Volume 64, Issue 2, 2020
Journal of Scientific Research
Institute of Science,
Banaras Hindu University, Varanasi, India.

CNN Model for Image Classification on MNIST and Fashion-MNIST Dataset

Shivam S. Kadam, Amol C. Adamuthe*, and Ashwini B. Patil

Dept. of CS & IT, RIT, Rajaramnagar, Sangli, MS, India.
kadamshivam007@gmail.com, amol.admuthe@gmail.com*, ashwini.patil@ritindia.edu

Abstract: **Paper presents application of convolutional neural network for image classification problem. MNIST and Fashion-MNIST datasets used to test the performance of CNN model. Paper presents five different architectures with varying convolutional layers, filter size and fully connected layers. Experiments conducted with varying hyper-parameters namely activation function, optimizer, learning rate, dropout rate and batch size. Results show that selection of activation function, optimizer and dropout rate has impact on accuracy of results. All architectures give accuracy more than 99% for MNIST dataset. Fashion-MNIST dataset is complex than MNIST. For Fashion-MNIST dataset architecture 3 gives better results. Review of obtained results and from literature shows that CNN is suitable for image classification for MNIST and Fashion-MNIST dataset.**

Index Terms: **Convolutional neural network, fashion-MNIST, image classification, MNIST.**

I. INTRODUCTION

Albawi et al. (2017) state that Convolutional neural network (CNN) is one of the most popular deep neural networks. O'Shea & Nash (2015) said convolutional neural network (CNN) is similar to artificial neural network (ANN) which is comprised of neurons that self-optimize through learning. CNN has multiple layers namely convolutional layer, pooling layer and fully connected layer. The main objective of convolutional layer is to obtain the features of an image by sliding smaller matrix (kernel or filter) over the entire image and generate the feature maps. The pooling layer used to retain the most important aspect by reducing the feature maps. Fully connected layer interconnect every neuron in the layer to the neurons from the previous and next layer, to take the matrix inputs from the previous layers and flatten it to pass on to the output layer, which will make the prediction. Due to such architecture, it will take fewer parameter to learn and reduce the amount of data required to train the model. CNNs inspired by time-delay neural networks in which the weights are shared in temporal dimension for reduction in computation. Successful training of the hierarchical layers leads CNN as the first successful deep learning architecture proved by Liu et al. (2017). CNN has shown excellent performance for machine learning applications. Due to practical benefits of CNN, it gives better accuracy and improve the performance of system in applications like image classification, computer vision, natural language processing (Albawi et al., 2017), age and gender classification (Levi & Hassner, 2015), text classification (Lai et al. 2015), facial expression recognition (Mollahosseini et al., 2016), speech recognition (Abdel-Hamid et al., 2014), visual document analysis (Simard et al., 2003).

Image classification means assigning the label to input image from fixed set of categories. The image classification includes variety of application like designing robots, objects identifications, automatic cars, traffic signal processing. Feature extraction is more important task for the image representation in classification problem. CNN applied on large-scale datasets to learn images representation and reuse it for the classification. It gives better accuracy and performance in image classification problem as compared to other Neural Network on classifying the CIFAR-10 dataset proved by Wang & Xi (2015).

Objective of this paper is to apply convolutional neural network for image classification problem. Five different architectures with varying convolutional layers, filter size and fully connected layers are proposed. To test the performance of CNN we have used MNIST and Fashion-MNIST datasets.

The rest of this paper is organized as follows. Section II is literature review of different machine learning algorithms for

* Corresponding Author
DOI: 10.37398/JSR.2020.640251

Fashion-MNIST and MNIST. Section III demonstrated CNN of image classification. Experimental details, datasets, results and discussion are presented in section IV. Section V presents conclusions.

II. Literature Review

In literature, many techniques have experimented for image classification of Fashion-MNIST and MNIST.

Levi & Hassner (2015), proposed CNN architecture to overcome the overfitting problem due to less number of images. Result shows that CNN provide improved gender and age classification result even in case of smaller size of contemporary unconstrained image sets. For large-scale image classification using CNN with dataset of 1 million YouTube videos belonging to 487 classes is experimented by Karpathy et al. (2014). Authors found that CNN learn the powerful features even from weakly-labeled data. The performance of proposed model compared using UCF-101 Action Recognition dataset. Significant performance improvement up to 63.3 % is presented. Jmour et al. (2018) trained the CNN on ImageNet dataset for traffic sign classification system. CNN is used to learn features and classify RGB-D images task. Various parameter has effect on accuracy of training results. Authors presented the effect of mini-batch-size on training model. The model given 93.33% accuracy on the test set with a minimum batch size of 10. CNN for document image classification is presented in paper by Kang et al. (2014). Authors used CNN with rectified linear units and trained with dropout. Results obtained using CNN are better than traditional methods. Tobacco litigation and NIST tax-form dataset are used for experimentation. For Tobacco dataset, 80% of accuracy is achieved for training and 20% for validation for 10 classes of images. The median accuracy of 65.37% is achieved for 100 samples. A median accuracy of 100% is achieved through 100 partitions of training and test on NIST tax-form dataset. CNN is used for house number digit classification in paper published by Sermanet et al. (2012). They improved the traditional CNN by multistage features and use of Lp pooling method. Obtained accuracy of 94.85% using SVHN dataset for 45.2% error improvement.

Hu et al. (2015) proposed , five-layer CNN architecture is experimented on several hyperspectral image data sets to classify hyperspectral images directly in spectral domain, which given better performance. CNN framework such as Caffe used to reduce training and testing time and achieved 90% accuracy on MNIST dataset. Manessi & Rozza (2018), has introduced two approaches to learn the different combination of base activation function namely identity function, ReLU and tanh. The proposed approaches compared with well-known architectures namely LeNet-5, AlexNet, and ResNet-56 using three standard datasets (Fashion-MNIST, CIFAR-10 and ILSVRC-2012). Results show substantial improvements in the overall performance, such as an increase in the top-1 accuracy for AlexNet on ILSVRC-2012 of 3.01 percentage points. Tang, Y. (2013), proved that the replacement of softmax layer with SVMs is useful for classification tasks. The experimentation carried out on MNIST and CIFAR-10 datasets. Using SVM, cross validation accuracy for testing phase is increased up to 68.9% which was 67.6% using softmax. Results of RNN for classification of Fashion-MNIST dataset presented by Zhang. This model developed using Long-Short Term Memory technique to reduce the risk of gradient vanishing the traditional RNN faces. Cross validation detects and prevents overfitting and decrease of score caused due to overfitting. The proposed model achieved accuracy more than 89% which is comparatively better than other models. Xiao et al. (2017), presented accuracy of 89.70% and 97.30% for fashion-MNIST dataset and MNIST dataset respectively using SVC classifier.

Bhatnagar et al. (2017) proposed three different CNN architecture for solving Fashion-MNIST dataset, for classification of fashion article images. Authors obtained an accuracy of 92.54% by using a two layer CNN along with batch normalization and skip connections. The architecture proposed by Tang (2013) was emulated by Agarap (2017), by combining the convolutional neural network (CNN) and linear SVM for image classification on MNIST dataset. The result shows that the test accuracy of CNN-SVM model and CNN-Softmax for MNIST dataset is approximately 99.04% and 99.23% respectively. For Fashion-MNIST dataset, CNN-SVM and CNN-Softmax given approximately 90.72% and 91.86% respectively. The random erasing method for training the convolutional neural network (CNN) is used in paper presented by Zhong et al. (2017), which randomly selects a rectangle region in an image and erases its pixels with random values. In this process, training images with various levels of occlusion are generated, which reduces the risk of over-fitting and makes the model robust to occlusion. Experiment conducted on CIFAR10, CIFAR100, and Fashion-MNIST datasets, with various architectures, with good performance on various recognition tasks. ReLU is used as classification function in a deep neural network instead of activation function by Agarap (2018). He implemented feed-forward neural network (FFNN) and convolutional models with two different classification functions, i.e. (1) softmax, and (2) ReLU. The predictive performance of DL-ReLU models is compared with DL-Softmax models on MNIST, Fashion- MNIST and Wisconsin Diagnostic Breast Cancer datasets. CNN-ReLU had the most number of correct predictions per class. Conversely, with its faster convergence, CNN-Softmax had the higher cumulative correct predictions per class. Accuracy obtained on MNIST dataset by using CNN with

softmax activation function is 95.36% whereas using CNN with relu activation function is 91.74%. Hierarchical convolutional neural network using VGGNet on Fashion-MNIST dataset is presented by Seo & Shin (2019). Results of SGD optimizer, 128 batch size, 60 epochs by varying learning rate as 0.001, 0.0002, and 0.00005 are presented. Obtained training and testing accuracy is 100% and 93.52% respectively.

A Novel neural classifier Limited Receptive Area (LIRA) is proposed by Kussul & Baidyk (2004) for image recognition, which contains 3 neuron layers as sensor, associative and output layers. The proposed classifier tested on MNIST databases. The classifier showed 99.41% accuracy. The problem of handwritten digit recognition in optical character recognition is addressed by Deng (2012). The freely available MNIST database of handwritten digits is used. While experimenting on MNIST, artificially distorted versions of the original training samples were introduced in training set that involved random combinations of jittering, shifts, scaling, deskewing, deslanting, blurring, and compression.

The neural networks are emphasized as a dominant technology for analysis of visual inputs obtained from documents. Simard et al. (2003) highlighted use of convolutional neural networks over fully connected networks as the flexible architecture of CNN is suitable for intended problem of visual document. The performance is observed to be optimum when new general set of elastic distortions was created. As a result of implementation on both convolutional neural networks and fully connected networks, simple convolutional neural networks with introduction to elastic distortion yields 0.45 error which is best for MNIST database. Results obtained by using 2 layer MLP (CE) with no distortion 1.6% error rate, 2 layer MLP (CE) with affine distortion 1.1% error rate, 2 layer MLP (MSE) with elastic distortion 0.9% error rate, 2 layer MLP (CE) with elastic distortion 0.7% error rate, simple convolutional network (CE) with affine distortion 0.6% error rate and simple convolutional network (CE) with elastic distortion 0.4% error rate.

Wu trained the CNN on MNIST handwritten digital database using a back-propagation algorithm LeNet-5 convolutional network, which can be used for identifying extreme changing patterns. Input layer is built with 28*28 neurons, representing the size of images. Hidden layer has 100 neurons with sigmoid activation function along with output layer of 10 neurons, representing the classes of handwriting images. Stochastic Gradient Descent (SGD) training algorithm is used for random gradient descent training to minimize the losses. Results obtained are 94.00% as testing accuracy in 100 epochs. Palvanov & Cho (2013) used 2 convolution and 2 fully connected layers with 50 batch size given 98.10% testing accuracy for MNIST dataset.

III. CNN FOR IMAGE CLASSIFICATION

This section presents methodology and implementation details of convolutional neural network for image classification.

Artificial neural networks are computer systems that vaguely resemble the way human brain works. ANNs are basically frameworks consisting a network of computational nodes, which collectively work in distributed manner to process complex data inputs in order to optimize the final output said by O'Shea & Nash (2015). CNNs in a way reflect traditional ANN in aspect of composition, as it comprises Self Optimizing Neurons, of which each neuron receives an input and perform an operation yielding a scalar product accompanied with non-linear function. CNNs are regularized versions of multilayer perceptron's which usually refer to fully connected networks, that is, each neuron in one layer is connected to all neurons in the next layer, last layer of which contains loss functions associated with the classes, where the entire of the network will still express a single perceptive score function (the weight). As per O'Shea & Nash (2015), CNN inherits all of the regular tips and tricks which can be applied for traditional ANNs. There are three types of layers in CNN: convolutional layer, pooling layer, and fully connected layer. All these layers perform different task on the input data. In the convolutional layer, filters are applied to extract the features. Pooling layer perform the max pooling or average pooling, which extract the maximum value in the filter region or average value in filter region respectively. The fully connected layer aggregate the information from feature maps and generate the final classification. Fig. 1 shows CNN for image classification. Table 1 shows the proposed architecture for image classification. Experiments are conducted with five different architectures described in table 1.

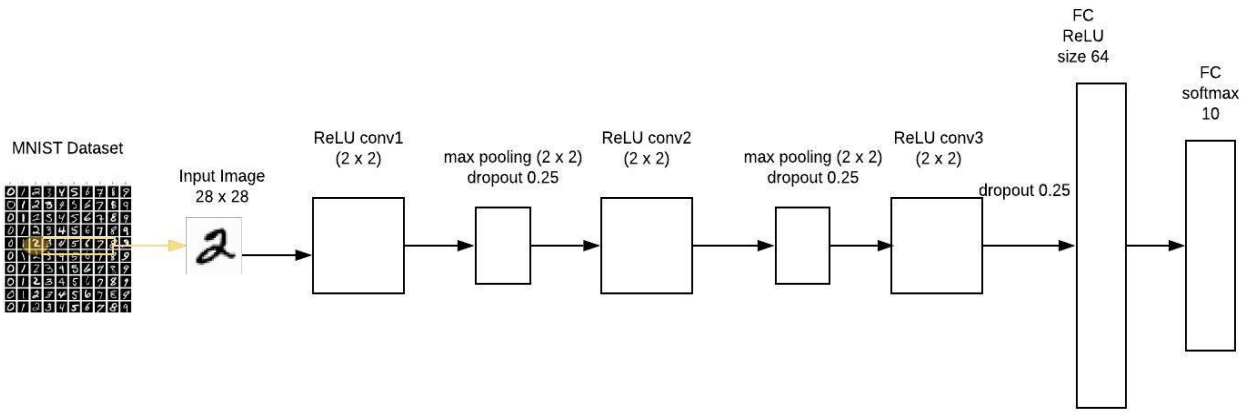

Fig. 1. CNN for image classification

Table 1. Different CNN Architecture for image classification

Architecture 1	Architecture 2	Architecture 3	Architecture 4	Architecture 5
only one input layer and two fully connected layers	2 convolutional layers with (2 x 2) filter size and 2 fully connected layers	3 convolutional layers with (2 x 2) filter size and 2 fully connected layers	4 convolutional layers with (2 x 2) filter size and 2 fully connected layers	4 convolutional layers with (3 x 3) filter size and 2 fully connected layers
(1) INPUT:28×28×1 (2) FC:10 Output Classes	(1) INPUT:28×28×1 (2) FC:10 Output Classes	(1) INPUT:28×28×1 (2) FC:10 Output Classes	(1) INPUT:28×28×1 (2) FC:10 Output Classes	(1) INPUT:28×28×1 (2) FC:10 Output Classes
(3) FC:128 Hidden Neurons	(3) CONV2D:2×2 size,64 filters (4) POOL:2×2 size (5) DROPOUT: = 0.25 (6) CONV2D :2×2 size,64 filters (7) DROPOUT: = 0.25 (8) FC:64 Hidden Neurons (9) DROPOUT: = 0.25	(3) CONV2D:2×2 size,64 filters (4) POOL:2×2 size (5) DROPOUT: = 0.25 (6) CONV2D:2×2 size,64 filters (7) POOL:2×2 size (8) DROPOUT: = 0.25 (9) CONV2D :2×2 size,64 filters (10) DROPOUT: = 0.25 (11) FC:64 Hidden Neurons (12) DROPOUT: = 0.25	(3) CONV2D:2×2 size,64 filters (4) POOL:2×2 size (5) DROPOUT: = 0.25 (6) CONV2D:2×2 size,64 filters (7) POOL:2×2 size (8) DROPOUT: = 0.25 (9) CONV2D:2×2 size,64 filters (10) POOL:2×2 size (11) DROPOUT: = 0.25 (12) CONV2D :2×2 size,64 filters (13) DROPOUT: = 0.25 (14) FC:64 Hidden Neurons (15) DROPOUT: = 0.25	(3) CONV2D:3×3 size,32 filters (4) CONV2D:3×3 size,32 filters (4) POOL:2×2 size (5) DROPOUT: = 0.25 (6) CONV2D:3×3 size,64 filters (7) CONV2D:3×3 size,64 filters (8) POOL:2×2 size (9) DROPOUT: = 0.25 (10) FC:512 Hidden Neurons (11) DROPOUT: = 0.5

All experiments in this study were conducted on a laptop computer with Intel i5 processor, 8 GB of DDR3 RAM by using google colaboratory. Colaboratory is a research tool for machine learning education and research. A Jupyter notebook environment requires no setup to use and runs entirely in the cloud. With Colaboratory you can write and execute code, save and share your analyses, and access powerful computing resources, all for free from your browser.

IV. DATASET, RESULTS AND DISCUSSION

This section presents datasets, obtained results and discussion.

A. Datasets

MNIST: MNIST by LeCun et al. (2010), is one of the established standard datasets for benchmarking deep learning models. MNIST dataset consists of 28x28 grey-scale images of handwritten digits. Training dataset consists of 60000 images and testing dataset consists of 10000 images of different classes such as '0', '1', '2', '3', '4', '5', '6', '7', '8', and '9'.

Fashion-MNIST. Xiao et al. (2017) presented the new Fashion-MNIST dataset as an alternative to the conventional MNIST. Fashion-MNIST, a fashion product images dataset, which comprises of 28x28 grayscale images of 70,000 images of which 60,000 are used for training and 10,000 are used for

testing purpose. This paper consists of 10 fashion item classes such as T-shirt/Top, Trouser, Pullover, Dress, Coat, Sandals, Shirt, Sneaker, Bag, Ankle boots which has been classified using Convolutional Neural Network. This dataset is compatible with any machine-learning package capable of working with the original MNIST dataset.

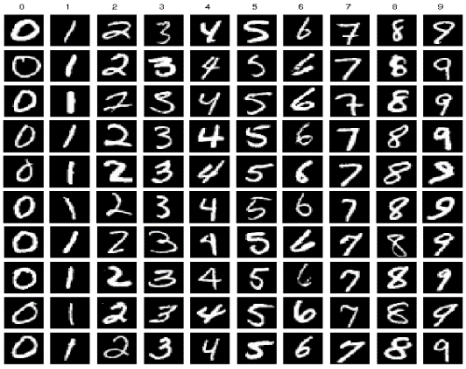

Fig. 2. MNIST dataset image

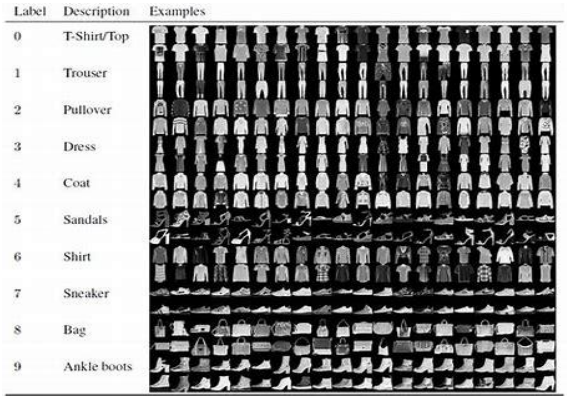

Fig. 3. Fashion-MNIST dataset image

B. Results of Architecture 1

During the experimentation performance of sigmoid, softmax, relu and tanh activation functions are tested. Results are taken with different optimizers namely Adam, Adagrad, Adadelta, SGD and RMSprop.

Table 2 shows the performance of different optimizers.

Table 2. Performance of optimizers

		Fashion-MNIST dataset		**MNIST dataset**	
Optimizer	**epoch**	**Training accuracy**	**Testing accuracy**	**Training accuracy**	**Testing accuracy**
SGD	50	87.99%	86.53%	99.97%	98.38%
RMSprop	50	94.25%	88.76%	99.70%	97.99%
Adagrad	50	97.04%	89.62%	99.92%	98.15%
	60	99.60%	89.65%	99.92%	98.20%
Adadelta	60	94.72%	89.00%	99.93%	98.20%
	70	96.90%	89.04%	99.94%	98.22%
	100	97.99%	88.87%	99.96%	98.23%
Adam	50	97.38%	88.59%	99.20%	98.12%
	60	98.10%	88.69%	99.46%	98.34%
	70	98.39%	89.03%	99.69%	98.43%
	80	98.57%	89.16%	99.71%	98.54%
	100	98.80%	89.18%	99.78%	98.60%

Table 3 shows the result with varying batch size. Sigmoid activation function, 0.1 dropout and Adam optimizer is used. Increasing batch size has no significant impact on testing/training accuracy.

Table 3. Batch size experimentation

	Fashion-MNIST dataset		**MNIST dataset**	
Batch size	**Training accuracy**	**Testing accuracy**	**Training accuracy**	**Testing accuracy**
16	93.88%	88.39%	99.65%	98.09%
32	94.41%	88.74%	99.67%	98.03%
64	94.73%	89.30%	99.84%	98.29%
100	94.52%	89.48%	99.91%	98.48%
120	94.56%	88.39%	99.92%	98.14%
140	94.50%	89.05%	99.90%	98.40%
150	94.60%	89.38%	99.90%	98.41%
160	96.63%	89.05%	99.94%	98.34%
200	94.48%	88.82%	99.93%	98.27%

Table 4. Learning rate experimentation

	Fashion-MNIST dataset		**MNIST dataset**	
Learning rate	**Training accuracy**	**Testing accuracy**	**Training accuracy**	**Testing accuracy**
0.01	87.50%	85.43%	98.69%	97.85%
0.02	84.50%	84.01%	99.68%	98.18%
0.001	92.59	88.67	99.73%	98.19%
0.002	92.93	89.21	99.75%	98.18%

Table 4 shows result of adam optimizer, sigmoid activation function, 100 Batch size, 60 epochs with varying learning rate. Fashion-MNIST dataset is found more difficult than MNIST dataset. Accuracy on MNIST dataset are more than 99% for majority of cases. For Fashion-MNIST dataset 0.002 learning rate works better.

Table 5 and 6 shows the result with varying dropout for MNIST and Fashion-MNIST dataset respectively. For both datasets, training accuracy decrease with increase in dropout rate. For both datasets, increase in dropout shows slight increase in testing accuracy followed by decrease.

C. Results of Architecture 2

For MNIST dataset, best training accuracy and testing accuracy obtained are 98.86% and 98.96% respectively. The best result obtained with 128 batch size, softmax activation function, adam optimizer, 0.25 dropout after each pooling layer, 50 epochs and 2x2 kernel size.

For Fashion-MNIST dataset, best training accuracy and testing accuracy obtained are 92.02% and 92.76% respectively. The best result obtained with 128 Batch size, softmax activation function,

adam optimizer, 0.25 dropout after each pooling layer, 50 epochs and 2x2 kernel size.

Table 5. Result of MNIST dataset

Batch size	Epoch	Dropout	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7
50	50	Training	100%	99.70%	99.45%	99.03%	98.44%	97.78%	96.68%	94.71%
		Testing	98.09%	98.09%	98.21%	98.25%	98.35%	98.18%	97.86%	97.57%
	100	Training	99.95%	99.76%	99.63%	99.30%	98.75%	98.05%	97.04%	95.17%
		Testing	97.89%	98.05%	98.47%	98.37%	98.21%	98.19%	98.02%	97.62%
100	50	Training	99.78%	99.74%	99.46%	99.02%	98.55%	97.72%	96.57%	94.74%
		Testing	97.57%	98.13%	98.22%	98.34%	98.10%	98.19%	98.05%	97.32%
	100	Training	100%	99.80%	99.65%	99.26%	98.82%	98.20%	97.08%	95.34%
		Testing	98.10%	98.10%	98.40%	98.37%	98.46%	98.26%	98.11%	97.78%

Table 6. Result of Fashion-MNIST dataset

Batch size	Epoch	Dropout	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7
50	50	Training	96.34%	94.58%	93.60%	92.26%	91.25%	89.32%	87.71%	84.93%
		Testing	88.84%	89.15%	89.32%	89.11%	89.17%	88.23%	88.04%	87.07%
	100	Training	98.17%	96.51%	95.14%	93.72%	92.26%	90.71%	88.73%	86.16%
		Testing	88.14%	88.77%	88.64%	89.15%	89.45%	88.84%	88.19%	87.86%
100	50	Training	96.21%	94.67%	93.58%	92.36%	91.18%	89.79%	88.31%	85.89%
		Testing	88.77%	89.11%	88.94%	88.92%	89.18%	88.35%	88.51%	87.53%
	100	Training	98.32%	96.45%	95.39%	94.20%	92.58%	90.91%	88.77%	86.67%
		Testing	89.13%	89.37%	89.48%	89.43%	89.03%	89.03%	88.81%	87.92%

D. Results of Architecture 3

For MNIST dataset, best training accuracy and testing accuracy obtained are 99.60% and 99.37% respectively. The best result obtained with 128 batch size, softmax activation function, adam optimizer, 0.25 dropout after each pooling layer, 50 epochs and 2x2 kernel size.

For Fashion-MNIST dataset, best training accuracy and testing accuracy obtained are 93.09% and 93.56% respectively. The best result obtained with 128 Batch size, softmax activation function, adam optimizer, 0.25 dropout after each pooling layer, 50 epochs and 2x2 kernel size.

E. Results of Architecture 4

In architecture 4, the optimal parameter are 128 batch size, 50 epochs, Softmax activation function, adam optimizer, 2x2 kernel size and 0.25 dropout after each pooling layers. For MNIST dataset, best obtained training accuracy and testing accuracy are 99.02% and 99.03% respectively. For Fashion-MNIST dataset, best obtained training accuracy and testing accuracy are 93.17% and 92.94% respectively.

F. Results of Architecture 5

In architecture 5, the optimal parameter are 128 batch size, 50 epochs, softmax activation function, RMSprop optimizer, (2x2) kernel size and 0.25 dropout after each pooling layers. For MNIST dataset, best obtained training accuracy and testing accuracy are 99.26% % and 99.29% % respectively. For Fashion-MNIST dataset, best obtained training accuracy and testing accuracy are 92.67 % and 92.86 % respectively.

G. Comparison of results

This subsection presents comparison of obtained results of proposed CNN architectures. Further, the obtained results are compared with literature.

Results indicates that CNN is suitable to solve MNIST dataset. The results obtained with all architectures are close to or better than 99%. Training and testing accuracy are same.

For Fashion-MNIST dataset, architecture 1 gives more than

99.6-% training accuracy and 89.65% testing accuracy. Training accuracy given by architecture is better than all others but fails in testing accuracy. Architecture 3 gives highest testing accuracy.

Table 7. Comparison of obtained results

	Fashion-MNIST		**MNIST**	
	Training accuracy	**Testing accuracy**	**Training accuracy**	**Testing accuracy**
Architecture 1	99.60%	89.65%	99.91%	98.48%
Architecture 2	92.02%	92.76%	98.86%	98.96%
Architecture 3	93.09%	93.56%	99.60%	99.37%
Architecture 4	93.17%	92.94%	99.02%	99.03%
Architecture 5 with Adam optimizer	93.12%	93.56%	99.48%	99.55%
Architecture 5 with RMSprop optimizer	92.67%	92.86%	99.26%	99.29%

Table 8. Comparison of results

Technique used	**Fahion-MNIST Testing Accuracy**	**MNIST Testing Accuracy**
Support Vector Classifier with rbf kernel Bhatnagar et al. (2017)	89.70%	-
Decision Tree Classifier Xiao et al. (2017)	79.80%	87.30%
Extra Tree Classifier Xiao et al. (2017)	77.50%	80.60%
Gradient Boosting Classifier Xiao et al. (2017)	88.00%	96.90%
Kneighbors Classifier Xiao et al. (2017)	85.40%	95.90%
Linear SVC Xiao et al. (2017)	83.60%	91.70%
Logistic Regression Xiao et al. (2017)	84.20%	91.70%
MLP Classifier Xiao et al. (2017)	87.10%	97.20%
Passive Aggressive Classifier Xiao et al. (2017)	77.60%	87.70%
Perceptron Xiao et al. (2017)	78.20%	88.70%
Random Forest Classifier Xiao et al. (2017)	87.30%	97.00%
SGD Classifier Xiao et al. (2017)	81.90%	91.40%
SVC Xiao et al. (2017)	89.70%	97.30%
Gaussian NB Xiao et al. (2017)	51.10%	52.40%
Three-layer Neural Network (Zhang).	87.23%	
ResNet-20 Zhong et al. (2017)	-	% of error rate 4.02±0.07
ResNet-32 Zhong et al. (2017)	-	% of error rate 3.80±0.05
ResNet-44 Zhong et al. (2017)	-	% of error rate 4.01±0.14
ResNet-56 Zhong et al. (2017)	-	% of error rate 4.13±0.42
ResNet-110 Zhong et al. (2017)	-	% of error rate 4.01±0.13
ResNet-20-PreAct Zhong et al. (2017)	-	% of error rate 4.02±0.09
ResNet-32-PreAct Zhong et al. (2017)	-	% of error rate 4.00±0.05
ResNet-44-PreAct Zhong et al. (2017)	-	% of error rate 4.23±0.15
ResNet-56-PreAct Zhong et al. (2017)	-	% of error rate 3.99±0.08
ResNet-110-PreAct Zhong et al. (2017)	-	% of error rate 4.19±0.15
ResNet-18-PreAct Zhong et al. (2017)	-	% of error rate 3.90±0.06
WRN-28-10 Zhong et al. (2017)	-	% of error rate 3.65±0.03
ResNeXt-8-64 Zhong et al. (2017)	-	% of error rate 3.79±0.06
FFNN with softmax Agarap (2018)	89.35%	97.98%
FFNN with relu Agarap (2018)	89.06%	97.77%
Proposed classifier LIRA grayscale (neural net) Kussul & Baidyk (2004)	-	% of error rate 0.61
2 layer MLP (CE) with no distortion (Simard et al. 2003).	-	% of error rate 1.6
2 layer MLP (CE) with affine distortion (Simard et al. 2003).	-	% of error rate 1.1
2 layer MLP (MSE) with elastic distortion Simard et al. 2003).	-	% of error rate 0.9
2 layer MLP (CE) with elastic distortion (Simard et al. 2003).	-	% of error rate 0.7
Regression model Palvanov & Cho (2013)	-	92.10%
ResNet Palvanov & Cho (2013)	-	97.30%
CapsNet Palvanov & Cho (2013)	-	99.40%
Reduced set SVM poly 5 Seo & Shin , 2019	-	% of error rate 1.0
LeNet-5 (neural net) Seo & Shin , 2019	-	% of error rate 0.95
Virtual SVM poly 9 [distortions] Seo & Shin , 2019	-	% of error rate 0.8
LeNet-5 [distortions] (neural net) Seo & Shin , 2019	-	% of error rate 0.8

Boosted LeNet-4 [distortions] (neural net) Seo & Shin , 2019	-	% of error rate 0.7
2 layer MLP (MSE) with affine distortion. LeCun et al (1998)	-	% of error rate 1.6
Tangent dist. with affine+thick distortion LeCun et al (1998)	-	% of error rate 1.1
Lenet5 (MSE) with affine distortion [29]	-	% of error rate 0.8
Boost Lenet4 MSE with affine distortion, LeCun et al (1998)	-	% of error rate 0.7
SVM with affine distortion Decoste & Schölkopf (2002)	-	% of error rate 1.4
Virtual SVM with affine distortion Decoste & Schölkopf (2002)	-	% of error rate 0.6
Shape matching + 3-NN Belongie et al. ,2001; Belongie et al. ,2002	-	% of error rate 0.63
SVC-rbf_grayscale Liu et al. (2002)	-	% of error rate 0.42

For MNIST SVC presented by Xiao et al. (2017) gave high testing accuracy of 97.30% but Liu et al. (2002), have added rbf kernel to SVC which gave him increase in accuracy i.e. 0.42% error rate. As per

Simard et al. 2003, distortion is playing major role in defining testing accuracy for MNIST as 2 layer MLP (CE) with no distortion gave 1.6% of error rate whereas adding affine distortion gave 1.1% of error rate whereas adding elastic distortion gave 0.7% of error rate. This states elastic distortion works better as LeCun et al. (1998) with affine distortion also worked poor compared to elastic distortion.

ResNet-X is a residual network where X represents number of total layers where in paper by Zhong et al. (2017), ResNet with 32 layers worked great with just % error rate of 3.80±0.05. Whereas LeNet by Seo & Shin (2019), is CNN architecture of 2 CNN layers followed by fully connected layers where first layer has 20 filters and second layer has 50 filters of size (5 x 5). Also Max pooling of (2x2) is used after every CNN layer but dropout is omitted in LeNet.

KerasNet is a CNN in keras framework which is made of 4 CNN layers and fully connected layers and contains max pooling with dropout. Here KerasNet performed well as compared with LeNet-5 (Manessi & Rozza, 2018).

Results reported in table 8, indicates that neural network, multiplayer perceptron and deep learning methods are better than other classifiers.

Table 9 and 10, compares various factors such as activation function, optimizer, batch size, epochs, no of layers, dropout, and learning rate for different CNN architectures proposed and presented in literature for datasets under consideration.

Table 9. Comparing CNN Architectures for Fashion-MNIST

Reference	Method	Activation Function	Optimizer	Batch size	Epoch	Layers	Dropout	Learning rate	Training accuracy	Testing accuracy
Bhatnagar et al. (2017)	CNN with Batch normalization and Residual skip	NA	NA	NA	NA	2 Convolution 2 fully connected	NA	NA	NA	92.54%
Bhatnagar et al. (2017)	CNN	SVM	NA	NA	NA	2 Convolution 2 fully connected	NA	NA	NA	90.72%
Bhatnagar et al. (2017)	CNN	softmax	NA	NA	NA	2 Convolution 2 fully connected	NA	NA	NA	91.86%
Bhatnagar et al. (2017)	CNN with Batchnor-malization	NA	NA	NA	NA	2 Convolution 2 fully connected	NA	NA	NA	92.22%
Manessi & Rozza, 2018	KerasNet	aff({id,ReLU, tanh})	RMSprop	NA	NA	4 convolution 2 fully connected	NA	0.0001	NA	94.41%
Manessi & Rozza, 2018	KerasNet	id	RMSprop	NA	NA	4 convolution 2 fully connected	NA	0.0001	NA	90.51%
Manessi & Rozza, 2018	KerasNet	ReLU	RMSprop	NA	NA	4 convolution 2 fully connected	NA	0.0001	NA	90.79%
Manessi & Rozza, 2018	KerasNet	Tanh	RMSprop	NA	NA	4 convolution 2 fully connected	NA	0.0001	NA	93.43%
Manessi & Rozza, 2018	KerasNet	LReLU	RMSprop	NA	NA	4 convolution 2 fully connected	NA	0.0001	NA	91.13%
Manessi & Rozza, 2018	KerasNet	conv({id,ReLU})	RMSprop	NA	NA	4 convolution 2 fully connected	NA	0.0001	NA	92.39%
Manessi &	KerasNet	conv({id,tanh})	RMSprop	NA	NA	4 convolution	NA	0.0001	NA	93.64%

Rozza, 2018						2 fully connected				
Manessi & Rozza, 2018	KerasNet	conv({tanh,ReLU})	RMSprop	NA	NA	4 convolution 2 fully connected	NA	0.0001	NA	92.04%
Manessi & Rozza, 2018	KerasNet	conv({id,ReLU,tanh})	RMSprop	NA	NA	4 convolution 2 fully connected	NA	0.0001	NA	92.94%
Manessi & Rozza, 2018	KerasNet	aff({id,ReLU})	RMSprop	NA	NA	4 convolution 2 fully connected	NA	0.0001	NA	93.37%
Manessi & Rozza, 2018	KerasNet	aff({id, tanh})	RMSprop	NA	NA	4 convolution 2 fully connected	NA	0.0001	NA	94.41%
Manessi & Rozza, 2018	KerasNet	aff({tanh,ReLU})	RMSprop	NA	NA	4 convolution 2 fully connected	NA	0.0001	NA	93.48%
Manessi & Rozza, 2018	LeNet-5	aff({tanh,ReLU })	RMSprop	NA	NA	4 convolution 2 fully connected	NA	0.0001	NA	93.02%
Manessi & Rozza, 2018	LeNet-5	id	RMSprop	NA	NA	4 convolution 2 fully connected	NA	0.0001	NA	90.50%
Manessi & Rozza, 2018	LeNet-5	ReLU	RMSprop	NA	NA	4 convolution 2 fully connected	NA	0.0001	NA	91.06%
Manessi & Rozza, 2018	LeNet-5	Tanh	RMSprop	NA	NA	4 convolution 2 fully connected	NA	0.0001	NA	92.33%
Manessi & Rozza, 2018	LeNet-5	LReLU	RMSprop	NA	NA	4 convolution 2 fully connected	NA	0.0001	NA	91.03%
Manessi & Rozza, 2018	LeNet-5	conv({id,ReLU})	RMSprop	NA	NA	4 convolution 2 fully connected	NA	0.0001	NA	91.87%
Manessi & Rozza, 2018	LeNet-5	conv({id,tanh})	RMSprop	NA	NA	4 convolution 2 fully connected	NA	0.0001	NA	92.36%
Manessi & Rozza, 2018	LeNet-5	conv({tanh,ReLU})	RMSprop	NA	NA	4 convolution 2 fully connected	NA	0.0001	NA	92.56%
Manessi & Rozza, 2018	LeNet-5	conv({id,ReLU,tanh})	RMSprop	NA	NA	4 convolution 2 fully connected	NA	0.0001	NA	92.21%
Manessi & Rozza, 2018	LeNet-5	aff({id,ReLU})	RMSprop	NA	NA	4 convolution 2 fully connected	NA	0.0001	NA	92.83%
Manessi & Rozza, 2018	LeNet-5	aff({id, tanh})	RMSprop	NA	NA	4 convolution 2 fully connected	NA	0.0001	NA	92.65%
Manessi & Rozza, 2018	LeNet-5	aff({id,ReLU, tanh})	RMSprop	NA	NA	4 convolution 2 fully connected	NA	0.0001	NA	92.80%
Agarap (2017)	CNN	softmax	NA	128	10000 steps	NA	0.5	0.003	NA	91.86%
Agarap (2017)	CNN	SVM	NA	128	10000 steps	NA	0.5	0.003	NA	90.72%
Agarap (2018)	NA	softmax	NA	NA	NA	4 Convolution 2 fully connected	NA	NA	NA	86.08%
Agarap (2018)	NA	relu	NA	NA	NA	4 Convolution 2 fully connected	NA	NA	NA	85.84%
Seo & Shin , 2019	VGG16	relu	SGD	128	60	NA	NA	0.001 0.0002 0.00005	100%	93.52%
Seo & Shin , 2019	VGG19	relu	SGD	128	60	NA	NA	0.001 0.0002 0.00005	100%	93.33%
Architecture 3	CNN	softmax	adam	128	50	3 Convolution 2 fully connected	0.25	0.001	93.09%	93.56%

NA indicates not available

Table 10. Comparing CNN Architectures for MNIST

Reference	**Activation**	**Dropout**	**Learning rate**	**Batch size**	**Layers**	**Testing accuracy**
[16]	softmax	NA	NA	300	2 convolution layers	% of error rate 14.00
[16]	SVM	NA	NA	300	2 convolution layers	% of error rate 11.90
Agarap (2017)	softmax	0.5	0.003	128	NA	99.23%

Agarap (2017)	SVM	0.5	0.003	128	NA	99.04%
Agarap (2018)	softmax	NA	NA	NA	4 Convolution 2 fully connected	95.36%
Agarap (2018)	relu	NA	NA	NA	4 Convolution 2 fully connected	91.74%
Simard et al. 2003	NA	NA	NA	NA	2 Convolution 2 fully connected with elastic distortion	% of error rate 0.4
H. Wu	NA	NA	NA	NA	3 Convolution 1 fully connected	94%
Palvanov & Cho (2013)	NA	NA	NA	50	2 Convolution 2 fully connected	98.10%
Architecture 3	softmax	0.1	0.001	128	3Convolution 2 fully connected	99.37%

NA indicates not available

Figure 4 and 5, presents the training time required for proposed architectures for solving MNIST and Fashion-MNIST dataset. Architecture 1 with only one input layer and two fully connected layers requires 100 seconds for training the algorithm. Training time increases with increase in convolutional layers. Figure 4 and 5 indicates significant increase in training time with 3x3 filter size.

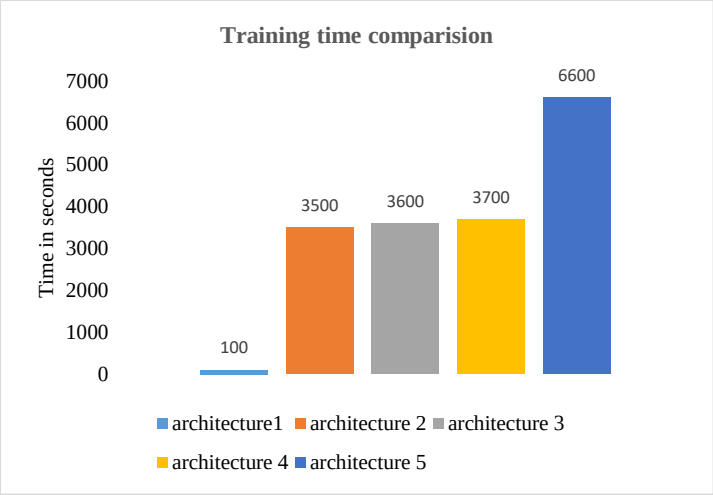

Fig. 4. Training time comparison for MNIST dataset

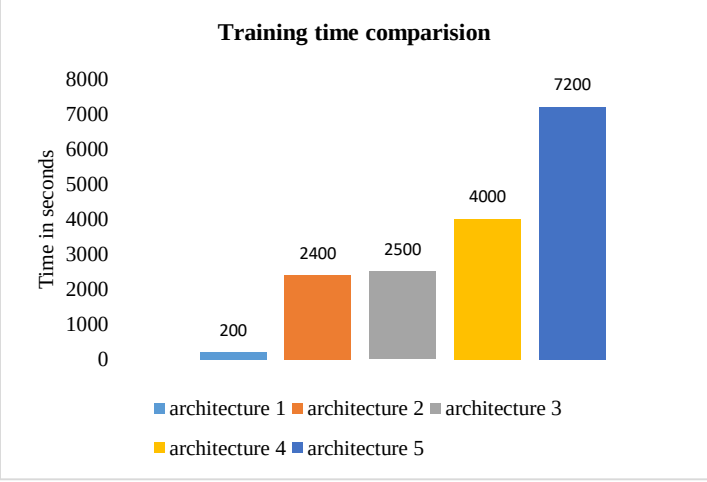

Fig. 5. Training time comparison for Fashion-MNIST dataset

CONCLUSIONS

Paper introduced five CNN architectures to solve image classification problem on MNIST and Fashion-MNIST dataset. Results indicates that any CNN model give 99% accuracy for MNIST dataset. Architecture 3 (3 convolutional layer and 2 fully connected layers) gives better testing accuracy for fashion-MNIST dataset. In Architecture 1 algorithm best optimizer observed is adagrad, best activation function is sigmoid, batch size is 64, epoch are 50 and dropout is 0.1. From Architecture 2 algorithm onwards best optimizer observed is adam, best activation function is softmax, batch size is 128, epoch are 50, dropout is 0.25 and kernel size is (2x2). Architecture 1 with only one input layer and two fully connected layers requires less time for training the algorithm. Training time increases with increase in convolutional layers. There is significant increase in training time with 3x3 filter size.

REFERENCES

Abdel-Hamid, O., Mohamed, A. R., Jiang, H., Deng, L., Penn, G., & Yu, D. (2014). Convolutional neural networks for speech recognition. IEEE/ACM Transactions on audio, speech, and language processing, 22(10), 1533-1545.

Agarap, A. F. (2017). An architecture combining convolutional neural network (CNN) and support vector machine (SVM) for image classification. *arXiv preprint arXiv:1712.03541*.

Agarap, A. F. (2018). Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*.

Albawi, S., Mohammed, T. A., & Al-Zawi, S. (2017, August). Understanding of a convolutional neural network. In *2017 International Conference on Engineering and Technology (ICET)* (pp. 1-6). IEEE.

Belongie, S., Malik, J., & Puzicha, J. (2001, July). Matching shapes. In *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001* (Vol. 1, pp. 454-461). IEEE.

Belongie, S., Malik, J., & Puzicha, J. (2002). Shape matching and object recognition using shape contexts. *IEEE transactions on pattern analysis and machine intelligence*, *24*(4), 509-522.

Bhatnagar, S., Ghosal, D., & Kolekar, M. H. (2017, December). Classification of fashion article images using convolutional

neural networks. In *2017 Fourth International Conference on Image Information Processing (ICIIP)* (pp. 1-6). IEEE.

Decoste, D., & Schölkopf, B. (2002). Training invariant support vector machines. *Machine learning*, *46*(1-3), 161-190.

Deng, L. (2012). The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Processing Magazine*, *29*(6), 141-142

H. Wu, "CNN-Based Recognition of Handwritten Digits in MNIST Database".

Hu, W., Huang, Y., Wei, L., Zhang, F., & Li, H. (2015). Deep convolutional neural networks for hyperspectral image classification. *Journal of Sensors*, *2015*.

Jmour, N., Zayen, S., & Abdelkrim, A. (2018, March). Convolutional neural networks for image classification. In *2018 International Conference on Advanced Systems and Electric Technologies (IC_ASET)* (pp. 397-402). IEEE.

Kang, L., Kumar, J., Ye, P., Li, Y., & Doermann, D. (2014, August). Convolutional neural networks for document image classification. In *2014 22nd International Conference on Pattern Recognition* (pp. 3168-3172). IEEE.

Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., & Fei-Fei, L. (2014). Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition* (pp. 1725-1732)

Kussul, E., & Baidyk, T. (2004). Improved method of handwritten digit recognition tested on MNIST database. *Image and Vision Computing*, *22*(12), 971-981.

Lai, S., Xu, L., Liu, K., & Zhao, J. (2015, February). Recurrent convolutional neural networks for text classification. In *Twenty-ninth AAAI conference on artificial intelligence*.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278-2324.

LeCun, Y., Cortes, C., & Burges, C. J. (2010). Mnist handwritten digit database. AT&T Labs.

Levi, G., & Hassner, T. (2015). Age and gender classification using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops* (pp. 34-42)

Liu, C. L., Nakashima, K., Sako, H., & Fujisawa, H. (2002, August). Handwritten digit recognition using state-of-the-art techniques. In Proceedings Eighth International Workshop on Frontiers in Handwriting Recognition (pp. 320-325). IEEE

Liu, W., Wang, Z., Liu, X., Zeng, N., Liu, Y., & Alsaadi, F. E. (2017). A survey of deep neural network architectures and their applications. *Neurocomputing*, *234*, 11-26.

Manessi, F., & Rozza, A. (2018, August). Learning combinations of activation functions. In *2018 24th International Conference on Pattern Recognition (ICPR)* (pp. 61-66). IEEE.

Mollahosseini, A., Chan, D., & Mahoor, M. H. (2016, March). Going deeper in facial expression recognition using deep neural networks. In *2016 IEEE Winter conference on applications of computer vision (WACV)* (pp. 1-10). IEEE.

O'Shea, K., & Nash, R. (2015). An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*.

Palvanov, A., & Im Cho, Y. (2018). Comparisons of deep learning algorithms for MNIST in real-time environment. *International Journal of Fuzzy Logic and Intelligent Systems*, *18*(2), 126-134.

Seo, Y., & Shin, K. S. (2019). Hierarchical convolutional neural networks for fashion image classification. *Expert Systems with Applications*, *116*, 328-339.

Sermanet, P., Chintala, S., & LeCun, Y. (2012, November). Convolutional neural networks applied to house numbers digit classification. In Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012) (pp. 3288-3291). IEEE.

Simard, P. Y., Steinkraus, D., & Platt, J. C. (2003, August). Best practices for convolutional neural networks applied to visual document analysis. In *Icdar* (Vol. 3, No. 2003).

Tang, Y. (2013). Deep learning using linear support vector machines. *arXiv preprint arXiv:1306.0239*.

Wang, C., & Xi, Y. (2015). Convolutional Neural Network for Image Classification. *Johns Hopkins University Baltimore, MD*, *21218*.

Xiao, H., Rasul, K., & Vollgraf, R. (2017). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*.

Zhang, K. LSTM: An Image Classification Model Based on Fashion-MNIST Dataset.

Zhong, Z., Zheng, L., Kang, G., Li, S., & Yang, Y. (2017). Random erasing data augmentation. *arXiv preprint arXiv:1708.04896*.

***