-α*(∂cost)/∂B,

Now for RMSProp, it is

W=W-αdw/ ($\sqrt{}$ (sdw) +ε) (here dw= (∂cost)/∂W)

b=b-αdb/ ($\sqrt{}$ (sdb) +ε)

Where sdw=β*sdwprev+(1-β)(dw)^2,sdb=β*sdbprev+(1-β)*(db)^2.

We are going to have faster training of our model and as we are taking the square of dw and then taking its root this algorithm is called root mean square propagation. The difference for change in gradient Descent and RMS crop is the way of how the gradients for slopes or derivative terms are calculated in each of them. RMS prop restricts or obstructs the oscillations in the vertical direction. Hence learning rates could be made Better by letting our algorithm consider larger or big steps in horizontal direction while reaching the local minimum Quicker as per the counter plot given above.

2.2.3 Adam

If you know about momentum then you will know that the weight updation is given by this equation. ForAdam optimizer we combine both the momentum as well as the rms prop into one single equation.

W=W-α·vdw/($\sqrt{}$(sdw)+ε) (here dw=(∂cost)/∂W)

b=b-α·vdb/($\sqrt{}$(sdb)+ε)
Where vdw=β·vdwprev+(1-β)·dw
vdb=β·vdbprev+(1-β)·db
sdw=β·sdwprev+(1-β)·(dw)^2
sdb=β·sdbprev+(1-β)·(db)^2

2.2.4 Adagrad

Since there is no concept of momentum in undergrad Optimizer it is simpler than stochastic gradient Descent but with a minor drawback. Adagrad has a separate learning rate for each iteration unlike the other optimization algorithms. Observing the concept and equation AdaGard, there is a presence of accumulation of squares of the gradients in the denominator so each and every term that is added is positive. This accumulated sum term might be growing/ increasing during training. This may lead to shrinking of the learning rate constantly after which the algorithm might not be able to learn any further. Therefore, the other algorithms like RMS prop and Adam overcomes this shortcoming of Adagard by using exponentially weighted moving average concept.

SGD⇒wt=w_ (t-1)-η ∂L/ (∂w_ (t-1))

Adagrad⇒w_t=w_ (t-1)-η_t1 ∂L/ (∂w_ (t-1))

Where η_t1=η/$\sqrt{}$ (α_t+ε), ε is a small " +ve" number to avoid divisibility by" 0 α_t=$\sum$_(i=1)^t(∂L/(∂w_(t-1)))^2, summation of gradient square

3 Result and Discussion

Considering both the aforementioned CNN and ANN architectural models for image classification of fashion MNIST dataset, it was observed that CNN worked better than ANN and has yielded an improvised accuracy for both training data and testing data against prominent optimizers (RmsProp, Adam) when compared to ANN. This is probably due to the power of convolution operations for feature consideration in CNN. All the results are tabulated in Table 1.

Table 1. Result comparison for ANN, CNN against different optimizers

Optimizer	**ANN**		**CNN**	
	Training Accuracy (%)	Testing Accuracy (%)	Testing Accuracy (%)	TESTING ACCURACY (%)
SGD	89	87	88	87
ADAGRAD	86	84	81	81
RMSPROP	88	87	93	89
ADAM	91	88	95	91

Diving deep into the individual performance against each considered optimizer, it is observed that for SGD, ANN has given accuracy of 89% on training data and 87% on test data respectively as shown-in Figure whereas CNN for SGD yielded an accuracy of 89% on training data and 81% on test data respectively.

AdaGrad optimizer gives 86% and 84% of training and testing accuracy respectively for ANN and 81% and 81% training and testing accuracy respectively for CNN.

Similarly, using the RMS optimizer, the ANN model for image classification gives an overall accuracy of 88% on the training data and 87% on the test data as shown in figure. CNN outperformed when used along with the same Optimizer giving a training accuracy of 93% and test accuracy of 89% respectively.

Lastly Adam Optimizer worked best among all the other optimizers discussed in this study. For ANN the training accuracy for image classification was about 91% and test accuracy was 88% as in figure Adam along with CNN yielded an accuracy of 95% for training and 91% on test data. Sample classification report for CNN with Adam optimizer is shown in Figure 2.

Hence it was found that although both models work well in classification of images using different Optimizers, comparatively CNN worked better than ANN and in order to reduce the overall error or cost while training the model the Optimizer that outperformed among all is the Adam optimizer. Therefore, CNN can fit well to diversify applications since they highly reduce the number of parameters to be trained that reduces the computational time and speeds up the training process.

A clear comparison of CNN and ANN along with their internal computations of this proposed model (such as number of parameters being trained after each layer, and the corresponding layers' input and output) is expressed in figure and figure. For ANN, as per Figure 3, the input in the first layer is flattened and multiplied with 3000 neurons that forms the first hidden layer

Classification Report:

	precision	recall	f1-score	support
0	0.84	0.88	0.86	1000
1	0.99	0.98	0.99	1000
2	0.85	0.90	0.87	1000
3	0.94	0.90	0.91	1000
4	0.86	0.86	0.86	1000
5	0.99	0.98	0.98	1000
6	0.76	0.73	0.75	1000
7	0.96	0.96	0.96	1000
8	0.98	0.98	0.98	1000
9	0.96	0.98	0.97	1000
accuracy			0.91	10000
macro avg	0.91	0.91	0.91	10000
weighted avg	0.91	0.91	0.91	10000

Fig 2. Classification report of CNN using ADAM

of ANN. The number of parameters being trained here are 2355000 followed by 1000 neurons in the second layer and finally 10 neurons in the output layer(since 10 classes are to be classified here).

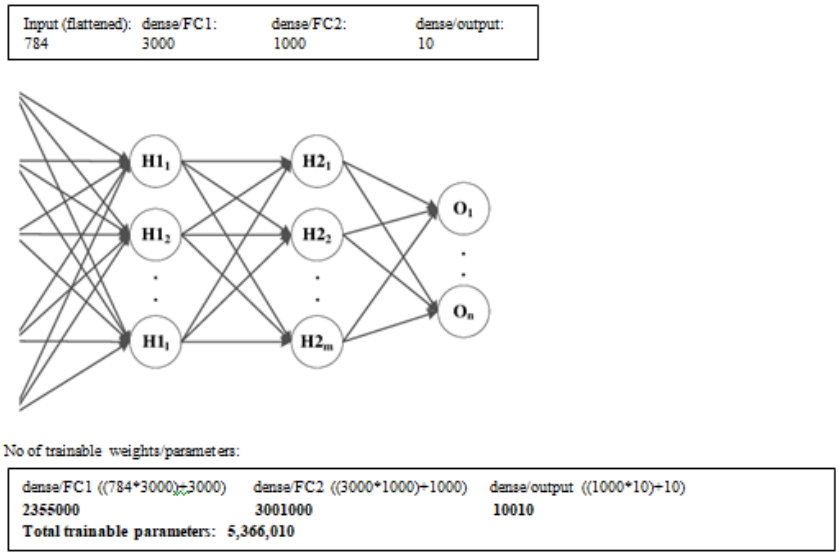

Fig 3. Parameter calculation in ANN

In Figure 4, the CNN takes an input image of size (28*28*1) which is passed through the first convolution/maxpool layer and the number of parameters trained at this step is 320 (considering the filter size, no' of-filters in previous-layer, current layer filters). The output then is passed through the subsequent/second convolution-layer, the no'-of-parameters trained at this layer are 18,496. The first full-connected/dense layer is made up of 1600 neurons-flattened as a single vector and multiplied with the number of neurons present in the previous layer ((current layer neurons*previous-layer neurons) +1*current layer-neurons). Number-of parameters trained after the first and the second fully connected layers is 102464 and 650 respectively.

Moreover, to be specific on image classification problems they require the best and most prominent features to be detected and uncovered, this can be achieved using CNN since it has the concept of convolution using filters at its Core. Hence CNN is highly recommended for such image classification applications than the traditional artificial-neural-networks.

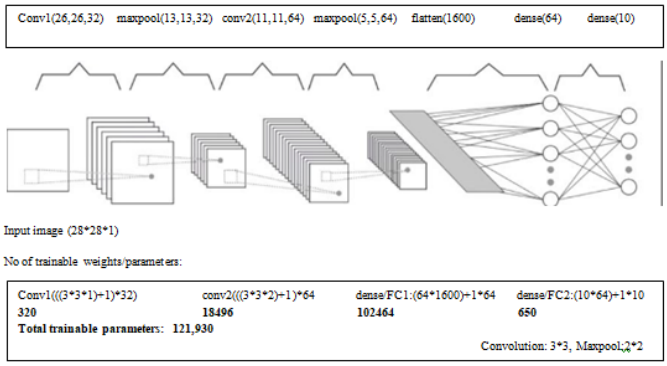

Fig 4. Parameters calculation in CNN

4 Conclusion

After the entire study, it was found that different optimization techniques work differently for ANN and CNN. Overall, Adam optimizer outperformed when used along with CNN architecture yielding maximum accuracy of about 95%. Moreover, another finding observed was the time taken and number of parameters while the architectures are being trained are much less in case of CNN than ANN. Therefore, CNN can be vastly used for diversified computer vision applications because of the power of convolution operation unlike the regular ANN. Because of this power of CNN, the scope of using CNN goes beyond and is not restricted to limited applications. In the future, this work can be extended to classify a lot more classes of textiles/clothing.

References

1) Hossain MA, Sajib MSA. Classification of Image using Convolutional Neural Network (CNN). *Global Journal of Computer Science and Technology*. 2019;19:13–18. Available from: https://computerresearch.org/index.php/computer/article/view/1821.
2) Sanghvi K. Fauna image classification using convolutional neural network. *International Journal of Future Generation Communication and Networking*. 2020;13:8–16. Available from: http://sersc.org/journals/index.php/IJFGCN/article/view/17733/8966.
3) Prilianti KR, Brotosudarmo THP, Anam S, Suryanto A. Performance comparison of the convolutional neural network optimizer for photosynthetic pigments prediction on plant digital image. *AIP Conference Proceedings*. 2019;2084. Available from: https://doi.org/10.1063/1.5094284.
4) Soydaner D. A Comparison of Optimization Algorithms for Deep Learning. *International Journal of Pattern Recognition and Artificial Intelligence*. 2020;34(13):2052013. Available from: https://doi.org/10.1142/S0218001420520138.
5) Agarwal M, Rajak A, Shrivastava AK. Assessment of optimizers impact on image recognition with convolutional neural network to adversarial datasets. *Journal of Physics: Conference Series*. 2021;1998(1). Available from: http://dx.doi.org/10.1088/1742-6596/1998/1/012008.
6) Kayed M, Anter A, Hadeermohamed. IEEE. . Available from: https://doi.org/10.1109/ITCE48509.2020.9047776.
7) Vaidehi K, Manivannan R. An Automated System to Preprocess and Classify Medical Digital X-Rays. *Pervasive Computing and Social Networking*. 2023. Available from: https://doi.org/10.1007/978-981-19-2840-6_59.
8) Vijayaraj A, Raj PTV, Jebakumar R, Senthilvel PG, Kumar N, Kumar RS, et al. Deep Learning Image Classification for Fashion Design. *Wireless Communications and Mobile Computing*. 2022;2022:1–13. Available from: https://doi.org/10.1155/2022/7549397.
9) Kadam SS, Adamuthe AC, Patil AB. CNN Model for Image Classification on MNIST and Fashion-MNIST Dataset. *Journal of scientific research*. 2020;64(02):374–384. Available from: https://doi.org/10.37398/jsr.2020.640251.