CQNN: Convolutional Quadratic Neural Networks

Pranav Mantini
Department of Computer Science
University of Houston
Houston, Texas 77004
Email: pmantini@uh.edu

Shishr K. Shah
Department of Computer Science
University of Houston
Houston, Texas 77004
Email: sshah@central.uh.edu

***Abstract*—Image classification is a fundamental task in computer vision. A variety of deep learning models based on the Convolutional Neural Network (CNN) architecture have proven to be an efficient solution. Numerous improvements have been proposed over the years, where broader, deeper, and denser networks have been constructed. However, the atomic operation for these models has remained a linear unit (single neuron). In this work, we pursue an alternative dimension by hypothesizing the atomic operation to be performed by a quadratic unit. We construct convolutional layers using quadratic neurons for feature extraction and subsequently use dense layers for classification. We perform analysis to quantify the implication of replacing linear neurons with quadratic units. Results show a keen improvement in classification accuracy with quadratic neurons over linear neurons.**

I. Introduction

Image classification is a fundamental task in computer vision that categorizes images according to visual content. While the idea has been lingering around since the early 1980's [14], [24], methods to accomplish this within practical bounds of performance and computational efficiency were not available until the 2000's [39], [17]. In the recent past, the availability of large scale image datasets [7], [20] along with the ability to perform parallel computations on graphical processing units (GPUs) have unleashed a spectacle of performance enhancement in visual recognition tasks that is on par with humans [12]. A variety of deep learning models have gained popularity by sequentially outperforming the previous: AlexNet [21], VGGNet [33], DenseNet [19], GoogleNet [36], and ResNet [15]. These models use a Convolutional Neural Network (CNN) at the core, which was proposed in the late 1980's [22].

CNNs are heavily motivated by the visual system in animals. Hubel and Wiesel demonstrated the existence of neurons that respond to visual stimuli. Their work identified two types of cells: simple and complex. The response of simple cells has been successfully modeled using linear functions (weighted sum of stimuli) [18]. Complex cells were noted to perform non-linear operations based on the output of other cells (Y Cells) [1]. The idea of a simple cell became an atomic unit of computation for neural networks and subsequently for modern CNNs. Fukushima [10] implemented a mechanism called "neocognitron" that has neurons that take as input a spatially correlated sequence (patches) of image pixels. The model introduced two layers, one a convolution layer consisting of neurons for transformation and the latter a down-sampling layer for classification. These have become the basic blocks of construction for the modern CNN architectures.

Lecun proposed LeNet-5 by stacking convolution layers followed by pooling and finally fully connected layers [23], which became a template for modern CNNs. Subsequent research was pursued along multiple dimensions to increase classification and recognition performance. AlexNet [21] proposed by Krizhevsky *et al.*demonstrated the use of overlapping pooling and Rectified Linear Unit (ReLU) activation functions to introduce non-linearity and learn complex mappings. Some researchers have hypothesized that deeper or broader models (more horizontal or vertical layers) can learn complex representations for classifications. VGG16/19 proposed by Simonyan and Zisserman consisted of 16/19 layers stacked vertically for image classification. Ciresan proposed multi-column CNN that stacked layers horizontally [5]. Other researchers have utilized the idea of the network in network [25] approach to form denser layers. Over the years, a multitude of tricks such as batch normalization, skipped connections (ResNet [15], Densenet [19]), dilated convolutions [40], etc.) have been introduced to improve performance. Other areas of research have focused on faster computations (Xception uses depth-wise separable convolutions [4]) and efficient optimization techniques for parameter estimation. However, the atomic operation for these models has remained a linear unit, representing simple cells.

In this work, we pursue an alternative dimension by hypothesizing the atomic operation to be performed by a quadratic unit that is representative of a complex cell. Figure 1 (a) shows an example of classification problem in Euclidean space. The data consists of two classes of points arranged in a spiral shape. We train two simple neural network models with two hidden layers (consisting of 4 and 3 neurons, respectively). The first model uses a ReLU activation function. In the second network, the neurons in the first hidden layer apply quadratic functions and utilize a sigmoid activation. Each network is trained for 2000 epochs, and Figure 1 (b) and (c) shows the decision boundary formed by the first and second architectures, respectively. The misclassified samples are indicated by a red boundary/arrow. The first network misclassified 31 blue samples, while the second network with quadratic neurons misclassified 2 samples (blue). This example demonstrates the capability to produce complex decision boundaries by equip-

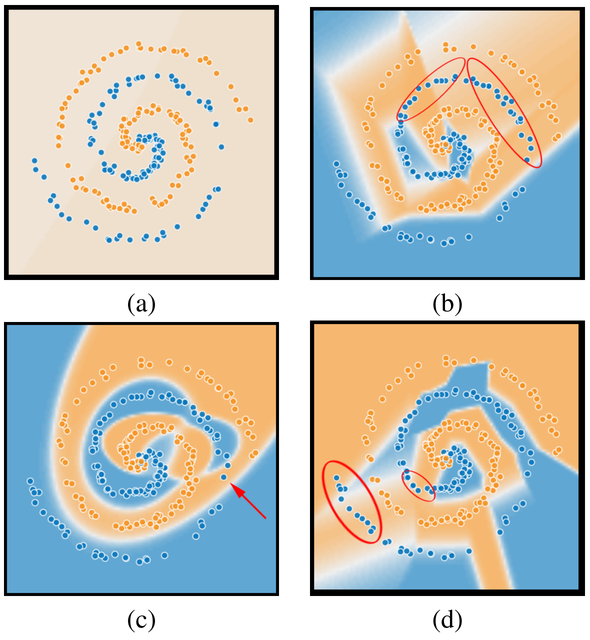

Fig. 1. Decision boundary of three models trained for classification (a) spiral shape toy data; (b) Neural network with linear neurons; (c) Neural network with quadratic neurons; (d) Neural network with linear neurons with 51 parameters.

ping neurons with quadratic functions (higher-order functions). In all fairness, we would like to note that the second model has a higher number of parameters (count of 36) as it is using quadratic functions, compared to the first model that has 24. Figure 1 (d) shows the decision boundary of a third network where the number of neurons in the first layer is increased to 8, each applying a linear function. Despite the fact that this network has more parameters (51) than the quadratic network, this network misclassified 17 samples (blue). This illustrates the point that scaling the order of neurons can produce complex functions that may not be possible to learn by scaling the number of linear neurons. The primary contributions of this work are:

- We propose Convolutional Quadratic neural networks (CQNN) for representation learning and demonstrate its application for image classification.
- We implement popular image classification architectures that consist of convolutional layers with quadratic neurons as processing units.
- We compare the image classification performance of the CQNN version against the existing architectures.

II. Related Work

CNNs: LeNet-5 [22] had a modular structure that allowed for scaling in multiple dimensions. AlexNet [21] achieved state-of-the-art recognition in the Imagenet Large Scale Visual Recognition Challenge (ILSVRC) in 2012. Following this, researchers have pursued multiple directions to improve recognition and classification accuracy on various datasets. ZFNet[41] optimized the kernel sizes (7X7), which improve the performance of AlexNet that used a combination of filter sizes. Lin *et al.* introduced the Network in Network (NIN) architecture that builds on micro-neural networks (a multi-layer perceptron network) with complex structure, thus introducing non-linearity to CNN layers which otherwise carry out linear operations [25]. This created denser layers that allowed for better representation of features for classification. Visual Geometry Group (VGG) scaled CNNs vertically to build deeper networks. VGG-11, VGG-16, and VGG-19 were introduced consisting of 11, 16, and 19 layers, respectively. However, the transition to deeper layers (16 to 19) introduced the vanishing gradient problem. GoogLeNet [36] introduced inception layers, where each layer is scaled horizontally using multiple convolutions layers with varying filter sizes.

He *et al.* [15] proposed skipped connections as a remedy for the vanishing gradient problem, which allowed them to build deeper networks. They introduced Residual networks (ResNet) consisting of 34, 50, 101, and 152 layers. Huang *et al.* [16] extended the idea of skipped connections and introduced DenseNet where the output of each layer is passed on to all of the following layers. Larsson *et al.*proposed Fractal networks as an alternative to residual networks, where each layer consists of parallel convolutional layers that aggregate results using join layers (element-wise mean). Networks consisting of 40 and 60 layers have shown comparable performance to ResNets. Other research contributions, such as dropout layers [34], batch normalization [15], data augmentation techniques [30], etc. have enabled the success of CNNs.

Most works have attempted to increase the representational capacity of CNN layers by scaling them along various dimensions: vertically (VGG), horizontally (GoogLeNet), and density (NiNs). We attempt to scale the neurons in the CNNs by employing higher-order functions. The underlying hypothesis is that equipping neurons with quadratic functions will improve the feature extraction capability of CNNs and will subsequently improve classification and recognition.

Bilinear CNN: Our work is closely related to bilinear classification models. Tanenbaum and Freeman [31] proposed bilinear model to capture variations in style and content of images. Pirsiavash [31] described a bilinear recognition algorithms for pedestrian detection, where the classifier is a product of two low rank matrices. Lin *et al.* [26] proposed a bilinear CNN for image classification, the model extracts features from two linear CNN networks, and then performs an outer product to create bilinear features, which are then used for classification. However the output of the outer product of two N dimensional features produces a vector of size N^2. A fully connected layer for large values of N requires a large number of parameters. Bilinear CNN's have shown to perform well for a wide variety of applications [2], [38].

Quadratic neural networks: Higher-order neural networks have been a topic of research since the 1990s. The linear neural function is a weighted sum (Sigma) of inputs. Giles and Maxwell [13] introduced Sigma-Pi networks where the function involved an additional term - a product (Pi) of inputs term - in the neural function. A variation of second-order function was used in the Pi-Sigma network that reduced the number of parameters. Milenkovic *et al.* [27] included the squared term as well in the quadratic functions. DeClaris and Su [6] proposed a quadratic junction and discussed the training properties of a multilayer feed-forward neural network and latter used it for clustering analysis. Cheng and Leung [3] explored the idea of rotational quadratic neural networks, which include inputs that are processed through a rotation function.

More recently with the renewed interest in neural networks, Yaparla *et al.* [11] demonstrated the feasibility to perform pattern classification using quadratic neurons. Fan *et al.* [8] revisited the quadratic neuron function, and later used this function in the CNN layer of a UNET [32] architecture to reduce metal artifacts in an x-ray computed tomography (CT) [9]. In this paper, we explore quadratic convolutional layers further for extracting image features and subsequently for image classification.

III. Convolutional Quadratic Neural Networks

Let $X^T = \{x_1, x_2, ..., x_d\}$ be the input vector with d dimensions, where $\{\}^t$ is the transpose. A neuron performing a linear function is represented as

$$f(X) = WX + b, \tag{1}$$

where $W = \{w_1, w_2, .., w_d\}$ are the weights and b is the bias. A generalized quadratic functions for a neuron can be defined as.

$$q(X) = X'^T W_q X', \tag{2}$$

where $X'^T = \{X^T|1\} = \{x_1, x_2, ..., x_d, 1\}$ is the augmented vector, and

$$W_q = \begin{bmatrix} w'_{1,1} & w'_{1,2} & . & . & w'_{1,d+1} \\ w'_{2,1} & . & . & . & . \\ . & . & . & . & . \\ w'_{d,1} & . & . & . & w'_{d+1,d+1} \end{bmatrix} \tag{3}$$

are the weights. Equation 2 is mathematically equivalent to the quadratic neuron defined in [9]. While it is logical to pursue research toward building higher order neural networks, these studies have largely been abandoned due to the exponential growth in parameters with input dimensions. Let d be the dimensions of the input vector; then a linear neuron is modeled using $(d+1)$ parameters. A generalized quadratic neuron as defined in Equation 2 has $(d+1)^2$ parameters. A multi-layer neural network with l layers, and with $\{n_1, n_2, ...n_l\}$ neurons in each layer has $(d+1)n_1 + \Sigma_{i=1}^{l}(n_i+1)n_{i+1}$ weights. A quadratic neural network with the same number of layers and neurons would have $(d+1)^2 n_1 + \Sigma_{i=1}^{l}(n_i+1)^2 n_{i+1}$ weights. To put things into perspective, consider an input vector of 1000 dimensions, a layer of 100 linear neurons has 100100 parameters. If the 100 linear neurons were replaced by 100 quadratic neurons, the number of parameters would increase to $100, 200, 100$. Considering networks where multiple layers would be stacked together, such networks are considered as large and time-consuming optimization problem given the current (2020) computational power.

CQNN: We propose to use quadratic neurons for representation learning rather than for classification. The idea is to use quadratic neurons for image representation learning and subsequently use linear neurons for classification. We build networks with a combination of quadratic and linear neurons for image classification where the convolutions layers used for extracting image representation are constructed using quadratic neurons, and the dense layers used in classification at the latter stages use linear neurons. The rationale for this choice is two-fold, first, the image features are extracted using higher-order functions allowing the network to learn complex representation (over linear functions), and the second is that the input dimension to the neurons is limited by the kernel size, hence producing networks with a reasonable number of parameters. Most CNNs have elected to use smaller kernel sizes for convolution (3X3, 5X5, etc.). Let I be an image and consider a filter of size NXN used in the convolutional layer of CNNs. Let $X_{i,j}^T = \{x_1, x_2, ..., x_{N^2}\}$ be the pixels in the receptive field of the image spanned by the kernel at location (i, j). Then the output of the quadratic neuron is computed as

$$q(X_{i,j}) = X'^T_{i,j} W_q X'_{i,j}. \tag{4}$$

A CNN layer with linear neurons consisting of k kernels of size NXN has $k(N^2+1)$ parameters, and a quadratic kernel has $k(N^2+1)^2$ parameters, however, in most cases, N is restricted to smaller numbers $(1, 3, 5)$. Hence allowing us to build quadratic networks with a manageable increase in the count of parameters.

IV. Experimental Design

We design experiments to quantify the capacity of CQNNs to perform image classification and compare them with CNNs. We consider two popular CNNs and construct CQNNs with similar architecture, and compare their performance.

Architectures: We consider AlexNet and ResNet. While there have been numerous CNN architectures that have been proposed since the introduction of AlexNet [21] in 2012, our objective is to conduct experiments to ascertain the effect of using quadratic convolutional layers in a CNN. AlexNet is a basic architecture, and replacing the neurons in the CNN layers with quadratic neurons will allow us to attribute the change in performance with the implication of using quadratic neurons. Furthermore, it is interesting to understand if the ideas proposed to improve the performance of regular CNNs would translate to CQNNs. ResNet introduces two concepts, skipped connections and batch normalization. Using ResNet allows us to understand if these benefits translate when using quadratic convolutional layers. For each architecture, we construct four variants.

1) **CQNN (Proposed):** The overall architecture of these networks is similar to the original CNN architectures. They utilize the same number of layers, each layer has an equal number of filters, and the dense layers are exactly the same. CQNNs have two key differences: first, the neurons

in the convolutional layer are replaced with quadratic neurons proposed in Equation 2, and second, the networks use Exponential Linear Units (ELu) for activation instead of the ReLu used in the original architectures.

2) **CQNNFan:** Fan *et al.* [8] proposed a quadratic neuron for neural networks. We use a variant of this function (for computational efficiency) that contains a quadratic term and a linear term, defined as:

$$f(x) = (W_1X + b_1)(W_2X + b_2) + (W_3X + b_3) \quad (5)$$

The overall architecture is similar to CQNNs, except for the neurons use the function defined in Equation 5.

3) **Bilinear Networks (BiCNNs)**: For comparison, we compare the performance of CQNN with bilinear CNNs. The bilinear implementation consists of two parallel networks consisting of only convolutional layers. The features extracted from the networks are used to compute an outer product. We finally perform classification on the outer product. For AlexNet and ResNet, we remove the final bottleneck network and create two parallel networks. The outer product of the features from the two networks is input to a dense layer for classification.

4) **Horizontally Scaled Networks (HSCNNs):** The quadratic variants of the CNN's have a larger number of parameters. Increasing the number of parameters either by scaling the network vertically (stacking more layers) or horizontally (more filters in each layer) results in an increase in performance. More parameters imply the capability to learn more complex functions. Tan and Le [37] conducted a study on the effect of scaling neural networks in various dimensions (depth, width, and resolutions) and demonstrated this idea. To perform a fair comparison, we scale each CNN horizontally (i.e. increase the number of filters) to create horizontally scaled versions, such that they have the same number of parameters as the CQNN versions. This would allow us to quantify the effect of increasing the parameters by scaling the network horizontally vs. scaling the order of the neural functions in each layer.

Table I (a) show the architecture of a simple AlexNet, it's CQNN, and the horizontally scaled counterparts. The AlexNet and quadratic AlexNet have 32 filters in the first two layers, 64 filters in layers three and four, while the horizontally scaled version uses a scaling factor of 15 and has 480 filters in the first two layers, and 960 filters in third and fourth. The dense layers are the same for all three AlexNet variants, as we employ quadratic neurons for representation learning or feature extraction and not for classification. Similarly, we employ a ResNet architecture with 21 layers shown in Table I (b) (similar to ResNet-20 architecture). The architecture uses three ResNet stacks; each stack begins with a convolutional layer that has a stride value of 2 and performs downsampling, followed by 3 sets of two convolutional layers. Each convolution layer is followed by batch normalization and activation layer. The number of filters is doubled for each stack that is deeper into the network. The quadratic version has the exact same structure, but each neuron performs a quadratic operation. We use a scaling factor of 22 to create the horizontally scaled version. To create CQNNFan versions we replace the neurons in CQNN to perform the function defined in Equation 5. To create the bilinear versions, we create two parallel networks using the CNN architectures and remove the final bottle next layers. Then the outer product of the features from the parallel network are input a final fully connected classification layer.

Dataset: We conduct experiments on two image classification datasets, Cifar-10 and Cifar-100. Both datasets contain a total of 60,000 images. Cifar-10 has 10 classes with 6,000 images per class. Each class has 5000 train images, and 1000 test images. Cifar-100 is similar to Cifar 10, however, it is much more challenging to solve with 100 classes. There are 600 images available for each class, where 500 are used for training and 100 for testing. We train and test the capability of AlexNet, ResNet, and their variants to classify images and compare their training behavior and performance.

V. Results

Each network, AlexNet, and ResNet and their horizontally scaled and quadratic variants are trained on the Cifar-10 and Cifar-100 datasets for 200 epochs. We compare their training behavior and the testing accuracy to ascertain the implication of horizontal scaling vs quadratic scaling of the neural function.

Training Behavior

Cifar-10 dataset: Figure 2 (a), (b) shows the train accuracy and loss of the AlexNet and its variants on the Cifar-10 dataset. AlexNet shows to overfit the data after 100 epochs, an optimal learning rate or limiting the training epochs can address the problem. The BiCNN and CQNNFan versions of the AlexNet shows a keen improvement in the learning process. The HSCNN and CQNN versions outperform the other variants. CQNN performs slightly better than HSCNN variant. Figure 2 (c), (d) shows the training accuracy and loss on the ResNet variants. Overall ResNet models perform better than AlexNet models. HSCNN and BiCNN variants of ResNet show no improvement in the learning process. However, the quadratic version shows a clear improvement in the training accuracy and loss.

Cifar-100 dataset: Cifar-100 is a more complex dataset than the Cifar-10. The effects of HSCNN and CQNNs are much more evident in this case. Figure 3 shows the training behavior on the Cifar-100 dataset. The HSCNN and CQNN of AlexNet shows a keen improvement in the training accuracy and loss. HSCNN and BiCNN of version of ResNet on the Cifar-10 showed little improvement, however, the same network shows an improvement in learning the complex Cifar-100 dataset. In both cases, the quadratic version shows an improvement in the training accuracy and achieves a lower cross-entropy loss. Overall, scaling the order of neurons shows an improvement over horizontal scaling.

Table I. Architecture of (a) AlexNet; (b) ResNet, and their quadratic, and horizontally scaled variants, where QConv2D is a 2d convolution layer with quadratic neurons, QResStack is a ResNet stack with ResNet layers consisting of quadratic convolutional layers, s indicates the scaling factor, and ||3X3, 16, 2|| X 3 is a layer consisting of 16 filters of size 3X3, and the result is computed using 2 stride, and the last number implies that the layer is repeated 3 times.

(a) AlexNet Variants

CNN	description	CQNN	description	HSCNN (s=15)	description
Conv2D	||3X3, 32, 1||	QConv2D	||3X3, 32, 1||	Conv2D	||3X3, 480, 1||
Activation	ReLu	Activation	ELu	Activation	ReLu
Conv2D	||3X3, 32, 1||	QConv2D	||3X3, 32, 1||	Conv2D	||3X3, 480, 1||
Activation	ReLu	Activation	ELu	Activation	ReLu
Maxpooling	2X2	Maxpooling	2X2	Maxpooling	2X2
Dropout	0.25 rate	Dropout	0.25 rate	Dropout	0.25 rate
Conv2D	||3X3, 64, 1||	QConv2D	||3X3, 64, 1||	Conv2D	||3X3, 960, 1||
Activation	ReLu	Activation	ELu	Activation	ReLu
Conv2D	||3X3, 64, 1||	QConv2D	||3X3, 64, 1||	Conv2D	||3X3, 960, 1||
Activation	ReLu	Activation	ELu	Activation	ReLu
Maxpooling	2X2	Maxpooling	2X2	Maxpooling	2X2
Dropout	0.25 rate	Dropout	0.25 rate	Dropout	0.25 rate
Flatten	-	Flatten	-	Flatten	-
Dense	512 Neurons	Dense	512 Neurons	Dense	512 Neurons
Activation	ReLu	Activation	ELU	Activation	ReLu
Dropout	0.25 rate	Dropout	0.25 rate	Dropout	0.25 rate
Dense	-	Dense	-	Dense	-

(b) ResNet Variants

CNN	description	CQNN	description	HSCNN (s=22)	description
ResStack	$\left[\begin{array}{\|c\|c} 3X3, 16, 2 & X1 \\ \begin{array}{c} 3X3, 16, 1 \\ 3X3, 16, 1 \end{array} & X3 \end{array}\right]$	QResStack	$\left[\begin{array}{\|c\|c} 3X3, 16, 2 & X1 \\ \begin{array}{c} 3X3, 16, 1 \\ 3X3, 16, 1 \end{array} & X3 \end{array}\right]$	ResStack	$\left[\begin{array}{\|c\|c} 3X3, 352, 2 & X1 \\ \begin{array}{c} 3X3, 352, 1 \\ 3X3, 352, 1 \end{array} & X3 \end{array}\right]$
ResStack	$\left[\begin{array}{\|c\|c} 3X3, 32, 2 & X1 \\ \begin{array}{c} 3X3, 32, 1 \\ 3X3, 32, 1 \end{array} & X3 \end{array}\right]$	QResStack	$\left[\begin{array}{\|c\|c} 3X3, 32, 2 & X1 \\ \begin{array}{c} 3X3, 32, 1 \\ 3X3, 32, 1 \end{array} & X3 \end{array}\right]$	ResStack	$\left[\begin{array}{\|c\|c} 3X3, 704, 2 & X1 \\ \begin{array}{c} 3X3, 704, 1 \\ 3X3, 704, 1 \end{array} & X3 \end{array}\right]$
ResStack	$\left[\begin{array}{\|c\|c} 3X3, 64, 2 & X1 \\ \begin{array}{c} 3X3, 64, 1 \\ 3X3, 64, 1 \end{array} & X3 \end{array}\right]$	QResStack	$\left[\begin{array}{\|c\|c} 3X3, 64, 2 & X1 \\ \begin{array}{c} 3X3, 64, 1 \\ 3X3, 64, 1 \end{array} & X3 \end{array}\right]$	ResStack	$\left[\begin{array}{\|c\|c} 3X3, 1408, 2 & X1 \\ \begin{array}{c} 3X3, 1408, 1 \\ 3X3, 1408, 1 \end{array} & X3 \end{array}\right]$
AveragePooling	2X2	AveragePooling	2X2	AveragePooling	2X2
Flatten	-	Flatten	-	Flatten	-
Dense	-	Dense	-	Dense	-

Performance: Table II shows the testing accuracy of the two datasets and the number of parameters used in each model. In all the cases, the quadratic version uses a lower number of parameters than the horizontally scaled versions. The horizontally scaled AlexNet variant uses $0.8M$ more parameters than the quadratic version, while the horizontally scaled Resent variant use over $2M$ more parameters than the quadratic version. Overall, the quadratic version shows a clear improvement in performance over the horizontally scaled versions.

Cifar-10: On this dataset, the BiCNN AlexNet improves the performance by 13%, CQNNFan AlexNet improves the performance by 24%, and the HSCNN and the CQNN AlexNet versions perform equally well and show a 25% increase in the accuracy. The BiCNN and CQNNFan version of ResNet show no improvement in accuracy. The ResNet CQNN version shows a 6% improvement in accuracy compared to the HSCNN that shows a 2% improvement.

Cifar-100: For this dataset, the CNN layers of AlexNet produce a feature vector of length 2304, and the outer product has 2304^2 dimensions. Creating an dense layer for 100 classes for BiCNN would requires a large number of parameters that makes is tedious to train. We add a additional CNN layer with 32 Filters at the end to create a feature vector of length 800 and use it to create the BiCNN. However, the network does not perform any better than the AlexNet. CQNNFan increases the performance by 17%. The advantage of using the proposed quadratic function is more evident in the results for the Cifar-100 dataset. The quadratic version of the AlexNet shows 22% improvement in accuracy compared to the horizontally scaled version that has a 15% improvement of the accuracy. The HSCNN, BiCNN and CQNNFan versions of ResNet, does not show an improvement in the testing accuracy on this dataset. However, CQNN version of ResNet shows a 6% improvement of the accuracy.

These results illustrate the effect of employing the proposed quadratic neurons. We train and test the existing architectures on the Cifar-100 dataset: VGG16[33], Inception ResNet [35], DenseNet [16], and Xception [4]. Most architectures are optimized to perform well on larger image sizes. Since the Cifar-100 dataset provides images of size $32X32$ and resolution has an effect on the performance of these algorithms, similar to the quadratic architectures, we train these networks from scratch with an input size of 32X32. All the architectures are trained

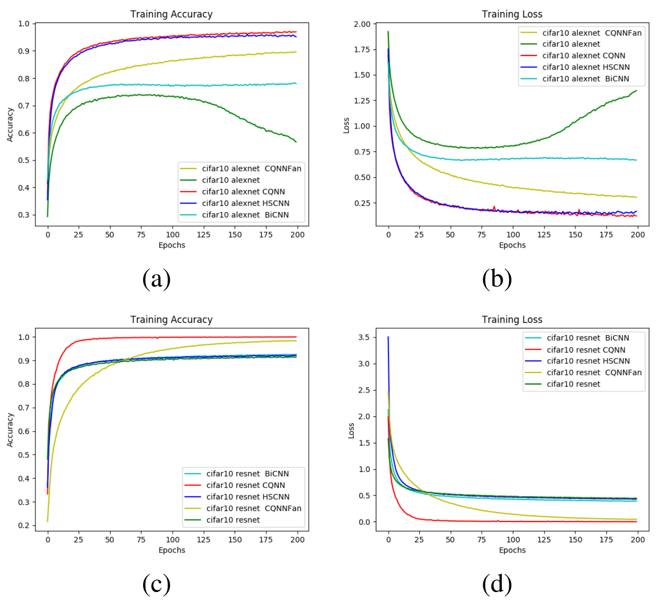

Fig. 2. Accuracy and Training Loss on Cifar-10 dataset: (a) Accuracy of AlexNet variants; (b) Crossentropy loss of AlexNet variants; (c) Accuracy of ResNet variants; (d) Crossentropy loss of ResNet variants

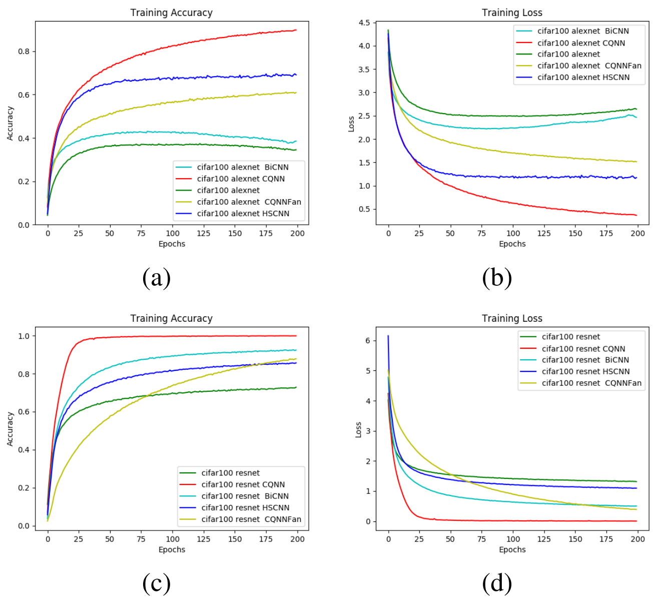

Fig. 3. Accuracy and Training Loss on Cifar-100 dataset: (a) Accuracy of AlexNet variants; (b) Crossentropy loss of AlexNet variants; (c) Accuracy of ResNet variants; (d) Crossentropy loss of ResNet variants

for 200 epochs using RMSProp with a learning rate of 1e-3, except densnet121 and Xception that are optimized using Stochastic Gradient Descent (SGD), as they failed to learn steadily using RMSProp. A multitude of training tricks can improve the accuracy [29] considerable, we refrain from using these methods and focus on the ability of the neurons to learn representation and classification. Table III shows a comparison of the count of parameters used by each architecture and the accuracy. Experiments show that using quadratic neurons in a

Table II. Testing accuracy of Existing architectures on Cifar-100 dataset.

Dataset	Model	Parameters	Accuracy
Cifar - 10	AlexNet	1.2M	0.63
	AlexNet HSCNN	32.2M	**0.88**
	AlexNet BiCNN	53.2M	0.79
	AlexNet CQNNFan	2.2M	0.87
	AlexNet CQNN	31.4M	**0.88**
	ResNet	.2M	0.85
	ResNet HSCNN	130.6M	0.87
	ResNet BiCNN	0.5M	0.84
	ResNet CQNNFan	0.8M	0.84
	ResNet CQNN	128M	**0.91**
Cifar - 100	AlexNet	1.2M	0.41
	AlexNet HSCNNs	32.2M	0.56
	AlexNet BiCNN	64M	0.41
	AlexNet CQNNFan	2.3M	0.58
	AlexNet Quadratic	31.4M	**0.63**
	ResNet	.2M	0.60
	ResNet HSCNN	130.8M	0.59
	ResNet BiCNN	0.9M	0.51
	ResNet CQNNFan	0.8M	0.55
	ResNet CQNN	128M	**0.66**

Table III. Testing accuracy of current methods on Cifar-100 dataset.

Model	Optimizer	Params	Acc.
VGG16[33]	RMSProp (lr=1e-3)	34M	0.44
Inception ResNet[35]	RMSProp (lr=1e-3)	54.5M	0.51
Densenet121[16]	SGD (lr=1e-2, d=1e-6,nm=0.9)	12.5M	0.52
Xception[4]	SGD (lr=1e-2, d=1e-6,nm=0.9)	21M	0.60

simple architecture such as AlexNet can outperform existing complex architectures.

Ablation Study: The quadratic versions have two key differences, first, they use quadratic neurons, and second, they use ELu activations, unlike the linear version that use ReLu.

The quadratic version outperforms the linear version, and studies have shown that using Elu activation has advantages over ReLu as an activation function [28]. We conduct experiments on the Cifar-100 dataset to understand the contribution of the quadratic neurons and ELu activation towards the performance gain. We train two alternative networks based on the AlexNet architecture, the first uses quadratic neurons and ReLu activation functions, and the latter uses linear neurons and Elu activation. Table IV shows that using ELu activation function instead of ReLu with linear neurons increases the performance by 8%. Using quadratic neurons with ReLu activation also produces an 8% increase in performance. However, using quadratic neurons with ELu activation produces a 22% increase in performance. We conclude that ReLu activation

Table IV. Accuracy of AlexNet variants with combination of linear/quadratic neurons and ReLu/Elu activation functions.

Neuron\Activation	ReLu	Elu
Linear neurons	0.41	0.49
Quadratic neurons	0.49	**0.63**

function is unsuitable when using quadratic neurons, rather quadratic neurons when combined with ELu activation perform better.

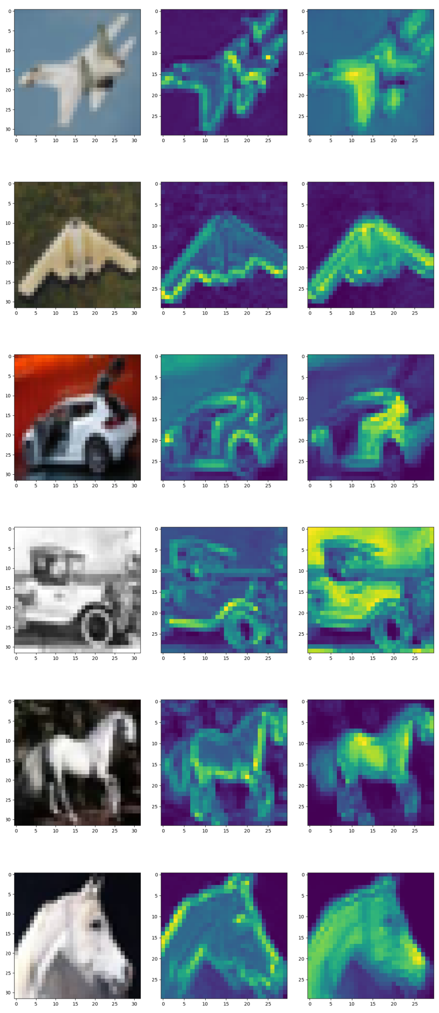
Fig. 4. Activations from first convolutional layer of AlexNet variants: (left) image, (middle) linear (right) quadratic

Discussion and Future Work: The convolutional layers are designed to extract features followed by which the dense layer performs classification (by computing a posterior). Figure 4 shows the features extracted from the first convolutional layer in the linear and quadratic AlexNet architectures. The features extracted by each filter (after activation) are summed together to form the activation images. The left column shows the input images, the middle column shows the activations from the linear version and the right shows the activation from the quadratic version. It is evident that linear convolutions layers learn to extract edges in the initial stages that are later used for classification. On the other hand, the quadratic version is capable of extracting the whole object of interest and then encode its features for classification. In an environment with multiple objects and cluttered background, edges alone are not sufficient to perform classification.

Quadratic neurons show a clear improvement in extracting relevant features for classification, however, there is much work needed toward making them practical. Experiments show that the quadratic version of the AlexNet consumes 25 times more parameters and floating-point operations (FLOPS) over the linear version, and the ResNet20 like architecture consumes 220 times more parameters and flops.

VI. Conclusion

We have proposed a quadratic function for higher-order neurons and constructed convolutional layers consisting of them. We implemented CQNN equivalents of AlexNet and ResNet architecture consisting of quadratic convolutional layers and trained them for the task of image classification. Experiments on two public datasets, Cifar-10 and Cifar-100 show a clear improvement in training behavior and testing accuracy compared to conventional CNN architecture. We perform a comparison against existing networks, and experiments show that CQNN improves the accuracy of image classification.

References

[1] Matteo Carandini. What simple and complex cells compute. *The Journal of physiology*, 577(Pt 2):463, 2006.

[2] Hesen Chen, Jingyu Wang, Qi Qi, Yujian Li, and Haifeng Sun. Bilinear cnn models for food recognition. In *2017 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, pages 1–6. IEEE, 2017.

[3] KF Cheung and CS Leung. Rotational quadratic function neural networks. In *[Proceedings] 1991 IEEE International Joint Conference on Neural Networks*, pages 869–874. IEEE, 1991.

[4] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258, 2017.

[5] Dan CireşAn, Ueli Meier, Jonathan Masci, and Jürgen Schmidhuber. Multi-column deep neural network for traffic sign classification. *Neural networks*, 32:333–338, 2012.

[6] Nicholas DeClaris and Mu-chun Su. A novel class of neural networks with quadratic junctions. In *Conference Proceedings 1991 IEEE International Conference on Systems, Man, and Cybernetics*, pages 1557–1562. IEEE, 1991.

[7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[8] Fenglei Fan, Wenxiang Cong, and Ge Wang. A new type of neurons for machine learning. *International journal for numerical methods in biomedical engineering*, 34(2):e2920, 2018.

[9] Fenglei Fan, Hongming Shan, Lars Gjesteby, and Ge Wang. Quadratic neural networks for ct metal artifact reduction. In *Developments in X-Ray Tomography XII*, volume 11113, page 111130W. International Society for Optics and Photonics, 2019.

[10] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 36(4):193–202, 1980.

[11] Yaparla Ganesh, Rhishi Pratap Singh, and Garimella Rama Murthy. Pattern classification using quadratic neuron: An experimental study. In *2017 8th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, pages 1–6. IEEE, 2017.

[12] Robert Geirhos, David HJ Janssen, Heiko H Schütt, Jonas Rauber, Matthias Bethge, and Felix A Wichmann. Comparing deep neural networks against humans: object recognition when the signal gets weaker. *arXiv preprint arXiv:1706.06969*, 2017.

[13] C Lee Giles and Tom Maxwell. Learning, invariance, and generalization in high-order neural networks. *Applied optics*, 26(23):4972–4978, 1987.

[14] Robert M Haralick, Karthikeyan Shanmugam, and Its' Hak Dinstein. Textural features for image classification. *IEEE Transactions on systems, man, and cybernetics*, (6):610–621, 1973.

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[16] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[17] Yongzhen Huang, Kaiqi Huang, Yinan Yu, and Tieniu Tan. Salient coding for image classification. In *CVPR 2011*, pages 1753–1760. IEEE, 2011.

[18] David H Hubel and Torsten N Wiesel. Receptive fields of single neurones in the cat's striate cortex. *The Journal of physiology*, 148(3):574–591, 1959.

[19] Forrest Iandola, Matt Moskewicz, Sergey Karayev, Ross Girshick, Trevor Darrell, and Kurt Keutzer. Densenet: Implementing efficient convnet descriptor pyramids, 2014.

[20] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. The cifar-10 dataset. *online: http://www. cs. toronto. edu/kriz/cifar. html*, 55, 2014.

[21] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[22] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[23] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[24] James R Leger and Sing H Lee. Image classification by an optical implementation of the fukunaga–koontz transform. *JOSA*, 72(5):556–564, 1982.

[25] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.

[26] Tsung-Yu Lin, Aruni RoyChowdhury, and Subhransu Maji. Bilinear cnn models for fine-grained visual recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 1449–1457, 2015.

[27] Srdjan Milenkovic, Zoran Obradovic, and Vanco Litovski. Annealing based dynamic learning in second-order neural networks. In *Proceedings of International Conference on Neural Networks (ICNN'96)*, volume 1, pages 458–463. IEEE, 1996.

[28] Dabal Pedamonti. Comparison of non-linear activation functions for deep neural networks on mnist classification task. *arXiv preprint arXiv:1804.02763*, 2018.

[29] Gabriel Pereyra, George Tucker, Jan Chorowski, Łukasz Kaiser, and Geoffrey Hinton. Regularizing neural networks by penalizing confident output distributions. *arXiv preprint arXiv:1701.06548*, 2017.

[30] Luis Perez and Jason Wang. The effectiveness of data augmentation in image classification using deep learning. *arXiv preprint arXiv:1712.04621*, 2017.

[31] Hamed Pirsiavash, Deva Ramanan, and Charless C Fowlkes. Bilinear classifiers for visual recognition. In *Advances in neural information processing systems*, pages 1482–1490, 2009.

[32] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[33] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[34] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[35] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-first AAAI conference on artificial intelligence*, 2017.

[36] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[37] Mingxing Tan and Quoc V Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*, 2019.

[38] Chaofeng Wang, Jun Shi, Qi Zhang, and Shihui Ying. Histopathological image classification with bilinear convolutional neural networks. In *2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 4050–4053. IEEE, 2017.

[39] Jinjun Wang, Jianchao Yang, Kai Yu, Fengjun Lv, Thomas Huang, and Yihong Gong. Locality-constrained linear coding for image classification. In *2010 IEEE computer society conference on computer vision and pattern recognition*, pages 3360–3367. IEEE, 2010.

[40] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.

[41] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.